AA128

13.56Mhz MFRC-522 RC522 RFID + S50 Card + Keychain

V1.0.23.10.20

Preface

OurCompany

KUONGSHUN Electronic Company is a supplier and manufacturer of electronic components, it is committed to board and starter kit for Arduino, Raspberry PI, Smart Robot Car, 3D printer. It is also a collection of scientific research, design, production, maintenance and sales of high-tech enterprises, in the field of automation with professional standards and mature technology, we rapid rise in the field of foreign trade. Relying on technology and development, continuing to provide users with high-tech products, is our constant pursuit. Fully introduction of foreign advanced technology to enhance the value of our products.

Company gains users' praise for supplying first-class quality product and superb technical services, has now become the first choice of domestic and international procurement company.

Official Website: https://www.kuongshun-ks.com

AA128 13.56Mhz MFRC-522 RC522 RFID + S50 Card + Keychain

Product Description

This kit mainly contains 1 RFID-RC522 module, 1 standard S50 blank card, 1 S50 shaped card (keychain shape), 1 straight 8-pin pin with a pitch of 2.54mm and 1 curved 8-pin pin with a pitch of 2.54mm. Before use, you can choose to weld straight rows of pins and curved rows of pins according to your own needs.S50 blank card and S50 shaped card are read-write, read-write capacity are 4KB.

RFID-RC522 module using PHILIPS MFRC 522 original chip design card reading circuit, easy to use, low cost, suitable for equipment development, card reader development and other advanced applications, the need to carry out the design of radio frequency card terminal / production users. This module can be directly loaded into a variety of card reader mold. Module voltage of 3.3V, through the SPI interface simple lines can be directly connected to any CPU motherboard communication with the user, can ensure stable and reliable operation of the module, card reading distance.

Product Parameters

Operating Current: 13-26mA/DC 3.3V

Idle Current: 10-13mA/DC 3.3V

Sleep Current: <80uA

Peak Current: <30mA

Operating frequency: 13.56MHz

Supported card types: mifare1S 50, mifare1S 70, mifare UltraLight, mifare Pro, mifare Des fire

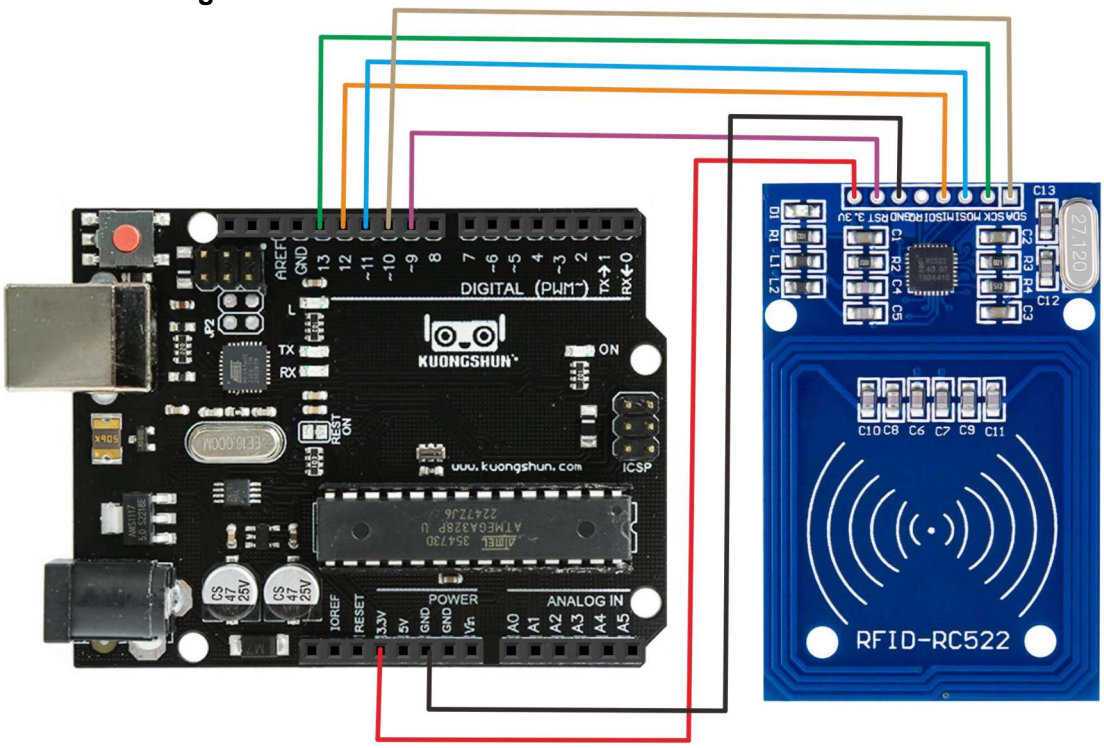
Size: 40mm×60mm

Ambient Operating Temperature: -20°C~+80°C

Ambient Storage Temperature: -40°C~+85°C

Data transfer rate: Maximum 10Mbit/s

Connection Diagram



Sample Code

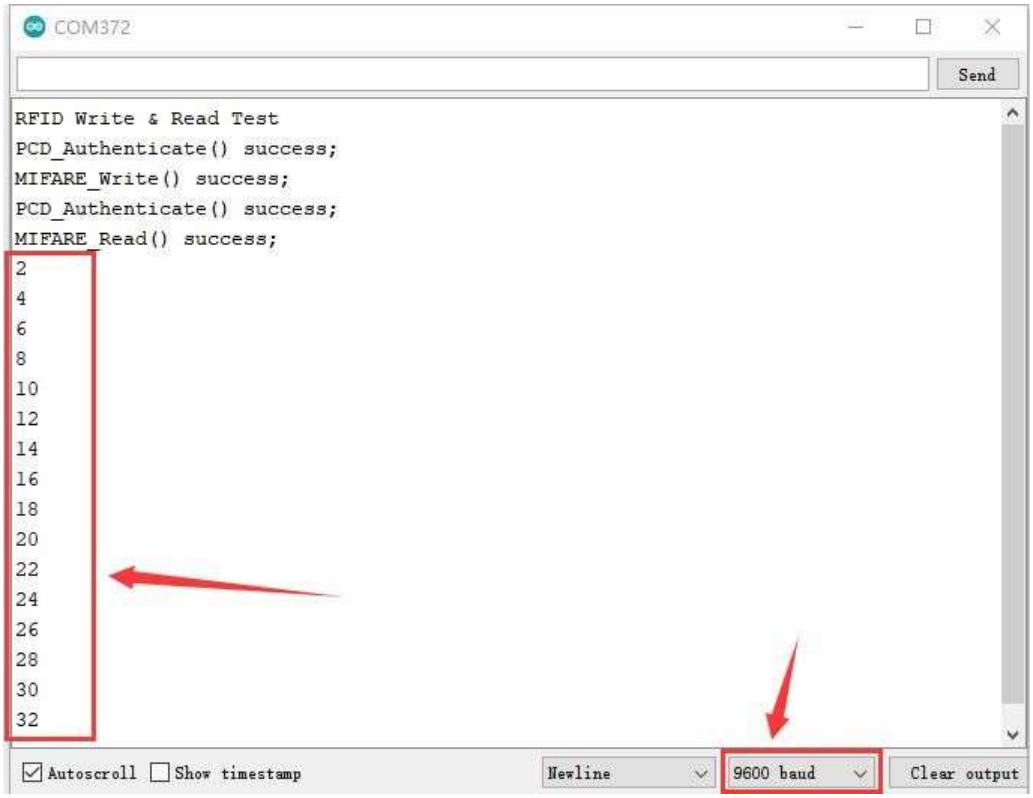
```
/*
https://www.kuongshun-ks.com/
*/
#include <SPI.h>
#include <MFRC522.h>

MFRC522 rfid(10, 9); byte
mylist[16]={2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32};

byte mylist2[16]={0};

boolean MFRC522_WriteCard(MFRC522 *_name, byte _block, byte *_buffer, byte _length){
  MFRC522::MIFARE_Key _key; for(byte i = 0; i < 6; i++)
    _key.keyByte[i] = 0xFF;
  MFRC522::StatusCode _status;
  _status = _name->PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, _block, &_key,
&(_name->uid)); if(_status !=
  MFRC522::STATUS_OK){
    Serial.print(F("PCD_Authenticate() failed: "));
    Serial.println(_name->GetStatusCodeName(_status));
    return false;
  }
```

```
    else{
      Serial.println(F("PCD_Authenticate() success;"));
    }
    _status = _name->MIFARE_Write(_block, _buffer, _length);
    if(_status != MFRC522::STATUS_OK){
      Serial.print(F("MIFARE_Write() failed: "));
      Serial.println(_name->GetStatusCodeName(_status));
      return false;
    }
    else{
      Serial.println(F("MIFARE_Write() success;"));
    }
    return true;
}

boolean MFRC522_ReadCard(MFRC522 *_name, byte _block, byte *_buffer, byte _length){
    MFRC522::MIFARE_Key _key; for(byte i = 0; i < 6; i++)
      _key.keyByte[i] = 0xFF;
    MFRC522::StatusCode _status;
    _status = _name->PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, _block, &_key,
&(_name->uid)); if(_status !=
    MFRC522::STATUS_OK){
      Serial.print(F("PCD_Authenticate() failed: "));
      Serial.println(_name->GetStatusCodeName(_status));
      return false;
    }
    else{
      Serial.println(F("PCD_Authenticate() success;"));
    }
    if(_length < 18){
      byte _Read_buffer[18]; byte
      _Read_buffer_length = 18;
      _status = _name->MIFARE_Read(_block, _Read_buffer, &_Read_buffer_length);
      if(_status != MFRC522::STATUS_OK){
        Serial.print(F("MIFARE_Read() failed: "));
        Serial.println(_name->GetStatusCodeName(_status));
        return false;
      }
      else{
        Serial.println(F("MIFARE_Read() success;"));
      }
      for(byte _i = 0; _i < _length; _i++)
        _buffer[_i] = _Read_buffer[_i];
```

```
    }
    else{
        _status = _name->MIFARE_Read(_block, _buffer, &_length);
        if(_status != MFRC522::STATUS_OK){
            Serial.print(F("MIFARE_Read() failed: "));
            Serial.println(_name->GetStatusCodeName(_status));
            return false;
        }
        else{
            Serial.println(F("MIFARE_Read() success;"));
        }
    }
    return true;
}

boolean MFRC522_IsNewCard(MFRC522 *_name){
    if(!_name->PICC_IsNewCardPresent())
        return false;
    if(!_name->PICC_ReadCardSerial())
        return false;
    return true;
}

void setup(){
    Serial.begin(9600);
    SPI.begin();
    rfid.PCD_Init();
    Serial.println("RFID Write & Read Test");
}

void loop(){
    if(MFRC522_IsNewCard(&rfid)){
        MFRC522_WriteCard(&rfid, 1, mylist, 16);
        MFRC522_ReadCard(&rfid, 1, mylist2, 16);
        for (int i = 1; i <= 16; i = i + (1)) {
            Serial.println(mylist2[(int)(i - 1)]);
        }
        rfid.PICC_HaltA();
        rfid.PCD_StopCrypto1();
    }
    delay(200);
}
```

Test Result

After connecting the wires according to the wiring method and uploading the code, open the serial monitor, set the baud rate to 9600, and put the S50 blank card (white card) or S50 shaped card (key fob) close to the RFID-RC522 module, the module will write an array of 16 values from 2 to 32 to the blank card or key fob, and at the same time, it will read the data of the blank card or the key and display the data on the serial monitor as shown in the figure below. data in the serial monitor, as shown in the figure below.



Resources

Related code and library files are linked below:

Xxxxxxxxxxx