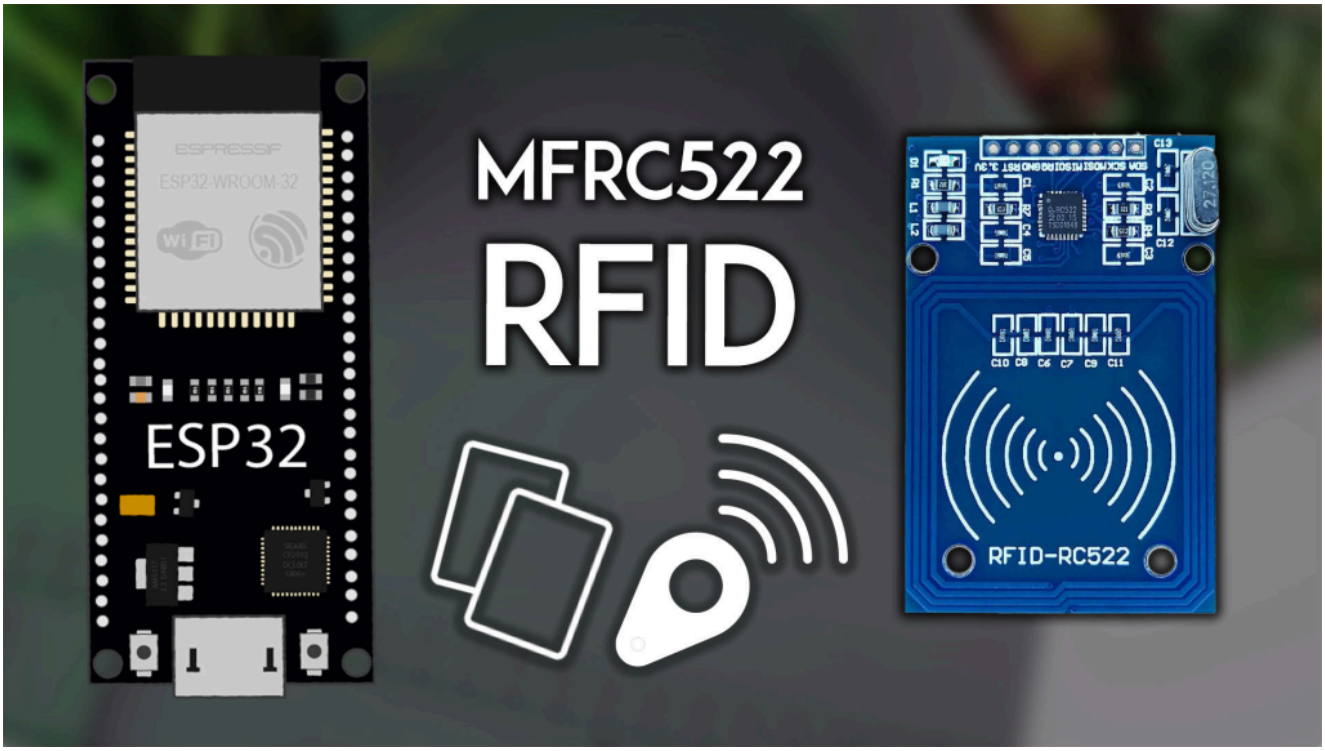
ESP32 with MFRC522 RFID Reader/Writer (Arduino IDE)

Learn how to interface the MFRC522 RFID reader module with the ESP32 board. You'll learn how to get raw RFID data, get the RFID card UID, and add personal data to the RFID cards. We'll use the `Arduino_MFRC522v2` library and the ESP32 will be programmed using Arduino IDE.



We have a similar guide for the ESP8266 NodeMCU with MFRC522 RFID Reader/Writer (Arduino IDE)

In summary, in this tutorial we'll cover:

- Introduction to the MFRC522 RFID Reader/Writer
- Wire the MFRC522 RFID Reader/Writer module to the ESP32
- Read RFID card raw data
- Read the RFID card UID
- Write/Read personal data to and from the RFID card

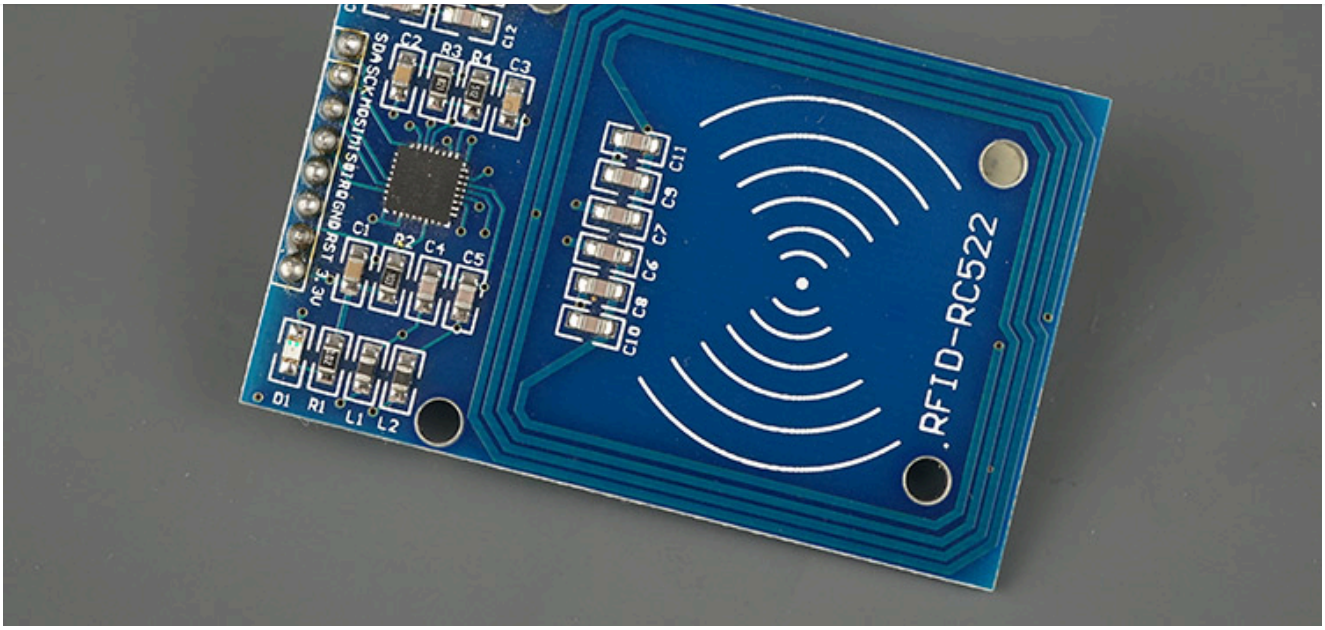
In this tutorial, we'll be using the MFRC522 RFID reader/writer and that's the one we recommend you getting to interface with the ESP32.

RFID means radio-frequency identification. RFID uses electromagnetic fields to transfer data over short distances and it's useful to identify people, make transactions, etc. Some stores also use RFID tags on their products' labels to identify them.

An RFID system needs tags and a reader:

- **Tags** (proximity integrated circuit cards—PICC) are attached to the object to be identified, in this example, we have a keychain and an electromagnetic card that come with the MFRC522 RFID Reader/Writer module. Each tag has a unique identification (UID).

- **Reader/Writer** (proximity coupling device—PCD) is a two-way radio transmitter-receiver that sends a signal to the tag and reads its response.



The MFRC522 RFID reader works at 3.3V and supports SPI and I2C communication protocols. The library we're going to use to control the RFID reader supports both protocols, but we'll be using the SPI.

Parts Required

Here's a list of the required components for this project:

- ESP32 DOIT DEVKIT V1 Board (read Best ESP32 Dev Boards)
- MFRC522 RFID Reader/Writer + tags
- Breadboard
- Jumper wires

You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

To learn more about the RFID reader with the Arduino read: Security Access using MFRC522 RFID Reader with Arduino

Prerequisites

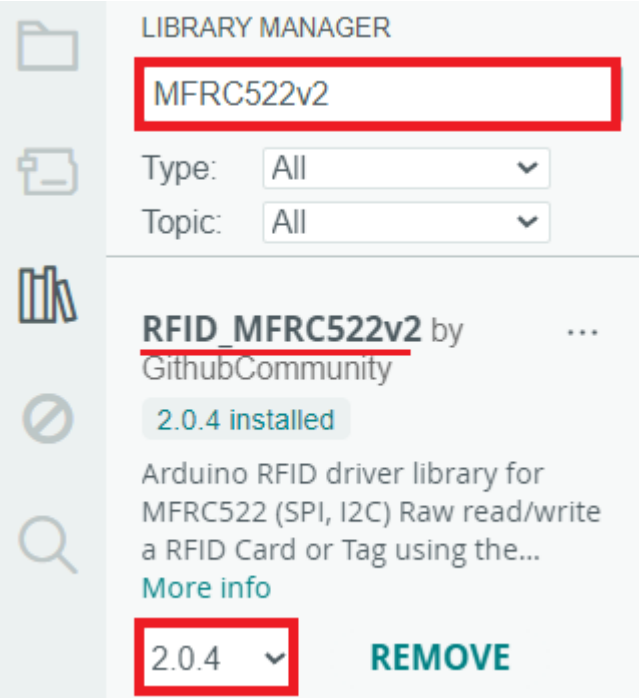
IDE. Follow the next tutorial to install the ESP32 on the Arduino IDE, if you haven't already.

- Installing the ESP32 Board in Arduino IDE (Windows, Mac OS X, and Linux instructions)

Installing the Arduino_MFRC522v2 Library

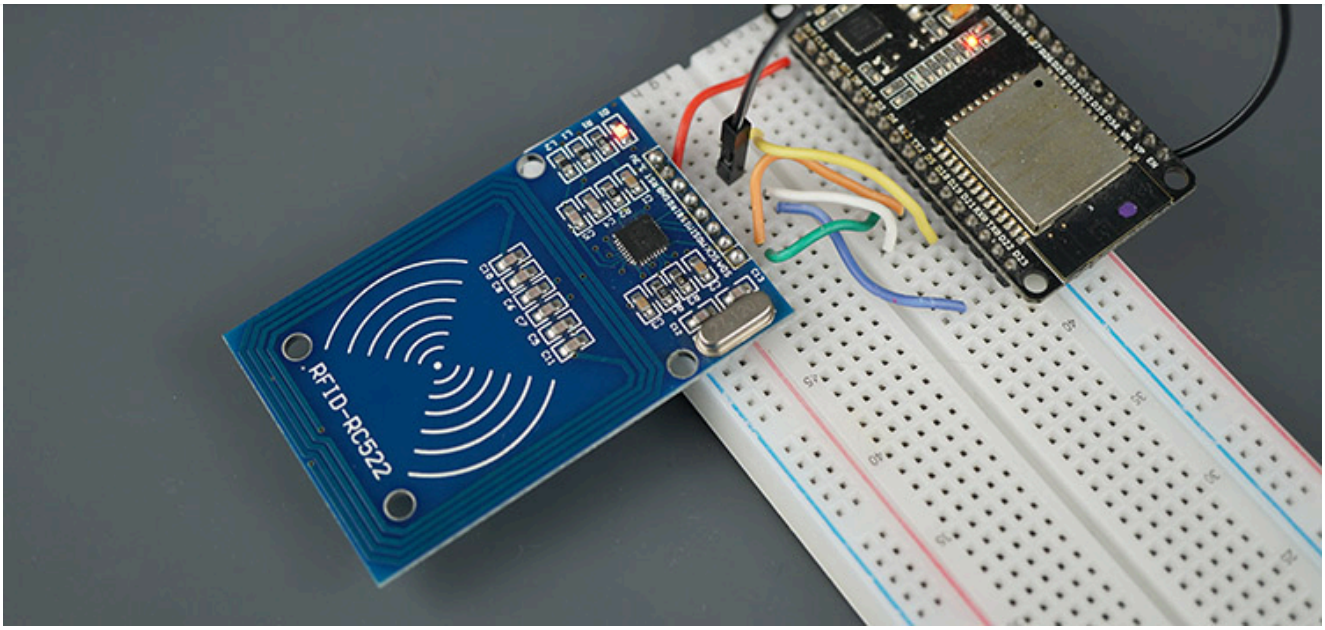
For this tutorial, we'll use the `MFRC522v2.h` library to control the RFID reader. In the Arduino IDE, go to **Sketch** > I**nclude Library** > **Manage Libraries** or click on the Library Manager icon at the left sidebar.

Search for **MFRC522v2** and install the library by GithubCommunity.



MFRC522 RFID Reader/Writer Pinout and Wiring to the ESP32

We'll connect the MFRC522 RFID Reader using the ESP32 default SPI pins. You can use the following table.



MFRC522 RFID Reader	ESP32	Description
SDA	GPIO 5	SPI signal input, I2C data line, or UART data input
SCK	GPIO 18	SPI clock
MOSI	GPIO 23	SPI data input
MISO	GPIO 19	SPI master-in-slave-out, I2C serial clock, or UART serial output
IRQ	Don’t connect	Interrupt pin; signals the microcontroller when an RFID tag is nearby
GND	GND	
RST	GPIO 21	LOW signal to put the module in power-down mode; send a HIGH signal to reset the module

Note: this library is also compatible with the ESP8266, Arduino, and other boards. If you are using a different board, you can check the recommended library pin assigment.

Read RFID Card Raw Data – Code

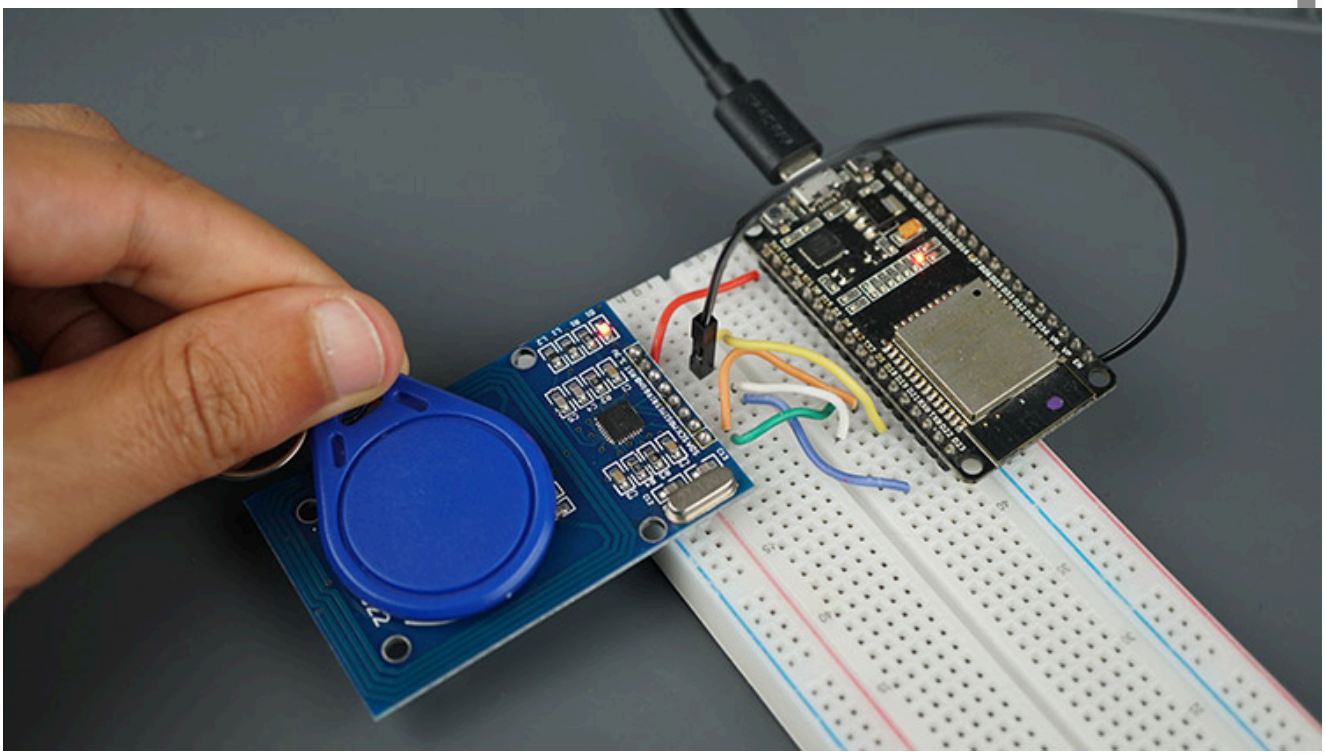
The following example reads raw data from the RFID tag. With this example, you'll better understand how data is stored on the MIFARE 1K tags.

```
/*
  Rui Santos & Sara Santos - Random Nerd Tutorials
  Complete project details at https://RandomNerdTutorials.com/esp32-mf
  Permission is hereby granted, free of charge, to any person obtainir
  The above copyright notice and this permission notice shall be inclu
*/

#include <MFRC522v2.h>
#include <MFRC522DriverSPI.h>
//#include <MFRC522DriverI2C.h>
#include <MFRC522DriverPinSimple.h>
#include <MFRC522Debug.h>

// Learn more about using SPI/I2C or check the pin assigment for your
MFRC522DriverPinSimple ss_pin(5);

MFRC522DriverSPI driver{ss_pin}; // Create SPI driver
//MFRC522DriverI2C driver{};     // Create I2C driver
MFRC522 mfrc522{driver};         // Create MFRC522 instance

void setup() {
  Serial.begin(115200);  // Initialize serial communication
  while (!Serial);       // Do nothing if no serial port is opened (ac

  mfrc522.PCD_Init();    // Init MFRC522 board.
```

View raw code

After wiring the circuit, upload the previous code to your board.

After uploading, open the Serial Monitor at a baud rate of 115200. Approximate your tag to the reader and see the tag info being displayed.



It displays the card UID, the tag type, and the memory blocks. The tag is of type MIFARE with 1KB of memory.

```
Firmware Version: 0x92 = v2.0
Scan PICC to see UID, SAK, type, and data blocks...
Card UID: 82 72 9F 0B
Card SAK: 08
PICC type: MIFARE 1KB
Sector Block   0  1  2  3   4  5  6  7   8  9 10 11  12 13 14 15  AccessBits
  15     63   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         62   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         61   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         60   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
  14     59   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         58   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         57   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         56   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         ...
   3     15   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         14   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         13   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         12   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
   2     11   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         10   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
          9   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
          8   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
   1      7   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
          6   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
          5   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
          4   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
   0      3   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
          2   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
          1   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
          0   82 72 9F 0B  64 08 04 00  01 1E C3 C1  52 E6 F5 1D  [ 0 0 0 ]
```

Here are some essential aspects you need to take into account about how data is stored on the MIFARE 1K tag.

Memory Layout

The memory is organized into 16 sectors (0 to 15), and each sector is divided into 4 blocks (0 to 3), which gives a total of 64 blocks. Each block can store 16 bytes of data (0 to 15).

Manufacturer Block

Sector 0, block 0 stores the UID (first 4 bytes), a CRC check byte, and manufacturer data. This is read-only and cannot be changed (except on UID-changeable cards). As you can see from the results in the Serial Monitor, the UUID of our tag is **82 72 9F 0B**.

The first three blocks of each sector can be used for storing data. The last block of each sector is a **sector trailer**. This sector trailer stores the 2 keys or passwords and controls access to the rest of the blocks in the sector. For example, block number 7 controls block 4, 5 and 6). You should not write on these sectors, otherwise, you'll probably brick your card and it will become unusable.

In summary, the Mifare 1K cards have a net storage capacity of:

```
(16 sectors/card x 3 data blocks/sector X 16 bytes/block) - 16 bytes (first block)
= 752 bytes/card
```

There are 16 sectors with 4 blocks each (64 blocks total). Each block is 16 bytes, giving 1024 bytes in total (1 KB). However, 16 blocks (one in each sector) are sector trailers that store access keys and permissions, not user data. This leaves 48 blocks available for user data. Subtracting 16 bytes for the read-only manufacturer block (Sector 0, Block 0), the effective user storage is 752 bytes.

Read the RFID Card UID – Code

The following example only reads the UID of the RFID tag. This can be useful if you just want to identify the card without caring about the rest of the stored data.

```
/*
  Rui Santos & Sara Santos - Random Nerd Tutorials
  Complete project details at https://RandomNerdTutorials.com/esp32-mf
  Permission is hereby granted, free of charge, to any person obtainir
  The above copyright notice and this permission notice shall be inclu
*/

#include <MFRC522v2.h>
#include <MFRC522DriverSPI.h>
//#include <MFRC522DriverI2C.h>
#include <MFRC522DriverPinSimple.h>
#include <MFRC522Debug.h>
```

```
MFRC522DriverSPI driver{ss_pin}; // Create SPI driver
//MFRC522DriverI2C driver{};     // Create I2C driver
MFRC522 mfrc522{driver};         // Create MFRC522 instance

void setup() {
  Serial.begin(115200);  // Initialize serial communication
  while (!Serial);       // Do nothing if no serial port is opened (ac

  mfrc522.PCD_Init();    // Init MFRC522 board.
  MFRC522Debug::PCD_DumpVersionToSerial(mfrc522, Serial);   // Show de
```

View raw code

This code is quite simple to understand. First, you initialize the reader on the `setup()`.

```
mfrc522.PCD_Init();    // Init MFRC522 board.
MFRC522Debug::PCD_DumpVersionToSerial(mfrc522, Serial); // Show details
Serial.println(F("Scan PICC to see UID"));
```

Then, in the `loop()`, check if there is a card present.

```
// Reset the loop if no new card present on the sensor/reader. This sav
if (!mfrc522.PICC_IsNewCardPresent()) {
  return;
}
```

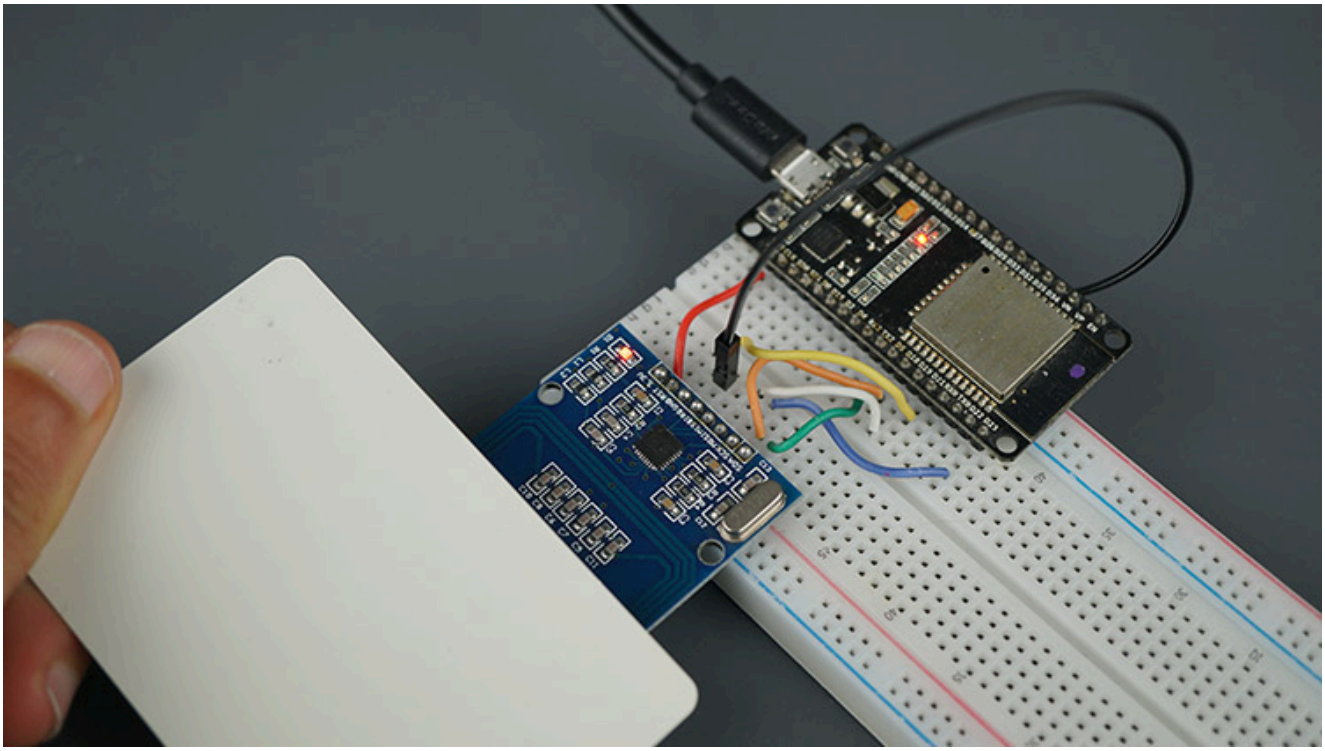
If so, read the card.

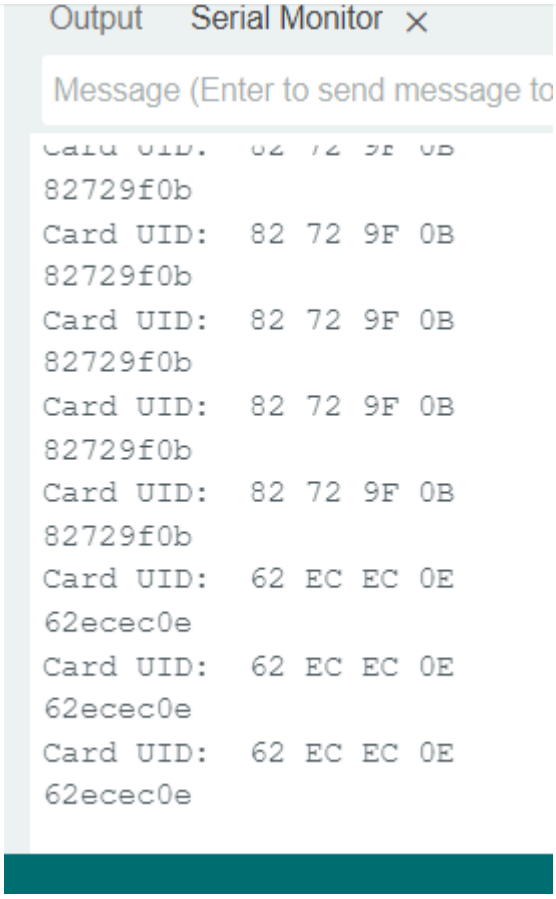
```
// Select one of the cards.
if (!mfrc522.PICC_ReadCardSerial()) {
```

Finally, print the UID and save it on a variable.

```
  Serial.print("Card UID: ");
  MFRC522Debug::PrintUID(Serial, (mfrc522.uid));
  Serial.println();

  // Save the UID on a String variable
  String uidString = "";
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    if (mfrc522.uid.uidByte[i] < 0x10) {
      uidString += "0";
    }
    uidString += String(mfrc522.uid.uidByte[i], HEX);
  }
  Serial.println(uidString);
}
```

Upload the code to your board. Open the Serial Monitor at a baud rate of 115200.





Write/Read Personal Data to and from the RFID Card – Code

In this example, we'll show you how to write custom data on specific blocks. Then, we'll also show you how to read the data from those same blocks.

```
/*
  Rui Santos & Sara Santos - Random Nerd Tutorials
  Complete project details at https://RandomNerdTutorials.com/esp32-mf
  Permission is hereby granted, free of charge, to any person obtainir
  The above copyright notice and this permission notice shall be inclu
*/

#include <MFRC522v2.h>
#include <MFRC522DriverSPI.h>
//#include <MFRC522DriverI2C.h>
#include <MFRC522DriverPinSimple.h>
#include <MFRC522Debug.h>
```

```
MFRC522DriverPinSimple ss_pin(5);

MFRC522DriverSPI driver{ss_pin}; // Create SPI driver
//MFRC522DriverI2C driver{};     // Create I2C driver
MFRC522 mfrc522{driver};         // Create MFRC522 instance

MFRC522::MIFARE_Key key;

byte blockAddress = 2;
byte newBlockData[17] = {"Rui Santos - RNT"};
//byte newBlockData[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};   // CLEA
byte bufferblocksize = 18;
byte blockDataRead[18];
```

View raw code

How Does the Code Work?

We start by including the necessary libraries to set up the RFID reader.

```
#include <MFRC522v2.h>
#include <MFRC522DriverSPI.h>
//#include <MFRC522DriverI2C.h>
#include <MFRC522DriverPinSimple.h>
#include <MFRC522Debug.h>
```

It defines the CS (Chip Select) pin as pin 5, which will be used for SPI communication with the reader.

```
// Learn more about using SPI/I2C or check the pin assigment for your b
MFRC522DriverPinSimple ss_pin(5);

MFRC522DriverSPI driver{ss_pin}; // Create SPI driver
//MFRC522DriverI2C driver{};     // Create I2C driver
MFRC522 mfrc522{driver};         // Create MFRC522 instance
```

Then, we define a variable called `blockAddress`. This specifies the block on the RFID tag where data will be written or read. In this example, `blockAddress` is set to 2, which means the code will interact with block 2 of the card's memory. You can change this value if you want to write to a different block.

```
byte blockAddress = 2;
```

> **Important:** the last block of each sector is a **sector trailer**. You should not write on that sector, otherwise, you'll probably brick your card.

Next, `newBlockData[17]` holds the data you want to write to the card. Right now, it's set to `Rui Santos – RNT`, but you can change it. Make sure it is no longer than 16 bytes.

```
byte newBlockData[17] = {"Rui Santos - RNT"};
```

If you want to clear the block data, you can pass the following bytes array:

```
byte newBlockData[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
```

In the `setup()`, initialize the MFRC522 module.

```
mfrc522.PCD_Init();    // Init MFRC522 board.
```

The default key for the RFID card is also set on the following lines. By default, all bytes are `0xFF` for the factory key. This key allows access to the card's data blocks.

```
  key.keyByte[i] = 0xFF;
}
```

The `loop()` checks if a new RFID card is detected. If a card is present, it reads and displays the card's UID.

```
if (!mfrc522.PICC_IsNewCardPresent() || !mfrc522.PICC_ReadCardSerial())
  delay(500);
  return;
}

// Display card UID
Serial.print("----------------\nCard UID: ");
MFRC522Debug::PrintUID(Serial, (mfrc522.uid));
Serial.println();
```

Next, the code tries to authenticate a specific block on the card using the default key (in this case, block 2).

```
// Authenticate the specified block using KEY_A = 0x60
if (mfrc522.PCD_Authenticate(0x60, blockAddress, &key, &(mfrc522.uid))
  Serial.println("Authentication failed.");
  return;
}
```

Note: `0x60` is a command that specifies the use of `KEY_A` to authenticate. `KEY_A` is one of two keys (`KEY_A` and `KEY_B`) available on RFID cards, each offering different permissions. Using `0x60` means the code is attempting to authenticate with `KEY_A`, which by default is `0xFF 0xFF 0xFF 0xFF 0xFF` on MIFARE RFID cards.

```
// Write data to the specified block
if (mfrc522.MIFARE_Write(blockAddress, newBlockData, 16) != 0) {
  Serial.println("Write failed.");
} else {
  Serial.print("Data written successfully in block: ");
  Serial.println(blockAddress);
}
```

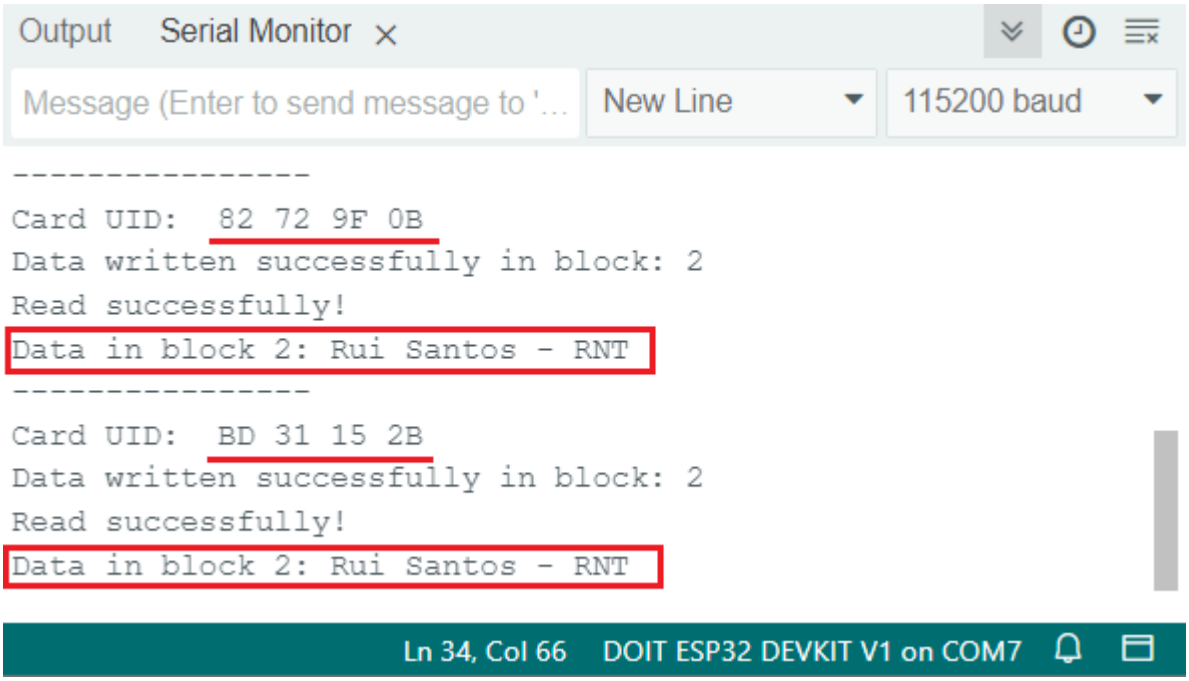
After writing, it reads the data back from the block and prints it to the Serial Monitor.

```
// Read data from the specified block
if (mfrc522.MIFARE_Read(blockAddress, blockDataRead, &bufferblocksize)
  Serial.println("Read failed.");
} else {
  Serial.println("Read successfully!");
  Serial.print("Data in block ");
  Serial.print(blockAddress);
  Serial.print(": ");
  for (byte i = 0; i < 16; i++) {
    Serial.print((char)blockDataRead[i]);  // Print as character
  }
  Serial.println();
}
```

Finally, the code halts communication with the card and stops encryption.

```
// Halt communication with the card
mfrc522.PICC_HaltA();
mfrc522.PCD_StopCrypto1();
```

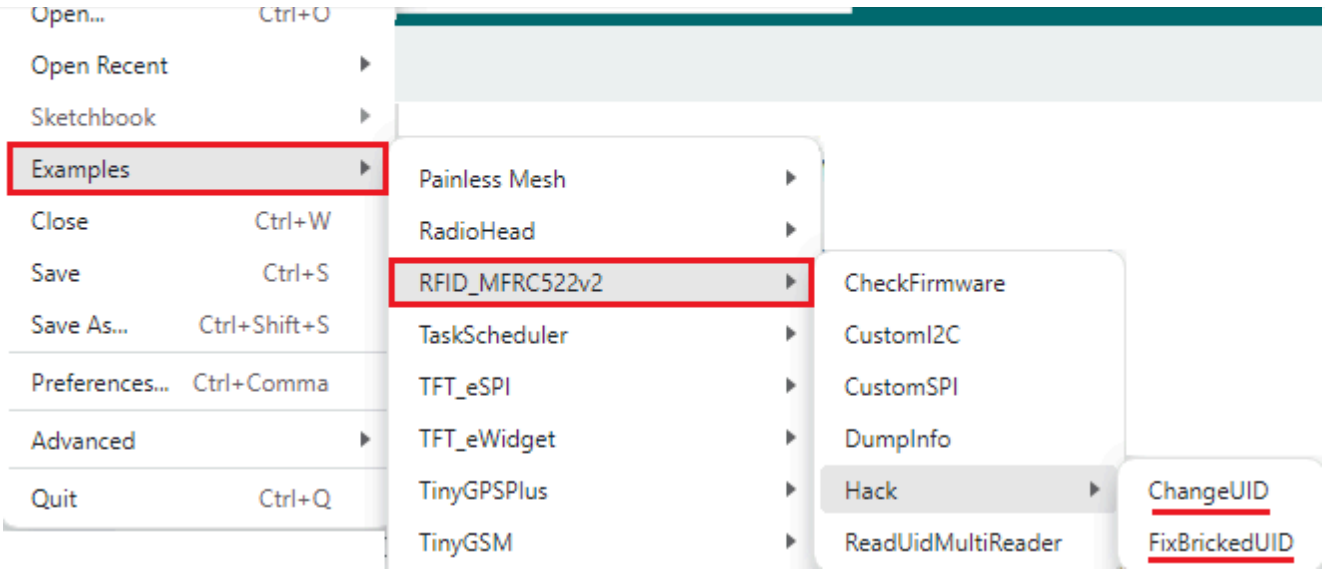
number 2 on two different RFID cards.



Other Useful Examples

There are few examples that come with the `Arduino_MFRC522v2` library that might be helpful. In the *Hack* submenu, you can find the *ChangeUID* and *FixBrickedUID* examples.

Important: most RFID cards don't allow you to modify the UID. This code only works on specific MIFARE Classic cards that have a writable UID block. Most RFID cards have the UID set as read-only, so if you try to change the UID it won't work.



Wrapping Up

In this guide, you learned how to interface the MFRC522 RFID Reader/Writer module with the ESP32. We've taken a look at how storage on Mifare 1K cards works and how to dump the information from the cards. We also learned how to get the card UID, write new data on a specific block and how to read data from a specific block.

We hope you found this guide for the ESP32 with MFRC522 RFID Reader/Writer useful. We have a similar guide for the Arduino board:

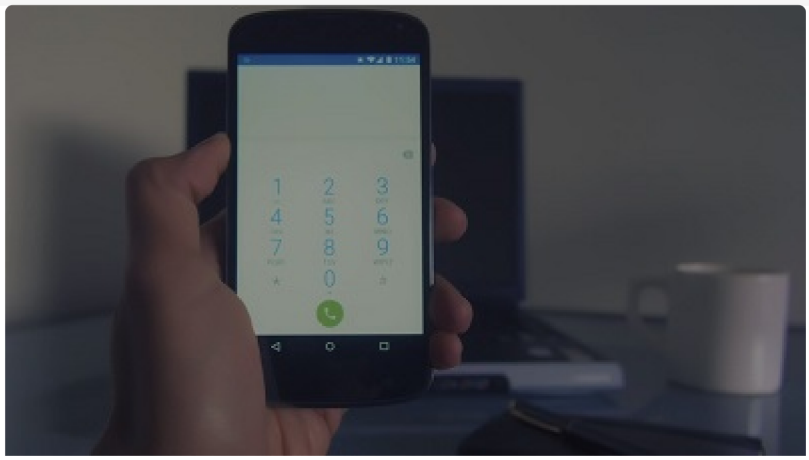
- Security Access using MFRC522 RFID Reader with Arduino

If you would like to learn more about the ESP32, and for inspiration for new projects, make sure to take a look at our resources:

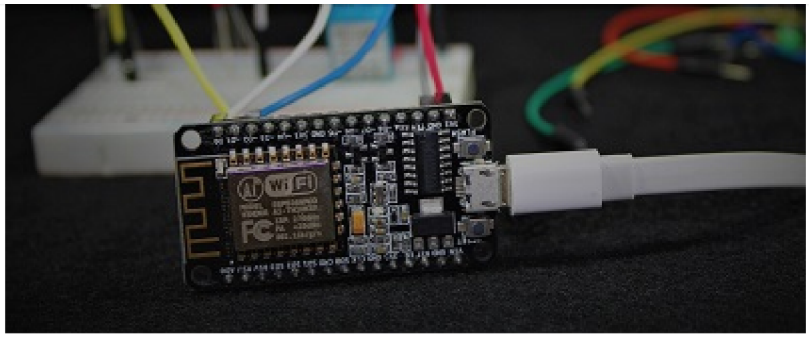
- Learn ESP32 with Arduino IDE (eBook)
- Free ESP32 Projects and Tutorials
- ESP32: 26 Free Guides for Sensors and Modules

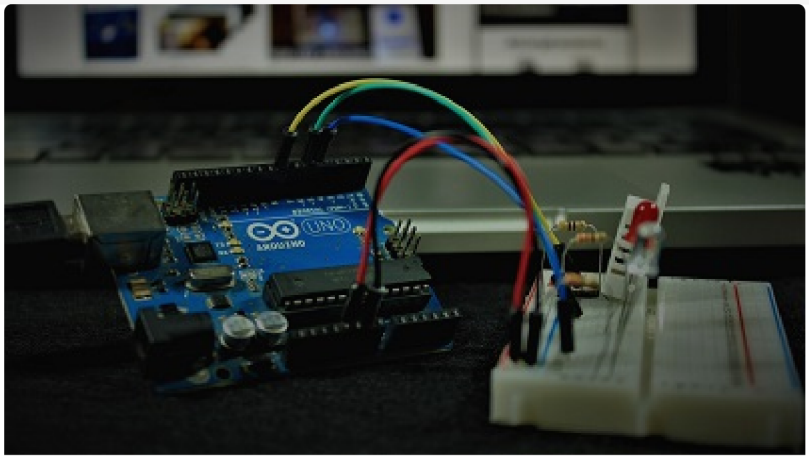
Thanks for reading.

Recommended Resources



Build a Home Automation System from Scratch » With Raspberry Pi, ESP8266, Arduino, and Node-RED.





What to Read Next…

ESP32 Save Data Permanently using Preferences Library

ESP32 Bluetooth Classic with Arduino IDE – Getting Started

ESP32 Web Server: Control Stepper Motor (HTML Form)

Enjoyed this project? Stay updated by subscribing our newsletter!

Your Email Address

SUBSCRIBE

21 thoughts on “ESP32 with MFRC522 RFID Reader/Writer (Arduino IDE)”

Mark

January 17, 2025 at 10:15 pm

Thanks for this great write-up on the MFRC522 RFID reader! It was fantastic for getting me oriented quickly.

I was hoping to use the SPI interface of the Cheap Yellow Display (CYD or ESP32-2432S028R) to control the MFRC522 board, but the fact that ESP32 pin 21 is used for reset was worrying since that is already a shared

library used in this article, it appears that in 25 Jun 2020, v2.0.0 the library quit using the hardware reset pin and now uses only the software reset. You can still find the commented-out hardware reset code near Arduino/libraries/RFID_MFRC522v2/src/MFRC522v2.cpp line 81.

I have not finished my project but I now think that it will be able to work. My next step will be to try your examples after removing the reset pin.

Reply

Mark

January 18, 2025 at 11:24 pm

Success – without connecting anything to the RST line I am able to operate the MFRC522 RFID reader with an ESP32 Dev Module using MIFARE 1KB cards, both read and write. Then I was able to use a SparkFun microSD Sniffer inserted in the Cheap Yellow Display microSD slot to connect to the MFRC522 RFID reader and again I was able to read and write MIFARE 1KB cards.
Note that this was tested using the “RFID_MFRC522v2 by Github Community” library with version “25 Dec 2024, v2.0.6” but I expect it would work without the reset pin using any version later than or equal to 2.0.0.
For anyone interested, here are the connections. Anything not listed is not connected.
MFRC522 RFID microSD Sniffer
——- ——-
CD SDA
CMD MOSI
GND GND
VCC 3.3V
CLK SCK
DAT0 MISO

Sara Santos

January 19, 2025 at 11:37 am

Thanks for sharing.
Regards,
Sara

Reply

Mark

January 19, 2025 at 6:36 am

The unmodified example code from this tutorial runs on the CYD with the MFRC522 RFID reader but when enabling LVGL and touchscreen it does not work. Back to the drawing board! My suspicion is that I will need to use a different “chip enable” signal.

Reply

Mark

January 30, 2025 at 6:47 am

I was able to the the SPI from the CYD MicroSD slot to control the RC522 RF IC Card Sensor Module while also using the CYD screen and touchscreen with the LVGL library.

If interested, details can be found here:
https://github.com/Mark-

Reply

Saimadhav

January 27, 2025 at 1:58 pm

https://randomnerdtutorials.com/esp32-mfrc522-rfid-reader-arduino/#:~:text=The%20following%20example%20reads%20raw%20data%20from%20the%20RFID%20tag.%20With%20this%20example%2C%20you%E2%80%99ll%20better%20understand%20how%20data%20is%20stored%20on%20the%20MIFARE%201K%20tags.

On running this first piece of code in the post, I am unable to see anything on the Serial Monitor. I tried adding a simple Serial.println("Hello World") after Serial.begin(9600). Even that is not getting printed. I have given connections as given in the post.

Also the serial communication works fine otherwise on the board. I had tested with Some simple LED blink code.

Any help on debugging highly appreciated. Thanks!

Reply

Sara Santos

January 27, 2025 at 6:15 pm

Hi.
You need to select 115200 baud rate in the Serial Monitor.
Regards,
Sara

Saimadhav S

January 28, 2025 at 2:17 am

Yes. 115200 baud rate was selected. But still no output in serial monitor.

Reply

Sara Santos

January 28, 2025 at 10:42 am

Hi.
Did you press the ESP32 Reset button after uploading the code?
Regards,
Sara

Reply

Saimadhav

January 28, 2025 at 12:48 pm

I had not pressed the reset button after uploading. It worked!

Thanks,
Regards,
Saimadhav

January 31, 2025 at 3:27 pm

Would have been useful to have read the persons' name and ID previously written to the card.

Reply

MK

February 8, 2025 at 12:58 pm

Hello, I uploaded the code, it did not go past the 'hard resetting via rst pin' part, it was stuck there, I did not recieve any output in my serial monitor

Reply

Sara Santos

February 9, 2025 at 9:42 am

Hi.
That message is normal after uploading the code.
Open the Serial monitor and press the ESP32 reset button.
Regards,
Sara

Reply

February 10, 2025 at 1:46 pm

Hello,
Thanks for responding,
I pressed the reset button but it did not work.
I think the problem is in my ESP32, my variant is the ESP-WROOM-32D.
I tried everything, including changing the baud rate and downloading correct UART bridge drivers.
I tested the UART bridge driver with PUTTY application but PUTTY was unable to access the bridge. It responded with a authentication error.

Reply

Sara Santos
February 12, 2025 at 10:19 am

Hi.
Maybe you need to try another ESP32 board.
Regards,
Sara

Reply

MK
February 12, 2025 at 10:27 am

Thank you

February 13, 2025 at 6:34 am

i had uploaded the code and pressed reset button ,but it doesn't working

Reply

Rui Santos

February 14, 2025 at 2:01 pm

What exactly is not working?

Reply

jack

February 17, 2025 at 12:11 pm

I have an ESP32-S3, and i am trying to get this working for I2C. But i am not receiving any signal on the card.
Please see my code

```
#include <Wire.h>
#include <MFRC522_I2C.h>

#define SDA_PIN 8
#define SCL_PIN 9
#define MFRC522_I2C_ADDRESS 0x28 // Adjust if needed based on your scanner results

MFRC522_I2C mfrc522(MFRC522_I2C_ADDRESS, 0xFF, &Wire); // Instance with I2C address and reset pin
```

```
Serial.println("Starting I2C MFRC522...");
Wire.begin(SDA_PIN, SCL_PIN);
Wire.setClock(400000);

Serial.println("Initializing MFRC522...");
mfrc522.PCD_Init(); // Just call PCD_Init(), no return check
Serial.println("MFRC522 initialized successfully.");
}

void loop() {
Serial.println("Checking for card...");
if (!mfrc522.PICC_IsNewCardPresent()) {
delay(500);
return;
}

if (!mfrc522.PICC_ReadCardSerial()) {
Serial.println("Failed to read card.");
delay(500);
return;
}
Serial.print("Card UID: ");
for (byte i = 0; i < mfrc522.uid.size; i++) {
Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? "0" : "");
Serial.print(mfrc522.uid.uidByte[i], HEX);
Serial.print(" ");
}
Serial.println();
delay(2000);
}
```

Reply

Mark

February 27, 2025 at 10:30 pm

Jack – from my investigations I think the RC522-based Sensor Module does not connect the pins of its controller chip such that I2C would work, even though the chip itself is listed as having this capability. It might be possible to perform surgery on the board to make it work but I have not

sure I am not misunderstanding something.
Look for Adrianotiger post #4 on Mar 2023 on
https://forum.arduino.cc/t/esp32-rfid-rc522-i2c/1100200/3

Reply

Juergen
February 19, 2025 at 9:13 am

Hello,
when compiling the example “Write/Read Personal Data to and from the RFID Card” I get the error message “Compilation error: ‘MIFARE_Key’ in ‘class MFRC522’ does not name a type”
I am using version 2.0.4 of “MFRC522v2”

Juergen

Reply

Leave a Comment

Email *

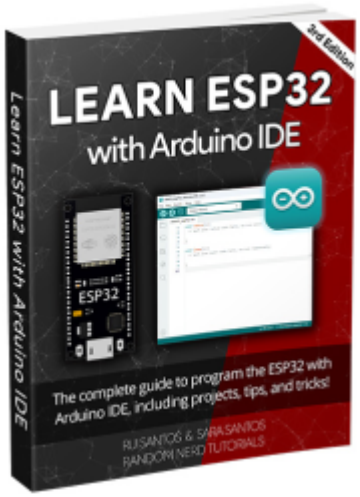
Website

Notify me of follow-up comments by email.

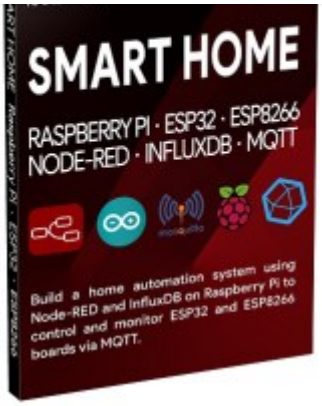
Notify me of new posts by email.

Post Comment

Affiliate Disclosure: *Random Nerd Tutorials is a participant in affiliate advertising programs designed to provide a means for us to earn fees by linking to Amazon, eBay, AliExpress, and other sites. We might be compensated for referring traffic and business to these companies.*



Learn ESP32 with Arduino IDE eBook »
Complete guide to program the ESP32 with Arduino IDE!



SMART HOME with Raspberry Pi, ESP32, and ESP8266 » learn how to build a complete home automation system.

Learn Raspberry Pi Pico/Pico W with MicroPython » The complete getting started guide to get the most out of the the Raspberry Pi Pico/Pico W (RP2040) microcontroller board using MicroPython programming language.

🔥 **Learn LVGL: Build GUIs for ESP32 Projects »** Learn how to build Graphical User Interfaces (GUIs) for ESP32 Projects using LVGL (Light Versatile Graphics Library) with the Arduino IDE.