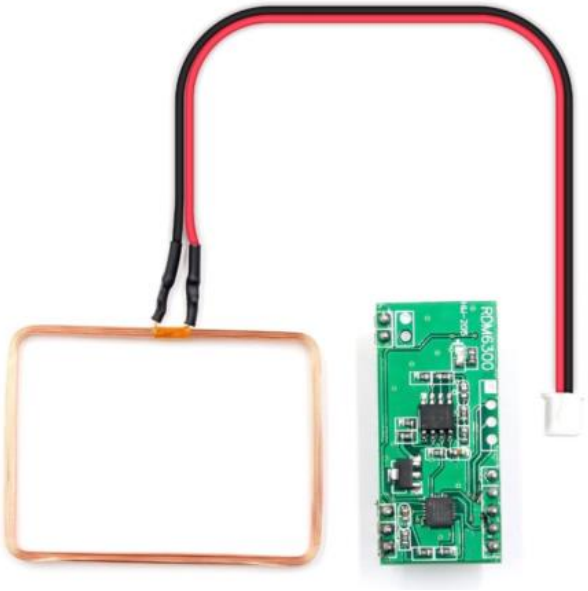
RDM6300 125KHz RFID Card Reader Module

RDM6300 125KHz RFID card reader module is designed for reading code from 125KHz card compatible read-only tags and read/write card . It can be applied in office/home security, personal identification, access control, anti-forgery, interactive toy and production control systems etc. Work at frequency of 125 kHz and enable to read EM4100-compatible tags.

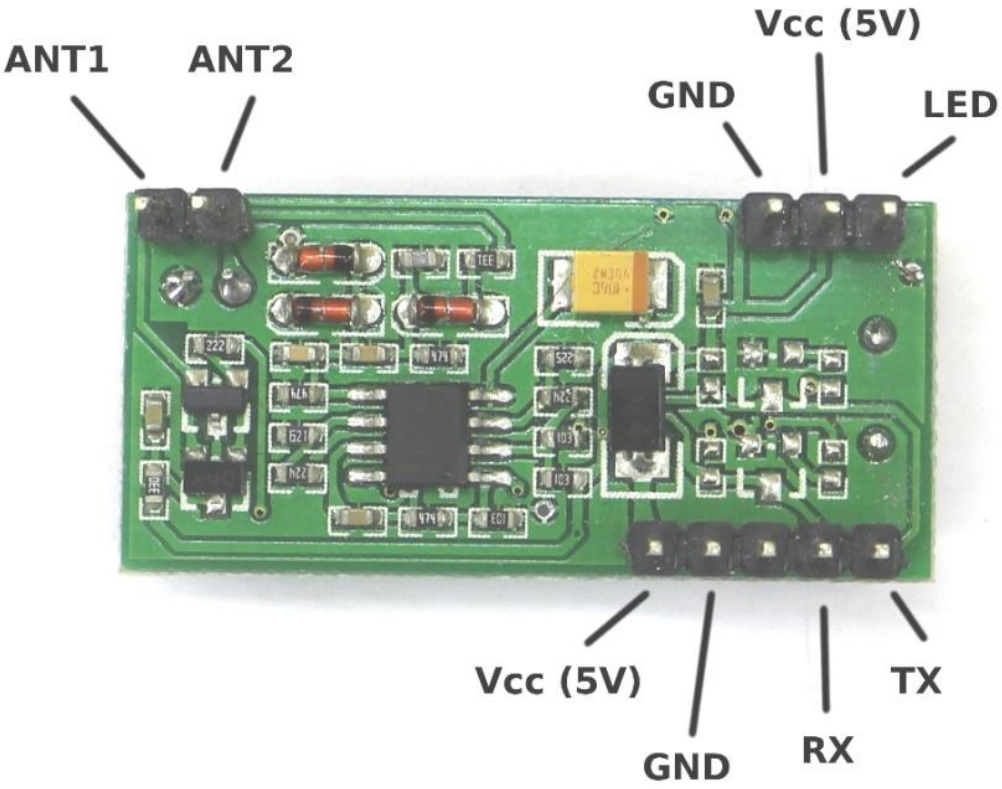


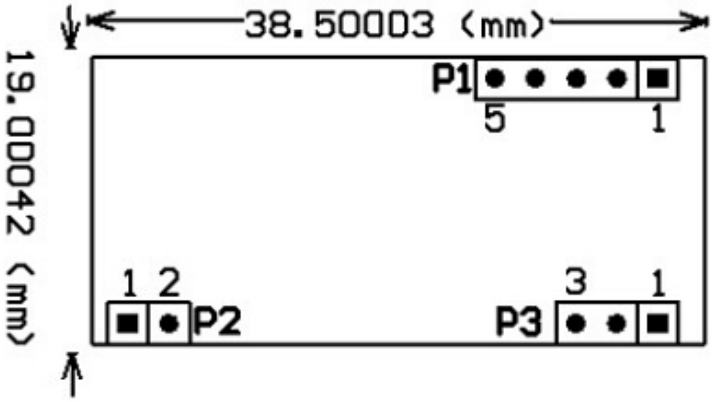
SKU: MDU1090

Brief Data:

- Operating Frequency: 125 kHz.
- Baud Rate: 9600.
- Interface: TTL level RS232 format.
- Working Voltage: 5Vdc(+/-5%).
- Working Current: <50mA.
- Receive Distance: 20~50mm.
- Working Temperature: -10°C~+70°C.
- Humidity: 0~95%.
- Loop Antenna: 46x33x3mm.
- Reader PCB size: 38x18 mm.

Interface Pins Function:

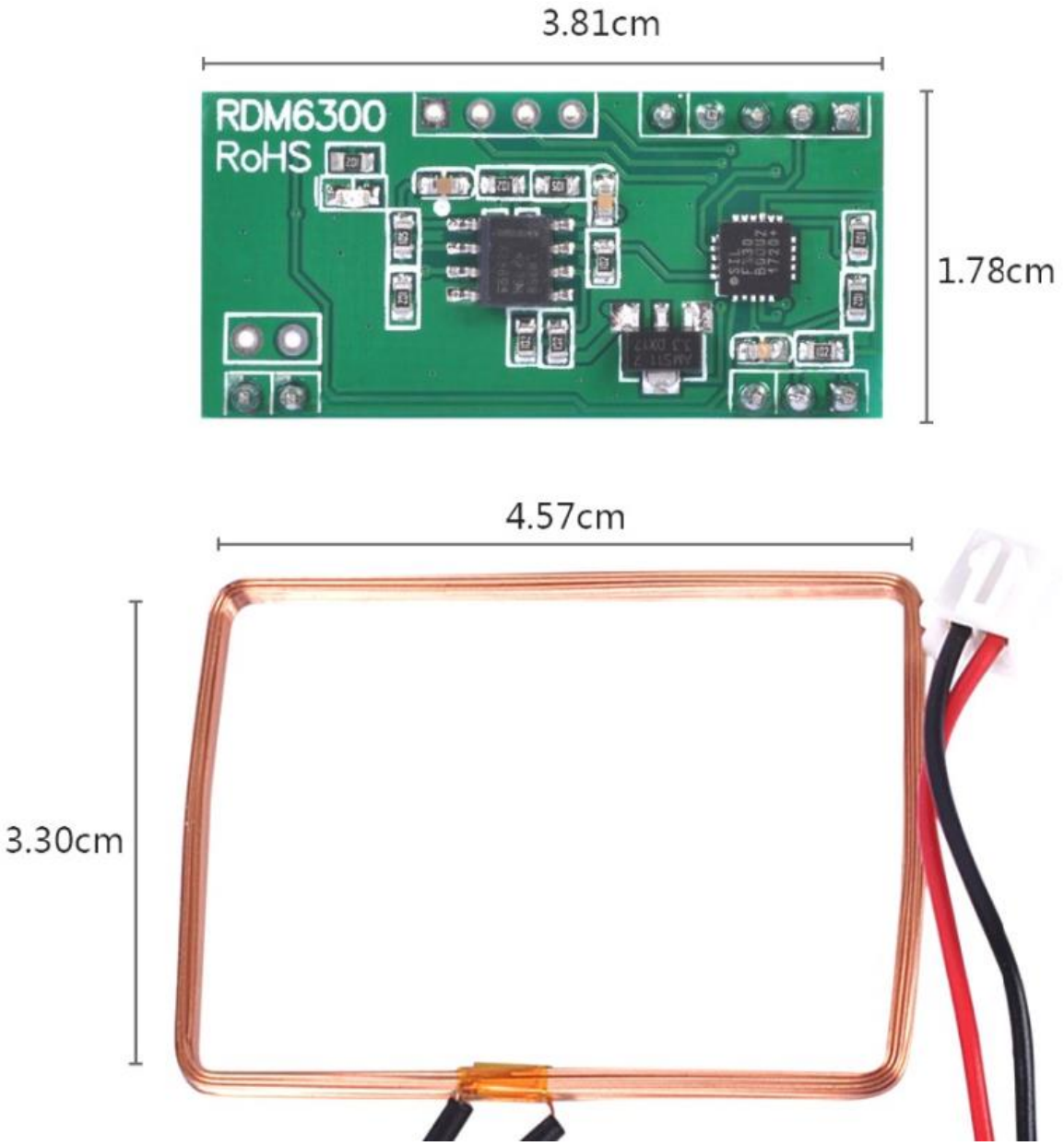




P1	P2	P3
1: TX	1: ANT1	1: LED
2: RX	2: ANT2	2: +5VDC
3: NC		3: GND
4: GND		
5: +5VDC		

Mechanical Dimension:

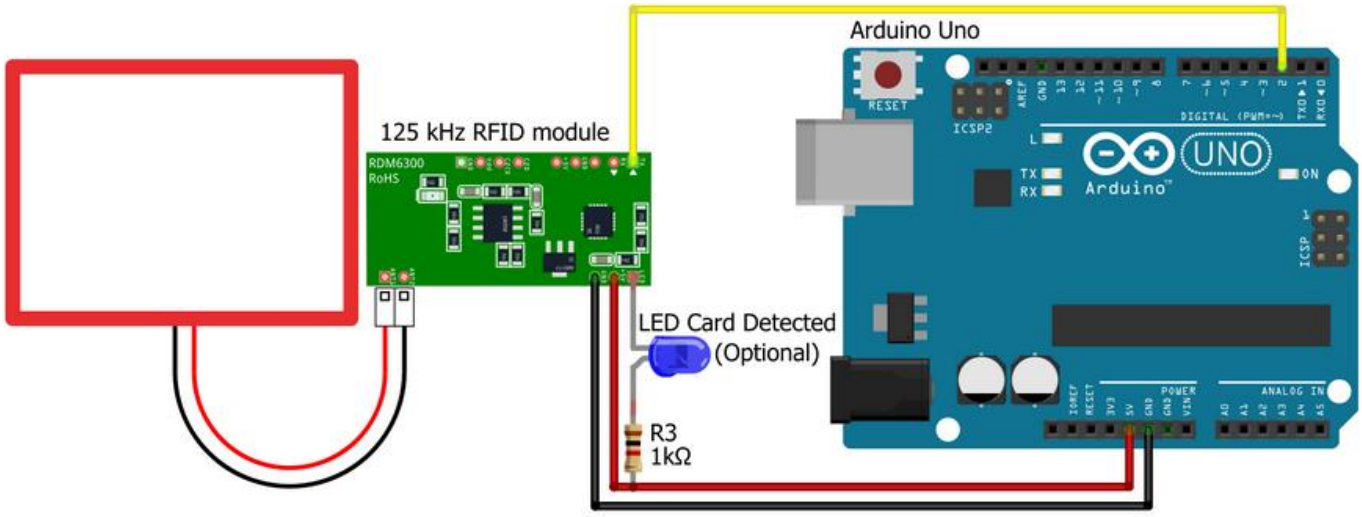
Unit: mm



Application Example with Arduino:

Arduino Circuit Connection:

In this demonstration, four pins of the RDM630/RDM6300 are wired to the Arduino Uno. Vcc has to be connected to the Arduino's 5V pin (red wire) and GND to the Arduino's GND (black wire). The TX pin has to be connected to digital pin #2 (yellow wire). Basically, the RX pin is not required as we do not send data to the RFID module in this tutorial. For the sake of completeness, RX is connected to digital pin #8. Lastly, the antenna is connected to ANT1 and ANT2 (polarity does not matter).



RDM6300	Wiring to Arduino Un
1: TX	Digital 2
2: RX	Digital 13

Reading Data from a RFID Tag:

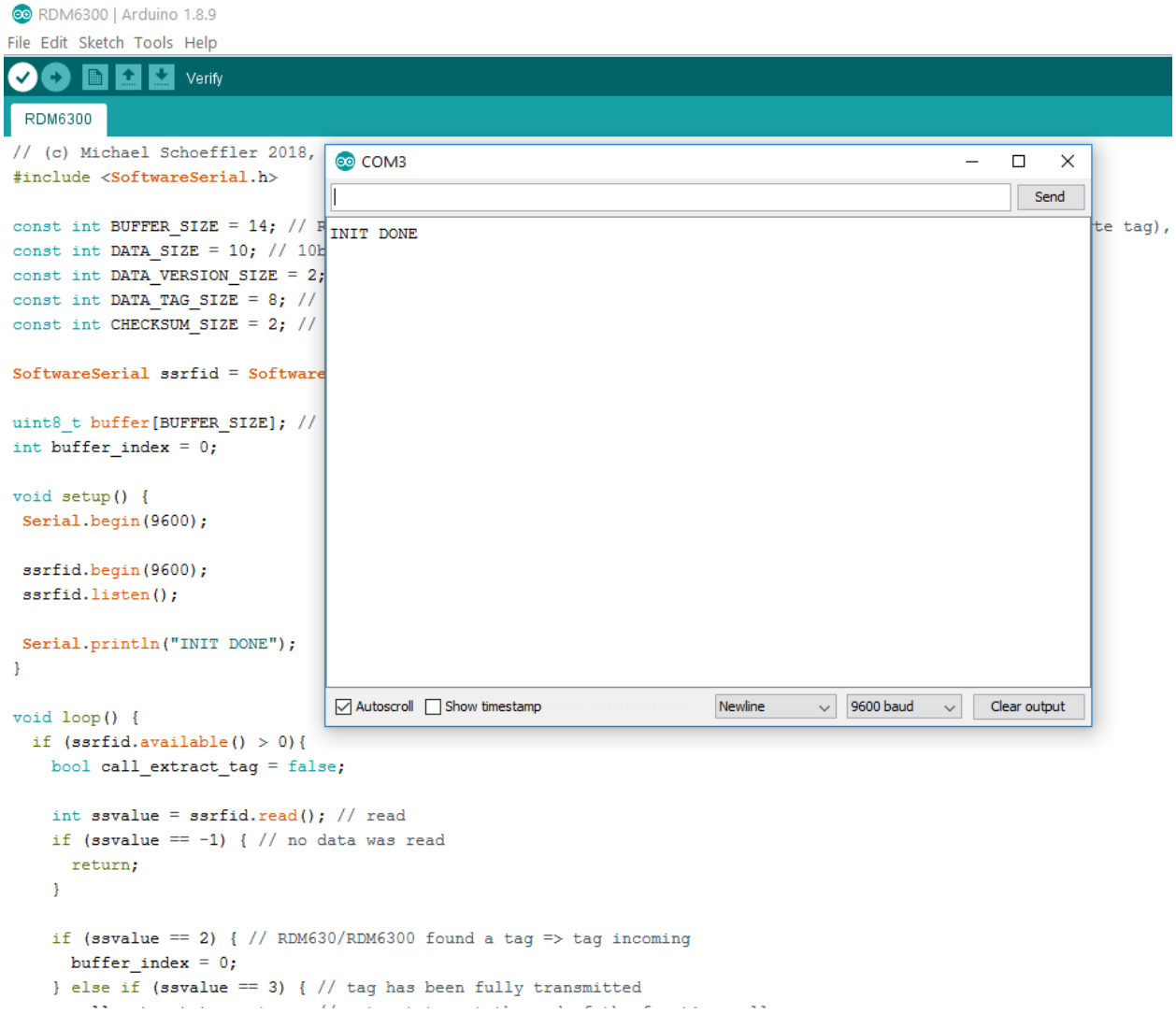
After having the circuit ready, upload the below sketch to Arduino Uno board:

```
/*================================================================================
//  Author      : Handson Technology
//  Project     : Arduino Uno
//  Description : RDM6300 125KHz RFID Card Reader Module
//  Source-Code : RDM6300.ino
//================================================================================
*/

#include <SoftwareSerial.h>

const int BUFFER_SIZE = 14; // RFID DATA FRAME FORMAT: 1byte head (value: 2), 10byte
data (2byte version + 8byte tag), 2byte checksum, 1byte tail (value: 3)
const int DATA_SIZE = 10; // 10byte data (2byte version + 8byte tag)
const int DATA_VERSION_SIZE = 2; // 2byte version (actual meaning of these two bytes
may vary)
const int DATA_TAG_SIZE = 8; // 8byte tag
const int CHECKSUM_SIZE = 2; // 2byte checksum

SoftwareSerial ssrfid = SoftwareSerial(2,8); RDM6300 TX pin to D2 Arduino Board.
```

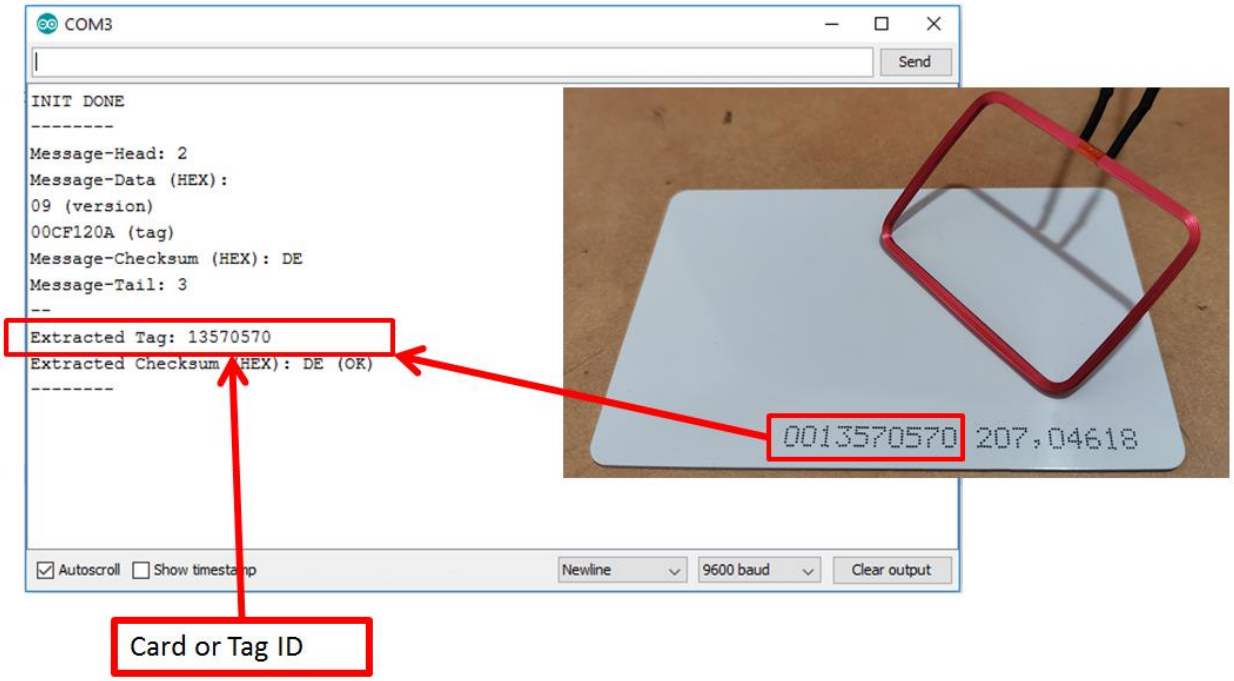
```
uint8_t buffer[BUFFER_SIZE]; // used to store an incoming data frame
int buffer_index = 0;

void setup() {
 Serial.begin(9600);

 ssrfid.begin(9600);
 ssrfid.listen();

 Serial.println("INIT DONE");
}

void loop() {
  if (ssrfid.available() > 0){
    bool call_extract_tag = false;

    int ssvalue = ssrfid.read(); // read
    if (ssvalue == -1) { // no data was read
      return;
    }

    if (ssvalue == 2) { // RDM630/RDM6300 found a tag => tag incoming
      buffer_index = 0;
    } else if (ssvalue == 3) { // tag has been fully transmitted
      call_extract_tag = true; // extract tag at the end of the function call
    }

    if (buffer_index >= BUFFER_SIZE) { // checking for a buffer overflow (It's very
unlikely that an buffer overflow comes up!)
      Serial.println("Error: Buffer overflow detected!");
      return;
    }

    buffer[buffer_index++] = ssvalue; // everything is alright => copy current value to
buffer

    if (call_extract_tag == true) {
      if (buffer_index == BUFFER_SIZE) {
        unsigned tag = extract_tag();
      } else { // something is wrong... start again looking for preamble (value: 2)
        buffer_index = 0;
        return;
      }
    }
  }
}

unsigned extract_tag() {
    uint8_t msg_head = buffer[0];
    uint8_t *msg_data = buffer + 1; // 10 byte => data contains 2byte version + 8byte
tag
    uint8_t *msg_data_version = msg_data;
    uint8_t *msg_data_tag = msg_data + 2;
    uint8_t *msg_checksum = buffer + 11; // 2 byte
    uint8_t msg_tail = buffer[13];

    // print message that was sent from RDM630/RDM6300
    Serial.println("--------");

    Serial.print("Message-Head: ");
    Serial.println(msg_head);

    Serial.println("Message-Data (HEX): ");
    for (int i = 0; i < DATA_VERSION_SIZE; ++i) {
```

```
      Serial.print(char(msg_data_version[i]));
    }
    Serial.println(" (version)");
    for (int i = 0; i < DATA_TAG_SIZE; ++i) {
      Serial.print(char(msg_data_tag[i]));
    }
    Serial.println(" (tag)");

    Serial.print("Message-Checksum (HEX): ");
    for (int i = 0; i < CHECKSUM_SIZE; ++i) {
      Serial.print(char(msg_checksum[i]));
    }
    Serial.println("");

    Serial.print("Message-Tail: ");
    Serial.println(msg_tail);

    Serial.println("--");

    long tag = hexstr_to_value(msg_data_tag, DATA_TAG_SIZE);
    Serial.print("Extracted Tag: ");
    Serial.println(tag);

    long checksum = 0;
    for (int i = 0; i < DATA_SIZE; i+= CHECKSUM_SIZE) {
      long val = hexstr_to_value(msg_data + i, CHECKSUM_SIZE);
      checksum ^= val;
    }
    Serial.print("Extracted Checksum (HEX): ");
    Serial.print(checksum, HEX);
    if (checksum == hexstr_to_value(msg_checksum, CHECKSUM_SIZE)) { // compare
calculated checksum to retrieved checksum
      Serial.print(" (OK)"); // calculated checksum corresponds to transmitted checksum!
    } else {
      Serial.print(" (NOT OK)"); // checksums do not match
    }

    Serial.println("");
    Serial.println("--------");

    return tag;
}

long hexstr_to_value(char *str, unsigned int length) { // converts a hexadecimal value
(encoded as ASCII string) to a numeric value
  char* copy = malloc((sizeof(char) * length) + 1);
  memcpy(copy, str, sizeof(char) * length);
  copy[length] = '\0';
  // the variable "copy" is a copy of the parameter "str". "copy" has an additional
'\0' element to make sure that "str" is null-terminated.
  long value = strtol(copy, NULL, 16);  // strtol converts a null-terminated string to
a long value
  free(copy); // clean up
  return value;
}
```

Open the Serial Monitor from Arduino IDE with baudrate set to 9600, the below window will pop up with "INIT DONE":



Put the RFID card or the keychain to the reader. Let the reader and the tag closer until all the information is displayed.



Web Resource:

- MDU1040 RC522
- https://www.mschoeffler.de/2018/01/05/arduino-tutorial-how-to-use-the-rdm630-rdm6300-rfid-reader/

HandsOn Technology provides a multimedia and interactive platform for everyone interested in electronics. From beginner to diehard, from student to lecturer. Information, education, inspiration and entertainment. Analog and digital, practical and theoretical; software and hardware.

open source hardware

Hands*On* Technology support Open Source Hardware (OSHW) Development Platform.

Learn : Design : Share

handsontec.com

The Face behind our product quality…

In a world of constant change and continuous technological development, a new or replacement product is never far away – and they all need to be tested.
Many vendors simply import and sell wihtout checks and this cannot be the ultimate interests of anyone, particularly the customer. Every part sell on Handsotec is fully tested. So when buying from Handsontec products range, you can be confident you're getting outstanding quality and value.

We keep adding the new parts so that you can get rolling on your next project.

Breakout Boards & Modules

Connectors

Electro-Mechanical Parts

Engineering Material

Mechanical Hardware

Electronics Components

Power Supply

Arduino Board & Shield

Tools & Accessory