# PINOUT RFID-RC522

| **Pin módulo** | **Pin Arduino** | **Comentario** |
| --- | --- | --- |
| **3.3V** | 3.3V | Alimenta el módulo con 3.3 V. ⚠ No aplicar 5 V; si el cable es largo, añade un condensador de 100 nF cerca del lector para estabilidad. |
| **RST** | D24 | Pin de **reset**: reinicia el MFRC522 por software o hardware. |
| **GND** | GND | **Tierra común**: referencia eléctrica compartida entre Arduino y módulo. |
| **IRQ/RQ** | D26 | **Interrupción**: el módulo avisa al micro de eventos (detección de tarjeta, fin de lectura...). Opcional, se puede dejar sin conectar si no se usa. |
| **MISO** | D50 | *Master In Slave Out*: datos **digitales** que envía el MFRC522 al Arduino por SPI. |
| **MOSI** | D51 | *Master Out Slave In*: datos **digitales** que envía Arduino al MFRC522 por SPI. |
| **SCK** | D52 | Señal de **reloj digital** SPI: marca el ritmo de intercambio de bits. |
| **SDA/SS** | D28 | **Selección de esclavo**: Arduino pone este pin a nivel bajo para comunicar con el MFRC522. |

Razón técnica

* **SPI hardware** en el Mega **no está en pines digitales “normales”** como en el UNO.
  + **MISO → pin 50**
  + **MOSI → pin 51**
  + **SCK → pin 52**
  + El pin **SS (Slave Select)** lo eliges tú, puede ser cualquier digital libre (en tu caso D28).
* Estos pines SPI están conectados directamente al hardware interno del microcontrolador ATmega2560. Usarlos:
  + **Maximiza velocidad** de transferencia (varios MHz).
  + Reduce errores de lectura/escritura en RFID.
  + Libera al micro de “emular” el protocolo por software, ahorrando recursos.
* **RST en D24**: es simplemente un pin digital que podemos usar para resetear el módulo por software, no requiere estar en un pin especial.

**3.3V alimentación**

* **Qué es:** Pin de entrada de alimentación del MFRC522 a 3.3 V. No es digital ni analógico; es una línea de potencia.
* **Para qué sirve:** Provee la energía al chip y su etapa de RF. Una alimentación limpia y estable reduce lecturas erráticas.
* **Tipo de señal:** No aplica (potencia). Trabaja a 3.3 V; las E/S del MFRC522 no son tolerantes a 5 V.
* **Cómo funciona:** El chip consume típicamente 15–30 mA; picos más altos al activar RF. Necesita desacoplo local.
* **Cómo se configura:** Conecta a 3V3 y añade condensadores cerca del módulo:
  + 100 nF cerámico en paralelo con 1–10 µF (electrolítico o cerámico).
  + Evita cables largos; si no hay remedio, refuerza el desacoplo.

**RST D24**

* **Qué es:** Entrada de reset del MFRC522 (activo bajo). Permite reiniciar el chip por hardware.
* **Para qué sirve:** Recuperar el estado conocido si el bus falla, re‑inicializar tras cambios de potencia o antes de configurar registros.
* **Tipo de señal:** Digital (entrada del MFRC522).
* **Cómo funciona:** Cuando el pin se lleva a nivel bajo durante unos micro/milisegundos, el chip se resetea y vuelve a su estado por defecto.
* **Cómo se configura:** Declara el pin del micro como salida y mantenlo en alto durante operación normal. Ejemplo:
  + Con Arduino:

cpp

const int PIN\_RST = 24; // D24

void setup() {

pinMode(PIN\_RST, OUTPUT);

digitalWrite(PIN\_RST, HIGH); // normal

// Reset por pulso (opcional)

digitalWrite(PIN\_RST, LOW); delay(5);

digitalWrite(PIN\_RST, HIGH); delay(5);

}

* + En la librería MFRC522: pasa el pin RST al constructor y llama a PCD\_Init() tras el reset.

**GND**

* **Qué es:** Referencia de tierra común entre el lector y el microcontrolador.
* **Para qué sirve:** Cierra el circuito eléctrico y define el 0 V para todos los niveles lógicos.
* **Tipo de señal:** No aplica (potencia/referencia).
* **Cómo funciona:** Sin una tierra común, los niveles “alto/bajo” no son interpretables y habrá fallos intermitentes.
* **Cómo se configura:** Conecta todos los GND (módulo, micro, fuente, periféricos). Usa un punto de estrella o planos de tierra para minimizar ruido.

**IRQ/RQ D26**

* **Qué es:** Salida de interrupción del MFRC522 (Interrupt Request).
* **Para qué sirve:** Notifica al micro eventos como detección de tarjeta, fin de transmisión o temporizadores, evitando sondeo continuo.
* **Tipo de señal:** Digital (salida del MFRC522 hacia entrada del micro).
* **Cómo funciona:** El MFRC522 establece la línea cuando ocurre un evento habilitado en sus registros de interrupción; se limpia leyendo el registro de estado y/o atendiendo el evento.
* **Cómo se configura:**
  + Conecta a un pin de interrupción del micro (ej. D2/D3 en UNO; en tu caso D26 si tu placa lo soporta).
  + En Arduino: habilita interrupciones y elige flanco según tu configuración del MFRC522.

cpp

const int PIN\_IRQ = 26;

void onRfidIrq() { /\* marcar flag, no hacer trabajo pesado aquí \*/ }

void setup() {

pinMode(PIN\_IRQ, INPUT\_PULLUP); // o INPUT según hardware

attachInterrupt(digitalPinToInterrupt(PIN\_IRQ), onRfidIrq, FALLING);

// En el MFRC522, habilita las IRQ deseadas (CommIEnReg/DivIEnReg) y limpia ComIrqReg.

}

* + Si no usas IRQ: déjalo sin conectar y realiza sondeo en loop.

**MISO D28**

* **Qué es:** Línea SPI Master‑In Slave‑Out. El MFRC522 envía datos al micro por aquí.
* **Para qué sirve:** Transmite al micro las respuestas a comandos SPI (registros, FIFOs, estados).
* **Tipo de señal:** Digital (datos en serie).
* **Cómo funciona:** En cada ciclo de reloj SCK, el MFRC522 desplaza un bit hacia el micro. El muestreo ocurre en el flanco definido por el modo SPI.
* **Cómo se configura:**
  + Usa los pines SPI hardware de tu placa. Importante: tus asignaciones D28/D30/D32/D34 deben corresponder a SPI hardware; si no, re‑cablea.
    - UNO/Nano: MISO D12.
    - Mega2560: MISO D50.
    - ESP32: pines configurables pero usa VSPI/HSPI.
  + En Arduino:

cpp

#include <SPI.h>

SPI.begin(); // inicializa MISO/MOSI/SCK hardware

SPI.beginTransaction(SPISettings(4000000, MSBFIRST, SPI\_MODE0));

// bajar SS, transferir, subir SS

SPI.endTransaction();

* + La librería MFRC522 asume SPI hardware; si no puedes usarlo, busca un wrapper de SoftSPI compatible o cambia de pines al bus hardware.

**MOSI D30**

* **Qué es:** Línea SPI Master‑Out Slave‑In. El micro envía datos/comandos al MFRC522 por aquí.
* **Para qué sirve:** Escribe registros, envía comandos y payloads a la FIFO del lector.
* **Tipo de señal:** Digital (datos en serie).
* **Cómo funciona:** El micro coloca un bit en MOSI; el MFRC522 lo captura según CPOL/CPHA del modo SPI durante SCK.
* **Cómo se configura:**
  + Usa el pin MOSI hardware de tu placa (UNO D11, Mega D51, etc.).
  + Configura SPI a 3.3 V lógicos y modo del MFRC522: MSBFIRST, SPI\_MODE0, frecuencia 1–10 MHz (4 MHz es conservador).

cpp

SPI.beginTransaction(SPISettings(4000000, MSBFIRST, SPI\_MODE0));

// transferencias SPI.transfer(...)

SPI.endTransaction();

**SCK D32**

* **Qué es:** Reloj del bus SPI generado por el maestro.
* **Para qué sirve:** Sincroniza el envío/recepción de bits entre micro y MFRC522.
* **Tipo de señal:** Digital (onda cuadrada alto/bajo).
* **Cómo funciona:** Marca los flancos en los que los datos se deben establecer y muestrear, definidos por CPOL/CPHA (SPI\_MODE0 para MFRC522).
* **Cómo se configura:**
  + Usa el SCK hardware de tu placa (UNO D13, Mega D52, etc.).
  + Selecciona una frecuencia segura: 1–4 MHz para prototipos con cables largos; hasta 10 MHz con cableado corto y limpio.
  + Configura SPI\_MODE0: CPOL=0, CPHA=0.

**SDA/SS D34**

* **Qué es:** Señal de selección de esclavo SPI (SS/CS/NSS). En muchos módulos aparece serigrafiado como “SDA”, pero en SPI es SS.
* **Para qué sirve:** Habilita al MFRC522 en el bus cuando está en nivel bajo. Cuando está alto, el lector ignora MOSI/SCK y libera MISO.
* **Tipo de señal:** Digital (control).
* **Cómo funciona:** El maestro baja SS antes de una transacción, realiza las transferencias y vuelve a subir SS al terminar; el estado de SS delimita comandos.
* **Cómo se configura:**
  + Declara el pin SS como salida en tu micro. En placas como Mega, mantén el pin SS hardware (D53) como salida para conservar el modo maestro.

cpp

const int PIN\_SS = 34; // tu cableado

void setup() {

pinMode(PIN\_SS, OUTPUT);

digitalWrite(PIN\_SS, HIGH); // inactivo

SPI.begin();

}

void xferByte(byte reg, byte val) {

digitalWrite(PIN\_SS, LOW);

SPI.transfer(reg);

SPI.transfer(val);

digitalWrite(PIN\_SS, HIGH);

}

* + En la librería MFRC522: pasa el pin SS al constructor. Asegúrate de que el resto de líneas SPI son las de hardware de tu placa.

**Recomendaciones de integración**

* **Coherencia de pines SPI:** Verifica que MISO/MOSI/SCK coinciden con los pines SPI hardware de tu placa. Si no, re‑cablea o usa una librería que soporte SoftSPI.
* **Niveles lógicos:** Todo a 3.3 V. Si tu micro es 5 V, usa adaptación de nivel hacia el RC522 en MOSI/SCK/SS/RST.
* **SPI settings:** MSBFIRST, SPI\_MODE0, 1–4 MHz para empezar; sube si el enlace es estable.
* **IRQ opcional:** Úsalo si necesitas bajar consumo o latencia; si no, el sondeo funciona bien para prototipos.