

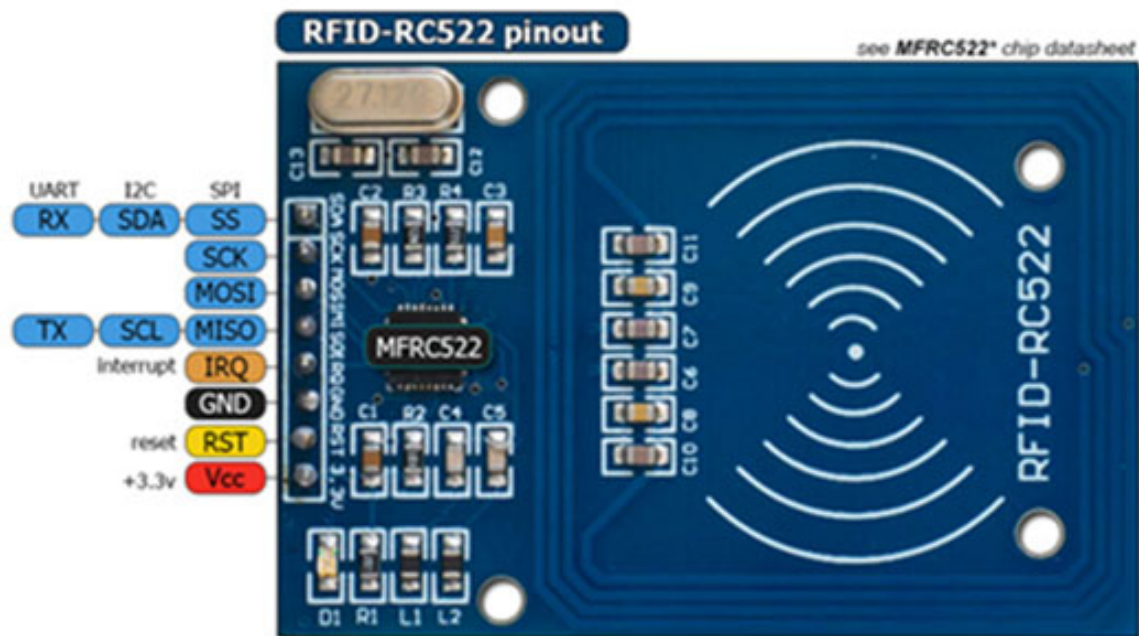


# RFID

# IDs in EEPROM

# MFRC522 RFID

MFRC-522 RC522 13.56Mhz SPI RFID Writer Reader Wireless modul

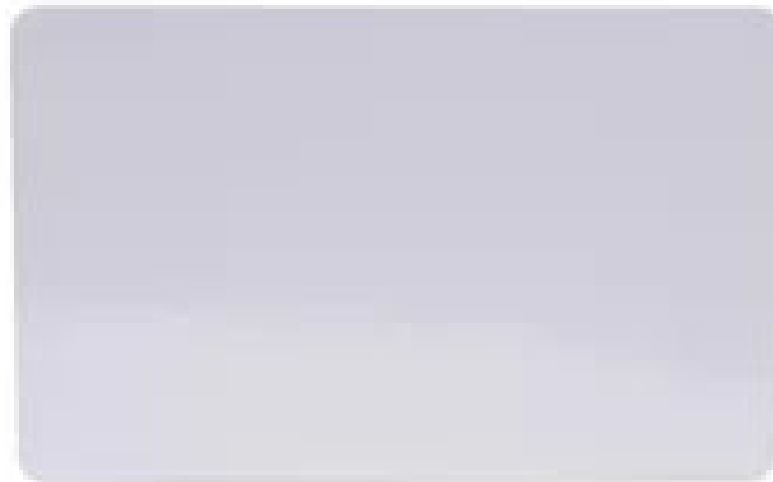


MFRC522 Chip IC radna frekvencija: 13.56MHz, Brzina razmjene podataka: Max. 10Mbit/s

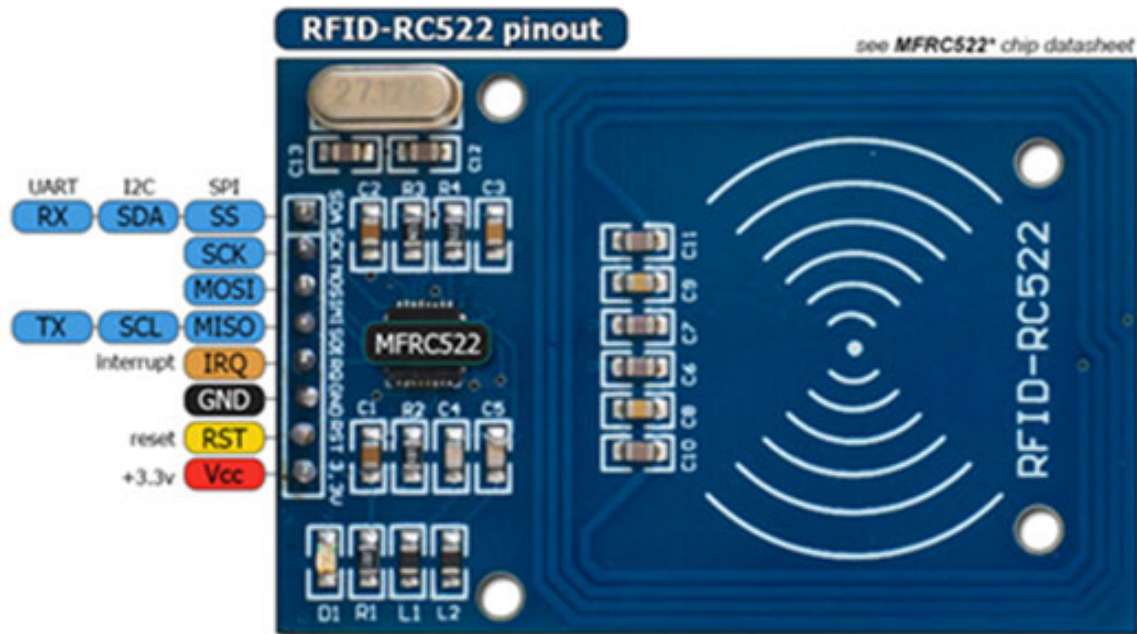
Podržava mifare1 S50 identifikatore

Dimenzije: 40mm × 60mm

# RFID IDENTIFIKATORI

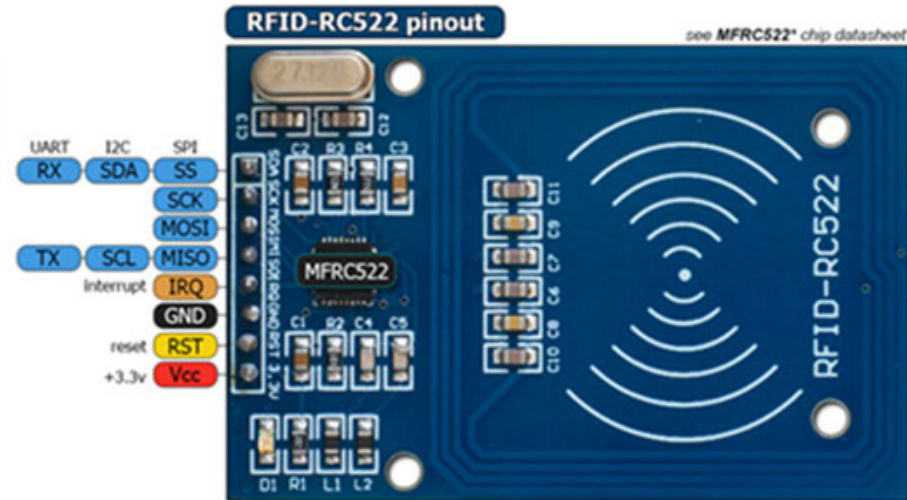
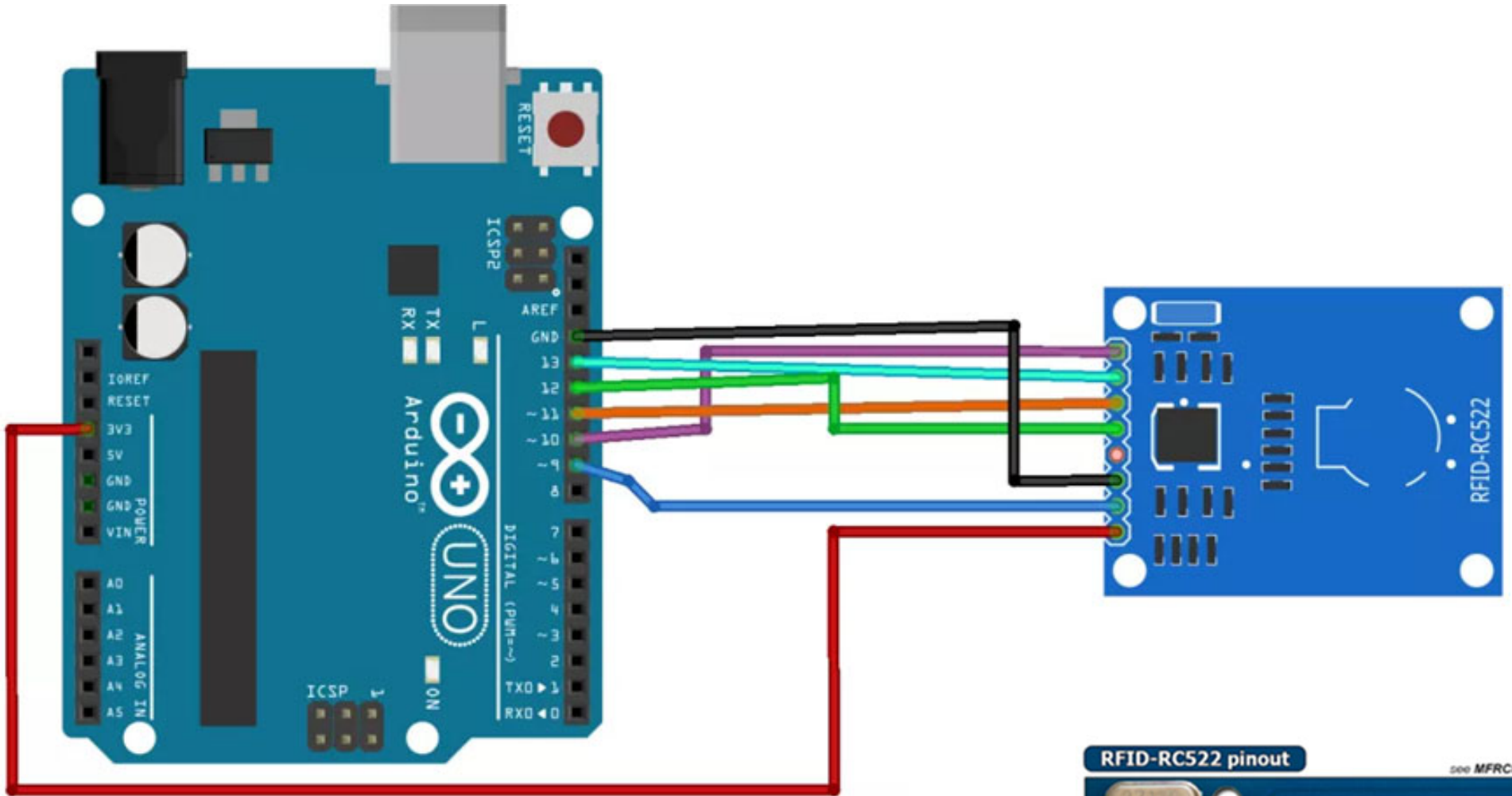


# POVEZIVANJE SA ARDUINO UNO



Pin čitača	Pin Arduino Uno
SDA	10
SCK	13
MOSI	11
MISO	12
IRQ	nepovezano
GND	GND
RST	9
3.3V	3.3V

# POVEZIVANJE SA ARDUINO UNO



# INSTALIRANJE BIBLIOTEKE

**Za rad sa MFRC522 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.

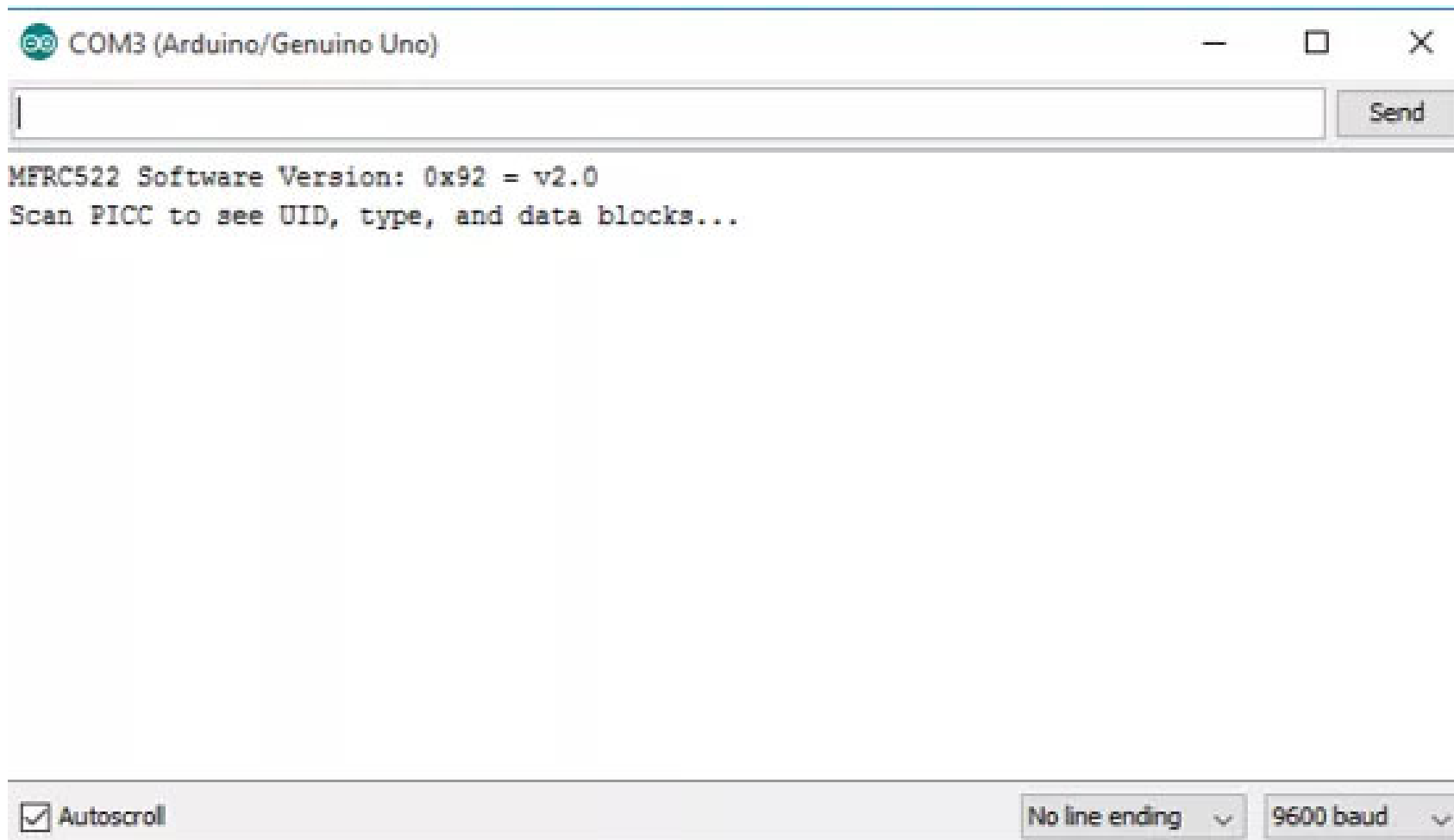
Otvoriti arduino IDE

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

# DUMPINFO



# DUMPINFO

COM3 (Arduino/Genuino Uno)

MFR522 Software Version: 0x92 = v2.0

Scan PICC to see UID, type, and data blocks...

Card UID: BD 31 15 2B

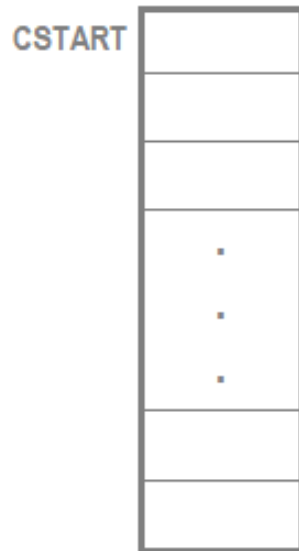
PICC type: MIFARE 1KB

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 ]
13	55	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[ 0 0 1 ]
	54	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
	53	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
	52	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
12	51	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[ 0 0 1 ]
	50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
	49	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
	48	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
11	47	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[ 0 0 1 ]
	46	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
	45	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
	44	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
10	43	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[ 0 0 1 ]
	42	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
	41	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
	40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]
9	39	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[ 0 0 1 ]
	38	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]

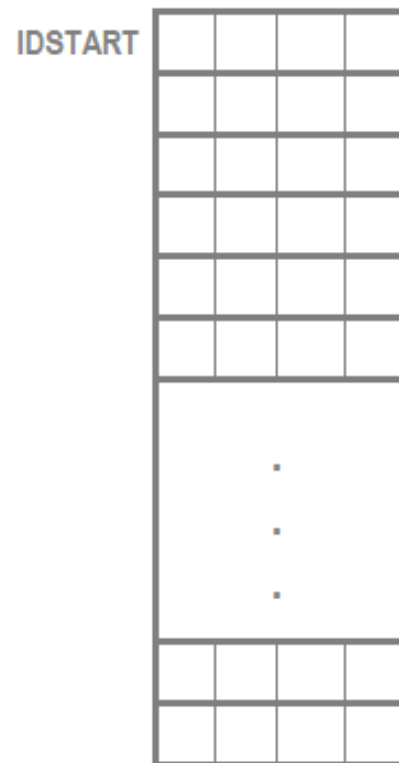


# EEPROM

## Konfiguracioni podaci



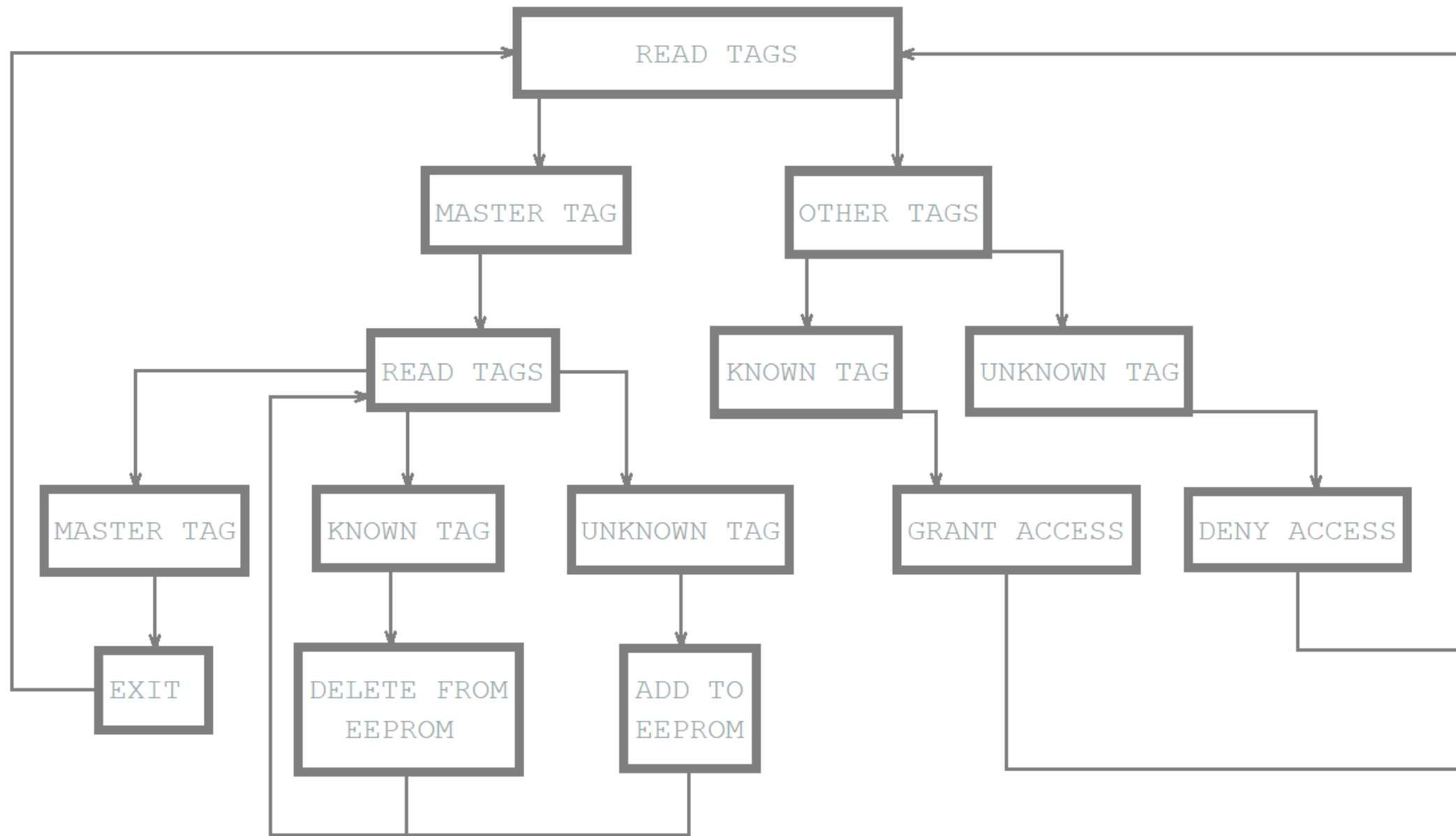
## ID brojevi kartica



## Događaji

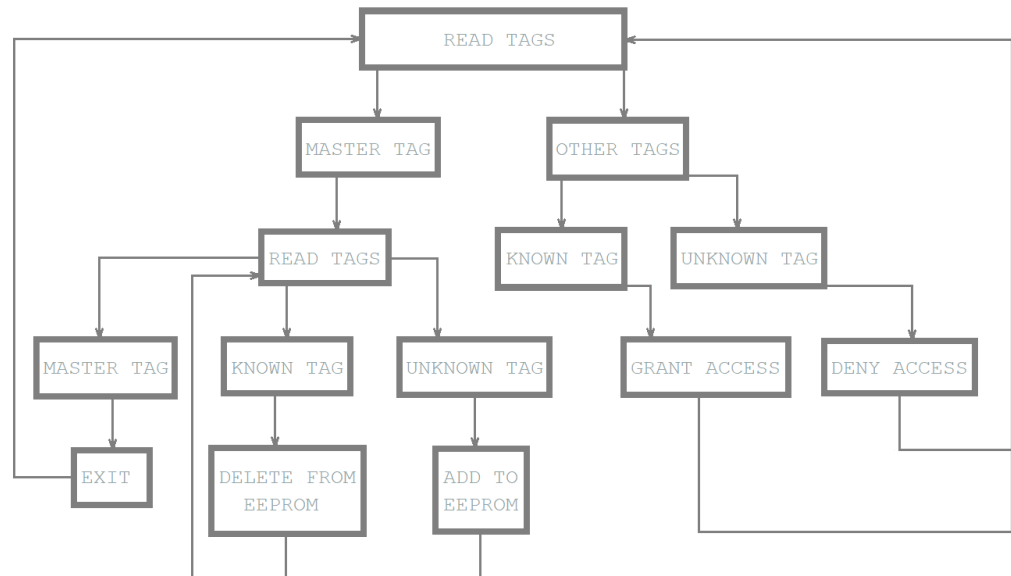


# DIJAGRAM TOKA



# READ TAGS

```
//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("");
  mfrc522.PICC_HaltA(); // Stop reading
  return 1;
}
```



```
// Find card ID in EEPROM
bool findID(byte* rCard) {
  uint8_t k;
  byte strCard[8]; // Stores an ID read from EEPROM
  uint8_t count = EEPROM.read(0); // Read the first byte of EEPROM

  for ( uint8_t i = 0; i < count; i++ ) { // Loop once for each EEPROM entry
    readID(i, strCard); // Read an ID from EEPROM, it is stored in storedID
    for ( k = 0; k < 4; k++ ) { // Loop 4 times
      if ( rCard[k] != strCard[k] ) { // IF a != b then false, because: one fails, all fail
        break;
      }
    }
    if ( k == 4 ) return true;
  }
  // If not, return false
  return false;
}
```

# KONFIGURACIONI PODACI

## Configuration data

CSTART = 0

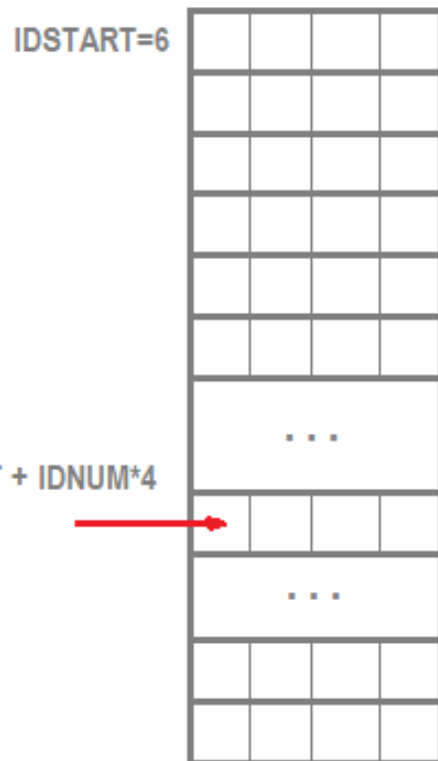
IDNUM
IDSTART
IDMST0
IDMST1
IDMST2
IDMST3

```
// Writing New Master ID to EEPROM
byte writeNewMasterID(char* 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 values to EEPROM in the right position
    }
    Serial.println(F("Succesfully wrote 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);
        delay(1);
    }
    Serial.println();
    Serial.println(F("Succesfully read Master ID record from EEPROM"));
}
```

# READ-WRITE ID

## ID brojevi kartica



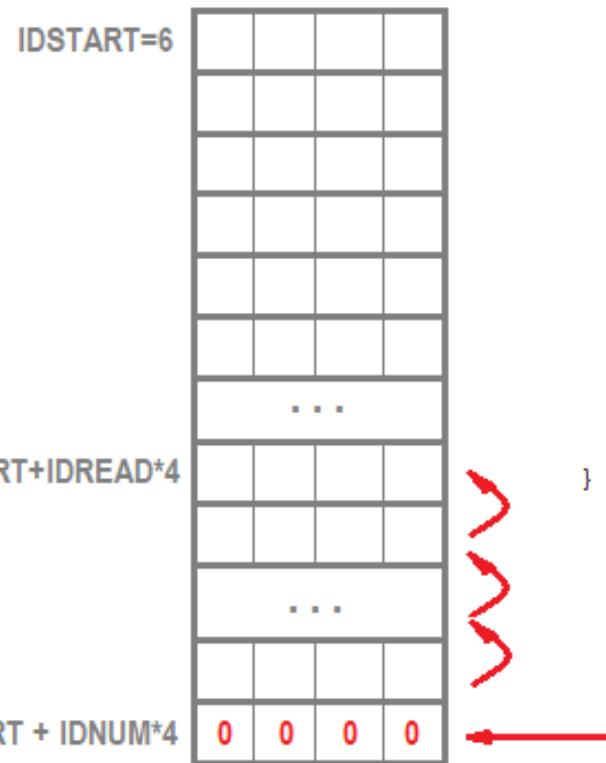
```
// Read an ID from EEPROM
void readID( uint8_t number, byte sCard[] ) {
    uint8_t start = ( number * 4 ) + 6; // Figure out starting position
    for ( uint8_t i = 0; i < 4; i++ ) { // Loop 4 times to get the 4 Bytes
        sCard[i] = EEPROM.read(start + i); // Assign values read from EEPROM to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

```
// Add tag(card) ID to EEPROM
byte writeID(byte* rCard) {
    uint8_t num = EEPROM.read(0); // Get the numer of used spaces, position
                                // 0 stores the number of ID cards

    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, rCard[j] ); // Write the array values
                                           // to EEPROM in the right position
    }
    Serial.println(F("Succesfully added ID record to EEPROM"));
    return (1);
}
```

# DELETE ID

ID brojevi  
kartica



```
// Find Slot
uint8_t findIDSLOT(byte* rCard) {
  uint8_t k;
  byte strCard[8]; // Stores an ID read from EEPROM
  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,strCard); // Read an ID from EEPROM, it is stored in storedCard[]
    for (k = 0; k < 4; k++ ) { // Loop 4 times
      if ( rCard[k] != strCard[k] ) { // IF a != b then false, because: one fails, all fail
        break;
      }
    }
    if (k == 4) return (i);
  }
  return (0);
}
```

```
// Remove ID from EEPROM
byte deleteID(byte* rCard) {
  ...
}
```

# DELETE ALL TAGS ID FROM EEPROM

```
//Delete all tags ID from EEPROM
void deleteEEPROM() {
    digitalWrite(RedLED, HIGH); // Red Led stays on to inform user we are going to wipe
    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) { //If EEPROM address 0
                // do nothing, already clear, go to the next address in order to save time and reduce writes to EEPROM
            }
            else {
                EEPROM.write(x, 0); // if not write 0 to clear, it takes 3.3mS
            }
        }
        Serial.println(F("EEPROM Successfully Wiped"));
        digitalWrite(RedLED, LOW); // visualize a successful wipe
        delay(200);
        digitalWrite(RedLED, HIGH);
        delay(200);
        digitalWrite(RedLED, LOW);
        delay(200);
        digitalWrite(RedLED, HIGH);
        delay(200);
        digitalWrite(RedLED, LOW);
    }
    else {
        // Show some feedback that the wipe button did not pressed for 10 seconds
        Serial.println(F("Wiping Cancelled"));
        digitalWrite(RedLED, LOW);
    }
}
```

# DELETE ALL TAGS ID FROM EEPROM

```
//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;
}
```



1. Modifikovati kod tako da se izmjena ID koda MASTER identifikatora inicira ukoliko se taster pritisne i otpusti 2 puta unutar 5 sekundi (2-1- SS poena).
2. Modifikovati deleteEEPROM funkciju tako da se EEPROM briše ukoliko se se taster pritisne i otpusti 3 puta unutar 5 sekundi (2-1- SS poena).

Ovoga puta poeni riješenih zadataka se sabiraju!