Part 11
-
RFID with RF522

Version: 2021-01-10

Basic Information

Through radio frequency identification or RFID (Radio Frequency Identification), it is possible to identify a person or object.
The system works with a card, key chain or a tag that has a chip inside it. Through the identification of the data on the chip, it is possible to do access control, a method widely used in records of employee points, public transportation, libraries, among others.
Our goal, therefore, is to create a program in which we can either read an RFID card or tag, or write the data to it. We use a ESP32 and an RFID-RC522 module.
It is important to note that we can store and retrieve data on these chips or cards remotely through devices, as these can have up to 1k of memory.

An RFID system basically consists of a transceiver with a decoder and an antenna, as well as a transponder, the card or tag. The transceiver constantly emits a RF signal. The card or tag has also an antenna inside. When the card approachs the trnasceiver, the emitted RF energy is captured by the card and converted to power (voltage and current). The energized card ot tag modulates the information stored in its memory and sends that data to the reader. The RFID reader receives the information sent by the tag, decodes and sends the data to the server application.

As mentioned, we have 1k of memory inside this type of chip. And, the EEPROM memory is organized as follows: there are 16 sectors of 4 blocks. Each block contains 16 bytes. Remember that within the source code, you only reference the block number.

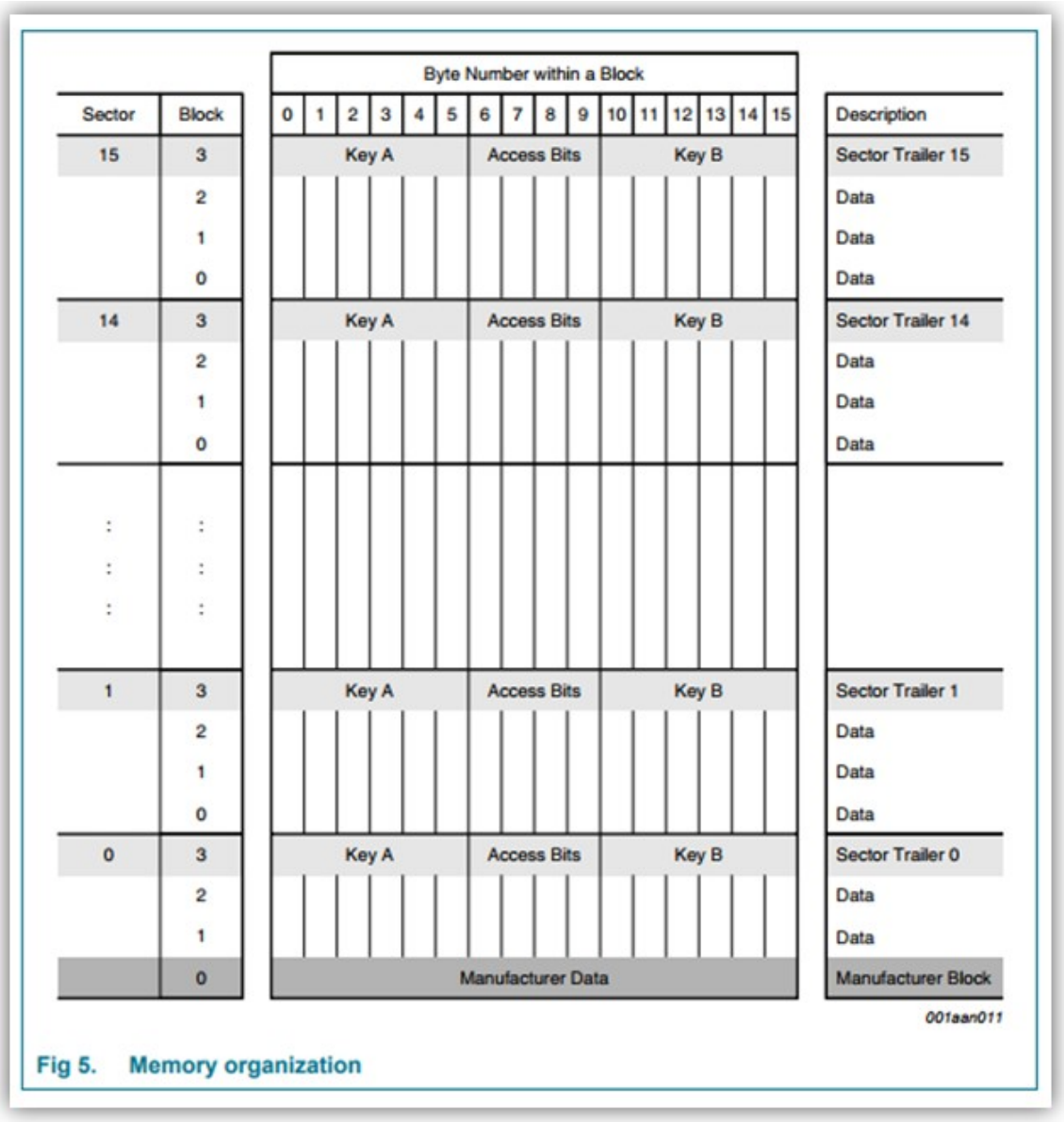
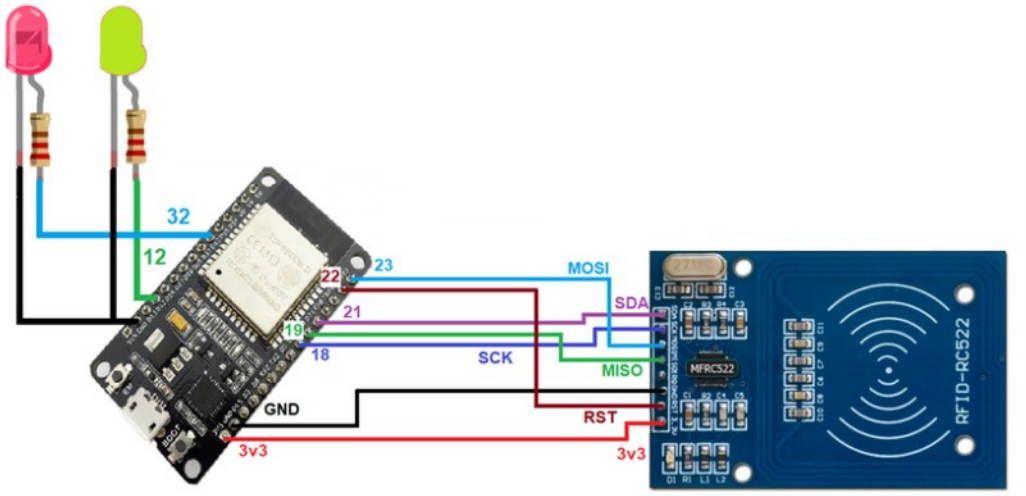


Fig 5. Memory organization

The RFID-RF522 module

The module can handle several interface protocols: serial (UART), I²C and SPI. The pins used are shown in the picture.
We will use the SPI interface.

In our setup we have the ESP32 powered by USB and hence connected via serial comport to the Arduino IDE, two leds to indicate whether the reading or writing was successful or not, and the RFID reader, the RC522. We have a card or tag.



The connections are as follows

ESP32 GPIO	RF522	LED green	LED red
21	SDA		
18	SCK		
23	MOSI		
19	MISO		
	IRQ		
GND	GND	Kathode -	Kathode -
22	RST		
3V3	3V3		
32			Anode + via 220 Ohm resistor
12		Anode + via 220 Ohm resistor	

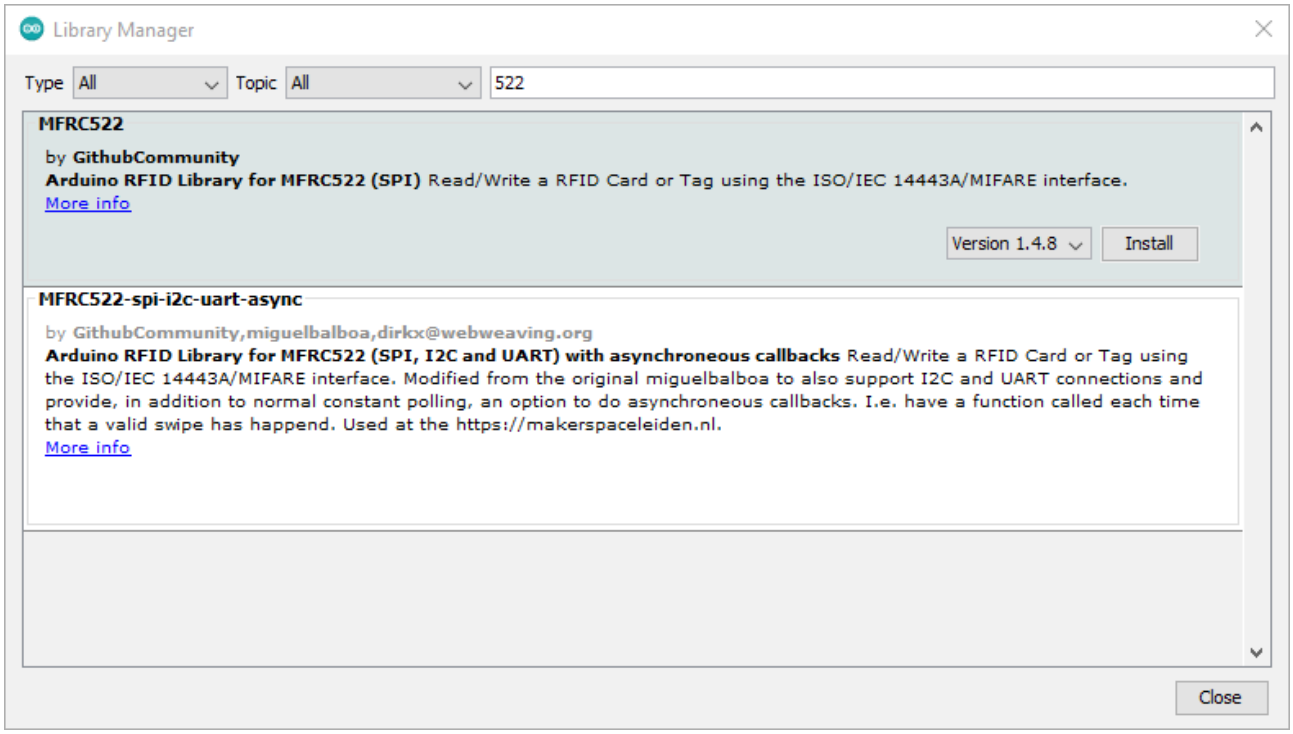
The Code

Placing the card or tag over the transceiver shows the option of 0 for reading data and 1 for recording this data. We do an example that shows that after reading the chip or card, if the green led is lit the reader recognized the number. In case of the red led, it means that some type of error occurred and the authentication was not performed.

Our program will work as follows: after starting, the program will wait for a card or tag to be identified. After that, a menu will appear for the user to choose between making a reading or recording something. Then the operation will be performed.

Add the library needed

Add the library "MFRC522" in the Arduino IDE through Tools → Manage libraries → 522



Write the code

We first include libraries and create objects.

```
//library for communicating of SPI bus
#include <SPI.h>
//library for communicating with the module RFID-RC522
#include <MFRC522.h>

#define SS_PIN          21
#define RST_PIN         22
#define SIZE_BUFFER     18
#define MAX_SIZE_BLOCK  16
#define LEDgreen        12
#define LEDred          32

// time to wait for serial input = 10 secs
#define TIME_OUT 10000L

//used in authentication
MFRC522::MIFARE_Key key;
//authentication return status
MFRC522::StatusCode status;

// Defined pins to module RC522
MFRC522 mfrc522(SS_PIN, RST_PIN);
```

In the `setup` we initialize the pins, serial and SPI communication, LEDs and antenna service.

```
void setup()
{
  Serial.begin(115200);

  // Init SPI bus
  SPI.begin();

  // set GPIO pins as output for LED
  pinMode(LEDgreen, OUTPUT);
  pinMode(LEDred  , OUTPUT);

  // Init MFRC522
  mfrc522.PCD_Init();

  delay(2000);
  // display info
  Serial.println("Approach your card or tag...");
  Serial.println();
}
```

In the `loop` we wait for the card to approach and select the same one.
In the menu, we offer the options of reading or writing data. We instruct this part when the device should leave the ACTIVE state for the STOP state. We have to use such a method to enable new readings.

```
void loop()
{
  //waiting the card approach
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
```

```
  // Select a card
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }

  // Dump debug info about the card
  // mfrc522.PICC_DumpToSerial(&(mfrc522.uid));

  //call menu function and retrieve the desired option
  int op = menu();

  if(op == 0)
  {
    readingData();
  }
  else
  {
    if(op == 1)
    {
      writingData();
    }
    else
    {
      if(op < 9)
      {
        Serial.println("Incorrect Option!");
      }
      else
      {
        Serial.println("Time out!");
      }
    }
  }
  Serial.println();
  Serial.println();
  Serial.println("Remove your card or tag...");
  Serial.println();
  delay(3000);

  // display info
  Serial.println();
  Serial.println();
  Serial.println("Approach your card or tag...");
  Serial.println();

  //instructs the PICC when in the ACTIVE state to go to a "STOP" state
  //mfrc522.PICC_HaltA();

  // "stop" the encryption of the PCD
  // it must be called after communication with authentication
  // otherwise new communications can not be initiated
  mfrc522.PCD_StopCrypto1();
}
```

In the `readingData` function, we have to prepare all the keys, handle the size of the buffer and authenticate the block that we are going to operate. Finally, we set the printing of the data read.

```
  //reads data from card/tag
  void readingData()
  {
    //prints the technical details of the card/tag
    mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));

    //prepare the key - all keys are set to FFFFFFFFFFFFh
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

    //buffer for read data
    byte buffer[SIZE_BUFFER] = {0};

    //the block to operate
    byte block = 1;
    byte size = SIZE_BUFFER;
    //authenticates the block to operate
    status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block,
  &key, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK)
    {
      Serial.print("Authentication failed: ");
      Serial.println(mfrc522.GetStatusCodeName(status));
      digitalWrite(LEDred, HIGH);
      delay(1000);
      digitalWrite(LEDred, LOW);
      return;
    }
    else
    {
      Serial.print("Authentication successful: ");
      Serial.println(mfrc522.GetStatusCodeName(status));
      digitalWrite(LEDgreen, HIGH);
      delay(1000);
      digitalWrite(LEDgreen, LOW);
    }

    //read data from block
    status = mfrc522.MIFARE_Read(block, buffer, &size);
    if (status != MFRC522::STATUS_OK)
    {
      Serial.print("Reading failed: ");
      Serial.println(mfrc522.GetStatusCodeName(status));
      digitalWrite(LEDred, HIGH);
      delay(1000);
      digitalWrite(LEDred, LOW);
      return;
    }
    else
    {
      Serial.print("Reading successful: ");
      Serial.println(mfrc522.GetStatusCodeName(status));
      digitalWrite(LEDgreen, HIGH);
      delay(1000);
      digitalWrite(LEDgreen, LOW);
    }

    Serial.print("\nData from block [");
    Serial.print(block);Serial.print("]: ");
```

```
  //prints read data
  for (uint8_t i = 0; i < MAX_SIZE_BLOCK; i++)
  {
      Serial.write(buffer[i]);
  }
  Serial.println(" ");
}
```

To write data to the card / tag we have to follow some steps. From the moment we choose the recording option, we have 30 seconds to make the data entry via serial. Enter the data to be written with the "#" character and prepare the key. You will need to clear the buffer and write to block 1, since in block 0 we have saved the card number, which is already in the factory. So we do not touch block 0.
We deal with the size of data and insert a command for authentication and enable secure communication. We also put error messages equal to the part of the reading for display in case of unauthenticated data. We recorded the data in the block due.

```
// write data to the card
void writingData()
{

  //prints thecnical details from of the card/tag
  mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));

  // waits 10 seconds for data entry via Serial
  Serial.setTimeout(TIME_OUT);
  Serial.println("Enter the data to be written");
  Serial.println("  with the '#' character at the end");
  Serial.println("  [maximum of 16 characters]:");

  //prepare the key - all keys are set to FFFFFFFFFFFFh
  for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

  //buffer para armazenamento dos dados que iremos gravar
  //buffer for storing data to write
  byte buffer[MAX_SIZE_BLOCK] = "";
  //the block to operate
  byte block;
  //size of data (bytes)
  byte dataSize;

  //recover on buffer the data from Serial
  //all characters before chacactere '#'
  dataSize = Serial.readBytesUntil('#', (char*)buffer, MAX_SIZE_BLOCK);

  //void positions that are left in the buffer; filled with whitespace
  for(byte i=dataSize; i < MAX_SIZE_BLOCK; i++)
  {
    buffer[i] = ' ';
  }

  //the block to operate
  block = 1;
  //transforms the buffer data in String
  String str = (char*)buffer;

  Serial.println(str);
  //authenticates the block to operate
  //Authenticate is a command to hability a secure communication
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
                                    block, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK)
  {
```

```
    Serial.print("PCD_Authenticate() failed: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    digitalWrite(LEDred, HIGH);
    delay(1000);
    digitalWrite(LEDred, LOW);
    return;
  }
  else
  {
    Serial.println("PCD_Authenticate() successful: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    digitalWrite(LEDgreen, HIGH);
    delay(1000);
    digitalWrite(LEDgreen, LOW);
  }

  //Writes in the block
  status = mfrc522.MIFARE_Write(block, buffer, MAX_SIZE_BLOCK);
  if (status != MFRC522::STATUS_OK)
  {
    Serial.print("MIFARE_Write() failed: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    digitalWrite(LEDred, HIGH);
    delay(1000);
    digitalWrite(LEDred, LOW);
    return;
  }
  else
  {
    Serial.println("MIFARE_Write() successful: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    digitalWrite(LEDgreen, HIGH);
    delay(1000);
    digitalWrite(LEDgreen, LOW);
  }
}
```

The `menu` function exposes all options and waits for data to be sent. When an option is selected, it removes 48 from the read value, which is 0 from the ASCII table. This table is old and not used on the PC, but on Arduino and microcontrollers you will have to deal with it.

```
//menu to operation choice
int menu()
{
  unsigned long timestart = millis();
  unsigned long now = millis();

  Serial.println("\nChoose an option:");
  Serial.println("  0 - Reading data");
  Serial.println("  1 - Writing data\n");
  //waits while the user does not start data
  while(!Serial.available())
  {
    now = millis();
    if(now - timestart > TIME_OUT)
    {
      return(9);
    }
 };

  //retrieves the chosen option
  int op = (int)Serial.read();

  //remove all characters after option (as \n per example)
  while(Serial.available())
  {
    if(Serial.read() == '\n') break;
    Serial.read();
  }

  //subtract 48 from read value, 48 is the zero from ascii table
  return (op-48);
}
```

The Full code

```
//library for communicating of SPI bus
#include <SPI.h>
//library for communicating with the module RFID-RC522
#include <MFRC522.h>

#define SS_PIN          21
#define RST_PIN         22
#define SIZE_BUFFER     18
#define MAX_SIZE_BLOCK  16
#define LEDgreen        12
#define LEDred          32

// time to wait for serial input = 10 secs
#define TIME_OUT 10000L

//used in authentication
MFRC522::MIFARE_Key key;
//authentication return status
MFRC522::StatusCode status;

// Defined pins to module RC522
MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup()
{
  Serial.begin(115200);

  // Init SPI bus
  SPI.begin();

  // set GPIO pins as output for LED
  pinMode(LEDgreen, OUTPUT);
  pinMode(LEDred  , OUTPUT);

  // Init MFRC522
  mfrc522.PCD_Init();

  delay(2000);
  // display info
  Serial.println("Approach your card or tag...");
  Serial.println();
}

void loop()
{
  //waiting the card approach
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }

  // Select a card
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }

  // Dump debug info about the card
  // mfrc522.PICC_DumpToSerial(&(mfrc522.uid));

  //call menu function and retrieve the desired option
  int op = menu();
```

```
  if(op == 0)
  {
    readingData();
  }
  else
  {
    if(op == 1)
    {
      writingData();
    }
    else
    {
      if(op < 9)
      {
        Serial.println("Incorrect Option!");
      }
      else
      {
        Serial.println("Time out!");
      }
    }
  }
  Serial.println();
  Serial.println();
  Serial.println("Remove your card or tag...");
  Serial.println();
  delay(3000);

  // display info
  Serial.println();
  Serial.println();
  Serial.println("Approach your card or tag...");
  Serial.println();

  //instructs the PICC when in the ACTIVE state to go to a "STOP" state
  //mfrc522.PICC_HaltA();

  // "stop" the encryption of the PCD
  // it must be called after communication with authentication
  // otherwise new communications can not be initiated
  mfrc522.PCD_StopCrypto1();
}

//reads data from card/tag
void readingData()
{
  //prints the technical details of the card/tag
  mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));

  //prepare the key - all keys are set to FFFFFFFFFFFFh
  for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

  //buffer for read data
  byte buffer[SIZE_BUFFER] = {0};

  //the block to operate
  byte block = 1;
  byte size = SIZE_BUFFER;
  //authenticates the block to operate
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block,
&key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK)
  {
```

```
    Serial.print("Authentication failed: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    digitalWrite(LEDred, HIGH);
    delay(1000);
    digitalWrite(LEDred, LOW);
    return;
  }
  else
  {
    Serial.print("Authentication successful: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    digitalWrite(LEDgreen, HIGH);
    delay(1000);
    digitalWrite(LEDgreen, LOW);
  }

  //read data from block
  status = mfrc522.MIFARE_Read(block, buffer, &size);
  if (status != MFRC522::STATUS_OK)
  {
    Serial.print("Reading failed: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    digitalWrite(LEDred, HIGH);
    delay(1000);
    digitalWrite(LEDred, LOW);
    return;
  }
  else
  {
    Serial.print("Reading successful: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    digitalWrite(LEDgreen, HIGH);
    delay(1000);
    digitalWrite(LEDgreen, LOW);
  }

  Serial.print("\nData from block [");
  Serial.print(block);Serial.print("]: ");

  //prints read data
  for (uint8_t i = 0; i < MAX_SIZE_BLOCK; i++)
  {
      Serial.write(buffer[i]);
  }
  Serial.println(" ");
}

// write data to the card
void writingData()
{

  //prints thecnical details from of the card/tag
  mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));

  // waits 10 seconds for data entry via Serial
  Serial.setTimeout(TIME_OUT);
  Serial.println("Enter the data to be written");
  Serial.println("  with the '#' character at the end");
  Serial.println("  [maximum of 16 characters]:");

  //prepare the key - all keys are set to FFFFFFFFFFFFh
  for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

  //buffer para armazenamento dos dados que iremos gravar
```

```
//buffer for storing data to write
byte buffer[MAX_SIZE_BLOCK] = "";
//the block to operate
byte block;
//size of data (bytes)
byte dataSize;

//recover on buffer the data from Serial
//all characters before chacactere '#'
dataSize = Serial.readBytesUntil('#', (char*)buffer, MAX_SIZE_BLOCK);

//void positions that are left in the buffer; filled with whitespace
for(byte i=dataSize; i < MAX_SIZE_BLOCK; i++)
{
  buffer[i] = ' ';
}

//the block to operate
block = 1;
//transforms the buffer data in String
String str = (char*)buffer;

Serial.println(str);
//authenticates the block to operate
//Authenticate is a command to hability a secure communication
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
                                  block, &key, &(mfrc522.uid));
if (status != MFRC522::STATUS_OK)
{
  Serial.print("PCD_Authenticate() failed: ");
  Serial.println(mfrc522.GetStatusCodeName(status));
  digitalWrite(LEDred, HIGH);
  delay(1000);
  digitalWrite(LEDred, LOW);
  return;
}
else
{
  Serial.println("PCD_Authenticate() successful: ");
  Serial.println(mfrc522.GetStatusCodeName(status));
  digitalWrite(LEDgreen, HIGH);
  delay(1000);
  digitalWrite(LEDgreen, LOW);
}

//Writes in the block
status = mfrc522.MIFARE_Write(block, buffer, MAX_SIZE_BLOCK);
if (status != MFRC522::STATUS_OK)
{
  Serial.print("MIFARE_Write() failed: ");
  Serial.println(mfrc522.GetStatusCodeName(status));
  digitalWrite(LEDred, HIGH);
  delay(1000);
  digitalWrite(LEDred, LOW);
  return;
}
else
{
  Serial.println("MIFARE_Write() successful: ");
  Serial.println(mfrc522.GetStatusCodeName(status));
  digitalWrite(LEDgreen, HIGH);
  delay(1000);
  digitalWrite(LEDgreen, LOW);
}
```

```
}

//menu to operation choice
int menu()
{
  unsigned long timestart = millis();
  unsigned long now = millis();

  Serial.println("\nChoose an option:");
  Serial.println("  0 - Reading data");
  Serial.println("  1 - Writing data\n");
  //waits while the user does not start data
  while(!Serial.available())
  {
    now = millis();
    if(now - timestart > TIME_OUT)
    {
      return(9);
    }
 };

  //retrieves the chosen option
  int op = (int)Serial.read();

  //remove all characters after option (as \n per example)
  while(Serial.available())
  {
    if(Serial.read() == '\n') break;
    Serial.read();
  }

  //subtract 48 from read value, 48 is the zero from ascii table
  return (op-48);
}
```