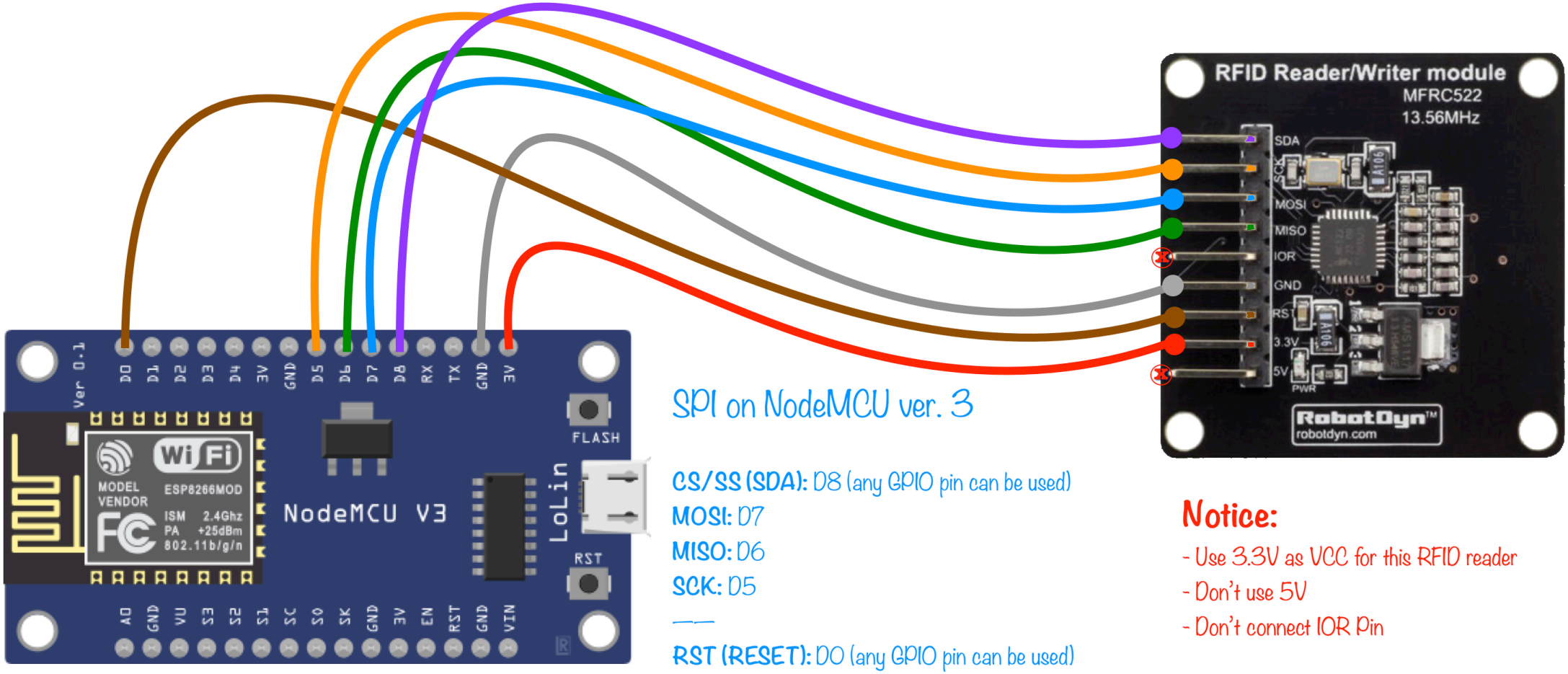
I. Hardware

Wiring Diagram



SPI on NodeMCU ver. 3

CS/SS (SDA): D8 (any GPIO pin can be used)
MOSI: D7
MISO: D6
SCK: D5

——

RST (RESET): D0 (any GPIO pin can be used)

Notice:

- Use 3.3V as VCC for this RFID reader
- Don't use 5V
- Don't connect IOR Pin

II. Software:: Environment Setup

Step 1: Install CP2102 on your PC and then reboot your PC

Silicon Labs CP2102 USB-to-Serial driver allows your computer to recognize and communicate with CP2102-based USB-to-UART (Universal Asynchronous Receiver-Transmitter) bridge devices. By installing the driver, your computer can detect and properly assign a COM port to the CP2102 USB-to-UART adapter, allowing serial communication.
Why Reboot the PC After Installation? After installing the driver, rebooting ensures the driver is properly loaded and initialized by the operating system.

Step 2: Integrate Arduino IDE with ESP8266 devices (NodeMCU v3 is a part of ESP8266 family)

a) Let's open the Arduino IDE.
b) Go to File > Preferences in the Arduino IDE
c) Copy the URL https://arduino.esp8266.com/stable/package_esp8266com_index.json and then paste it in the field "Additional Boards Manager URLs".
d) Click OK to close the preferences window.
e) Go to Tools > Board > Boards Manager
f) Navigate to esp8266 by ESP8266 Community and install it.

Step 3: Select your board

Go to Tools > Board > esp8266 > Generic ESP8266

Step 4: Select the port

Connect the NodeMCU to your PC.

On the Arduino IDE, go to: Tools > Port
You'll see a list of available serial ports. Disconnect the NodeMCU from your PC.

Go again to: Tools > Port
Note the ports that are currently listed.(Fewer?). Reconnect the NodeMCU to your PC.

Go again to: Tools > Port
A new port will appear. This is the one assigned to your device. Select that port.

Step 5: Libraries

The MFRC522 library for Arduino allows communication with the MFRC522 RFID module over SPI. It provides functions for reading card UID, authenticating, and reading/writing data to MIFARE cards.

📌 Install via Arduino Library Manager: Search for “MFRC522 by Miguel Balboa”.
📌 GitHub Repository: MFRC522 Library

III. Software: Code for NodeMCU v3

Link to the code:

https://github.com/benax-rw/RFID-Technology-Made-Easy/blob/main/IoT-for-RFID/transact-and-transfer/transact-and-transfer.ino

Highlights in the code:

Line 24: connectToWiFi("Benax-POP8A", "ben@kushi");

Benax-POP8A is the SSID name. Please change to your SSID you want to use.

ben@kushi is the WiFi key for Benax-POP8A SSID: Please change to your key according to the SSID you choose.

Line 216: transferData(mData, "/projects/4e8d42b606f70fa9d39741a93ed0356c/y2-2025/upload.php");

Of course you have a different address: If you don't remember how to check the address to your upload.php file, please watch the video https://www.youtube.com/watch?v=pLzxo7PBQDQ

Upload the code, open the Serial Monitor, make sure the baud rate is set to 115200, place the PICC with enough balance near the RFID card reader (in 2 cm or less) and then check what's being printed on the Serial Monitor.

Open your project link from https://foreach.benax.rw/?go=exhibition and check if data is uploaded. Otherwise, DEBUG, DEBUG and DEBUG until you make it.

What the code does (part 1): In Summary

This ESP8266-based RFID system reads and writes balance data from MIFARE cards using the MFRC522 module.

It connects to WiFi, authenticates the card, reads the balance, and deducts a transport fare if sufficient funds are available.

The new balance is then written back to the card.

Transaction data, including the card UID, initial balance, and fare, is sent to a remote server (iot.benax.rw) using an HTTP POST request.

The system confirms successful data transmission on the Serial Monitor, making it a streamlined RFID-based payment solution.

What the code does (part 2): Functions

Here is a list of functions in the code along with their purposes:

1. setup() – Initializes Serial communication, WiFi, SPI, and the MFRC522 RFID module.
2. loop() – Continuously checks for a new RFID card, reads its balance, processes the transaction, and updates the card.
3. connectToWiFi(const char* ssid, const char* passwd) – Connects the ESP8266 to a WiFi network.
4. connectToHost(const int httpPort) – Establishes a connection to the remote server.
5. transferData(String data, const char* filepath) – Sends transaction data to the remote server using HTTP POST.
6. getFeedback(String success_msg) – Reads and verifies the server's response to confirm if the transaction was successful.
7. readBalanceFromCard(byte blockNumber, byte readingBuffer[]) – Reads the balance stored in a specific RFID card block.
8. saveBalanceToCard(byte blockNumber, byte writingBuffer[]) – Writes the updated balance back to the RFID card.
9. operateData(byte blockNumber, String initialBalance) – Processes the transaction by checking the balance, deducting fare, updating the card, and sending data to the server.
10. getUUID() – Retrieves the unique ID (UID) of the RFID card.

Any Clarification needed?

Please, don't hesitate to reach me out and ask.

gabriel@benax.rw