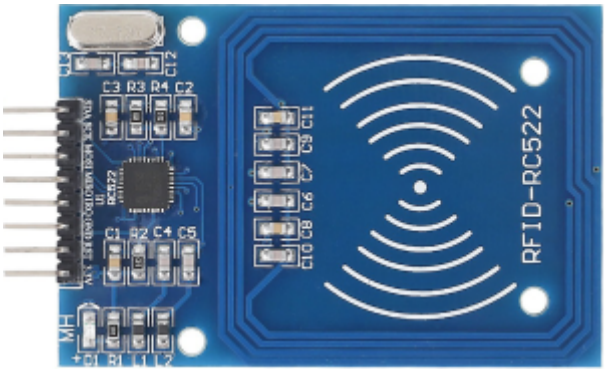
Pico with RFID-RCS522

Aug 2026

Introduction



RFID stands for radio-frequency identification, which is a wireless technology that uses radio waves to identify, track, and communicate information between a tag and a reader.

RC522 is a low-cost, 13.56 MHz radio frequency identification (RFID) reader and writer module powered by the NXP MFRC522 chip, commonly used with microcontrollers like Arduino and Raspberry Pi.

Key Specifications

- Operating Frequency: 13.56 MHz
- Operating Voltage: 2.5V to 3.3V (Note: Always power VCC with 3.3V; supplying 5V can damage the module, though data pins are mostly 5V tolerant)
- Operating Current: 13 mA to 26 mA
- Read Range: About 2 cm to 5 cm depending on the tag
- Supported Protocols: ISO/IEC 14443 Type A (such as MIFARE 1K, 4K, and Ultralight)
- Communication Interfaces: SPI (default), I2C, and UART

Pinout Configuration

- VCC: Power supply, strictly 3.3V
- RST: Reset pin
- GND: Ground reference
- IRQ: Interrupt pin to alert the microcontroller
- MISO / SCL / Tx: Master-In-Slave-Out for SPI
- MOSI: Master-Out-Slave-In for SPI
- SCK: Serial Clock for SPI
- SS / SDA / Rx: Slave Select / Signal input

RFID Frequencies

There are three main frequency bands (with a fourth microwave band sometimes included).

Low Frequency (LF)Range:

- 125 kHz to 134.2 kHz (or 120–150 kHz).
- Use cases: Animal tracking (pet microchips) and key fobs.

- Range distance: Short (up to 10 cm or a few inches).

High Frequency (HF)

- Range: 13.56 MHz.
- Use cases: Smart cards, payment badges, and NFC (Near Field Communication).
- Range distance: Short (up to 1 meter).

Ultra-High Frequency (UHF)

- Range: 860 MHz to 960 MHz (plus a 433 MHz band for active tags).
- Use cases: Warehouse inventory, toll passes, and supply chains.
- Range distance: Long (from a few meters up to 100 meters for active tags).

Microwave Frequency (Optional 4th Band)

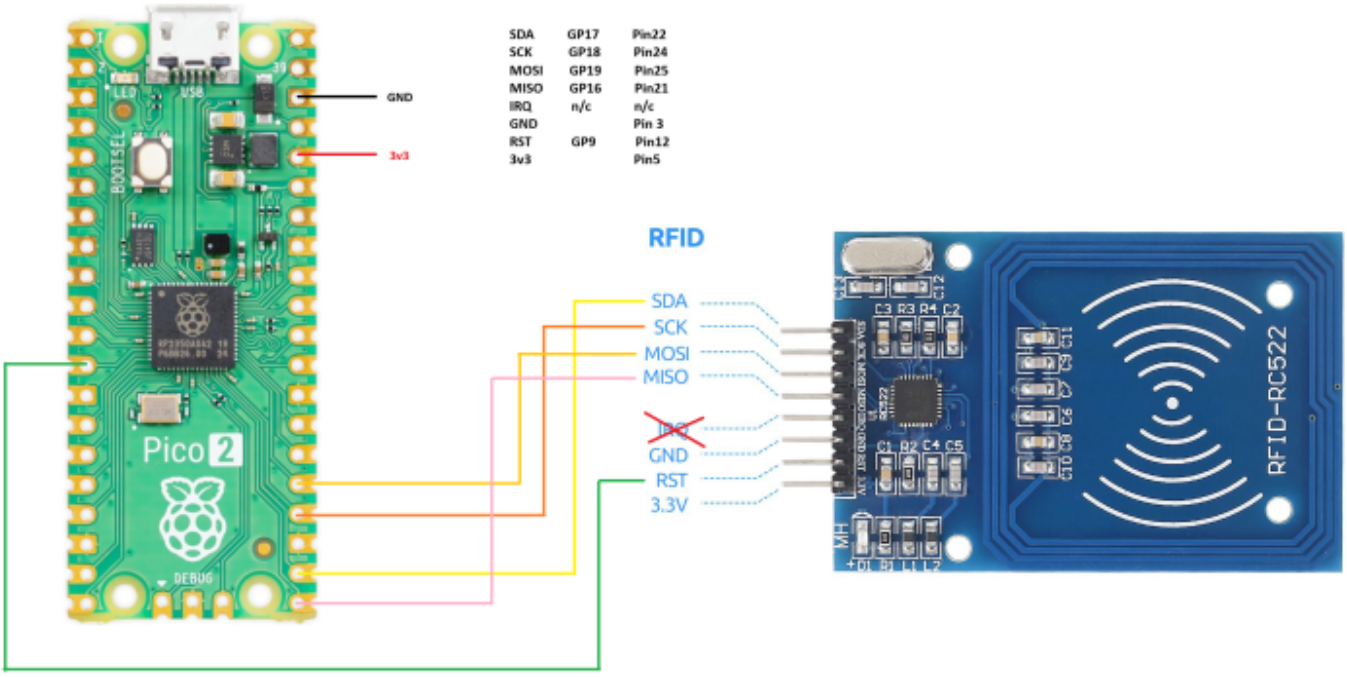
- Range: 2.45 GHz.
- Use cases: Long-range active tracking and toll collection.
- Range distance: Very long.

The device we will be using is the High Frequency (HF) 13.56 MHz Range.

Wiring for Raspberry Pi Pico

Here is the wiring for a Raspberry Pi Pico. Remember we are using:

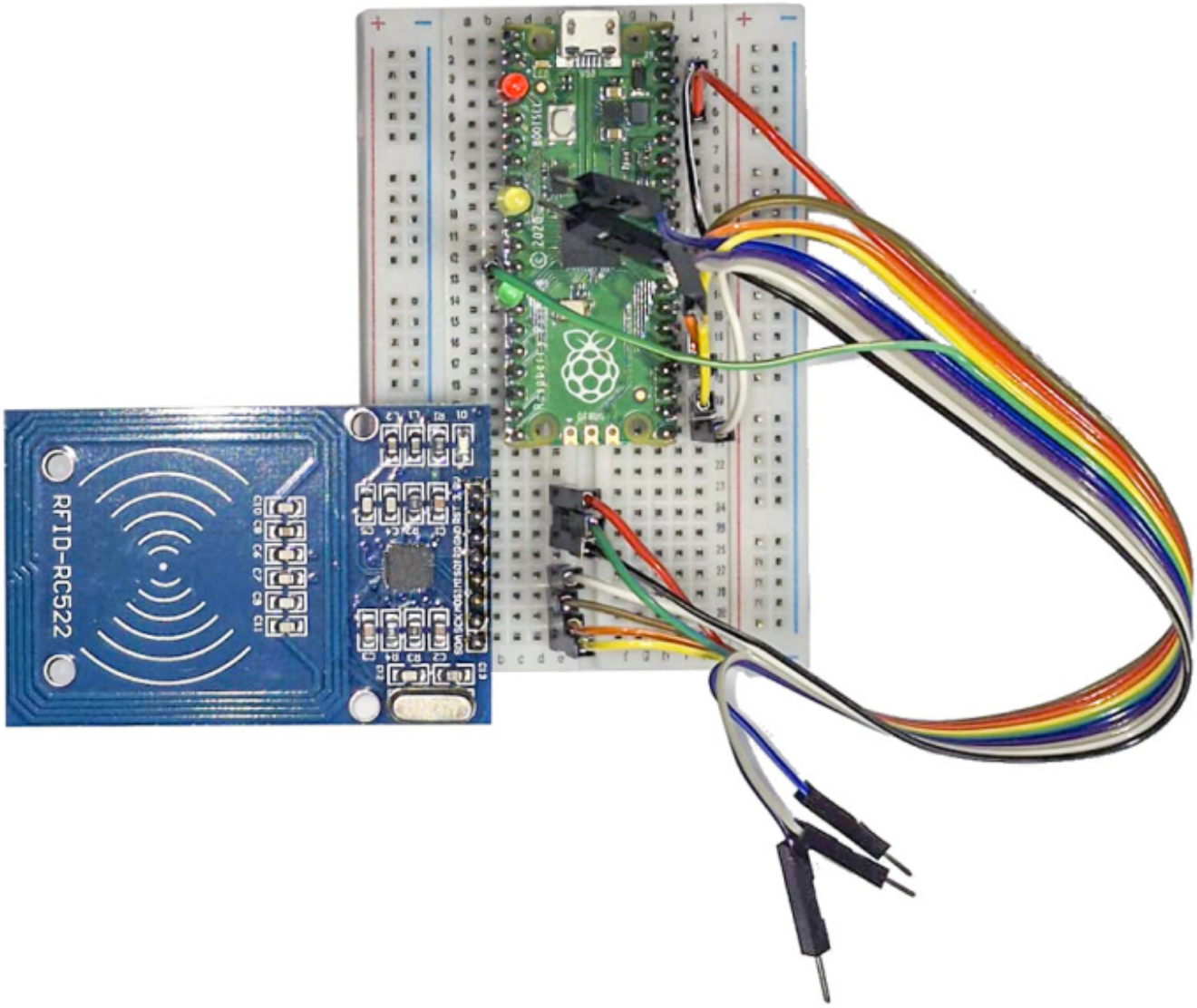
- SPI Protocol
- SPI Interface 0



Here are the pin connections, in case the diagram is not clear.

Signal	GPIO	Pico Pin Number
SDA	GP17	Pin 22
SCK	GP18	Pin 24
MOSI	GP19	Pin 25
MISO	GP16	Pin 21
IRQ	n/c	n/c
GND	—	Pin 3
RST	GP9	Pin 12
3V3	—	Pin 5

Once you have wired up your Pico and RFID-RCS522 we can move on to some code.



Some test Python Scripts

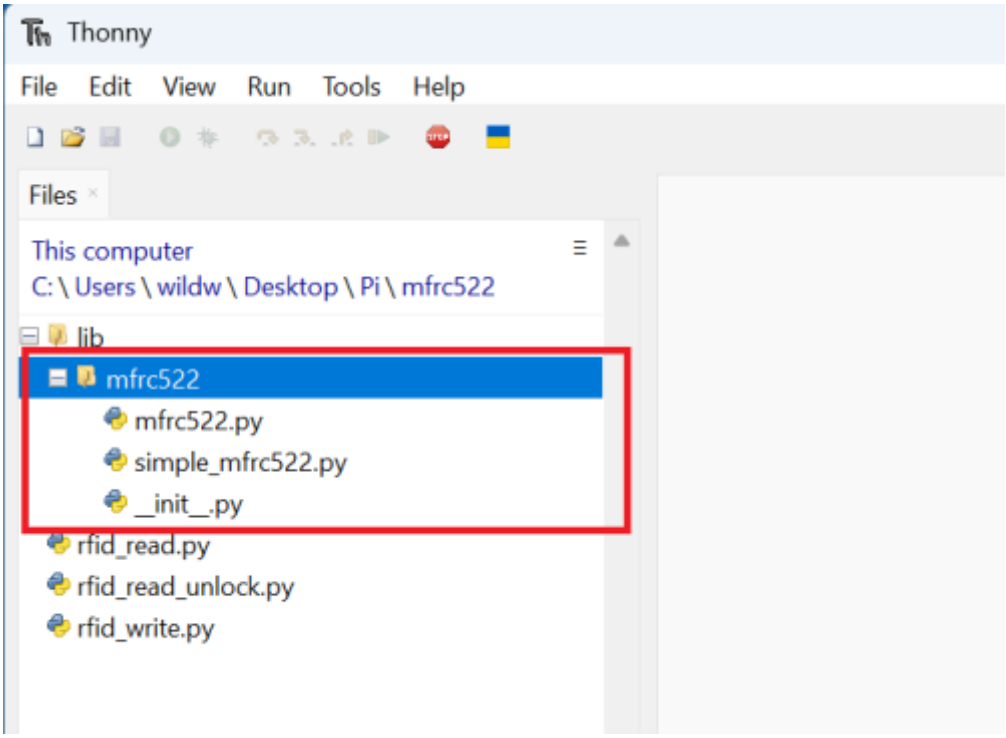
We will start with two scripts.

First script to Read an RFID Tag. Second Script to Write to an RFID Tag.

Once we have written to a tag we will use the first script to ensure that we can read what was written.

Before we begin, we have to copy some files to our Raspberry Pi Pico, driver files basically so that we can read and write to our RFID tags.

First of all, we need to create a folder on our Pico under the lib folder (if you don't have the lib folder, create this first). The folder name is mfrc522.



In this /lib/mfrc522 folder, we need to put the following three files:

- _init__.py
- mfrc522.py
- simple_mfrc522.py

These are the drivers for the RFID card we are using. You can download them from

here

.

In the top level folder, we need to add these two files:

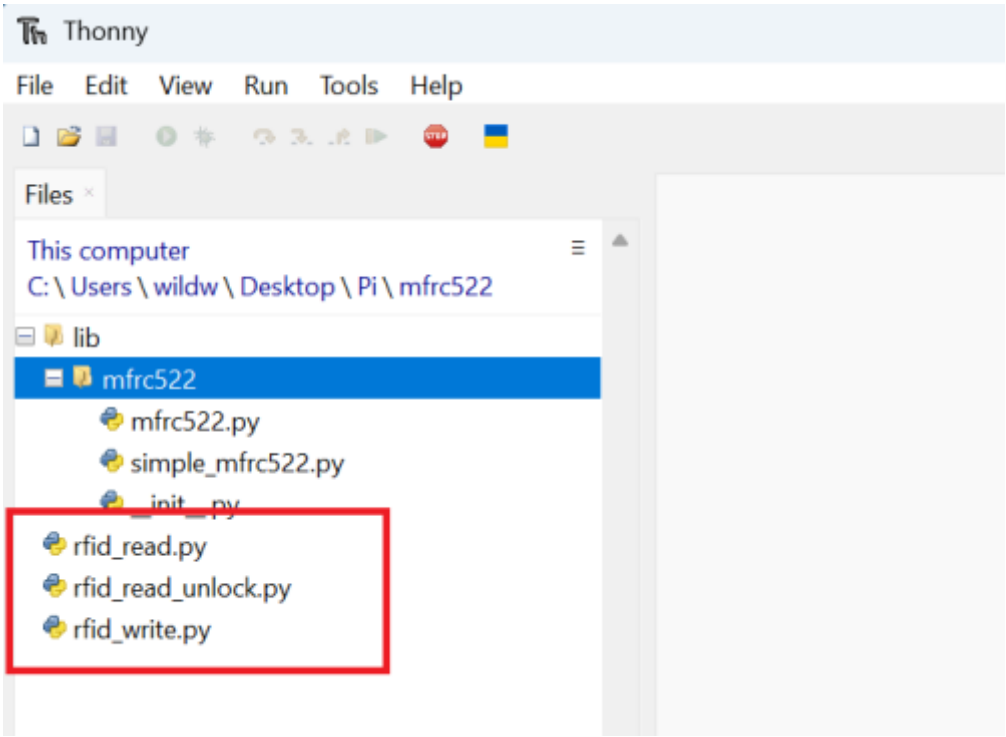
- rfid_read.py
- rfid_write.py

These are the two Python scripts we will use to read and write to and from our RFID Tags, you can download them

here

.

Here you can see the two files (plus one extra we will not use at this time).



At this point you should have a fully wired Pico/RFID board, plus all the software installed (in the correct folder structure).

A Quick Note About RFID Tags

If you purchased an RFID kit, then it likely came with a few items.

RFID Writer/Reader RFID Tag (Keyring) RFID Card.

If you don't have the tag or card at this time, then you need to acquire these to proceed with the testing.

These tags have various storage capabilities as laid out below:

RFID tag storage capacity ranges from 64 bits for basic identification tags up to 8 kilobytes or more for advanced data-storing tags.

Typical Capacities by Memory Type

- **Basic / Low-Memory Tags:** 64 to 128 bits. These store a basic serial number or electronic product code (EPC) that links to an external database.

- **Medium-Memory Tags:** 512 bits to 2 kilobytes (KB). These store extra item details, user data, or small web links (such as NFC NTAG chips with 144 to 888 bytes of user memory).
- **High-Memory Tags:** Up to 8 KB or more. Specialized industries like aerospace use these to store complete repair and parts histories directly on the item.

Remember:

- A 128-bit value equals 32 characters in hexadecimal or 16 characters in standard 8-bit text.
- A 1k bits (kilobits) equals 125 standard ASCII or UTF-8 characters using decimal math, or 128 characters using binary math.

Using the Read / Write Python Code

There are two main Python scripts here, one to read tags, one to write tags. Lets start with the tag reading Python code.

From Thonny, run the Python script rfid_read.py

Click the play button in Thonny.

Place your RFID Card/Keyring on the reader for a scan to occur.

Above we can see that the following was read from the RFID Tag.

- ID: 07 69 0D 64
- Text: mary had a little lamb

The ID is the Tag ID, the Text is something I have written to the RFID Tag previously.

Now we can write some data to the RFID Tag. Stop the running rfid_read.py and run the rfid_write.py.

Above we are being prompted to Enter a message, enter a short message, bearing in mind your Tag might not have a lot of memory. Press enter.

Now the Python rfid_write.py tells us it is writing to the card.

We now have to run the rfid_read.py again, to confirm that the write was successful.

Once running, tap your RFID card and you should see the message that you typed in during the rfid_write.py process.

At some point in the future we will come back to this and add some code to allow us to make real world changes when RFID tags are scanned, maybe turn on/off an LED or display an RFID tag ID: on an LCD screen.

Practical Example

This is a good example to show what you can do reading RFID tags. Now, I want to make something very, very clear. RFID tags have Unique IDs (UID) and you should be using this UID to ensure that only tags you have authorised can be used on your system. For this example, we are not using UIDs, because it's just a simple example of using an RFID tag to do something.

I am assuming you have read this entire article, if you have not, you will probably have a lot of issues, so I suggest you have a good read to understand the RFID Python scripts.

We are going to create 3 RFID tags. If you look in the previous section above, you can see we have used two Python scripts.

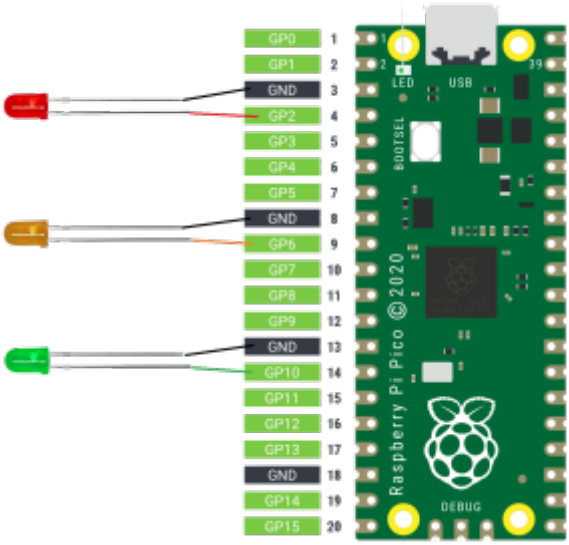
- rfid_read.py
- rfid_write.py

Initially, we are going to use the rfid_write.py. Using this Python script, create three RFID tags, RED, AMBER and GREEN.

What we are going to do is to turn on some LEDs depending on which RFID Tag is scanned.

The Hardware

Connect three LEDs, as shown in the diagram below.



If you used the same GPIO Pins that we used in this example to connect the RFID Reader/Writer, then the pins for the LEDs above will be free.

The Python Code

Below is the Python code we will use for this example, it's an expanded version of the rfid_read.py that we used earlier.

What the code does is very simple.

Reads the tag. Gets the value RED, AMBER or GREEN. Uses an if, elseif loop to turn on the corresponding LED (and turn the other off).

Here is the code, save it to your Pico as rfidled.py.

| download

```
from mfrc522 import SimpleMFRC522
import time
from machine import Pin

# Define GPIO pins for the LEDs
RED = Pin(2, Pin.OUT)
AMBER = Pin(6, Pin.OUT)
GREEN = Pin(10, Pin.OUT)

# Put them in an array
lights = [RED, AMBER, GREEN]

# Set the GPIO pins being used
reader = SimpleMFRC522(spi_id=0,sck=18,miso=16,mosi=19,cs=17,rst=9)

# Function to turn off all LEDs
def all_off():
    for light in lights:
        light.off()

# Read the RFID card/tag/sticker
def read():
    print("Reading...Please place the card...")  # print a message telling the user to tap the
card.
    id, text = reader.read()   # read in two fields from the tag, id and text.
    print("ID: %s\nText: %s" % (id,text))   # print the result in Thonny
#   return id, text
    return text.strip()  # this returns the contents of 'text' to the calling code, but strips off
any crap like spaces.
                         # Even if the tag is short (red) you still get 16 chars back from the
read, so we need to strip of what is not required.
```

```
# Call function 'all_off' to turn off all LEDs
all_off()

# Start a loop (forever)
while True:
    tag = read() #c all the function def read() and store the result in the varable tag
    print ("the scanned colour is =", tag)  #print the contents of the variable tag

    # Ched the read tag colour, turn off all LEDs then turn on the corresponding one. If no tags
match, nothing happens.
    if tag == "RED":
        all_off()
        RED.on()
    elif tag == "AMBER":
        all_off()
        AMBER.on()
    elif tag == "GREEN":
        all_off()
        GREEN.on()

    time.sleep(1) # The delay is to stop the tag being read multiple times per second.
```

Run the code, if you have wired your LEDs correctly, and programmed your tags, each LED will come on with each Tag scan.

From:
https://wiki.alanwalker.uk/ - **WalkerWiki - wiki.alanwalker.uk**

Permanent link:
https://wiki.alanwalker.uk/doku.php?id=pico_with_rfid-rcs522&rev=1788299358

Last update: **2026/09/01 21:49**