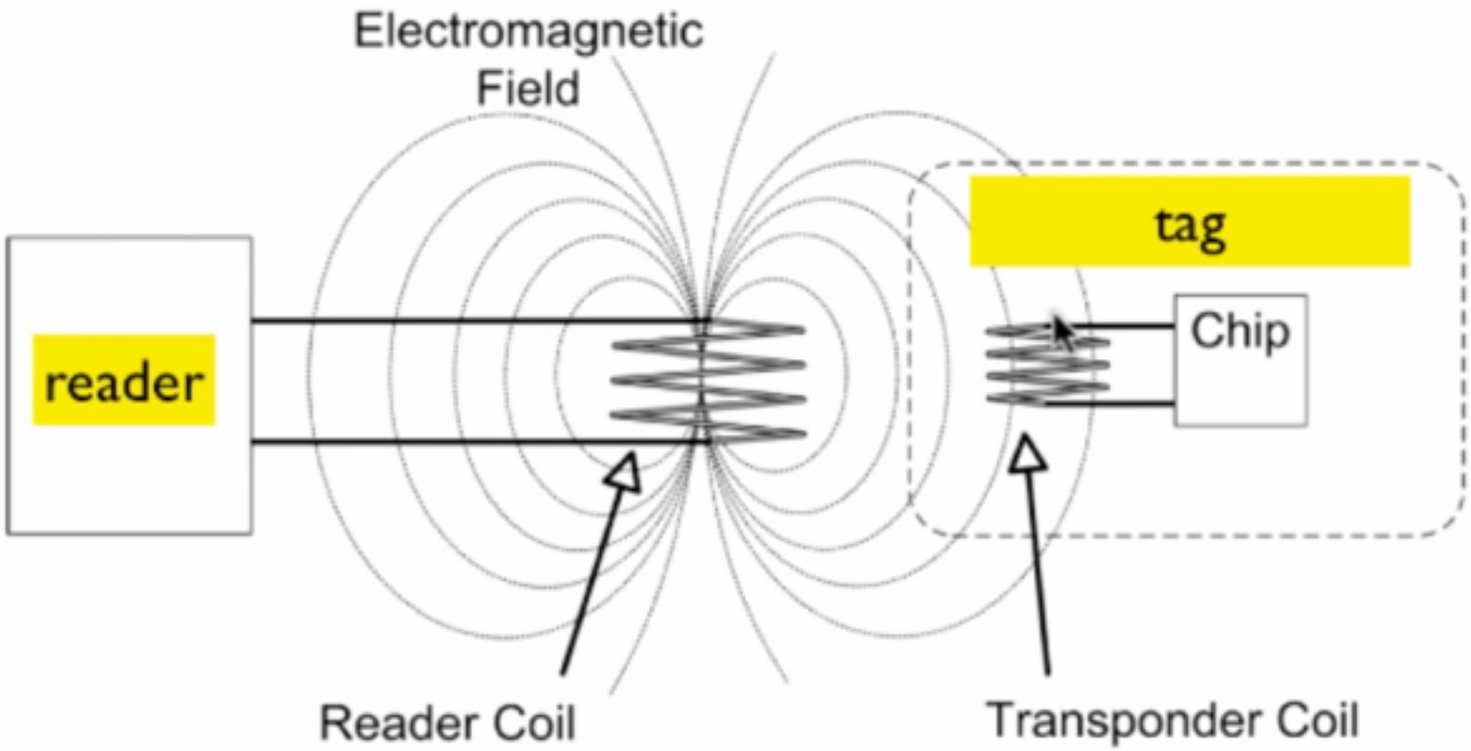
RFID Working Principle



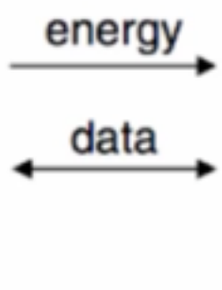
- RFID - Radio Frequency Identification
- The reader coil generates an electromagnetic field, which couples into the coil on the RFID tag ("transponder").
- The field generates a current in the transponder, which powers it, and also contains the transmitted data.

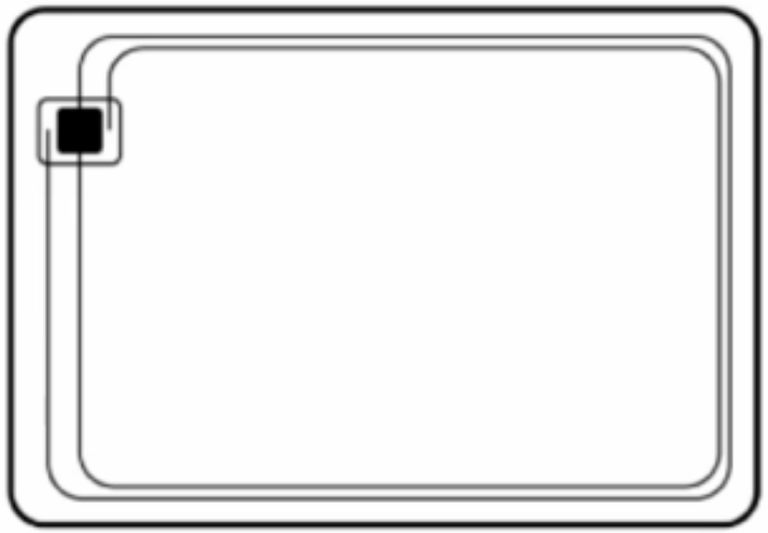
n: http://www.omicron-lab.com/fileadmin/assets/application_notes/App_Note_RFID_Resonance_Frequency_V1_0.pdf

MAKE: Hands-on Intro to Engineering Design

MIFARE RFID Card

- MIFARE MF1S503x 1kb RFID card
  - passive card, i.e. power comes via transmission.
- 1kb memory
- encryptable
- processing unit that can do arithmetic operations (to change $ $ amounts stored on the card, for example)
- Operating frequency of 13.56 MHz
- Operating distance up to 100 mm depending on antenna geometry and reader configuration
  - The Arduino Kit cards are for close proximity use (1-2 cm, as used for cash cards or building





MAKE: Hands-on Intro to Engineering Design

MIFARE RFID Card (1kb EEPROM)Structure

- MIFARE MF1S503x 1kb RFID card
- There are 16 sectors with 4 blocks each
- the 4th block of each sector ("sector trailer") contains the security keys and access bits
  - this is used to restrict and authenticate access to the stored data.
- The security keys are used to authenticate memory access
  - each time a block is being accessed the appropriate sector needs to be authenticated via one of the stored keys
- The access bits define what memory access is allowed.

8.6 Memory organization

The 1024 × 8 bit EEPROM memory is organized in 16 sectors of 4 blocks. One block contains 16 bytes.

Sector	Block	Byte Number within a Block 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Description
15	3	Key A						Access Bits				Key B						Sector Trailer 15
	2																	Data
	1																	Data
	0																	Data
14	3	Key A						Access Bits				Key B						Sector Trailer 14
	2																	Data
	1																	Data
	0																	Data
:	:																	
:	:																	
:	:																	
1	3	Key A						Access Bits				Key B						Sector Trailer 1
	2																	Data
	1																	Data
	0																	Data
0	3	Key A						Access Bits				Key B						Sector Trailer 0
	2																	Data
	1																	Data
	0	Manufacturer Data																Manufacturer Block

001aam011

Fig 5. Memory organization

Arduino SPI

Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances.

Typically there are three lines common to all the devices:

MISO (Master In Slave Out) - The Slave line for sending data to the master,
MOSI (Master Out Slave In) - The Master line for sending data to the peripherals,
SCK (Serial Clock) - The clock pulses which synchronize data transmission generated by the master

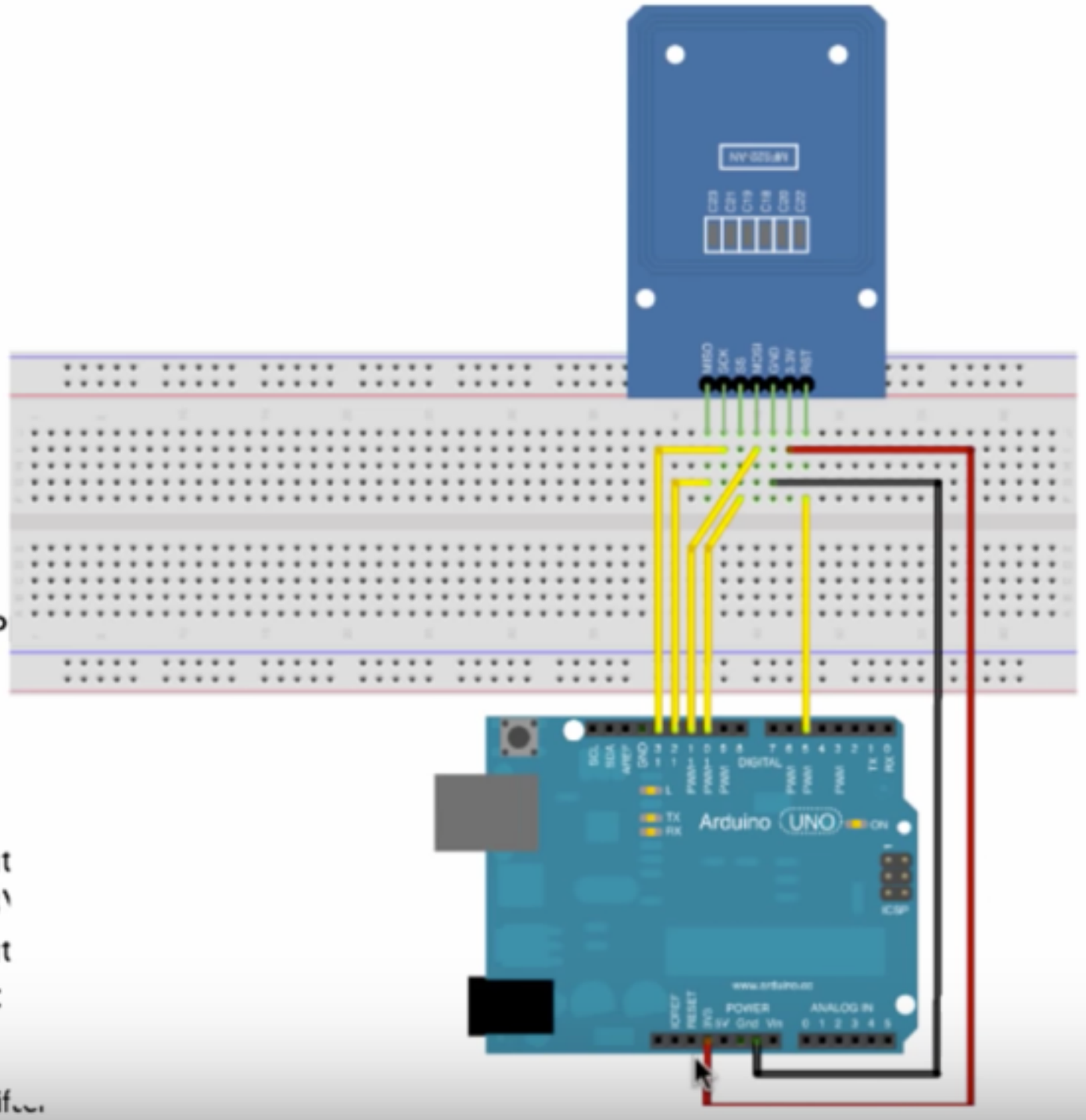
and one line specific for every device:

SS (Slave Select) - the pin on each device that the master can use to enable and disable specific devices.

MAKE: Hands-on Intro to Engineering Design

MF522 RFID Reader Setup with Arduino Uno

- The MF522 card is addressed via the SPI bus interface.
- The SPI bus has three lines:
  - MOSI (pin 11 on the UNO);Master Out Slave In
  - MISO (pin 12 on the UNO); Master In Slave Out
  - SCK (pin 13 on the UNO); Serial Clock
- Devices are selected via a 'slave select' (SS) line
  - Any remaining digital pin can be used.
  - Each device needs its own pin.
- Watch the SPI tutorial for further info on the SP bus.
- Note that the reader uses 3.3V
- The communication pins seem to be '5V tolerant This is not something to take for granted on 3.3\ devices. The data sheet is not specific on this, but many people on the internet seem to agree that works.
  - The proper way to do it would be to use a level-shif...

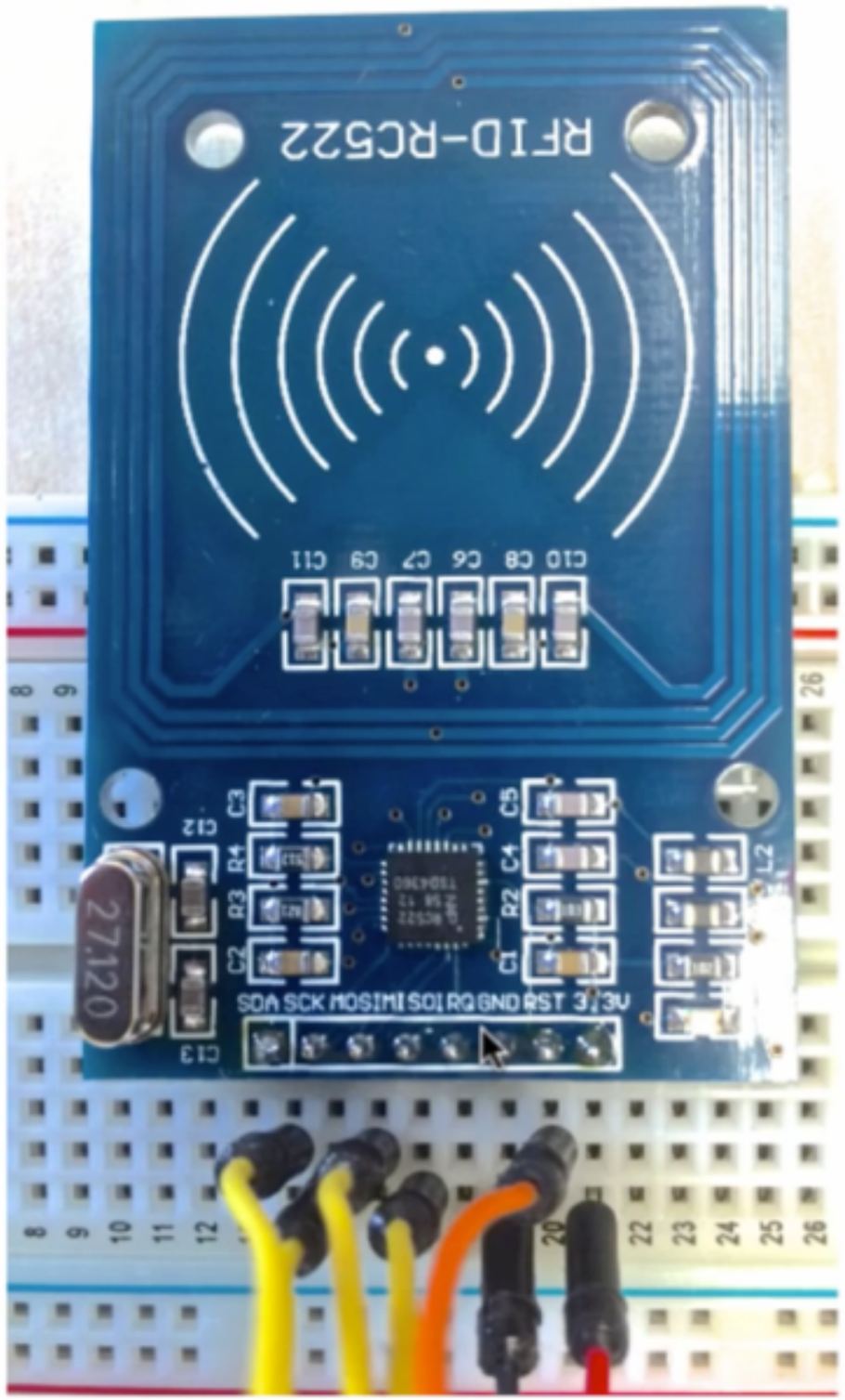


From: http://dkrgiesbammer.de/blog/nearfield-communication-rfid-prototype/

MAKE: Hands-on Intro to Engineering Design

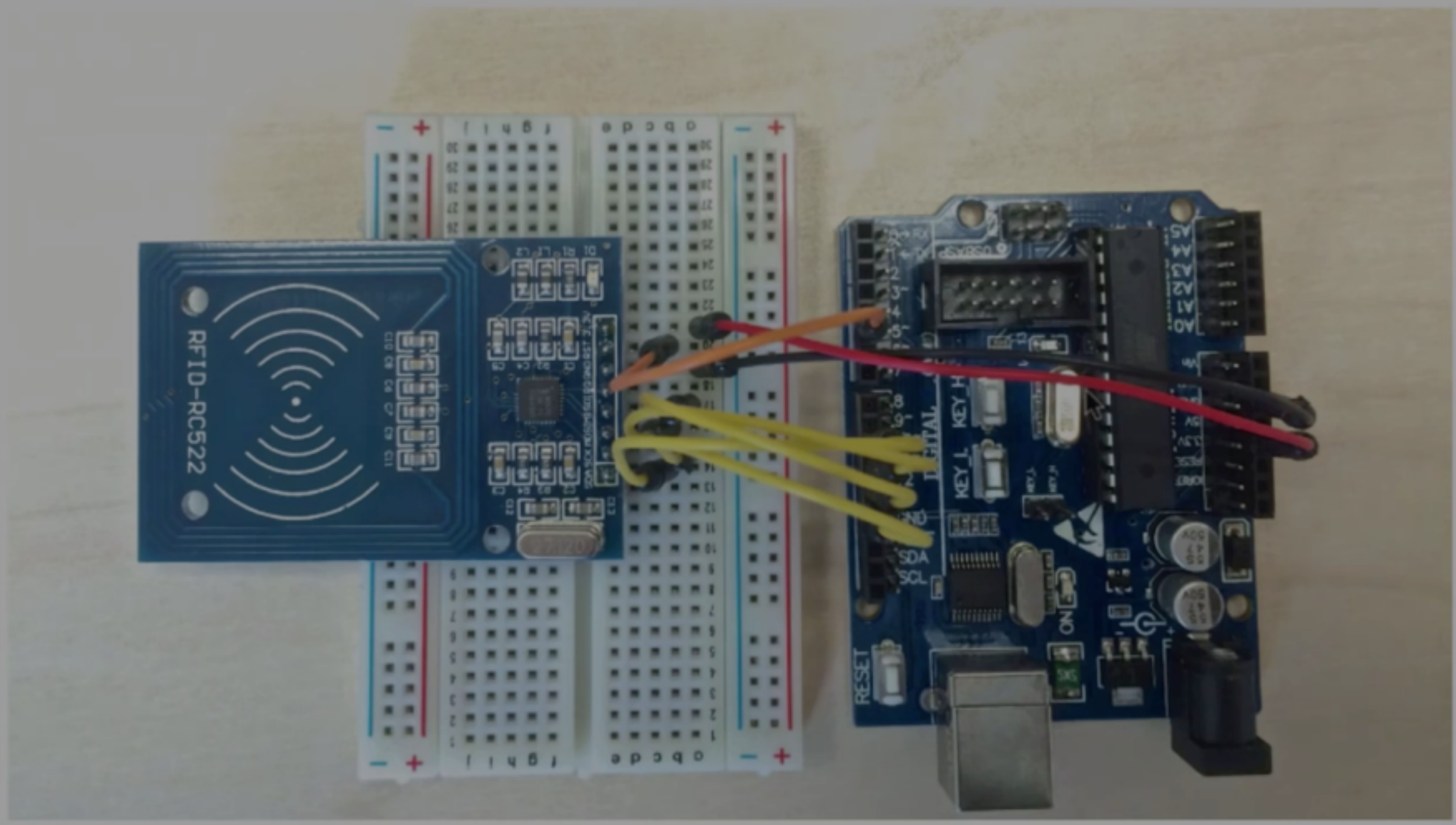
Different Reader Version With SDA Pin

- There is a slightly different version of RFID-RC522 shield.
- Difference: It also breaks out the I2C bus
  - Therefore there is a SDA pin (the pin on the left)
  - The SDA pin doubles as 'slave select' (SS) line for the SPI bus, i.e. connect SDA to pin 10 of the Arduino as SS line.
- There is also a "IRQ" pin (4th pin from the right)
  - This one is not used in our tutorial. Leave it unconnected.
- All other pins (SCK, MOSI, MISO,GND, RST, 3.3V are connected as for the other shield.



MAKE: Hands-on Intro to Engineering Design

Different Reader Version With SDA Pin



- This shows the setup with the Arduino Uno board

DumpInfo §

```
/* The reader can be found on eBay for around 5 dollars. Search for "mf-rc522" on ebay.com.
*/

#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 5
MFRC522 mfrc522(SS_PIN, RST_PIN);       // Create MFRC522 instance.

void setup() {
        Serial.begin(9600);     // Initialize serial communications with the PC
        SPI.begin();                    // Init SPI bus
        mfrc522.PCD_Init();     // Init MFRC522 card
        Serial.println("Scan PICC to see UID and type...");
}

void loop() {
        // Look for new cards
        if ( ! mfrc522.PICC_IsNewCardPresent()) {
                return;//go to start of loop if there is no card present
        }

        // Select one of the cards
        if ( ! mfrc522.PICC_ReadCardSerial()) {
                return;//if ReadCardSerial returns 1, the "uid" struct (see MFRC522.h lines 238-45)) contains the ID of the read card.
        }

        // Dump debug info about the card. PICC_HaltA() is automatically called.
        mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

Done uploading.

Binary sketch size: 9356 bytes (of a 32256 byte maximum)

The code..

```
void setup() {

	Serial.begin(9600);		// Initialize serial communications with the PC

	while (!Serial);		// Do nothing if no serial port is opened (added for Arduinos based on ATMEGA32U4)

	SPI.begin();			// Init SPI bus

	mfrc522.PCD_Init();		// Init MFRC522

	mfrc522.PCD_DumpVersionToSerial();	// Show details of PCD - MFRC522 Card Reader details

	Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks..."));

}
```

PCD: proximity coupling device (the card reader); PICC: proximity integrated circuit card

The code..

```
void loop() {
        // Look for new cards
        if ( ! mfrc522.PICC_IsNewCardPresent()) {
                return;
        }
        // Select one of the cards
        if ( ! mfrc522.PICC_ReadCardSerial()) {
                return;
        }
        // Dump debug info about the card; PICC_HaltA() is automatically called
        mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

DumpInfo §

```
 * The reader can be found on eBay for around 5 dollars. Search for "mf-rc522" on ebay.com.
 */

#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 5
MFRC522 mfrc522(SS_PIN, RST_PIN);        // Create MFRC522 instance.

void setup() {
        Serial.begin(9600);     // Initialize serial communications with the PC
        SPI.begin();                    // Init SPI bus
        mfrc522.PCD_Init();     // Init MFRC522 card
        Serial.println("Scan PICC to see UID and type...");
}

void loop() {
        // Look for new cards
        if ( ! mfrc522.PICC_IsNewCardPresent()) {
                return;//go to start of loop if there is no card present
        }

        // Select one of the cards
        if ( ! mfrc522.PICC_ReadCardSerial()) {
                return;//if ReadCardSerial returns 1, the "uid" struct (see MFRC522.h lines 23
        }

        // Dump debug info about the card. PICC_HaltA() is automatically called.
        mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

Done uploading.

```
Binary sketch size: 9356 bytes (of a 32256 byte maximum)
```

Send

```
Scan PICC to see UID and type...
Card UID: 54 C1 63 1A
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 ]
         37   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         36   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
   8     35   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         34   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         33   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         32   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
   7     31   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
         30   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         29   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
         28   00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  [ 0 0 0 ]
   6     27   00 00 00 00  00 00 FF 07  80 69 FF FF  FF FF FF FF  [ 0 0 1 ]
```

```
Card UID: B6 E1 AD 85
Card SAK: 08
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 ]
```

Read-Write

The setup code..

```
MFRC522::MIFARE_Key key;

for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
}

dump_byte_array(key.keyByte,MFRC522::MF_KEY_SIZE);
```

The loop code..

```
byte sector         = 1;

byte blockAddr      = 4;

byte dataBlock[]    = {

    0x01, 0x02, 0x03, 0x04, //  1,  2,   3,  4,

    0x05, 0x06, 0x07, 0x08, //  5,  6,   7,  8,

    0x08, 0x09, 0xff, 0x0b, //  9, 10, 255, 12,

    0x0c, 0x0d, 0x0e, 0x0f  // 13, 14,  15, 16

};

byte trailerBlock   = 7;
```

cont..

- **mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, trailerBlock, &key, &(mfrc522.uid));**

- **Returns STATUS_OK on success**

- **Read a sector: mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key, sector);**

Cont..

- **(MFRC522::StatusCode) mfrc522.MIFARE_Read(blockAddr, buffer, &size);**

- **(MFRC522::StatusCode) mfrc522.MIFARE_Write(blockAddr, dataBlock, 16);**

- **(MFRC522::StatusCode) mfrc522.MIFARE_Read(blockAddr, buffer, &size);**

Cont..

```
Serial.println(F("Checking result..."));

   byte count = 0;

   for (byte i = 0; i < 16; i++) {

        // Compare buffer (= what we've read) with dataBlock (= what we've
written)

        if (buffer[i] == dataBlock[i])

            count++;

    }

Correct if count is 16
```