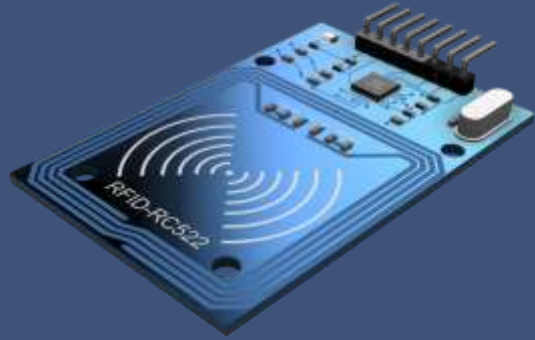


RFID

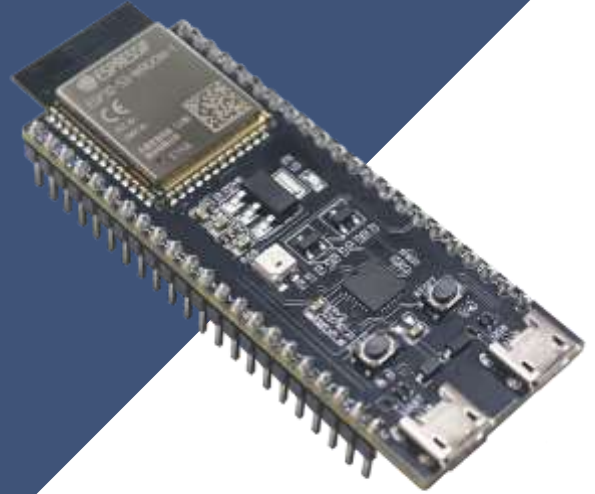
ID u EEPROM-u MIKROKONTROLERA



RFID RC522 READER

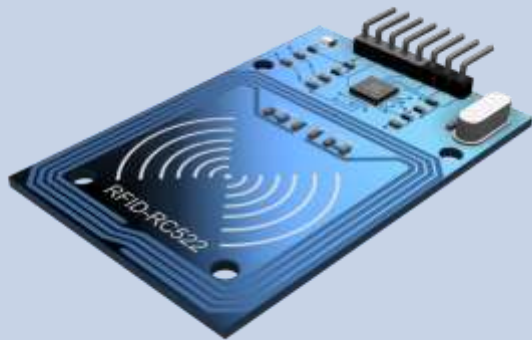


RFID TAGS AND CARDS



ESP32 S3 DEVELOPMENT BOARD

1



RFID RC522 READER



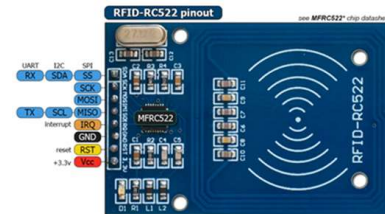
RC522 Chip IC radna frekvencija: 13.56MHz, Brzina razmjene podataka: Max. 10Mbit/s

Dimenzije: 40mm x 60mm





RFID RC522



RFID RC522 čitač je popularan RFID (Radio-Frequency Identification) modul koji radi na frekvenciji od 13,56 MHz.

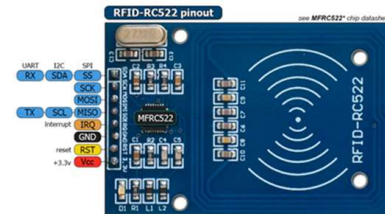
Obično se koristi za čitanje i pisanje RFID oznaka i kartica, posebno onih usklađenih sa standardom ISO/IEC 14443 tipa A, uključujući Mifare kartice.

Evo kratkog opisa čitača RC522:

- **Frekvencija:** radi na frekvenciji od 13,56 MHz, što je često korištena frekvencija za RFID komunikaciju.
- **Kompatibilnost:** Podržava različite RFID oznake i kartice u skladu sa standardom ISO/IEC 14443 tipa A, uključujući Mifare kartice kao što je Mifare 1K.
- **Komunikacijski interfejs:** Obično se povezuje s mikrokontrolerima ili razvojnim pločama kao što je Arduino putem **SPI (Serial Peripheral Interface)** protokola, što olakšava integraciju u različite projekte.
- **Domet očitavanja:** Učinkoviti domet očitavanja zavisi o faktorima kao što su dizajn antene i napajanje, ali obično je unutar nekoliko centimetara.
- **Funkcionalnost:** Mogućnost čitanja i pisanja podataka na RFID oznake i kartice. Može čitati jedinstvene identifikacijske brojeve (UID) pohranjene na oznakama/karticama i u nekim slučajevima dodatne podatke pohranjene u memorijskim sektorima.



RFID RC522



- **Radni napon:** radi na 3.3 V.
- **Antena:** Obično dolazi s ugrađenom antenom, iako se vanjske antene mogu spojiti za prošireni domet ili posebne primjene
- **Primjene:** Obično se koriste u raznim projektima i aplikacijama uključujući sisteme kontrole pristupa, upravljanje inventarom, sisteme praćenja prisutnosti i još mnogo toga.

Modul RC522 relativno je jednostavan za korištenje i pruža troškovno isplativo rješenje za projekte i aplikacije temeljene na RFID-u. Stekao je popularnost u zajednici proizvođača zbog svoje svestranosti i lakoće integracije s platformama mikrokontrolera, poput Arduina.



RFID RC522

Za rad sa RC522 čitačem iz Arduino razvojnog okruženja potrebno je instalirati biblioteku, koja se može preuzeti sa linka:

<https://github.com/miguelbalboa/rfid>

Za instaliranje biblioteke potrebno je odraditi sljedeća tri koraka:

Dodajte biblioteku selektovanjem Add ZIP u SKETCH meniju, INCLUDE Library opcija.

Otvoriti arduino IDE

Zatim, selektovati .zip fajl sa lokacije na kojoj je fajl sačuvan.

Detaljnije informacije o biblioteci mogu se vidijeti na adresi:

http://www.neilkolban.com/esp32/docs/cpp_utils/html/class_m_f_r_c522.html

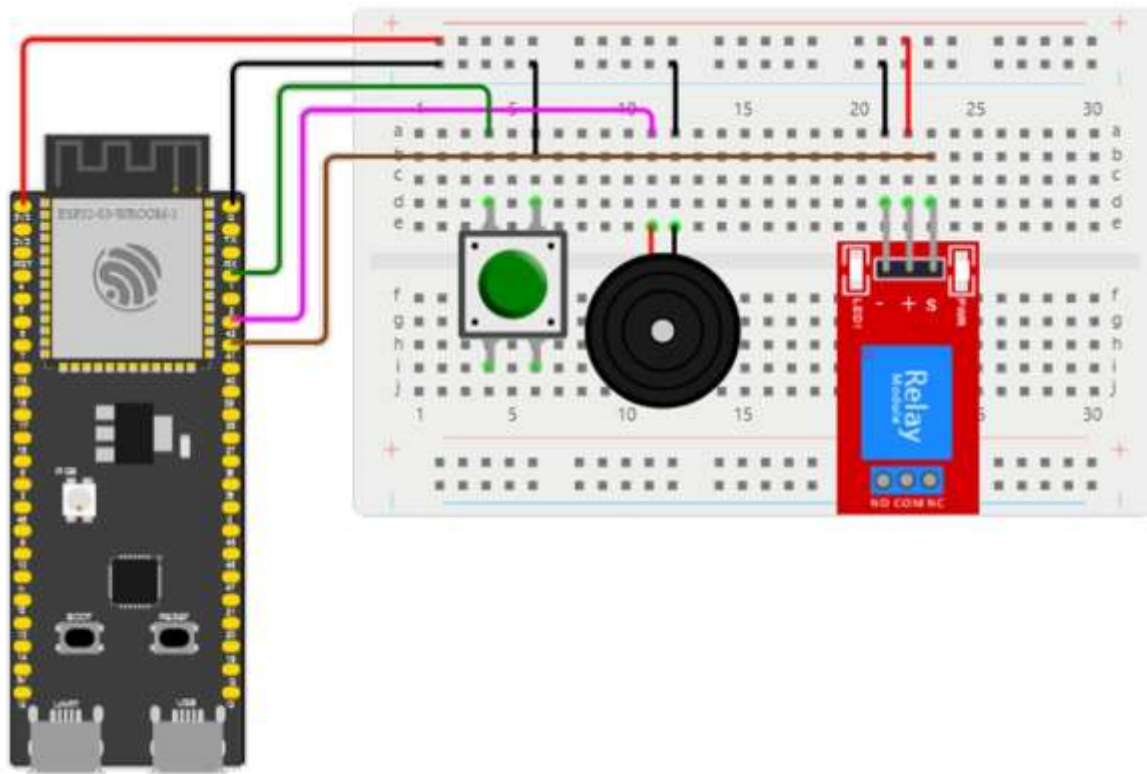
2

POVEZIVANJE

RFID RC 522 i ESP32S3



KAKO POVEZATI TASTER, BUZER I RELE



Uspješnost povezivanja
RFID-RC522 sa ESP32S3,
kao i tastera, bazera, relea
provjeriti pomoću skeča:

[IDCardBasic.ino](https://www.idcardbasic.ino)

3



ID u EEPROM-u MIKROKONTROLERA

Konfigurazioni podaci

CSTART	
	▪
	▪
	▪

ID brojevi
kartica

[illegible]

Događaji



KONFIGURACIONI PODACI

Configuration data

CSTART = 0

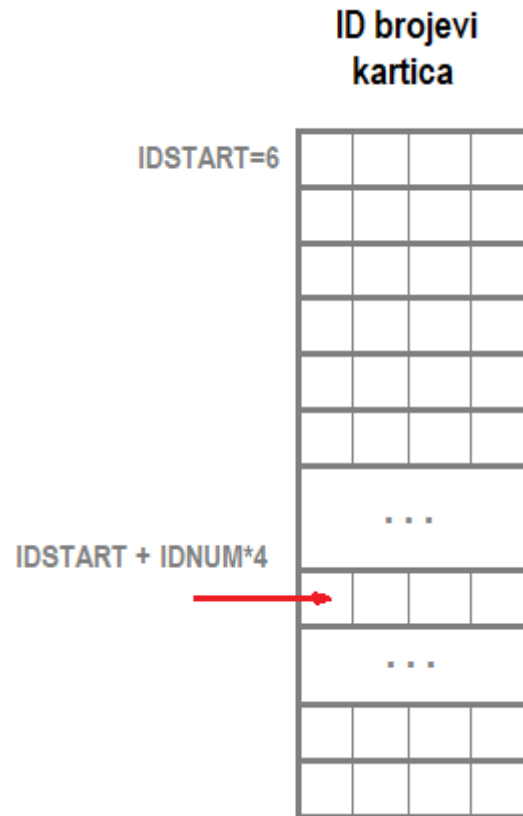
IDNUM
IDSTART
IDMST0
IDMST1
IDMST2
IDMST3

```
// Writing New Master ID to EEPROM
byte writeNewMasterID(byte* newMasterCard) {
    uint8_t start = 2; // Figure out where the next slot starts
    for ( uint8_t j = 0; j < 4; j++ ) { // Loop 4 times
        EEPROM.write( start + j, newMasterCard[j] ); // Write the array
    }
    EEPROM.commit();
    Serial.println(F("Succesfully writed New Master ID record to EEPROM"));
    return (1);
}

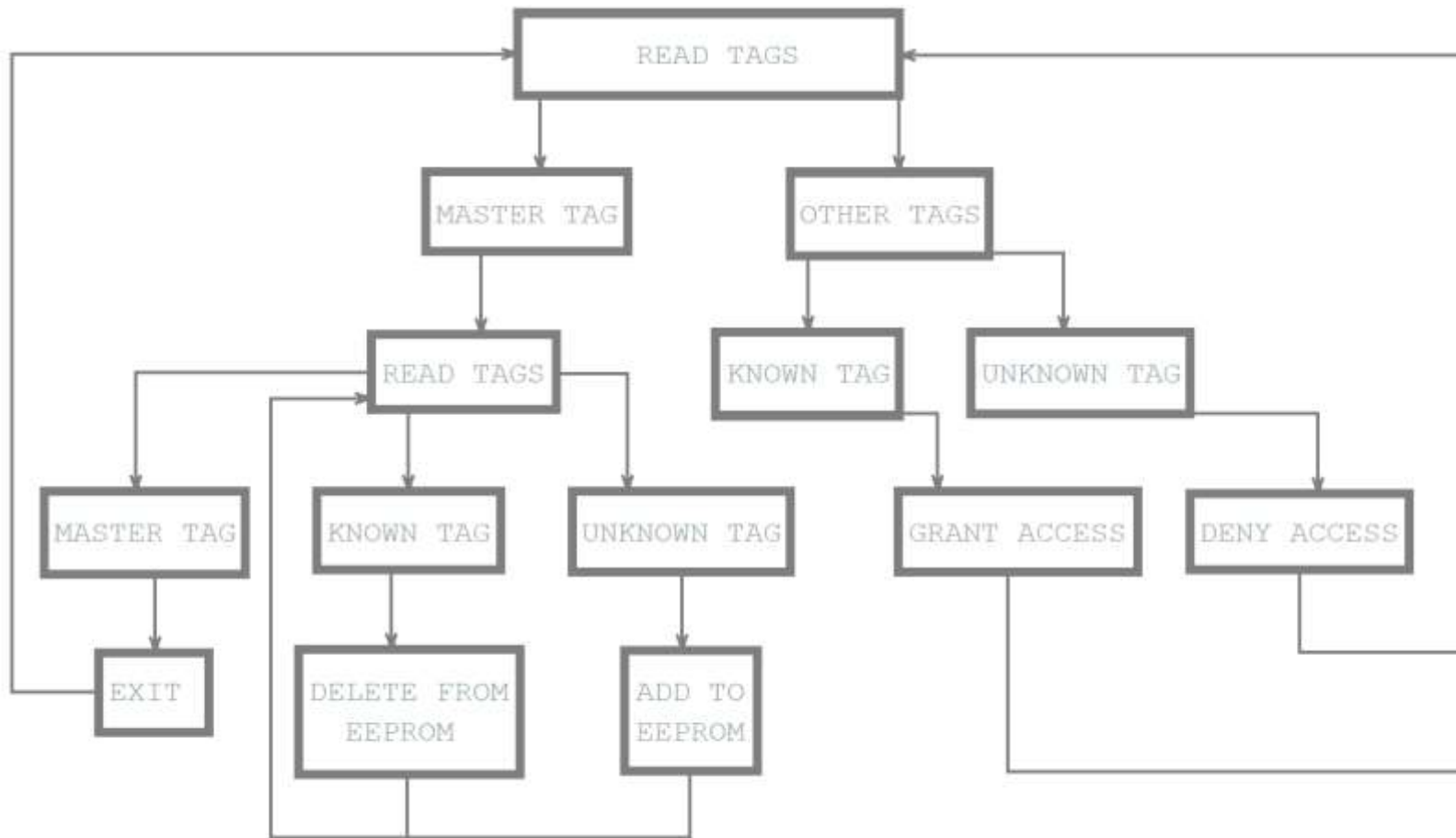
// Reading Master ID from EEPROM
void readMasterID(byte* masterTag) {
    uint8_t start = 2; // Figure out starting position
    for ( uint8_t i = 0; i < 4; i++ ) { // Loop 4 times to get the 4 Bytes
        masterTag[i] = EEPROM.read(start + i); // Assign values read from EEPROM to array
        Serial.print(masterTag[i], HEX); Serial.print(" ");
    }
    Serial.println();
    Serial.println(F("Succesfully read Master ID record from EEPROM"));
}

// Check to see if the ID passed is the master proگرامing card
bool isMaster( byte test[] ) {
    return checkTwo(test, masterCard);
}
```

ID BROJEVI IDENTIFIKATORA

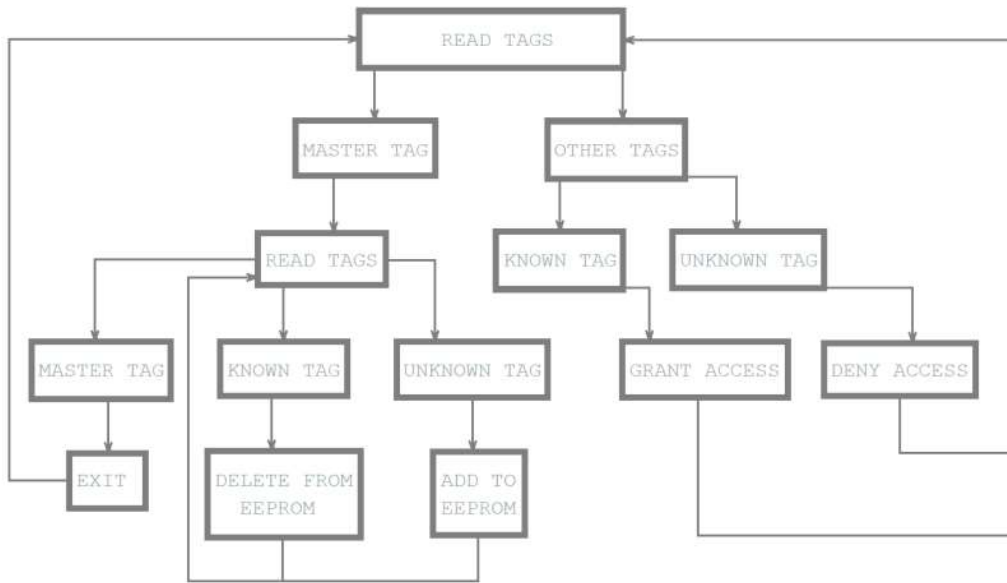


DIJAGAM TOKA PROGRAMA U MIKROKONTOLERU



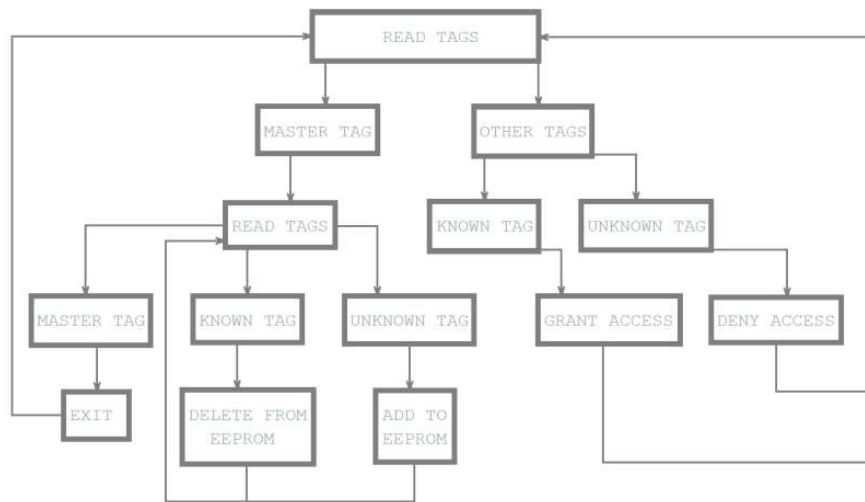
ČITANJE ID BROJA IDENTIFIKATORA

```
//Read card ID
uint8_t ReadCardID(byte* readTag) {
    if ( ! mfrc522.PICC_ReadCardSerial()) { //Since a PICC placed get Serial and continue
        return 0;
    }
    // We support 4 byte card ID
    Serial.println(F("Scanned card ID:"));
    for ( uint8_t i = 0; i < 4; i++) {
        readTag[i] = mfrc522.uid.uidByte[i];
        Serial.print(readTag[i], HEX);
    }
    readTag[4] = '\0';
    Serial.println("");
    return 1;
}
```



PRONALAZENJE ID BROJA IDENTIFIKATORA U MEMORIJI

```
// Find card ID in EEPROM
bool findID() {
    uint8_t k;
    uint8_t count = EEPROM.read(0); // Read the first Byte of EEPROM that
    for ( uint8_t i = 0; i < count; i++ ) { // Loop once for each EEPROM entry
        readID(i); // Read an ID from EEPROM, it is stored in storedID
        for (k = 0; k < 4; k++ ) { // Loop 4 times
            if ( readCard[k] != storedCard[k] ) { // IF a != b then false (one fails, all fail)
                break;
            }
        }
        if (k == 4) return true;
    }
    // If not, return false
    return false;
}
```



UPISIVANJE I ČITANJE ID BROJA IDENTIFIKATORA

```
// Add tag(card) ID to EEPROM
byte writeID() {
    uint8_t num = EEPROM.read(0); // Get the numer of used spaces,
    uint8_t start = ( num * 4 ) + 6; // Figure out where the next slot starts
    num++; // Increment the counter by one
    EEPROM.write( 0, num ); // Write the new count to the counter
    for ( uint8_t j = 0; j < 4; j++ ) { // Loop 4 times
        EEPROM.write( start + j, readCard[j] ); // Write the array values to EEPROM
    }
    EEPROM.commit();
    Serial.println(F("Succesfully added ID record to EEPROM"));
    return (1);
}

// Read an ID from EEPROM
void readID( uint8_t number) {
    uint8_t start = (number * 4 ) + 6; // Figure out starting position
    for ( uint8_t i = 0; i < 4; i++ ) { // Loop 4 times to get the 4 Bytes
        storedCard[i] = EEPROM.read(start + i); // Assign values read from EEPROM to array
        delay(1);
    }
}
```

BRISANJE ID BROJA IDENTIFIKATORA

```
// Remove ID from EEPROM
```

```
byte deleteID() {
    uint8_t num = EEPROM.read(0);    // Get the number of used spaces
    uint8_t slot;                    // Figure out the slot number of the card
    uint8_t start;                   // = ( num * 4 ) + 6; // Figure out where the next slot start
    uint8_t looping;                 // The number of times the loop repeats
    uint8_t j;
```

```
slot = findIDSLLOT(); // Figure out the slot number of the card to delete
```

```
start = (slot * 4) + 6;
```

```
looping = ((num - slot) * 4);
```

```
num--;    // Decrement the counter by one
```

```
EEPROM.write( 0, num );    // Write the new count to the counter
```

```
for ( j = 0; j < looping; j++ ) {           // Loop the card shift times
```

```
EEPROM.write( start + j, EEPROM.read(start + 4 + j)); // Shift the array values to 4 places
```

```
for ( uint8_t k = 0; k < 4; k++ ) { // Shifting loop. Write zero to the last four character in EEPROM
    EEPROM.write( start + j + k, 0);
```

```
EEPROM.commit();
```

```
Serial.println(F("Successfully removed ID record from EEPROM"));
```

```
return (1);
```

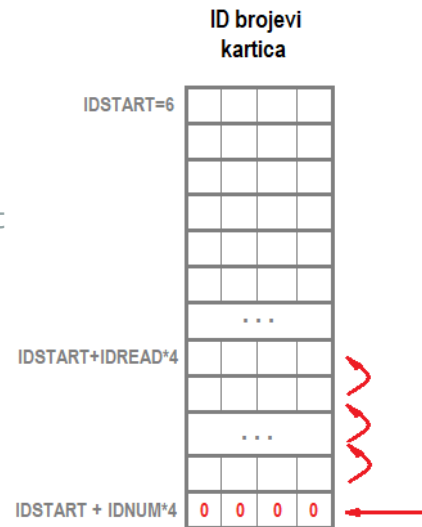
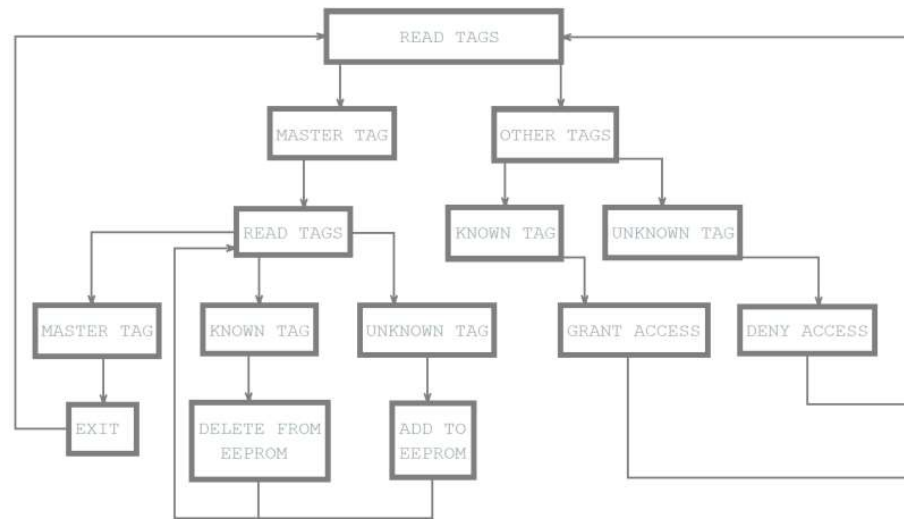


Diagram illustrating the mapping of a row in a 4x4 grid to a specific ID value:

- The grid is labeled "ID brojevi kartica".
- The first row is associated with $IDSTART=6$.
- The fourth row is associated with $IDSTART + IDNUM*4$, indicated by a red arrow pointing to the first cell of that row.
- Ellipses (...) indicate that the grid contains more rows and columns than shown.



BRISANJE SVIH ID BROJEVA IDENTIFIKATORA

```
//Delete all tags ID from EEPROM
void deleteEEPROM() {
  neopixelWrite(RGB_BUILTIN,RGB_BRIGHTNESS,0,0); // Red light
  Serial.println(F("Wipe Button Pressed"));
  Serial.println(F("You have 10 seconds to Cancel"));
  Serial.println(F("This will be remove all records and cannot be undone"));
  bool buttonState = monitorWipeButton(10000); // Give user enough time to cancel operation
  if (buttonState == true && digitalRead(wipeB) == LOW) { // If button still be pressed, wipe EEPROM
    Serial.println(F("Starting Wiping EEPROM"));
    for (uint16_t x = 0; x < EEPROM.length(); x = x + 1) { //Loop end of EEPROM address
      if (EEPROM.read(x) != 0) EEPROM.write(x, 0); //If EEPROM value on address != 0 clear it, it takes 3.3mS
    }
    EEPROM.commit();
    Serial.println(F("EEPROM Successfully Wiped"));
    // visualize a successful wipe
    neopixelWrite(RGB_BUILTIN,RGB_BRIGHTNESS,0,0); delay(200); // Red light
    neopixelWrite(RGB_BUILTIN,RGB_BRIGHTNESS,RGB_BRIGHTNESS,RGB_BRIGHTNESS); delay(200); // White light
    neopixelWrite(RGB_BUILTIN,RGB_BRIGHTNESS,0,0); delay(200); // Red light
    neopixelWrite(RGB_BUILTIN,RGB_BRIGHTNESS,RGB_BRIGHTNESS,RGB_BRIGHTNESS); delay(200); // Red light
    neopixelWrite(RGB_BUILTIN,RGB_BRIGHTNESS,0,0); // Red light
  }
  else {
    Serial.println(F("Wiping Cancelled")); // Show some feedback that the wipe button did not pressed for 15 seconds
    neopixelWrite(RGB_BUILTIN,RGB_BRIGHTNESS,0,0); // Red light
  }
}
```

MONITORING TASTERA

```
//Monitoring of the Wipe button
bool monitorWipeButton(uint32_t interval) {
    uint32_t now = (uint32_t)millis();
    while ((uint32_t)millis() - now < interval) {
        // check on every half a second
        if (((uint32_t)millis() % 500) == 0) {
            if (digitalRead(wipeB) != LOW)
                return false;
        }
    }
    return true;
}
```



ZA VJEŽBU

1. Modifikovati kod tako da se u MASTER mod ulazi svaki put nakon brisanja EEPROM-a (1)
2. Modifikovati kod tako da se u MASTER mod ulazi ukoliko ID kod MASTER identifikatora nije upisan u EEPROM mikrokontrolera, u za njega predviđenom prostoru (očitanje sve 0 ili sve FF) (1)
3. Modifikovati kod tako da se izmjena ID koda MASTER identifikatora inicira zadržavanjem na čitaču važećeg MASTER identifikatora, duže od 4 sekunde (3-2-1).
4. Modifikovati deleteEEPROM funkciju tako da se EEPROM briše ukoliko se MASTER identifikator zadrži na čitaču duže od 10 sekundi (2-1).