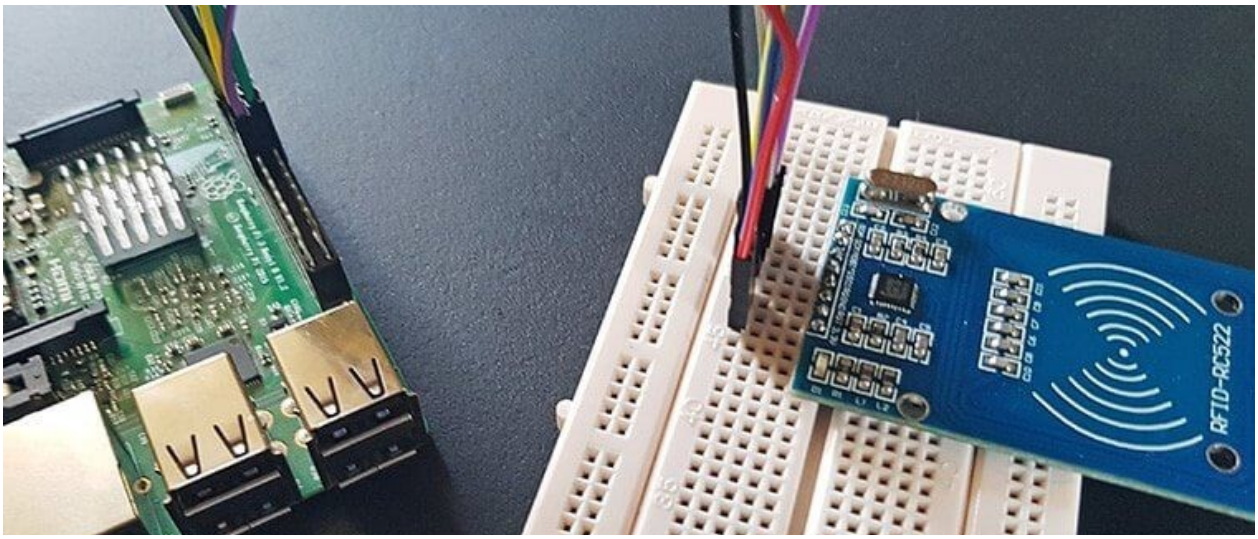
Part 70
-
RFID RC522

Version: 2021-01-07

How to setup a Raspberry Pi RFID RC522 Chip

In this Raspberry Pi RFID RC522 tutorial, I will be walking you through the steps on how to set up and wire the RFID RC522 chip with the Raspberry Pi.



This project opens up to quite a wide variety of different projects from using it as an attendance system to using it to open a lock.
The RFID RC522 is a very low-cost RFID (Radio-frequency identification) reader and writer that is based on the MFRC522 microcontroller.
The MRFC522 chip is controllable via SPI or I²C protocol.
This microcontroller provides its data through the selected protocol and works by creating a 13.56MHz electromagnetic field that it uses to communicate with the RFID tags.
Make sure that the tags you purchase for the RFID RC522 operate on the 13.56MHz frequency otherwise we will fail to read them.

Supportted cards or tags are:

- MIFARE1 S50
- MIFARE1 S70
- MIFARE Ultralight
- MIFARE Pro
- MIFARE DESFire

"MF RC522 is applied to the highly integrated read and write 13.56MHz contactless communication card chip, NXP launched by the company for the "table" application of a low-voltage, low-cost, small size of the non-contact card chip to read and write, smart meters and portable handheld devices developed better choice. The MF RC522 use of advanced modulation and demodulation concept completely integrated in all types of 13.56MHz passive contactless communication methods and protocols. 14443A compatible transponder signals. The digital part of to handle the ISO14443A frames and error detection. In addition, support rapid CRYPTO1 encryption algorithm, terminology validation MIFARE products. MFRC522 support MIFARE series of high-speed non-contact communication, two-way data transmission rate up to 424kbit/s. As new members of the 13.56MHz reader card series of highly integrated chip family, MF RC522 MF RC500 MF RC530 There are a lot of similarities, but also have many of the characteristics and differences. Communication between it and the host SPI mode helps to reduce the connection narrow PCB board volume, reduce costs."

RFID module:

The MF522-AN module uses the original Philip MFRC522 chip design circuit card reader, easy to use, low cost, and applies to the user equipment development, the reader and the development of advanced applications, the need for the user RF card terminal design/production. This module can be directly loaded into the various reader molds. Utilizes a voltage of 3.3V, through the SPI interface simple few lines directly with any user CPU motherboard connected communication can ensure that the module is stable and reliable work, distance card reader

Electrical parameters:

Operating current	:	13-26mA @ 3.3V
Idle current	:	10-13mA @ 3.3V
Sleep current	:	<80μA
Peak current	:	<30mA
Operating Frequency	:	13.56MHz
Module Size	:	40mm×60mm
Operating temperature	:	-20 to +80°C
Storage Temperature	:	-40 to +85°C
Relative humidity	:	5%-95%
SPI Data transfer rate	:	maximum 10Mbit/s

Wiring the RFID RC522

On the RFID RC522 you will notice that there are 8 possible connections on it, these being SDA (Serial Data Signal), SCK (Serial Clock), MOSI (Master Out Slave In), MISO (Master In Slave Out), IRQ (Interrupt Request), GND (Ground Power), RST (Reset-Circuit) and 3.3V (3.3V Power In).

We are going to use the SPI set-up as the module is set for this protocol and the Python library is written for this protocol.

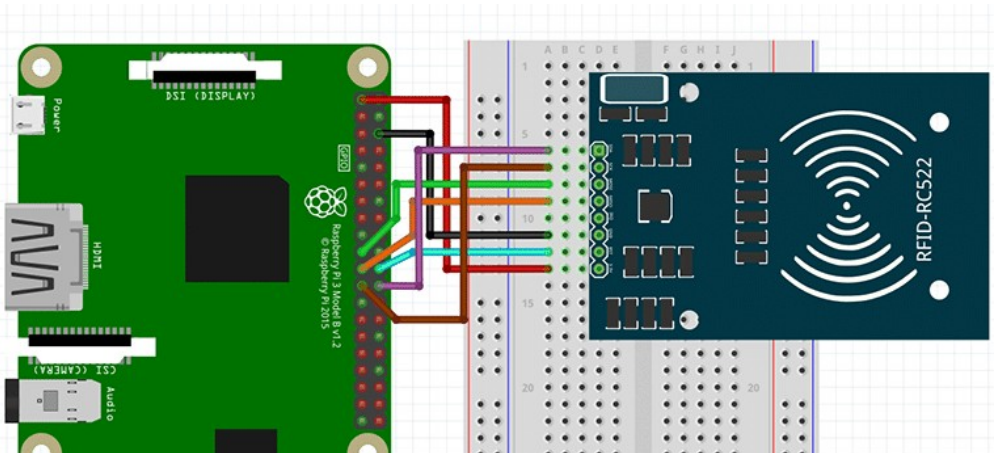
As the module only use 3.3V we can wire directly with the GPIO pins as these except 3.3V and can only handle 3.3V.

So wire all of these but the IRQ to our Raspberry Pi's GPIO pins.

You can either wire these directly to the GPIO Pins or plug the RFID RC522 into a breadboard then wire from there to the Raspberry Pi's GPIO Pins.

Wiring the RFID RC522 to the Raspberry Pi is fairly simple, with it requiring you to connect just 7 of the GPIO Pins directly to the RFID reader. Follow the table below, and check out our GPIO guide to see the positions of the GPIO pins that you need to connect the RC522 to.

- SDA connects to Pin 24
- SCK connects to Pin 23
- MOSI connects to Pin 19
- MISO connects to Pin 21
- GND connects to Pin 6
- RST connects to Pin 22
- 3.3V connects to Pin 1



Setting up Raspbian for the RFID RC522
Before beginning the process of utilizing the RFID RC522 on our Raspberry Pi, we will first have to make changes to its configuration. By default, the Raspberry Pi has the SPI (Serial Peripheral Interface) disabled.
Start the raspi-config tool

```
sudo raspi-config
```

This tool will load up a screen showing a variety of different options.
Select "`Interfacing Options`" and press `Enter`.
Now on the next screen, select "`P4 SPI`", again press `Enter`. You will now be asked if you want to enable the SPI Interface, select `Yes` with the arrow keys and press `Enter` to proceed.

Once the SPI interface has been successfully enabled by the raspi-config tool you should see the following text appear on the screen, "`The SPI interface is enabled`".
Before the SPI Interface is fully enabled we will first have to restart the Raspberry Pi. To do this first get back to the terminal by pressing `Enter` and then `Esc`. Now reboot the Raspberry Pi.

```
sudo reboot
```

Once the Raspberry Pi has rebooted, check to make sure that SPI is enabled. The easiest way to do this is to run the following command

```
lsmod | grep spi
```

If you see `spi_bcm2835`, then you can proceed and skip on to the next steps. If for some reason it does not appear, try following the next steps.

If for some reason the SPI module has not activated, we can edit the boot configuration file manually by running the following command.

```
sudo nano /boot/config.txt
```

Within the configuration file, find "`dtparam=spi=on`".
If you have found it, check to see if there is a `#` in front of it. If there is, remove it as this is commenting out the activation line. If you can't find the line at all, add "`dtparam=spi=on`" to the bottom of the file. Once you have made the changes, you can press `Ctrl + X` then pressing `Y` and then `Enter` to save the changes. Reboot the Pi and check again to see if the module has been enabled.

Getting Python ready for the RFID RC522
Now that we have wired up our RFID RC522 circuit to the Raspberry Pi we can now power it on and begin the process of programming simple scripts in Python to interact with the chip.
The scripts that we will be showing you how to write will show you how to read data from the RFID chips and how to write to them. These will give you the basic idea of how data is dealt with and will be the basis of further RFID RC522 tutorials.

Before we start programming, we first need to update our Raspberry Pi to ensure it's running the latest version of all the software. Run the following two commands on the Raspberry Pi to update it.

```
sudo apt -y update
sudo apt -y upgrade
```

Next is to install `python3-dev`, `python-pip` and `git` packages.

```
sudo apt -y install python3-dev python3-pip git
```

We must install the Python Library `spidev` to our Raspberry Pi using the python "`pip`" tool that we downloaded in the previous step. The `spidev` library helps handle interactions with the SPI and is a key component to this tutorial as we need it for the Raspberry Pi to interact with the RFID RC522. Run the following command to install `spidev`.

```
sudo pip3 install spidev
```

Now that we have installed the `spidev` library we can proceed to install the MFRC522 library using `pip` as well.
There are two files that are included within our MFRC522 library that we make use of:
`MFRC522.py` which is an implementation of the RFID RC522 interface, this library handles all the heavy lifting for talking with the RFID over the Pi's SPI Interface.
`SimpleMFRC522.py` that takes the `MFRC522.py` file and greatly simplifies it by making you only have to deal with a couple of functions instead of several.
To install the MFRC522 library, run the following command.

```
sudo pip3 install mfrc522
```

With the library installed, we can begin programming for our RFID RC522. To start off with we will be showing you how to write data to the RFID cards by using the RC522. Simply go onto our next section to begin programming our first Python script.

Writing with the RFID RC522
For our first Python script, we will be showing you how to write data from the RC522 to the RFID tags. Thanks to the `SimpleMFRC522` script this will be relatively simple, but we will still go into how each part of the code works.

Lets start off by making a folder where we will be storing our couple of scripts.
We will be calling this folder "`pi-rfid`", create it by running the following command.

```
mkdir ~/rfid
```

Begin by changing directory into our newly cloned folder, and begin writing our `Write.py` Python script.

```
cd ~/rfid
sudo nano rfid-write.py
```

Within this file, write the following lines of code. This code will basically ask you for text to input and then write that text to the RFID Tag.

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522
```

The first line of this segment of code helps tell the terminal how to interpret the file, and it lets it know that it should use Python when executing it and not something else such as Bash.
Our first import, RPi.GPIO has all the functions needed to interact with the GPIO Pins, and we need this to make sure they are cleared when the script finishes running.
The second import, imports in our `SimpleMFRC522` library, this is what we will use actually to talk with the RFID RC522, it greatly simplifies dealing with the chip compared to the base MFRC522 library.

```
rfid = SimpleMFRC522()
```

This line creates a copy of the `SimpleMFRC522` as an object, runs its setup function then stores it all in our `rfid` variable.

```
try:
    text = input('New data:')
    print("Now place the tag to write")
    rfid.write(text)
    print("Written")
```

Our next block of code we keep within a `try` statement, this is so we can catch any exceptions and clean up properly.
The second line here reads in an input from the command line, and we use input in Python 3 to read in all input and store it in our text variable.
With the third line, we utilize `print()` to notify the user that they can now place their RFID tag down onto the reader for writing.
Afterward, on our fourth line of code we use our `reader` object to write the values we stored in the text variable to the RFID tag, this will tell the RFID RC522 Circuit to write the text values to a certain sector.
Finally, on the 5th line of code, we use `print()` again to notify the user that we have successfully written to the RFID tag.

```
finally:
    GPIO.cleanup()
```

Our final two lines of code handle exiting of the script. Finally, always occurs after the try statement, meaning no matter what we run the `GPIO.cleanup()` function. These lines are crucial as failing to clean up can prevent other scripts from working correctly.

Once you have finished writing in the script, it should look something like below.

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522

rfid = SimpleMFRC522()

try:
    text = input('New data:')
    print("Now place the tag to write")
    rfid.write(text)
    print("Written")

finally:
    GPIO.cleanup()
```

Once you are happy that the code looks correct, you can save the file by pressing `Ctrl + X` then pressing `Y` and then finally hitting `Enter`.

Now that we have written our script, we will want to test it out. Before testing out the script make sure that you have an RFID tag handy. Once ready, type the following command into the Raspberry Pi's terminal.

```
sudo python3 rfid-write.py
```

You will be asked to write in the new data, in our case we are going to just type in pimylifeup as its short and simple. Press `Enter` when you are happy with what you have written.

With that done, simply place the RFID Tag on top of the RFID RC522 circuit. As soon as it detects it, it will immediately write the new data to the tag. You should see "`Written`" appear in the command line if it was successful.
You can look at our example output below to see what a successful run looks like.

```
pi@raspberrypi:~/rfid $ sudo python3 rfid-write.py
New data:rfidtest
Now place the tag to write
Written
```

You have now successfully written the rfid-w`rite.py` script, and we can now proceed to show you how to read data from the RFID RC522.

Reading with the RFID RC522
Now that we have written our script to write to RFID tags using our RC522 we can now write a script that will read this data back off the tag.

Let's start off by changing the directory to make sure we are in the right place, and then we can run `nano` to begin writing our `Read.py` script.

```
cd ~/rfid
sudo nano rfid-read.py
```

Within this file, write the following lines of code. This script will basically sit and wait till you put the RFID tag on the RFID RC522 reader, it will then output the data it reads off the tag.

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522
rfid = SimpleMFRC522()

try:
    id, text = rfid.read()
    print(id)
    print(text)

finally:
    GPIO.cleanup()
```

The code is contained within a try statement, and we use this so we can catch any exceptions that might occur and deal with them nicely.
It tells the circuit to begin reading any RFID tag that is placed on top of the RC522 reader.
We utilize `print()` to print out the information that we received from reading the RFID Chip, this includes the ID associated with the RFID tag and the text that is stored on the tag.

Once you are sure you have entered the code correctly, you can save the file by pressing `Ctrl + X` then pressing `Y` and then finally hitting `Enter`.

Now that we have finally finished our `rfid-read.py` script we can test it out.

```
sudo python3 rfid-read.py
```

With the script now running, all you need to do is place the RFID Tag on top of the RFID RC522 circuit. As soon as the Python script detects the RFID tag being placed on top, it will immediately read the data and print it back out to you.
An example of what a successful output would look like is displayed below.

```
pi@raspberrypi:~/rfid $ sudo python3 rfid-read.py
827843705425
rfidtest
```

If you successfully receive data back from the `rfid-read.py` script with the text that you pushed to the card using the `Write.py` script.

Using the unique RFID ID-numer, you can identify each card and act accordingly.