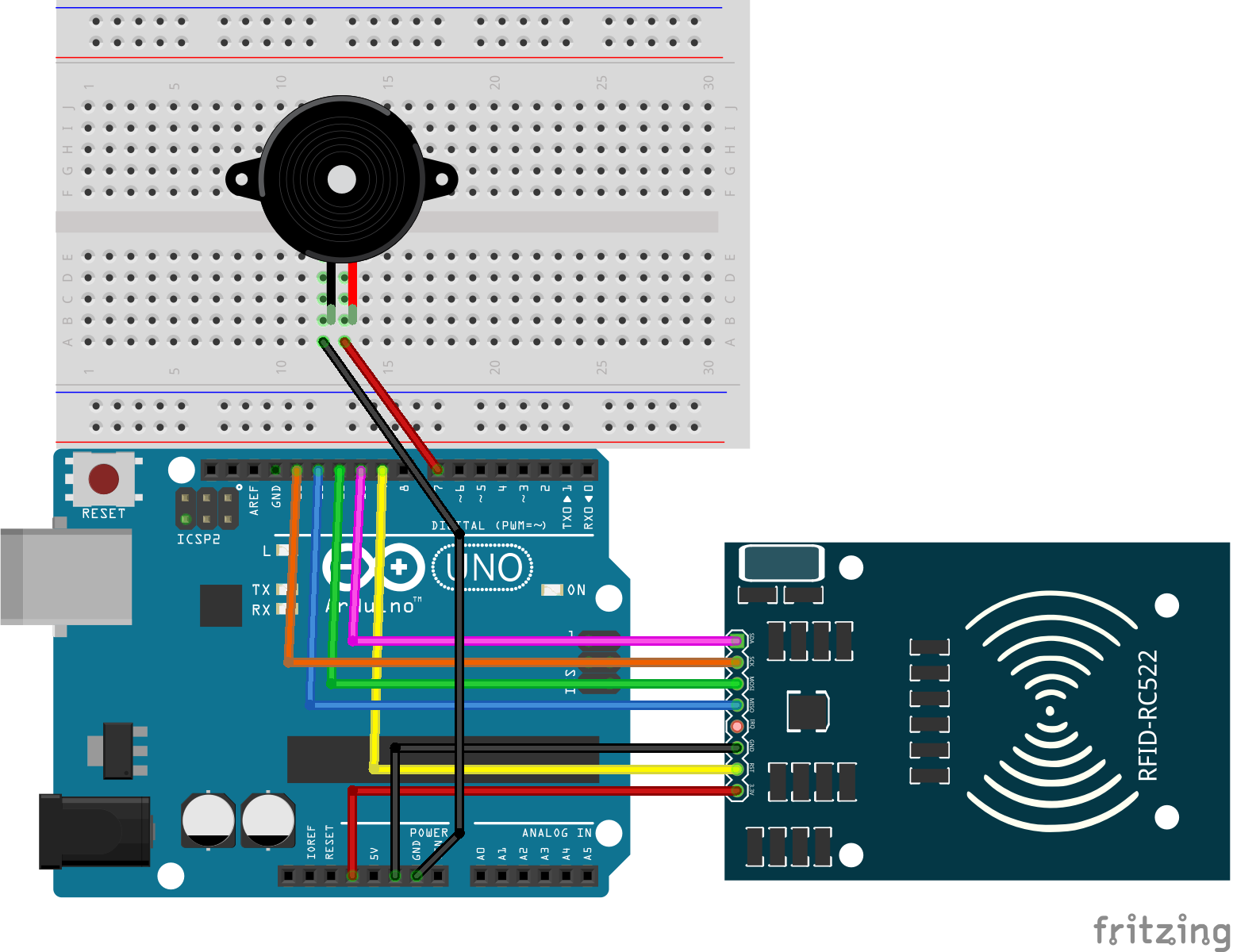
Danh sách linh kiện

| Stt | Tên linh kiện | Số lượng | Chức năng |
| --- | --- | --- | --- |
| 1 | Mạch Arduino Uno/Vietduino | 1 |  |
| 2 | Mạch RFID NFC 13.56MHz RC522 | 1 | đọc/viết thông tin lên móc khóa từ |
| 3 | Dây cắm đực-cái | 7 | kết nối mạch RFID với mạch Arduino |
| 4 | Dây cắm đực-đực | 2 | kết nối buzzer với mạch Arduino |
| 5 | Buzzer | 1 | phát tiếng mỗi khi mạch RFID đọc/viết thành công |
| 6 | Breadboard | 1 | chứa buzzer |

Mạch điện



Danh sách các thư viện sử dụng (nếu không có trên libraries manager thì dẫn link)

[GitHub - miguelbalboa/rfid: Arduino RFID Library for MFRC522](https://github.com/miguelbalboa/rfid)

Các phần mềm cần cài đặt

Tên: **PLX-DAQ-v2.11.zip**

[PLX-DAQ version 2 - now with 64 bit support! (and further new features) - Using Arduino / Interfacing w/ Software on the Computer - Arduino Forum](https://forum.arduino.cc/t/plx-daq-version-2-now-with-64-bit-support-and-further-new-features/420628/72#msg3251256)

Code

### read.ino

#include <SPI.h>

#include <MFRC522.h>

#define RST\_PIN 9

#define SS\_PIN 10

const int buzzer = 7;

MFRC522 mfrc522(SS\_PIN, RST\_PIN);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

void setup() {

pinMode(buzzer, OUTPUT);

**Serial**.begin(9600);

SPI.begin();

mfrc522.PCD\_Init();

**Serial**.println("CLEARDATA");

**Serial**.println("LABEL,CLOCK,ID");

}

void loop(){

if ( ! mfrc522.PICC\_IsNewCardPresent()) {

return;

}

if ( ! mfrc522.PICC\_ReadCardSerial()) {

return;

}

byte block = 4;

byte len = 18;

byte buffer1[18];

MFRC522::StatusCode status;

MFRC522::MIFARE\_Key key;

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

if (get\_authenticate(status, key, block) == 0) return;

if (read\_ID(status, block, buffer1, len) == 0) return;

**Serial**.print("DATA,TIME,");

String ID = get\_ID(buffer1);

**Serial**.print(ID);

**Serial**.println();

tone(buzzer, 3136, 250);

delay(1000); //change value if you want to read cards faster

mfrc522.PICC\_HaltA(); // Halt PICC

mfrc522.PCD\_StopCrypto1(); // Stop encryption on PCD

}

bool get\_authenticate(MFRC522::StatusCode status, MFRC522::MIFARE\_Key key, byte block){

status = mfrc522.PCD\_Authenticate(MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A, block, &key, &(mfrc522.uid));

if (status != MFRC522::STATUS\_OK) {

**Serial**.print(F("PCD\_Authenticate() failed: "));

**Serial**.println(mfrc522.GetStatusCodeName(status));

return 0;

}

else

{

**Serial**.println(F("PCD\_Authenticate() success "));

return 1;

}

}

bool read\_ID(MFRC522::StatusCode status, byte block, byte buffer1[18], byte len){

status = mfrc522.MIFARE\_Read(block, buffer1, &len);

if (status != MFRC522::STATUS\_OK) {

**Serial**.print(F("Reading failed: "));

**Serial**.println(mfrc522.GetStatusCodeName(status));

return 0;

}

return 1;

}

String get\_ID(byte buffer1[18]){

String ID = "";

for (int i = 0; i < 16; i++)

if((char)buffer1[i] == '\*') break;

else ID = ID + (char)buffer1[i];

return ID;

}

### write.ino

#include <SPI.h>

#include <MFRC522.h>

#define RST\_PIN 9

#define SS\_PIN 10

MFRC522 mfrc522(SS\_PIN, RST\_PIN);

const int buzzer = 7;

String IDs[5] = {"Thang","Ngoc","Minh","Thu","Nghia"};

int index = 0;

int tags = 5;

void setup() {

pinMode(buzzer, OUTPUT);

**Serial**.begin(9600);

SPI.begin();

mfrc522.PCD\_Init();

**Serial**.println(F("Write personal data on a MIFARE PICC "));

}

void loop() {

if(tags > 0){

if ( ! mfrc522.PICC\_IsNewCardPresent()) {

return;

}

if ( ! mfrc522.PICC\_ReadCardSerial()) {

return;

}

card\_info();

MFRC522::StatusCode status;

MFRC522::MIFARE\_Key key;

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

byte block = 4;

if (get\_authenticate(status, key, block) == 0) return;

byte len = 16;

byte buffer[len];

preprocess(buffer, len);

**Serial**.println();

write\_info(status, block, buffer, len);

**Serial**.print("Card ID: ");

**Serial**.print(IDs[index]);

**Serial**.println(" ");

mfrc522.PICC\_HaltA(); // Halt PICC

mfrc522.PCD\_StopCrypto1(); // Stop encryption on PCD

index++;

--tags;

}

}

void card\_info(){

**Serial**.println();

**Serial**.print(F("Card UID:")); //Dump UID

for (byte i = 0; i < mfrc522.uid.size; i++) {

**Serial**.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");

**Serial**.print(mfrc522.uid.uidByte[i], HEX);

}

}

bool get\_authenticate(MFRC522::StatusCode status, MFRC522::MIFARE\_Key key, byte block){

status = mfrc522.PCD\_Authenticate(MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A, block, &key, &(mfrc522.uid));

if (status != MFRC522::STATUS\_OK) {

**Serial**.print(F("PCD\_Authenticate() failed: "));

**Serial**.println(mfrc522.GetStatusCodeName(status));

return 0;

}

else

{

**Serial**.println(F("PCD\_Authenticate() success "));

return 1;

}

}

void preprocess(byte buffer[16], byte len){

String temp = (IDs[index]);

for(int i = 0; i < len; ++i)

buffer[i] = 0;

for(int i = 0; i < temp.length(); ++i)

buffer[i] = temp[i];

buffer[temp.length()] = '\*';

}

void write\_info(MFRC522::StatusCode status, byte block, byte buffer[16], byte len){

status = mfrc522.MIFARE\_Write(block, buffer, len);

if (status != MFRC522::STATUS\_OK) {

**Serial**.print(F("MIFARE\_Write() failed: "));

**Serial**.println(mfrc522.GetStatusCodeName(status));

return;

}

else{

**Serial**.println(F("MIFARE\_Write() success "));

tone(buzzer, 3136, 250);

}

}