Group T: Jirad Al Massri (32323), Nour-Hamed-Raafat Hamed (31348) and Tarik Aydin (26751)

Chicken Check

1. Introduction

Written by Nour-Hamed-Raafat Hamed

In monitoring chickens there's a lot of different parameters are measured to gain insight on chicken movement and behaviour and a more complex objective of monitoring health indicators like feeding and nesting habits. Where are the chicken? In the chicken coop? Sitting on the nest? Breeding? Outside?.

Not only monitoring the chickens but also to track the long-term development of behaviour tracking for example it can help to register alarming trends and changes early on such that appropriate actions can be taken. This makes chicken checker a topic of importance in sustainable farming The Chicken Checker project uses Radio Frequency Identification (RFID) technology to monitor and track the activity of chickens on a farm as shown in figure 1. By connecting two antennas to a ESP32 and placing them at the start and end of the entrance of the chicken coop Microcontrollers are a valuable tool for farm owners and agricultural industries as they are cheap, small in size and easy to set up and use. They offer a lot of possibilities to help maintain a sustainable farming environment like reading results and transmitting the data to database where it can be assessed. The aim of this project is to scheme and program a small monitoring station for a chicken coop and learn how such technology works and its upsides and downside and how to set it up.

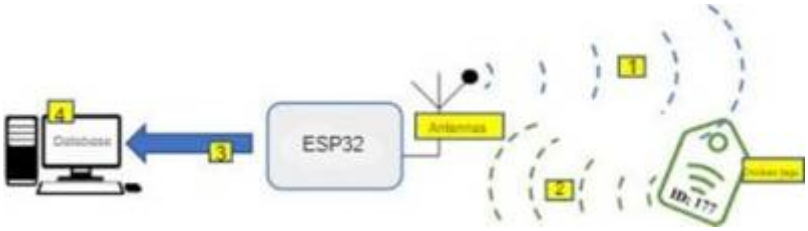


Figure 1 scheme of hardware on the farm

2. Materials & Methods

Written by Tarik Aydin

For this project the following materials were used:

1) PN532 Adafruit board

The PN532 Adafruit board operates on a 13.56MHz frequency. This board creates an electromagnetic field which can reach approximately 10cm range according to Adafruit, tested around 7.5cm where reached. In total two board were used for this project, the boards act as gates which detect a nfc card

when the nfc card passes the gates.

2) NFC cards type ISO14443 tags
The NFC cards acted passing chickens, the card consist of a small chip which is connected to a wire which is curled

3) Esp32 S3 Dev Module
The Esp32 S3 Dev Module worked as the brain of this project, the microcontroller controls the two PN532 boards and also sends the data to the computer.

4) Jumper wires
Jumper wires connect the parts used electronically. In total five male to male Jumper wires and twelve male to female jumper wires were used.

5) Bread board
The bread board was used to as a base to connect all the parts together.

6) USB-A to USB 2.0 micro B cable
This cable was used a simple data transmission cable from the computers serial port to the Esp32 S3 Dev module to program the microcontroller and to execute the code written in Arduino. (1)

7) Soldering Iron
The PN532 boards were delivered with lose pins, in order to secure the pins in the wanted place an soldering iron was used.

8) ChatGPT
ChatGPT is an AI tool which is able to answer questions and enables for a faster development of the Arduino code to program the ESP32 microcontroller and the PN532 boards.

3. Results

Written by Tarik Aydin

The code used for this project:

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_PN532.h>

// Define the slave select (SS) pins for the PN532 modules
#define PN532_SS1 5
```

```
#define PN532_SS2 1

// Create PN532 SPI instances
Adafruit_PN532 nfc1(PN532_SS1);
Adafruit_PN532 nfc2(PN532_SS2);

struct UIDCounter {
  uint8_t uid[7];
  uint8_t uidLength;
  unsigned long count;
};

UIDCounter uidCounters1[10]; // Array to store UIDs and counts for PN532 #1
UIDCounter uidCounters2[10]; // Array to store UIDs and counts for PN532 #2

void setup(void) {
  Serial.begin(115200);
  Serial.println("Hello!");

  // Set up the SPI pins
  SPI.begin(13, 12, 11); // SCK, MISO, MOSI

  // Initialize both PN532 modules
  nfc1.begin();
  nfc2.begin();

  // Check if PN532 modules are ready
  uint32_t versiondata = nfc1.getFirmwareVersion();
  if (!versiondata) {
    Serial.println("Didn't find PN532 module #1");
    while (1);
  }

  versiondata = nfc2.getFirmwareVersion();
  if (!versiondata) {
    Serial.println("Didn't find PN532 module #2");
    while (1);
  }

  // Configure PN532 modules to read RFID cards
  nfc1.SAMConfig();
  nfc2.SAMConfig();

  // Initially set passive activation retries to zero (turn off antenna) for
both modules
  nfc1.setPassiveActivationRetries(0x00);
  nfc2.setPassiveActivationRetries(0x00);
}

void loop(void) {
  uint8_t success1;
```

```
  uint8_t uid1[7] = { 0 };  // Buffer to store the returned UID
  uint8_t uidLength1;  // Length of the UID (4 or 7 bytes depending on
ISO14443A card type)

  uint8_t success2;
  uint8_t uid2[7] = { 0 };  // Buffer to store the returned UID
  uint8_t uidLength2;  // Length of the UID (4 or 7 bytes depending on
ISO14443A card type)

  // Activate the first PN532 module and read
  nfc1.setPassiveActivationRetries(0x0A); // Set retries to 10 for the first
module
  digitalWrite(PN532_SS1, LOW);
  success1 = nfc1.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid1,
&uidLength1);
  digitalWrite(PN532_SS1, HIGH);
  nfc1.setPassiveActivationRetries(0x00); // Disable antenna for the first
module

  if (success1) {
    updateUIDCounter(uidCounters1, uid1, uidLength1);
    printUIDs(uidCounters1, "PN532 #1");
  }

  // Add a small delay to ensure the second board does not activate too soon
  delay(50);

  // Activate the second PN532 module and read
  nfc2.setPassiveActivationRetries(0x0A); // Set retries to 10 for the
second module
  digitalWrite(PN532_SS2, LOW);
  success2 = nfc2.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid2,
&uidLength2);
  digitalWrite(PN532_SS2, HIGH);
  nfc2.setPassiveActivationRetries(0x00); // Disable antenna for the second
module

  if (success2) {
    updateUIDCounter(uidCounters2, uid2, uidLength2);
    printUIDs(uidCounters2, "PN532 #2");
  }

  // Add a delay to control the loop timing

}

void updateUIDCounter(UIDCounter *uidCounters, uint8_t *uid, uint8_t
uidLength) {
  for (int i = 0; i < 10; i++) {
    if (uidCounters[i].count == 0) {
```

```
      // Empty slot, add new UID
      memcpy(uidCounters[i].uid, uid, uidLength);
      uidCounters[i].uidLength = uidLength;
      uidCounters[i].count = 1;
      return;
    } else if (memcmp(uidCounters[i].uid, uid, uidLength) == 0) {
      // UID found, increment counter
      uidCounters[i].count++;
      return;
    }
  }
}

void printUIDs(UIDCounter *uidCounters, const char *label) {
  Serial.print(label);
  Serial.println(" - Detected UIDs:");
  for (int i = 0; i < 10; i++) {
    if (uidCounters[i].count > 0) {
      Serial.print("UID: ");
      for (uint8_t j = 0; j < uidCounters[i].uidLength; j++) {
        if (uidCounters[i].uid[j] <= 0xF)
          Serial.print("0");
        Serial.print(uidCounters[i].uid[j], HEX);
        if (j < uidCounters[i].uidLength - 1)
          Serial.print(" ");
      }
      Serial.print(" Count: ");
      Serial.println(uidCounters[i].count);
    }
  }
  Serial.println("--------------");
}
```

Explanation of the code:

1. In the first part of the code the libraries and the chip select pins are defined.
    1. The libraries used are <Wire.h>, <SPI.h> and <Adafruit_PN532.h>
    2. The chip select pins are set at pin 5 for the first PN532 board and at pin 1 for the second PN532 board is set.
2. The Adafruit_PN532 nfc function creates instances which define which pin has to be used for which board. Two PN532 boards were used in this project so two instances have to be created. This enables to distinguish the two boards.
3. Then the struct UIDCounter is defined which enables to store read UIDs and also tracks how many times a specific UID has been read.
    1. The UIDCounter uidCounters1[10] and the UIDCounter uidCounters2[10] are storing for each board individually the UIDs detected for the corresponding board.
4. The SPI pins are defined in the set up.
    1. Pin 13 is defined for the serial clock.

   2. Pin 12 is defined for MISO.
   3. Pin 11 is defined for MOSI.
5. The nfc1.begin() and nfc2.begin() functions initializes the communication for the boards.
6. The code uint32_t versiondata = nfc1.getFirmwareVersion(); if (!versiondata) checks if the boards are connected correctly and returns an message in the serial monitor when the boards not found.
7. The functions nfc1.SAMConfig() and nfc2.SAMConfig() configures the board to use the Secure Access Module which enables to read RFID cards.
8. The last functions nfc1.setPassiveActivationRetries(0x00) and nfc2.setPassiveActivationRetries(0x00) in the void set up turns the antenna of the PN532 boards off.
9. The void loop contains variable storing functions success1, success2 as well as control of the two PN532 boards. As in 8. The function nfc.setPassiveActivationRetries() is used. Here in the beginning the function sets the PN532 board up to try ten times to read a nfc card. The digitalWrite(PN532_SS1, LOW) pulls the SPI signal down starting the SPI communication of the PN532 board. After digitalWrite(PN532_SS1, LOW) comes a variable which is called Success1 which is using the nfc1.ReadPassiveTargetID(), which will read the card when the card is close enough. After the Success1 comes the function digitalWrite(PN532_SS1, HIGH) which stops the SPI communication, after that the antenna is turned off again with the nfc.setPassiveActivationRetries() function. If a card is successfully read the functions updateUIDCounter and printUIDs , the updateUIDCounter updates the UID count at the perspective board and the function printUIDs will print the UIDs in the Serial monitor.
10. Small delay of 50ms is added to make sure that the second board starts bit later which ensures that the boards do not interfere with each other. Point nine is then done again just with changed variable names for the second board.

Generation of the code:

The code used for the project was generated in a conversation with ChatGPT, small adaptations were made in the code. The AI added a delay which wasn’t necessary, as well as the delay of the second board was set a 100ms which would lead to interference of the two boards, because the first board tries ten times a second to read a card. The delay for the second board was changed to 50ms.

Results in the serial monitor in Arduino

In figure 2 the serial monitor is displayed. Here is shown that the boards each have an individual counter for the different UIDs and that the UIDs have an individual counter as well. This enables to see in which direction the chicken walks, because one of the boards will be placed before the exit/entrance and the other board will be placed after the exit/entrance.

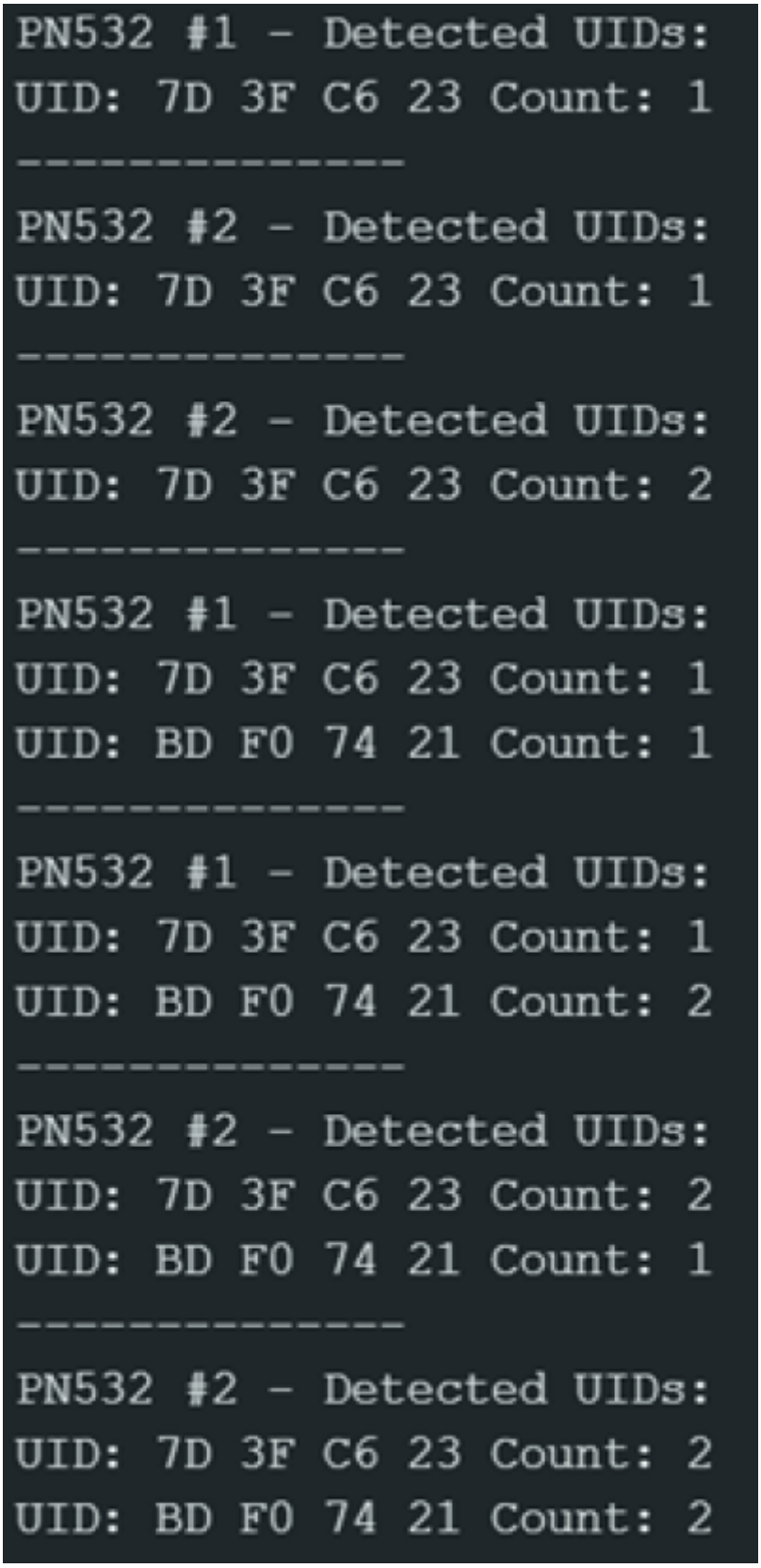


Figure 2 Serial Monitor in Arduino

Code that did not work for the project

The function nfc.readPassiveTargetID() is set as a standard to 0xFF [6], this means the function will

run for ever. This is a problem when two PN532 boards are used, because in the void loop the function will be needed two times once for board one and once for board two. When the code is set up in the way that the first board comes first and the second board comes second, code will run in the sequence first board one and then board two. So, this allows only to start reading with the first boards and the code also will only read one signal at the time at one board. One way to avoid to be stuck in the sequence is to use the freeRTOS library, which will allow to create tasks in Arduino. In each task the function nfc.readPassiveTargetID() is used and still operates in the standard setting 0xFF. Here the two tasks will work in parallel, avoiding to sequence the nfc card. Using the freeRTOS library only solves the sequencing issue, the bigger issue here is then that the two boars will be permanently on. Which is causing electromagnetic interference.

4. Discussion

Written by Jihad Al Massri

5. Conclusion

Written by Jihad Al Massri

6. References

1. https://www.anker.com/blogs/cables/how-to-identify-different-types-of-usb-cables-a-brief-guide
2. https://www.adafruit.com/product/789
3. https://www.adafruit.com/product/359
4. https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/hw-reference/esp32s3/user-guide-devkitc-1.html
5. https://openai.com/index/chatgpt/
6. https://github.com/adafruit/Adafruit-PN532/blob/master/Adafruit_PN532.cpp

First Draft Ideas

- RFID bird ringing, bird banding, tagging
- 13.56 MHz HF band (not LF nor UHF)
- https://learn.adafruit.com/adafruit-pn532-rfid-nfc?view=====all

About RFID

- https://www.rfid-grundlagen.de/
- https://www.smart-tec.com/de/faq

HF RFID Technology (13.56 MHz)

- Matt Rose and Jon Kurtz (2016): **NFC - A Closer Look**
  Future Electronics, original download

ISO 14443: Proximity Communication (typ. range: 7-15 cm)

1. **Frequency**: Both ISO 14443 and ISO 15693 operate at **13.56 MHz**.
2. **Purpose**: ISO 14443 is commonly used in contactless smart cards and NFC-enabled devices.
3. **Read Range**: ISO 14443 has a shorter read range of **7-15 cm** (approximately 2.8-5.9 inches).
4. **Data Transfer Rate**: It offers a higher data transfer rate with a **106 kbps** Baud rate.
5. **Applications**: ISO 14443 is suitable for financial transactions, access control, parking solutions, and attendance systems[2].

ISO 15693: Vicinity Communication (typ. range: < 100 cm)

1. **Purpose**: ISO 15693 defines the communication protocol for RFID tags operating at **13.56 MHz**.
2. **Read Range**: ISO 15693 provides an extended read range of **up to 1 meter** (approximately 3.3 feet).
3. **Data Transfer Rate**: It has a slower data rate with a **26 kbps** Baud rate.
4. **Applications**: ISO 15693 is useful for applications like inventory management, asset tracking, and libraries[15].

In summary, ISO 14443 is ideal for short-range applications like wireless payments, while ISO 15693 suits scenarios where longer read distances are necessary. [12]

Source: Conversation with Copilot, 30/06/2024

(1) NFC Reader Guide - Shop NFC. https://bing.com/search?q=iso+14443+vs+iso+15693.
(2) The difference between ISO15693 and ISO14443 - RFID Card.
https://www.rfidcard.com/the-difference-between-iso15693-and-iso14443/.
(3) Understanding the ISO15693 Protocol: An In-depth Look.
https://www.rfidlabel.com/understanding-the-iso15693-protocol-an-in-depth-look/.
(4) NFC Reader Guide - Shop NFC. https://www.shopnfc.com/en/content/31-nfc-reader-guide.
(5) What is ISO/IEC 15693? RFID Communication Standard for HF and NFC.
https://www.encstore.com/blog/5722-what-is-iso-iec-15693-rfid-communication-standard-for-hf-and-nfc.

Differences between ISO/IEC 15693-26 and ISO/IEC 15693-53 (Collected by COPILOT)

ISO/IEC 15693-26:

1. **Modulation Scheme**: ISO/IEC 15693-26 uses a subcarrier frequency of 423.75 kHz.
2. **Data Rate**: The data rate for ISO/IEC 15693-26 is **26.48 kbps.**

3. **Application**: This variant is commonly used in applications such as inventory management, asset tracking, and library systems.
4. **Read Range**: The read range is typically up to 1 meter (approximately 3.3 feet).

ISO/IEC 15693-53:

1. **Modulation Scheme**: ISO/IEC 15693-53 uses a subcarrier frequency of 847.5 kHz.
2. **Data Rate**: The data rate for ISO/IEC 15693-53 is **53.98 kbps.**
3. **Application**: It is also used for inventory management, asset tracking, and similar applications.
4. **Read Range**: Similar to ISO/IEC 15693-26, the read range is up to 1 meter.

In summary, both variants are part of the ISO/IEC 15693 standard and share similar use cases. The main difference lies in their modulation frequencies and data rates. □□□

Source: Conversation with Copilot, 30/06/2024

GAO RFID

- 13.56_MHz_HF_High-powered_RFID_Reader_233016.pdf
- Range: up to 90 cm with single antenna
- Speed: up to 60 readings per second
- Separation: 300 tags at once

NXP NFC Solutions

- **START HERE!**

NXP CLRC663 plus (CLRC66303HN) - High performance multi-protocol NFC frontend

https://www.nxp.com/products/rfid-nfc/nfc-hf/nfc-readers/clrc663-iplus-i-family-high-performance-nfc-frontends:CLRC66303HN

Source: **NXP CLRC663 product page**

Evaluation Board **CLEV6630B** (NXP)	Development Kit **OM26630FDKM** (NXP, including CLEV6630B)

- **NXP CLRC663 Product Page**

- Chip Mame: **CLRC66303HN** (I_T(max) 350-500 mA, up to 2 Watts)
- Eval Board Name: **CLEV6630B**
- Dev Kit Name: **OM26630FDKM** (includes CLEV6630B)

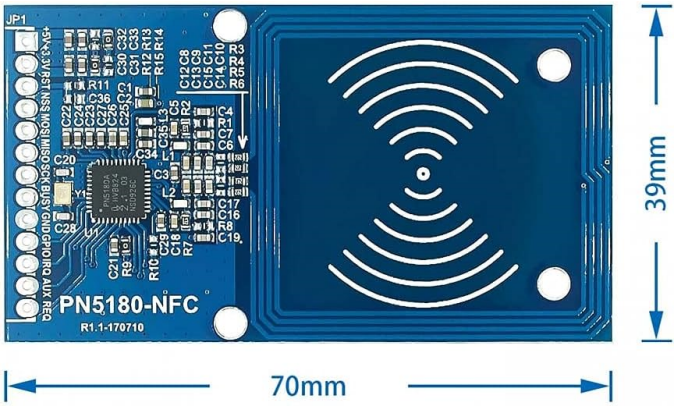
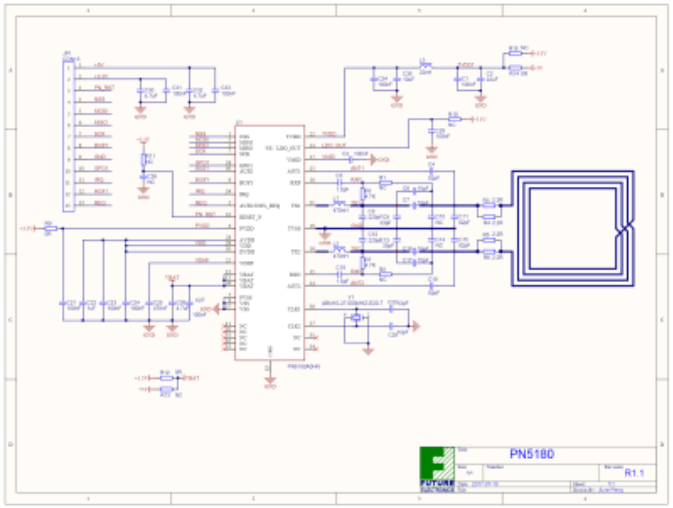
- Datasheet **CLRC663:** NXP CLRC663 - High performance multi-protocol NFC frontend CLRC663 and CLRC663 plus

- Datasheet **SLRC610:** NXP SLRC610 - High-performance ICODE frontend SLRC610 and SLRC610 plus
- AN11022 CLRC663 evaluation board quick start guide
- AN12657 Using the RC663 without library

NXP PN5180

OM25180FDK Dev. Kit + Extras from NXP	 NNEV5180BM product picture by Farnell.

- **NXP PN5180 Product Page**
- Chip Name: **PN5180B** (I_T(max) 250 mA) (PN5180A0HN???)
- Eval Board Name: **PNEV5180B**
- Dev Kit Name: **OM25180FDKM** (includes PNEV5180B)
- NXP NFC Antenna Tool
- Datasheet PN5180, **C3,C4** (Rev. 4.1, 2023-03-13): NXP PN5180A0xx/C3,C4 Rev. 4.1
- Datasheet PN5180, **C1,C2** (Rev. 3.6, 2018-05-07): NXP PN5180A0xx/C1/C2 Rev. 3.6

 A cheap PN5180 board (R1.1-170710) for Arduino-like projects.	 Schematic by Future Electronics (Download: A. Trappmann's Github)

PN5180 Application Notes

- AN11744 PN5180 Evaluation board quick start guide

- UM10954 PN5180 SW quick start guide

- AN11740 PN5180 Antenna design guide
- AN11741 How to design an antenna with DPC
- AN11906 Starting a product development with PN5180
- AN12650 Using the PN5180 without library
- AN12810 How to use the NanoVNA for the NFC reader antenna design

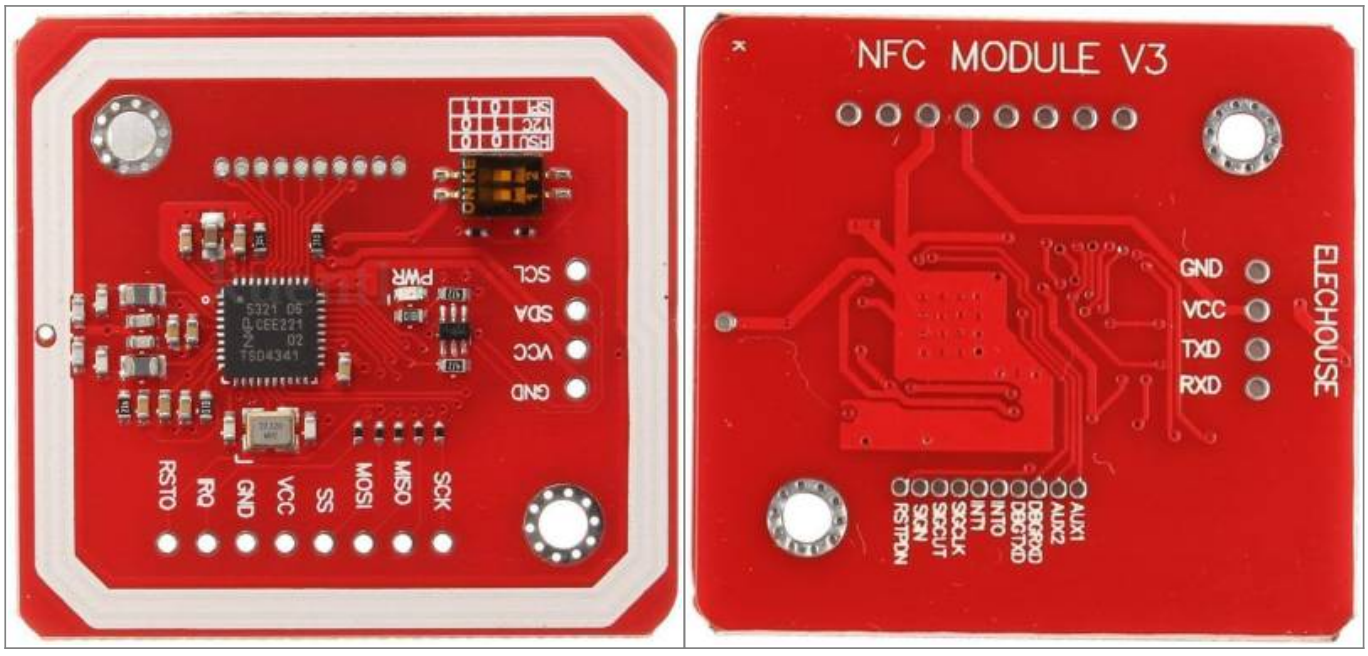
PN5180 Arduino Libraries

- https://github.com/ATrappmann/PN5180-Library
- https://github.com/L4M0S/PN5180-Library-14443
- https://github.com/playfultechnology/arduino-rfid-PN5180
- https://github.com/playfultechnology/PN5180-Library

PN532 Module V3 by Elechouse (5V, red board)

This is the cheap version (the red board with 8 + 4 pin headers / connectors) you still can buy everywhere. Drawback: Only the 5V is exposed to the connector and not the 3.3V output. The board has to be modified to be usable with a 3.3V voltage source, e.g. by desoldering the voltage regulator and building wire bridge from 3.3V to the power pin on the connector (origially named 5V).

- PN532_ Manual_V3.pdf
- https://github.com/elechouse/PN532



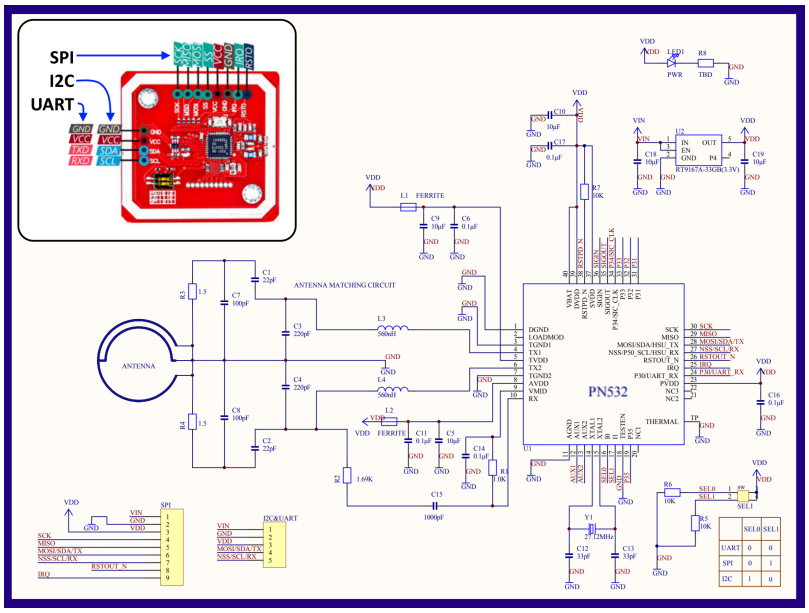
Adafruit RFID/NFC PN532 Breakout

Adafruit PN532 RFID Breakout

Other version of the red board (3.3V, 5V)

Wiki: http://wiki.sunfounder.cc/index.php?title=PN532_NFC_RFID_Module

Instructables: https://www.instructables.com/HackerBox-0072-Tagger/



PN532_Schematic.pdf

Terminology

Bird ringing is the term used in the UK and in some other parts of Europe and the world. Bird banding is the term used in the US. Organised ringing efforts are called ringing or banding schemes, and the organisations that run them are ringing or banding authorities. (Birds are ringed rather than rung) Those who ring or band are known as ringers or banders, and they are typically active at ringing or

banding stations.
https://en.wikipedia.org/wiki/Bird_ringing