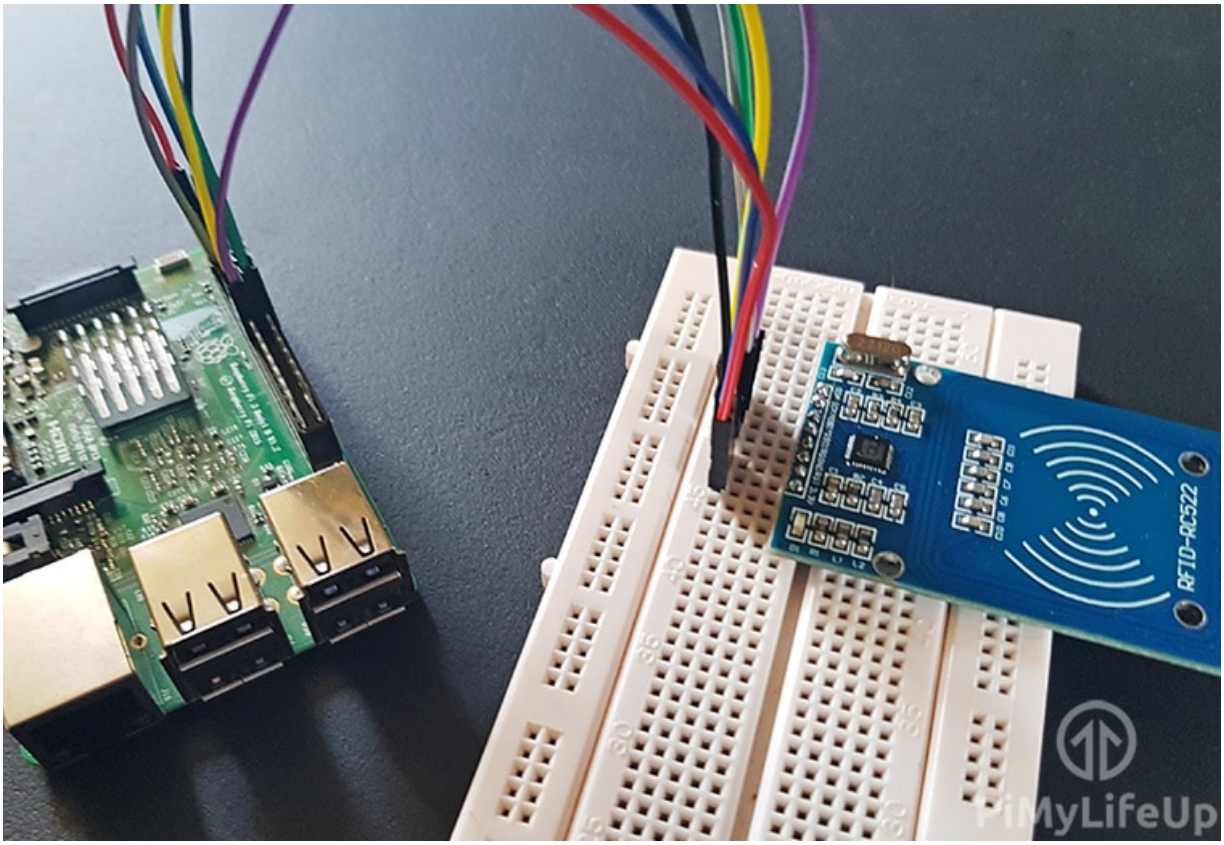
PiMyLifeUp

How to setup a Raspberry Pi RFID RC522 Chip

by Gus | Oct 3, 2017 | Beginner, Sensors | 5 comments



In this Raspberry Pi RFID RC522 tutorial, I will be walking you through the steps on how to setup and wire the RFID RC522 chip with your Raspberry Pi. This is a cool circuit to play around with and opens you 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. This microcontroller provides its data through the SPI 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 your RFID RC522 operate on the 13.56MHz frequency otherwise we will fail to read them.

We will be showing you how to wire up the RC522 as well as showing you how to write Python scripts to interact with the chip so you can both read and write your RFID Tags. You can extend this tutorial to use something like a 16×2 LCD for the Raspberry Pi, handy if you want to show some information.

Equipment List

Below are all the bits and pieces that I used for this Raspberry Pi RFID RC522 tutorial.

Recommended:

Raspberry Pi 2 or 3

Micro SD Card

RC522 RFID Reader

Breadboard

Breadboard Wire

Optional:

Raspberry Pi Case

Ethernet Network Connection or Wifi dongle (The Pi 3 has WiFi inbuilt)

Assembling the RFID RC522

One thing you will notice when purchasing an RFID RC522 Reader is that 90% of them don't come with the header pins already soldered in. This means you will have to do it yourself, luckily soldering header pins is a rather simple task, even for beginners.

1. First off if the header pins you received with your RC522 isn't the correct size snap them down so you only have a single row of 8 pins.

2. Place the header pins up through the holes of your RC522. One handy trick is to put the long side of the header pins into a breadboard and then putting the circuit over the top of the header pins. The breadboard will hold the pins tightly making it easier to solder them to the RFID RC522 circuit.

3. Now using a hot soldering iron and some solder, slowly solder each of the pins. Remember it

is best to heat the joint slightly before applying solder to it, this will ensure that the solder will adhere more to the joint and reduce the chances of creating a cold joint. We also recommend being careful with the amount of solder you apply.

4. With the header pins now soldered to your RFID circuit it is now ready to use and you can continue with the tutorial.

Wiring the RFID RC522

On your 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 will need to 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 like we did in this tutorial, plug the RFID RC522 into our Breadboard then wire from there to our Raspberry Pi's GPIO Pins.

Wiring your RFID RC522 to your 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 your RC522 to.

SDA connects to **Pin 24**.

SCK connects to **Pin 23**.

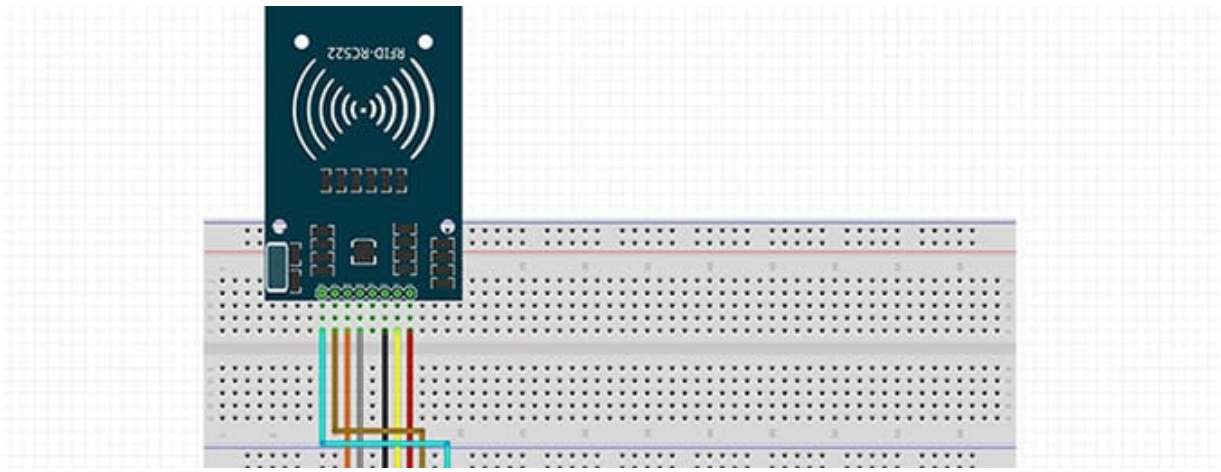
MOSI connects to **Pin 19**.

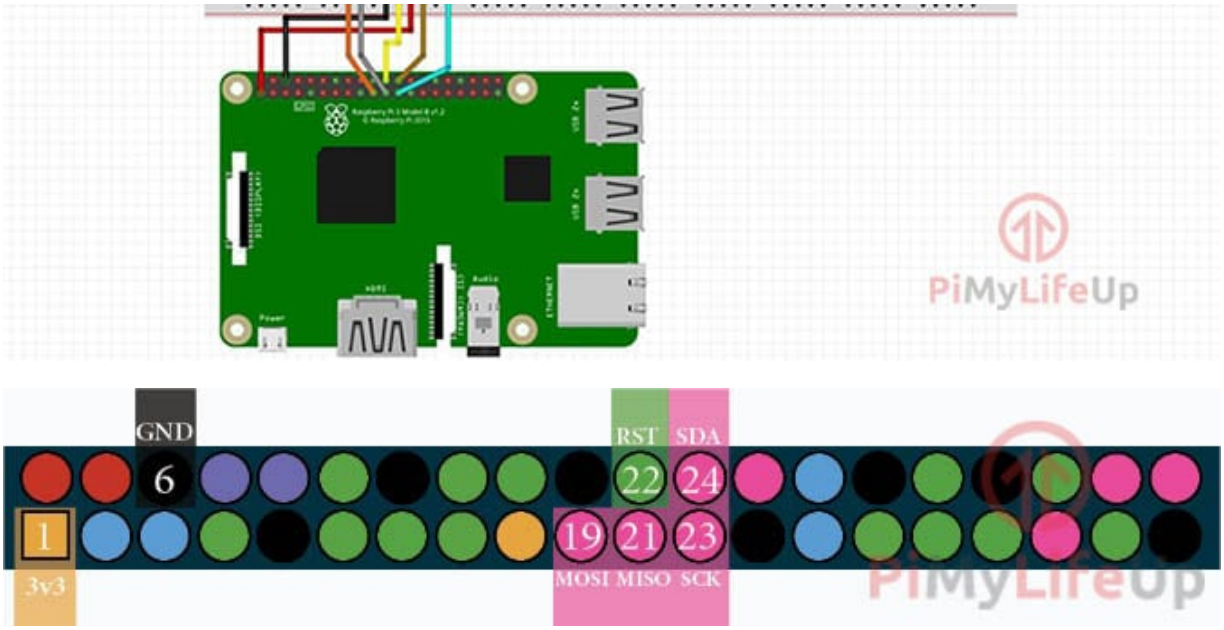
MISO connects to **Pin 21**.

GND connects to **Pin 6**.

RST connects to **Pin 22**.

3.3v connects to **Pin 1**.





Setting up Raspbian for the RFID RC522

Before we begin 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, which is a bit of a problem as that is what our RFID reader circuit runs through.

Don't worry though as it is fairly simple to re-enable this interface, just follow our steps below to configure your Raspberry Pi and Raspbian to utilize the SPI interface.

1. Let's begin by first opening the **raspi-config** tool, we can do this by opening terminal and running the following command:

```
sudo raspi-config
```

2. This tool will load up a screen showing a variety of different options. If you want a more in-depth look into these options you can check out our raspi-config guide that's located in the Getting Started section of this book.

On here use the **arrow keys** to select "**5 Interfacing Options**". Once you have this option selected, press **Enter**.

3. Now on this next screen you want to again use your **arrow keys** to select "**P4 SPI**", again press Enter to select the option once it is highlighted.

4. You will now be asked if you want to enable the SPI Interface, select **Yes** with your **arrow keys**

and press **Enter** to proceed. You will need to wait a little bit while the **raspi-config** tool does its thing in enabling SPI.

5. 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**.

Type the following linux command into the terminal on your Raspberry Pi to restart your Raspberry Pi.

```
sudo reboot
```

6. Once you Raspberry Pi has finished rebooting we can now check to make sure that it has in fact been enabled. The easiest way to do this is to run the following command to see if **spi_bcm2835** is listed.

```
lsmod | grep spi
```

If you see **spi_bcm2835** then you can proceed on with this tutorial and skip on to the next section. If for some reason it has not appeared when you entered the previous command, try following the next 3 steps.

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

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

8. Within the configuration file, use **Ctrl + W** to 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, just 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.

You can now proceed from **Step 5** again, rebooting your Raspberry Pi then checking 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 basically 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.

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

```
sudo apt-get update
sudo apt-get upgrade
```

2. Now the final thing we need before we can proceed is to install the **python2.7-dev** package, simply run the following command on your Raspberry Pi to install it.

```
sudo apt-get install python2.7-dev
```

3. To begin we must first clone the Python Library SPI Py and install it to our Raspberry Pi. This 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 two commands on your Raspberry Pi to clone the source code.

```
cd ~
git clone https://github.com/lthiery/SPI-Py.git
```

4. With the SPI Py Python Library now cloned to our Raspberry Pi we need to install it, this is incredibly simple as all we need to do is change into its directory and run a simple python command on our Raspberry Pi.

```
cd ~/SPI-Py
sudo python setup.py install
```

5. Now that we have installed SPI-Py we can now clone our RFID RC522 Python code from the PiMyLifeUp Github. There are two files included in this repository:

MFRC522.py which is an implementation of the RFID RC522 circuit.

SimpleMFRC522.py that takes the **MFRC522.py** file and greatly simplifies it.

To clone this repository, you can type the following two commands into your Raspberry Pi.

```
cd ~
git clone https://github.com/pimylifeup/MFRC522-python.git
```

6. With the repository now saved to our Raspberry Pi we can begin programming for our RFID RC522. To start off with we will be showing you how to write data to your 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 your RFID tags. Thanks to the SimpleMFRC522 script this will be relatively simple, but we will still go into how each part of the code words.

1. 1. Begin by changing directory into our newly cloned folder, and begin writing our **Write.py** python script.

```
cd ~/MFRC522-python
sudo nano Write.py
```

2. Within this file, write the following lines of code. This 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
import SimpleMFRC522
```

The first line of this segment of code helps tell the terminal how to interpret the file, 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, 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 to actually talk with the RFID RC522, it greatly simplifies dealing with the chip compared to the base MFRC522 library.

```
reader = SimpleMFRC522.SimpleMFRC522()
```

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

```
try:
        text = raw_input('New data:')
        print("Now place your tag to write")
        reader.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. Make sure that you retain the 'tabs' after try: as Python is whitespace sensitive, and it is how it differs between blocks of code.

The second line here reads in an input from the command line, we use raw_input in Python 2.7 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.

Afterwards 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 basically 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 basically handle exiting of the script. Finally, always occurs after the try statement, meaning no matter what we run the GPIO.cleanup() function. This is crucial as failing to cleanup can prevent other scripts from working correctly.

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

```
#!/usr/bin/env python

import RPi.GPIO as GPIO
import SimpleMFRC522

reader = SimpleMFRC522.SimpleMFRC522()

try:
        text = raw_input('New data:')
        print("Now place your tag to write")
        reader.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**.

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

```
sudo python Write.py
```

5. 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.

6. With that done, simply place your RFID Tag on top of your 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 your command line if it was successful.

You can look at our example output below to see what a successful run looks like.

```
pi@raspberrypi:~/MFRC522-python $ sudo python Write.py
New data:pimylifeup
Now place your tag to write
Written
```

7. You have now successfully written your **Write.py** script, we can now proceed to show you how to read data from the RFID RC522 in the next segment of this tutorial.

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.

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

```
cd ~/MFRC522-python
sudo nano Read.py
```

2. Within this file, write the following lines of code. This script will basically sit and wait till you put your 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
import SimpleMFRC522
```

The first line of code basically tells the operating system how to handle the file when a user executes it. Otherwise it will try and just run it as a normal script file and not a python file.

The first import is, **RPi.GPIO**. This library contains all the functions to deal with the Raspberry Pi's GPIO pins, we mainly import this to ensure that we cleanup when the script finishes executing.

The second import is, **SimpleMFRC522**. This script contains a few helper functions to make it an awful lot easier to deal with writing and reading from the RFID RC522, without it our simple scripts would become quite long.

```
reader = SimpleMFRC522.SimpleMFRC522()
```

This line is quite important as it calls SimpleMFRC522's creation function and then stores that into our reader variable as an object so we can interact with it later

```
try:
        id, text = reader.read()
```

```
        print(id)
        print(text)
```

This next block of code is contained within a **try** statement, we use this so we can catch any exceptions that might occur and deal with them nicely. You need to ensure that you use the '**tabs**' as shown after **try:** as Python is whitespace sensitive.

The second line in this block of code makes a call to our reader object, in this case it basically tells the circuit to begin reading any RFID tag that is placed on top of the RC522 reader.

With the third and fourth lines 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 was stored on the tag.

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

The two last lines of code handle the termination of the script. The finally statement always triggers after the try statement even if we get an exception.

This ensures that no matter what we run the **GPIO.cleanup()** function.

It is quite crucial as failing to clean up the GPIO can prevent other scripts from working correctly.

3. Now that you have finished writing your **Read.py** script for your RFID RC522 it should look something like what is shown below:

```
#!/usr/bin/env python

import RPi.GPIO as GPIO
import SimpleMFRC522

reader = SimpleMFRC522.SimpleMFRC522()

try:
        id, text = reader.read()
        print(id)
        print(text)
finally:
        GPIO.cleanup()
```

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**.

4. Now that we have finally finished our Read.py script we need to test it out. Before we test out the script, grab one of your RFID tags that you want to read. Once that you are ready, type the following command into your Raspberry Pi's terminal.

```
sudo python Read.py
```

5. With the script now running, all you need to do is place your RFID Tag on top of your 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:~/MFRC522-python $ sudo python Read.py
827843705425
pimylifeup
```

7. If you successfully receive data back from your Read.py script with the text that you pushed to the card using your Write.py script then you have successfully setup your Raspberry Pi to connect with your RFID RC522 Circuit.

We will be going into more depth with these scripts and the RFID chip in later tutorials. This includes exploring how to setup an attendance system among other cool DIY Pi projects.

If you have enjoyed this Raspberry RFID RC522 tutorial or have any feedback feel free to drop a comment below!

5

Raspberry Pi Motion Sensor using a PIR Sensor

Comments

He Lo on October 17, 2017 at 6:36 pm

Reply

Sir,

I couldn't write or read my tags (neither the credit card sized one nor the round keychaind one).

I've repeated 3 times re-installing as per your tutorial instructions, checked the wiring. I'm sure SPI is enabled (I checked several times, including `lsmode | grep spi` **and through the desktop gui "preferences/Raspberry Pi Configuration/Interfaces "options.**

Raspberry Pi ADC: MCP3008 Analog to Digital Converter

I created `Write.py` **as you instructed and ran the program, which did ask me for my input, which I supplied. It then asked me to place my tag, which I also did but nothing happened (expected a "written" output as programmed but no).**

What can possible be wrong?

Thank you.

SungKun Choi on October 18, 2017 at 6:51 pm

Reply

Thank you so much!!!

You are great.

Marvin on October 18, 2017 at 8:27 pm

Thank you!!!!!!!! Your Simple-Library is so helpful!

Reply

Andy on October 20, 2017 at 12:28 pm

Does this work on a Pi Zero W?

Reply

Gus on October 21, 2017 at 7:30 pm

Hey Andy,

Raspberry Pi Zero W uses the same GPIO pins as the Raspberry Pi 3 and Raspberry Pi 2 so the tutorial should work perfectly fine.

Cheers

Reply

Follow us on social

Search for Tutorials!

Search