RC522 – RFID MODULE

Imagine walking into a grocery store, filling your shopping cart with everything you need, and then simply walking out the front door without stopping at a checkout counter. No more waiting in long lines while a cashier scans each item one by one! This is now possible thanks to RFID technology.

RFID allows stores to create automatic checkout systems. With this technology, tiny RFID tags are attached to each product in the store. When you finish shopping, you can walk through a special area where RFID readers detect every item in your cart, almost instantly. The system automatically knows what you're buying and can charge you without you having to unload your cart or wait for someone to scan each item.

If you're interested in creating your own RFID project or system (maybe not a whole grocery store, but something smaller), the RC522

RFID reader/writer module is an excellent choice. The RC522 module can both read information from RFID tags and write new information to them, opening up many possibilities for creative projects with your Arduino.

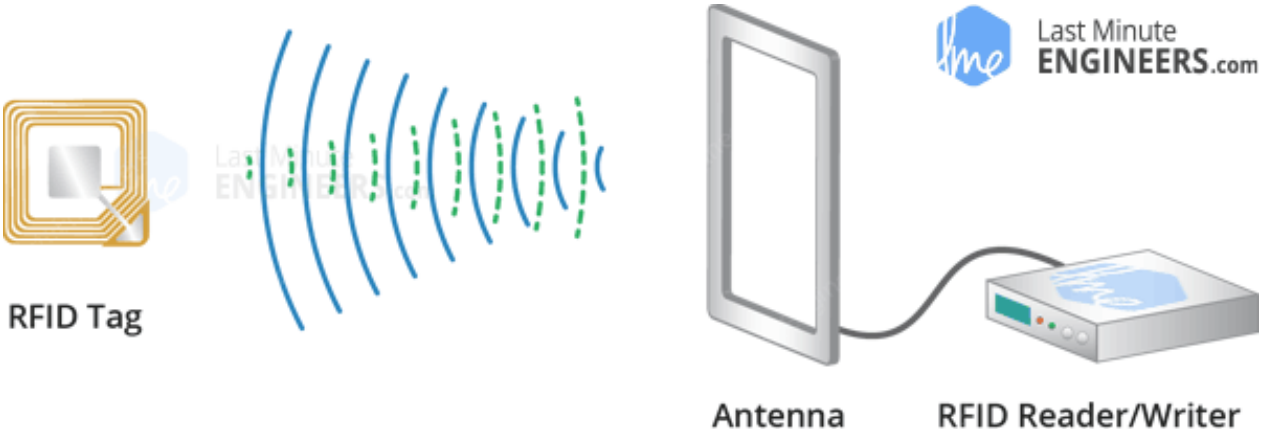
With just a few connections to your Arduino board, you can start building everything from custom access control systems to an inventory tracker, or something entirely unique. Let's get started!

What is RFID technology and how does it work?

An RFID (Radio Frequency Identification) system consists of two main components: a reader and a tag.

The RFID reader contains a radio frequency module and an antenna that creates a high-frequency electromagnetic field around it.

The RFID tag is attached to whatever object we want to identify or track. Most tags are passive, which means they don't have batteries or their own power source. Instead, they wait quietly until they enter the reader's electromagnetic field.



When a tag enters this field, the radio waves from the reader create a tiny electrical current in the tag's antenna. This small current gives

the tag just enough power to wake up its microchip. The microchip contains important information about the object the tag is attached to. Once powered up, the tag sends this information back to the reader. This process is called backscatter, where the tag doesn't generate its own radio signal but instead modifies (or "modulates", if you prefer the technical term) the reader's signal.

The reader picks up this modified signal, decodes the information, and sends it to a computer or microcontroller for further use.

While most RFID tags are passive, there are also active tags that have their own power source, like a small battery. These active tags can communicate over much greater distances because they don't need to rely on the reader's signal for power.

- **RFID Frequencies**

RFID systems operate at different radio frequencies, each with special advantages:

- **Low Frequency (LF):** These work at shorter ranges and are often used for animal tracking or access cards.
- **High Frequency (HF):** These offer a good balance of range and data transfer speed, making them perfect for library books, payment cards, and many hobbyist projects.
- **Ultra-High Frequency (UHF):** These can be read from farther away and are excellent for tracking inventory in warehouses or items moving quickly on assembly lines.

The RC522 RFID reader that we'll be working with operates in the High Frequency range, specifically at 13.56 MHz. This makes it ideal for

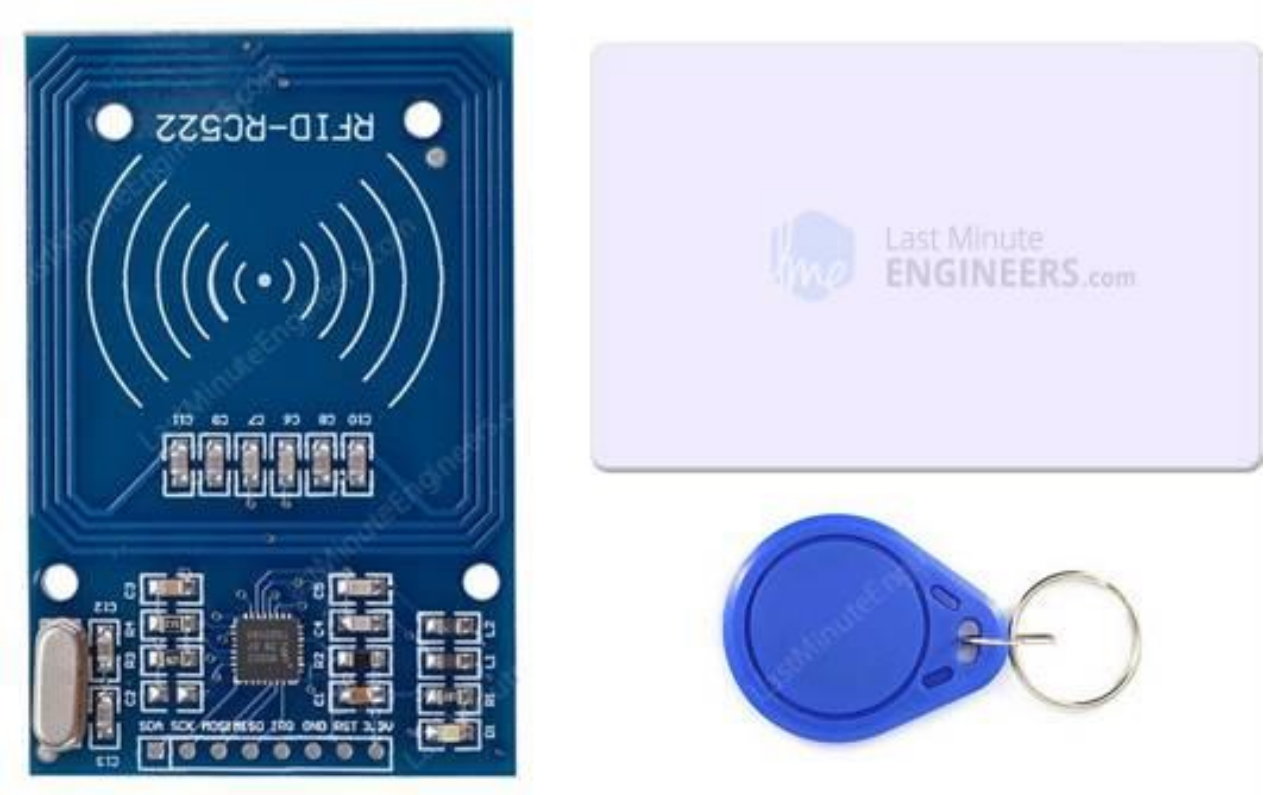
projects where you need to read tags from a few inches away, like electronic door locks, attendance systems, or interactive displays.

Hardware Overview

The RC522 RFID module, based on NXP's MFRC522 chip, is one of the most affordable RFID options available today, often costing less than four dollars. This makes it an excellent choice for hobby projects and experiments.

The module typically comes with two types of RFID tags:

- A card that looks similar to a credit card
- A key fob that can attach to your keychain



Both of these tags contain 1 kilobyte (1KB) of memory. While this might not sound like much compared to microSD cards, it's actually plenty of space for many projects. What's especially exciting is that the RC522 can not only read information from these tags but also write new information to them. This means you can store messages, codes, or identifiers of your choosing on the tags.

The RC522 RFID reader is designed to operate at 13.56 MHz and communicates with RFID tags that follow the ISO 14443A standard (a common set of rules for how RFID devices should work together).

When connecting the RC522 to a microcontroller like an Arduino, you have several options: SPI (fastest of the three), I2C, or UART. This flexibility means you can choose the communication method that works best for your particular project or the microcontroller you're using.

A particularly useful feature of the RC522 is its ability to generate interrupts. Instead of your Arduino constantly asking, "Is there a card nearby?" over and over again, the module itself can alert your Arduino when it detects a tag in range. This makes your projects more efficient and responsive.

The RC522 module itself operates at a voltage between 2.5V and 3.3V, but the good news is that its communication pins are "5V tolerant," meaning you can easily connect it directly to an Arduino or any other 5V microcontroller without needing a logic level converter.

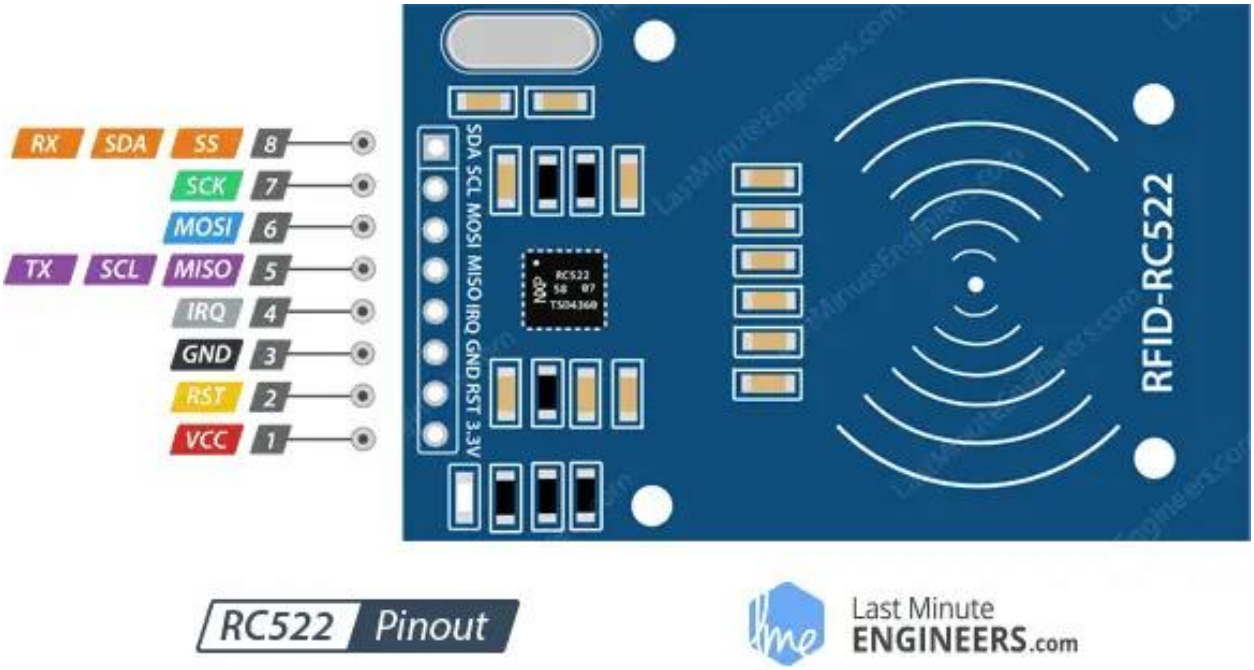
• Technical Specifications

Here are the specifications:

Frequency Range	13.56 MHz ISM Band
Host Interface	SPI / I2C / UART
Operating Supply Voltage	2.5 V to 3.3 V
Max. Operating Current	13-26mA
Min. Current(Power down)	10µA
Logic Inputs	5V Tolerant
Read Range	5 cm

• RC522 RFID Module Pinout

The RC522 module has a total of 8 pins that connect to your Arduino or other microcontroller. Each of these pins serves a specific purpose:



VCC supplies power to the module. It needs between 2.5 to 3.3 volts, so you can safely connect it to the 3.3V output of your Arduino. Be careful though—if you accidentally connect it to the 5V pin, you might permanently damage the module!

RST is the reset/power-down pin. When this pin is pulled low (connected to ground), the module enters power-down mode. In this mode, the oscillator shuts off, and the input pins are disconnected. When the signal changes from low to high (rising edge), the module resets and becomes active again.

GND is the ground pin and should be connected to the GND pin on your Arduino.

IRQ is the interrupt pin that signals your Arduino when an RFID tag is detected nearby.
MISO / SCL / Tx is a multi-purpose pin whose function changes depending on how you're communicating with the module:

- In SPI mode, it's the Master-In-Slave-Out (MISO) pin where data flows from the module to your Arduino
- In I2C mode, it's the Serial Clock (SCL) pin
- In UART mode, it's the transmit (Tx) pin where the module sends data

MOSI is where your Arduino sends data to the RC522 module when using SPI communication.

SCK (Serial Clock) is where the module receives clock pulses from your Arduino when using SPI communication.

SS / SDA / Rx is another multi-function pin:

- In SPI mode, it's the Slave Select (SS) pin
- In I2C mode, it's the Serial Data (SDA) pin

- In UART mode, it's the receive (Rx) pin where the module gets data

This pin often has a square marking on the circuit board, which helps you identify the other pins too.

Wiring an RC522 RFID Module to an Arduino

Now that we have a good understanding of the RC522 module, let's move on to connecting it to our Arduino!

Step 1: Power Connections

Start by connecting the power pins:

- Connect the VCC pin on the RC522 module to the 3.3V pin on your Arduino.
- Connect the GND pin to ground (GND) on the Arduino.

Step 2: Reset and Interrupt Pins

- The RST (Reset) pin can be connected to any digital pin on the Arduino; in our case, we'll use digital pin #9.
- The IRQ (Interrupt Request Output) pin will remain unconnected because the Arduino library we'll be using doesn't support interrupt functionality.

Step 3: SPI Communication Connections

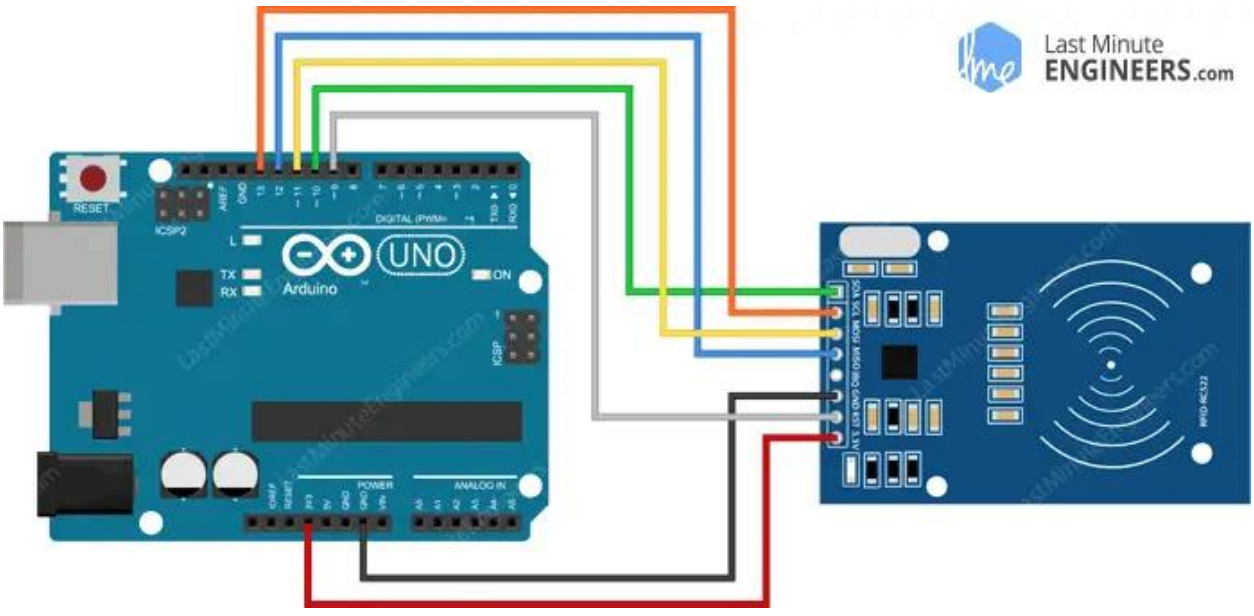
Next, we'll connect the pins used for SPI communication. Since the RC522 module needs to transfer data quickly, it works best when connected to the hardware SPI pins on your Arduino. For Arduino boards like the UNO or Nano V3.0, these pins are: digital 13 (SCK), 12 (MISO), 11 (MOSI), and 10 (CS).

Step 4: Quick References

To make things even easier, here's a quick reference table for the pin connections:

RC522 MODULE	ARDUINO
VCC	5V
GND	GND
RST	9
MISO / SCL / TX	12
MOSI	11
SCK	13
SS / SDA / RX	10

Below, you'll find an image showing how to wire the RC522 module to the Arduino.



Once everything is connected, you're all set and ready to begin coding!

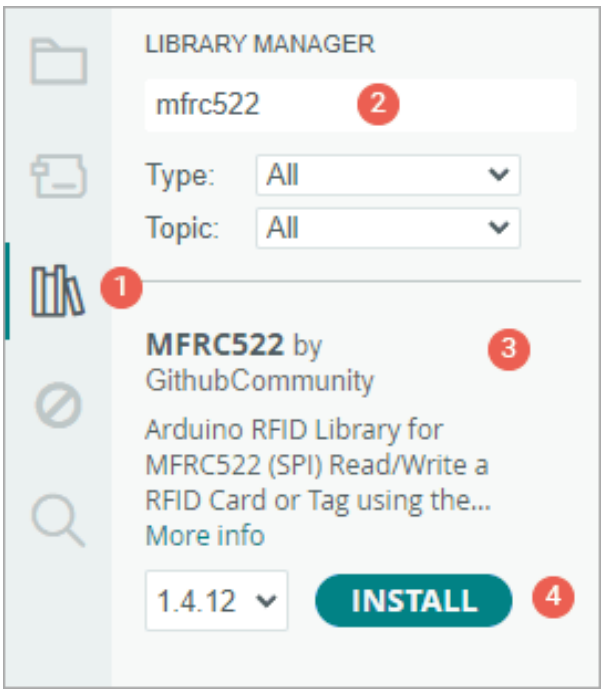
- **Library Installation**

Communicating with an RC522 RFID module might seem complicated at first, but don't worry! There's a library called the MFRC522 library that makes reading and writing RFID tags much easier.

However, this library doesn't come pre-installed with the Arduino IDE, so you'll need to add it yourself first.

To install the library,

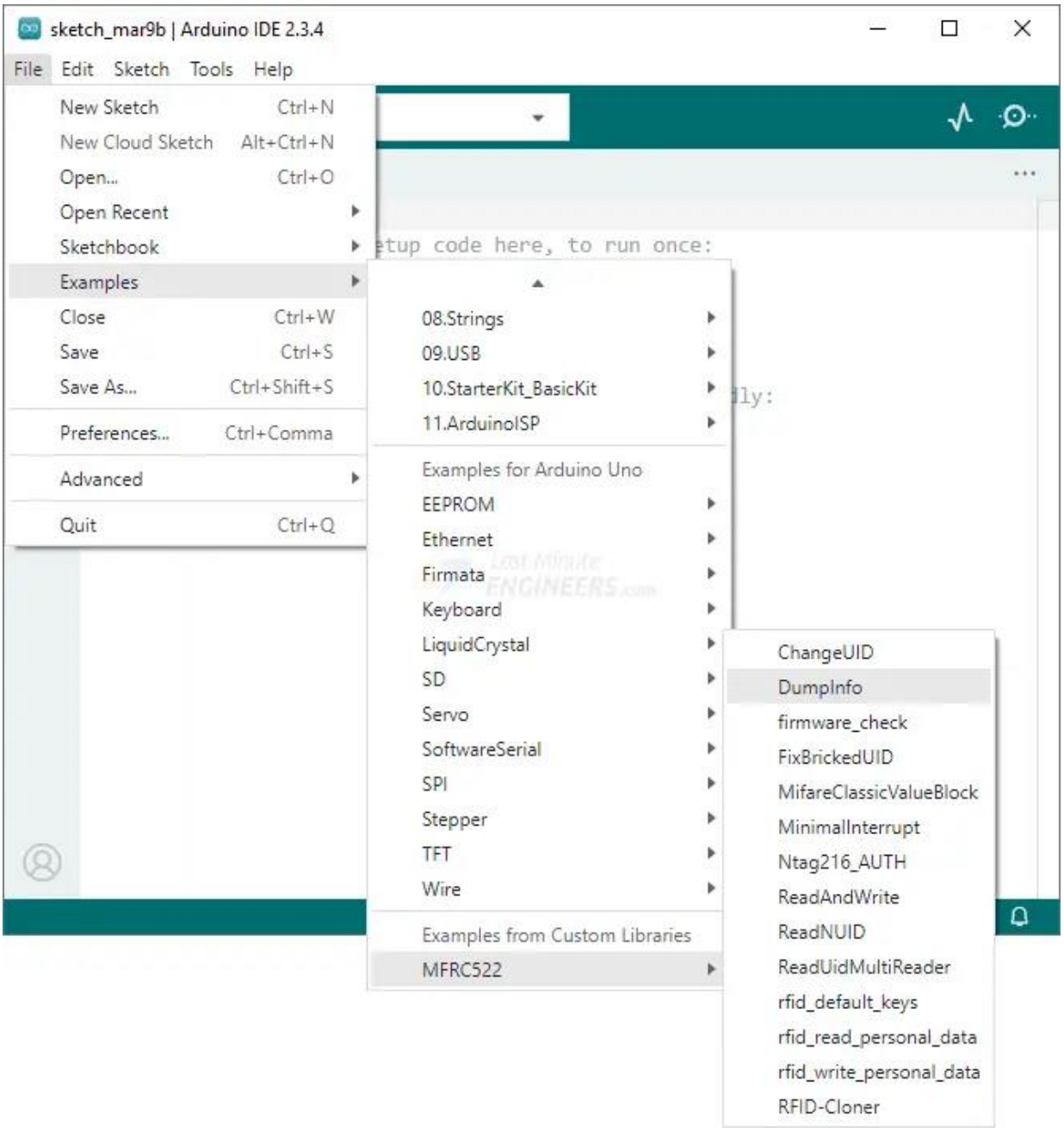
1. First open your Arduino IDE program. Then click on the Library Manager icon on the left sidebar.
2. Type "mfrc522" in the search box to filter your results.
3. Look for the library created by the Github Community
4. Click the Install button to add it to your Arduino IDE.



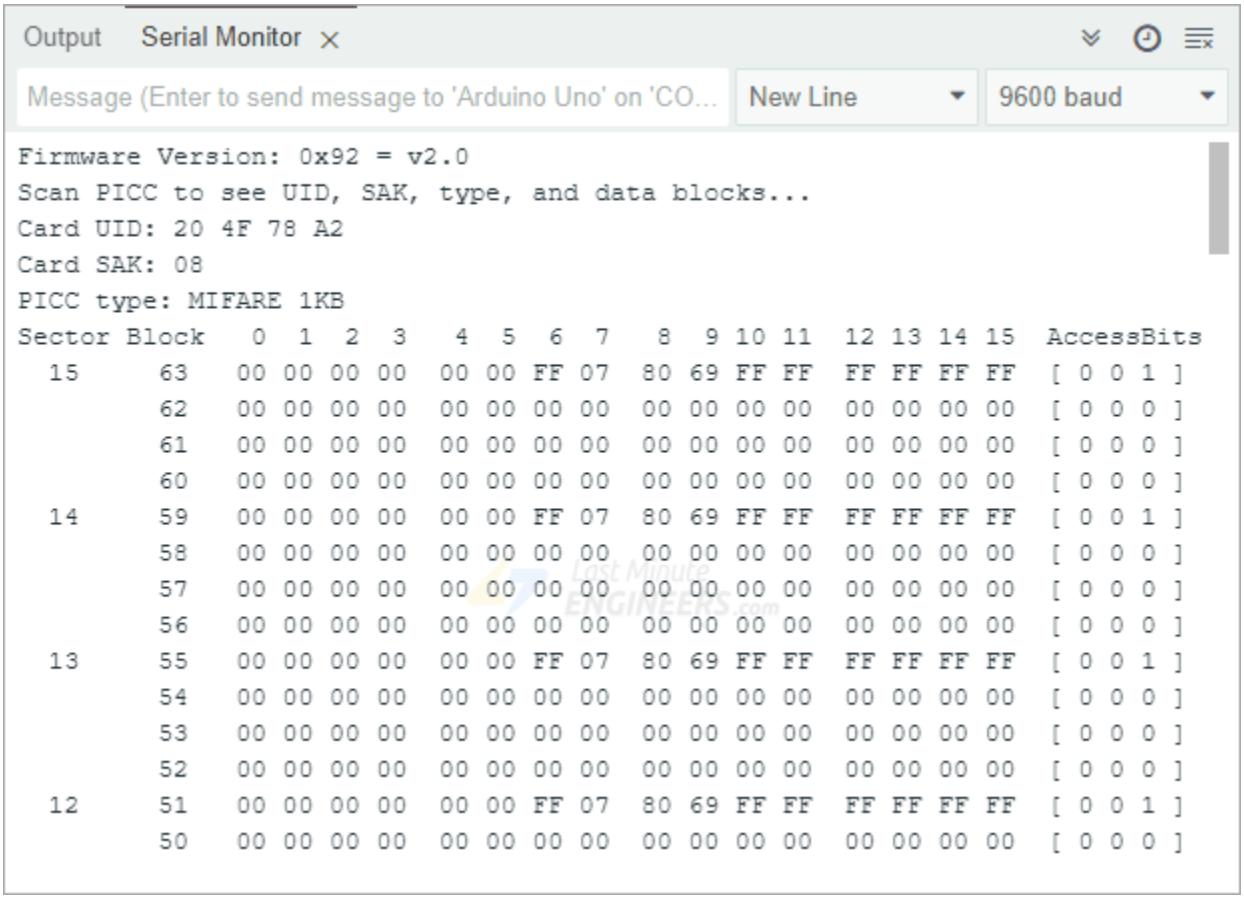
Arduino Code – Reading an RFID Tag

Now that the library is installed, let's move on to reading an RFID tag using an example sketch.

In the Arduino IDE, open the Examples menu, navigate to MFRC522, and select the DumpInfo sketch. This sketch reads an RFID tag and displays all the information stored on it. It's a great way to explore what's on your tags when you're just getting started.



Now, upload the sketch to your Arduino and open the Serial Monitor. When you bring your RFID tag (either the card or key fob) close to the module, you'll see information appear on your screen. This includes the tag's Unique ID (UID), how much memory it has, and even the contents of its entire 1K memory. Just make sure to hold the tag steady near the module until all the information has been read and displayed.



MIFARE Classic 1K Memory Layout

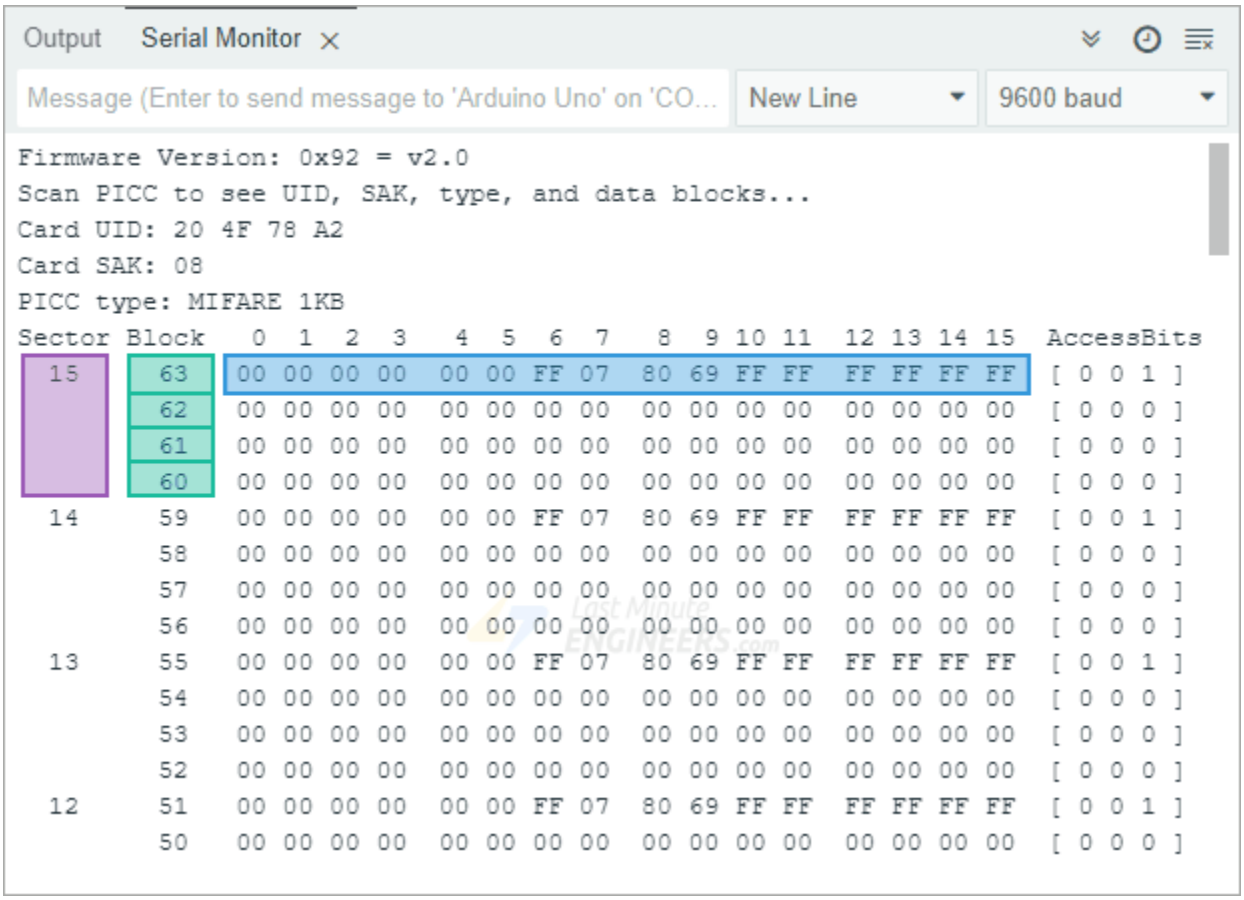
To use the tag effectively, it's important to understand how its memory is organized.

The tag's 1K memory is organized in a very specific way, similar to how an apartment building might be divided into floors, apartments, and rooms.

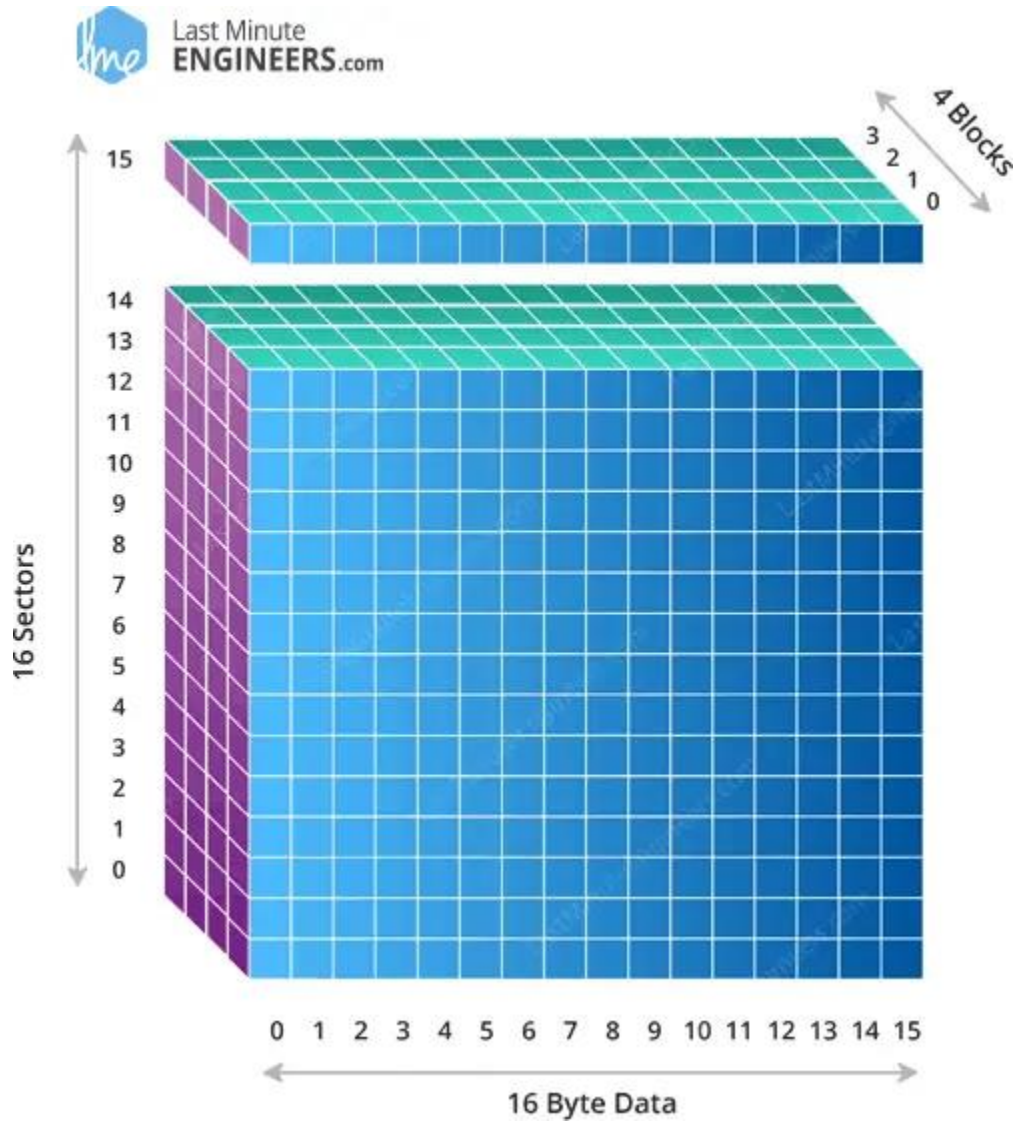
The entire tag memory is divided into 16 sectors (numbered from 0 to 15). Each sector is further divided into 4 blocks (blocks 0 to 3). And each block can store 16 bytes of data (from 0 to 15).

When we do the math, we can see where the "1K" name comes from:
16 sectors × 4 blocks × 16 bytes of data = 1024 bytes = 1K memory

The entire 1K of memory, with all its sectors, blocks, and data, is shown in the image below.

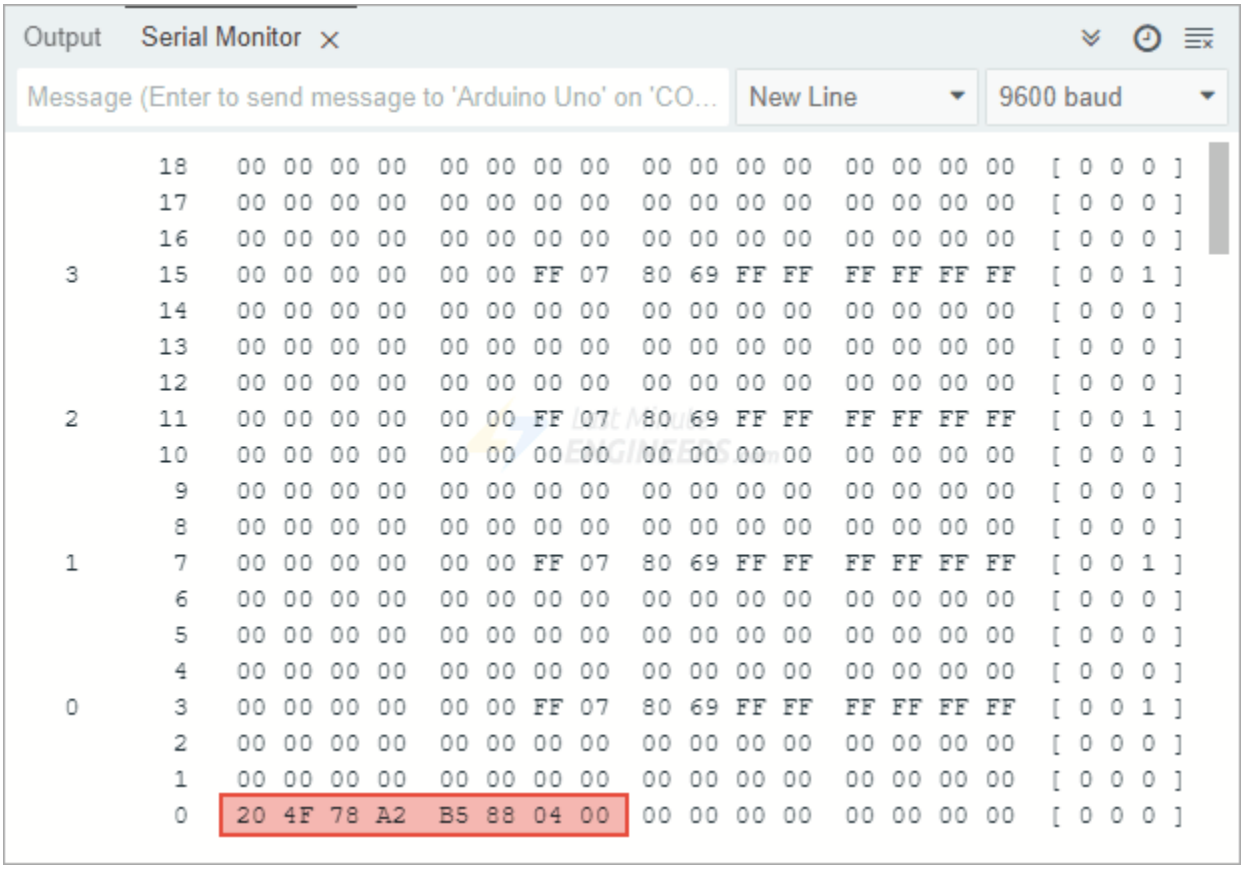


Here's a 3D representation of the MIFARE Classic 1K memory map layout.



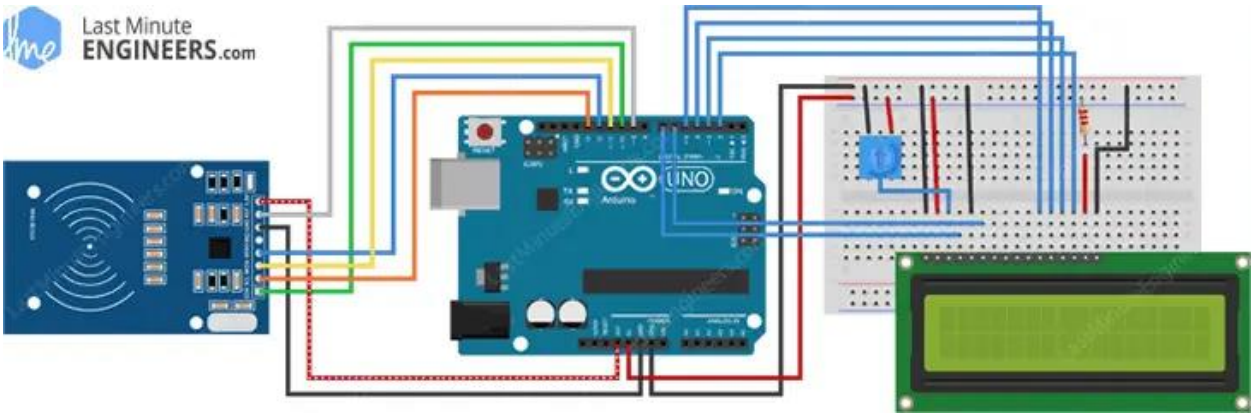
The last block (Block #3) of every sector is called a Sector Trailer. It contains special data known as Access Bits, which control the read and write permissions for the remaining three blocks in that sector. Because of this, only 3 blocks of each sector (Blocks #0, #1, and #2) are actually available for writing your own data. This means only 48 bytes per sector can be used for your own data.

Additionally, the very first block in the entire tag memory (Block #0 of Sector #0) is called the Manufacturer Block. This special block contains factory-programmed information, including the IC Manufacturer data and the Unique Identifier (UID) of the tag. The manufacturer block is highlighted in red in the image below.



Wiring

Before we upload the code and start scanning tags, let's take a quick look at the wiring. The RFID module is connected to the Arduino through the SPI interface, while the LCD screen is wired using digital I/O pins.



Arduino Code

Let's try the sketch below.

```
#include <SPI.h>
#include <MFRC522.h>
#include <LiquidCrystal.h>

#define RST_PIN 9
#define SS_PIN 10

byte readCard[4];
String MasterTag = "20C3935E";  // REPLACE this Tag ID with your Tag ID!!!
String tagID = "";

// Create instances
MFRC522 mfrc522(SS_PIN, RST_PIN);
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);  //Parameters: (rs, enable, d4, d5, d6,
d7)

void setup() {
```

```
  // Initiating
  SPI.begin();          // SPI bus
  mfrc522.PCD_Init();   // MFRC522
  lcd.begin(16, 2);     // LCD screen

  lcd.clear();
  lcd.print(" Access Control ");
  lcd.setCursor(0, 1);
  lcd.print("Scan Your Card>>");
}

void loop() {

  //Wait until new tag is available
  while (getID()) {
    lcd.clear();
    lcd.setCursor(0, 0);

    if (tagID == MasterTag) {

      lcd.print(" Access Granted!");
      // You can write any code here like opening doors, switching on a
relay, lighting up an LED, or anything else you can think of.
    } else {
      lcd.print(" Access Denied!");
    }

    lcd.setCursor(0, 1);
    lcd.print(" ID : ");
    lcd.print(tagID);

    delay(2000);

    lcd.clear();
    lcd.print(" Access Control ");
    lcd.setCursor(0, 1);
    lcd.print("Scan Your Card>>");
```

```
  }
}

//Read new tag if available
boolean getID() {
  // Getting ready for Reading PICCs
  if (!mfrc522.PICC_IsNewCardPresent()) {  //If a new PICC placed to RFID
reader continue
    return false;
  }
  if (!mfrc522.PICC_ReadCardSerial()) {  //Since a PICC placed get Serial and
continue
    return false;
  }
  tagID = "";
  for (uint8_t i = 0; i < 4; i++) {  // The MIFARE PICCs that we use have 4
byte UID
    //readCard[i] = mfrc522.uid.uidByte[i];
    tagID.concat(String(mfrc522.uid.uidByte[i], HEX));  // Adds the 4 bytes
in a single String variable
  }
  tagID.toUpperCase();
  mfrc522.PICC_HaltA();  // Stop reading
  return true;
}
```

Code Explanation:

The program is quite simple. First, we include the necessary libraries: SPI.h for communication with the RFID module, MFRC522.h for controlling the RFID module, and LiquidCrystal.h for the LCD display. We then define the pins used to connect the RFID module and create instances of the MFRC522 and LiquidCrystal objects.

```
#include <SPI.h>
#include <MFRC522.h>
#include <LiquidCrystal.h>
```

```
#define RST_PIN 9
#define SS_PIN 10

// Create instances
MFRC522 mfrc522(SS_PIN, RST_PIN);
LiquidCrystal lcd(7, 6, 5, 4, 3, 2); //Parameters: (rs, enable, d4, d5, d6,
d7)
```

We also set up the master tag, which is the authorized RFID card that grants access. This tag's unique ID is stored as a string. Whenever a card is scanned, its ID will be read and compared to this master tag. If the two match, access is granted; otherwise, access is denied.

```
byte readCard[4];
String MasterTag = "20C3935E";        // REPLACE this Tag ID with your Tag
ID!!!
String tagID = "";
```

Inside the setup() function, we initialize the SPI communication, set up the RFID reader, and configure the LCD display. The LCD screen starts by displaying a welcome message that prompts users to scan their cards.

```
void setup() {
  // Initiating
  SPI.begin();          // SPI bus
  mfrc522.PCD_Init();   // MFRC522
  lcd.begin(16, 2);     // LCD screen

  lcd.clear();
  lcd.print(" Access Control ");
  lcd.setCursor(0, 1);
  lcd.print("Scan Your Card>>");
}
```

The main loop() function continuously waits for a new RFID card to be scanned. When a tag is detected, its unique ID is read, and the program checks whether it matches the master tag. If it does, the LCD screen displays “Access Granted!” If the tag is not recognized, it shows “Access Denied!” The scanned tag’s ID is also displayed on the screen for reference. After a short delay, the system resets and prompts the next user to scan their card.

```
//Wait until new tag is available
  while (getID()) {
    lcd.clear();
    lcd.setCursor(0, 0);

    if (tagID == MasterTag) {

      lcd.print(" Access Granted!");
      // You can write any code here like opening doors, switching on a
relay, lighting up an LED, or anything else you can think of.
    } else {
      lcd.print(" Access Denied!");
    }

    lcd.setCursor(0, 1);
    lcd.print(" ID : ");
    lcd.print(tagID);

    delay(2000);

    lcd.clear();
    lcd.print(" Access Control ");
    lcd.setCursor(0, 1);
    lcd.print("Scan Your Card>>");
  }
}
```

The most important part of this code is the `getID()` function, which is responsible for reading the RFID tag's unique ID. It first checks if a new card is present. If a card is detected, it reads the card's serial number and converts the four-byte unique identifier (UID) into a string. This string is then stored and used for comparison with the master tag. Finally, the function stops reading the card until a new one is scanned.

```
//Read new tag if available
boolean getID() {
  // Getting ready for Reading PICCs
  if (!mfrc522.PICC_IsNewCardPresent()) {  //If a new PICC placed to RFID
reader continue
    return false;
  }
  if (!mfrc522.PICC_ReadCardSerial()) {  //Since a PICC placed get Serial and
continue
    return false;
  }
  tagID = "";
  for (uint8_t i = 0; i < 4; i++) {  // The MIFARE PICCs that we use have 4
byte UID
    //readCard[i] = mfrc522.uid.uidByte[i];
    tagID.concat(String(mfrc522.uid.uidByte[i], HEX));  // Adds the 4 bytes
in a single String variable
  }
  tagID.toUpperCase();
  mfrc522.PICC_HaltA();  // Stop reading
  return true;
}
```

This project is a great starting point for building a fully functional RFID-based access system. You can extend it by connecting a relay to control a door lock, adding a buzzer for sound feedback, or even storing multiple authorized IDs in memory. Try experimenting with the code to customize the system to your needs!

REFERENCE:

https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/