RFID

**RFID Fundamentals**

RFID stands for: Radio Frequency Identification RFID, or Radio Frequency Identification, is a system for transferring data over short distances (typically less than 6 inches). Often only one of the two devices need to be powered, while the other is a passive device. This allows for easy use in such things as credit cards, key fobs, and pet collars as there is no need to worry about battery life. The downside is that the reader and the information holder (ie credit card) must be very close, and can only hold small amounts of data

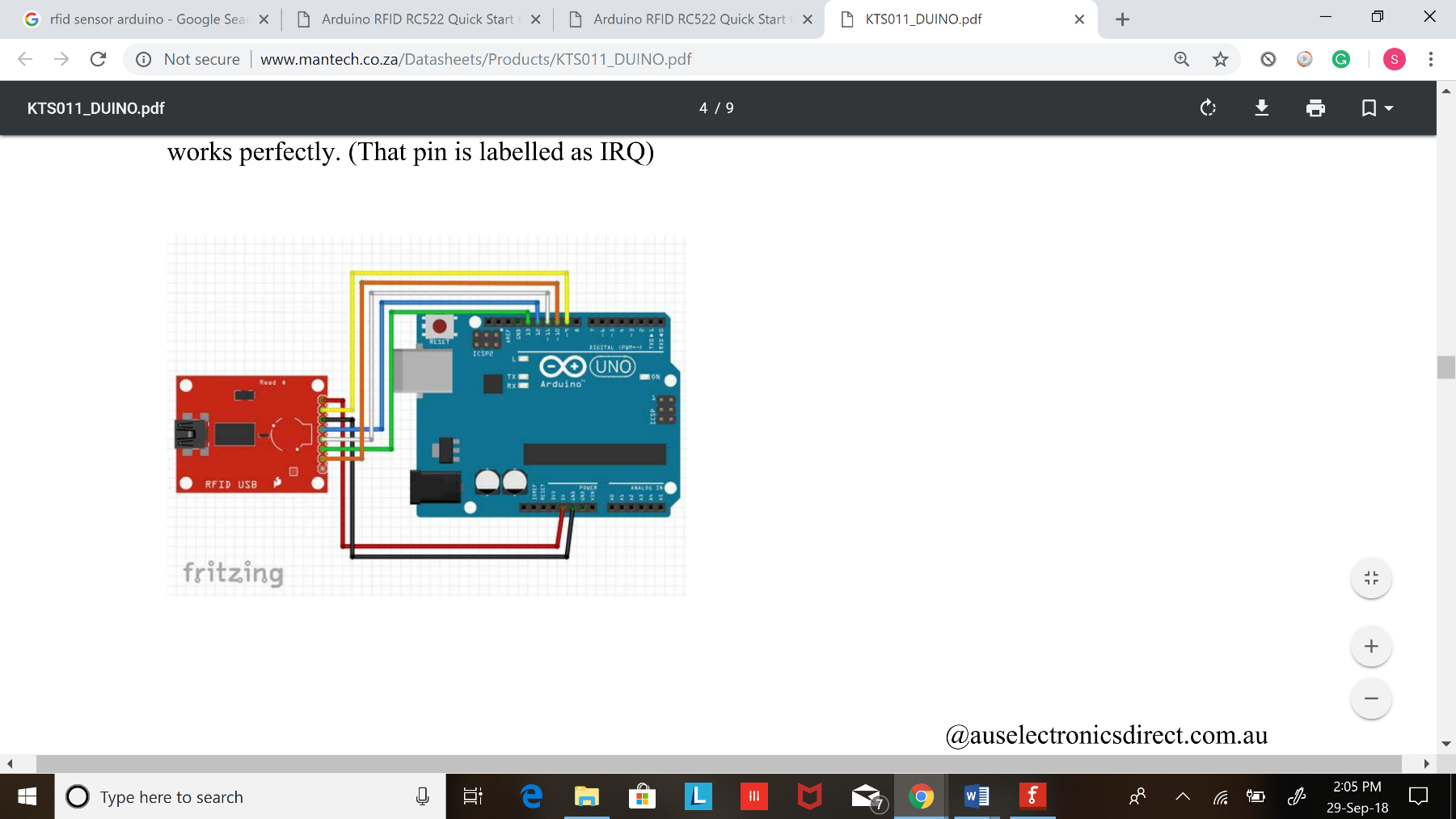
Each tag has a unique ID number that is read from the sensor and a microcontroller takes it from there. RFID cards/tags cannot hold data like a card with a magnetic strip, such as a metro card which keeps track of when it expires or runs out of money. However, a microcontroller can hold the data instead and have values increase and decrease for each time a recognizable card is swiped

In addition, unlike a magnetic strip card, RFID sensor do not have to make direct physical contact with the card and can sense a card from over and inch away.

**Wiring**

The following table shows the needed connections between the RFID and the Arduino Uno Cautions: **\*On the Arduino many of the pins are not swappable. Because this device uses the SPI bus, who’s pins cannot be moved around, pins 11, 12, 13 must remain as shown. RST and IRQ are user specified. \*This device is NOT a 5 volt powered device. You MUST power it with 3.3 volts. If you do not, you risk** overheating the RFID. Most Arduino boards include a 3.3V supply pin which can be used to power the RFID module. If 3.3 volts is not accessible, there are LD33V regulators available at Addicore.com that supply 3.3 volts.

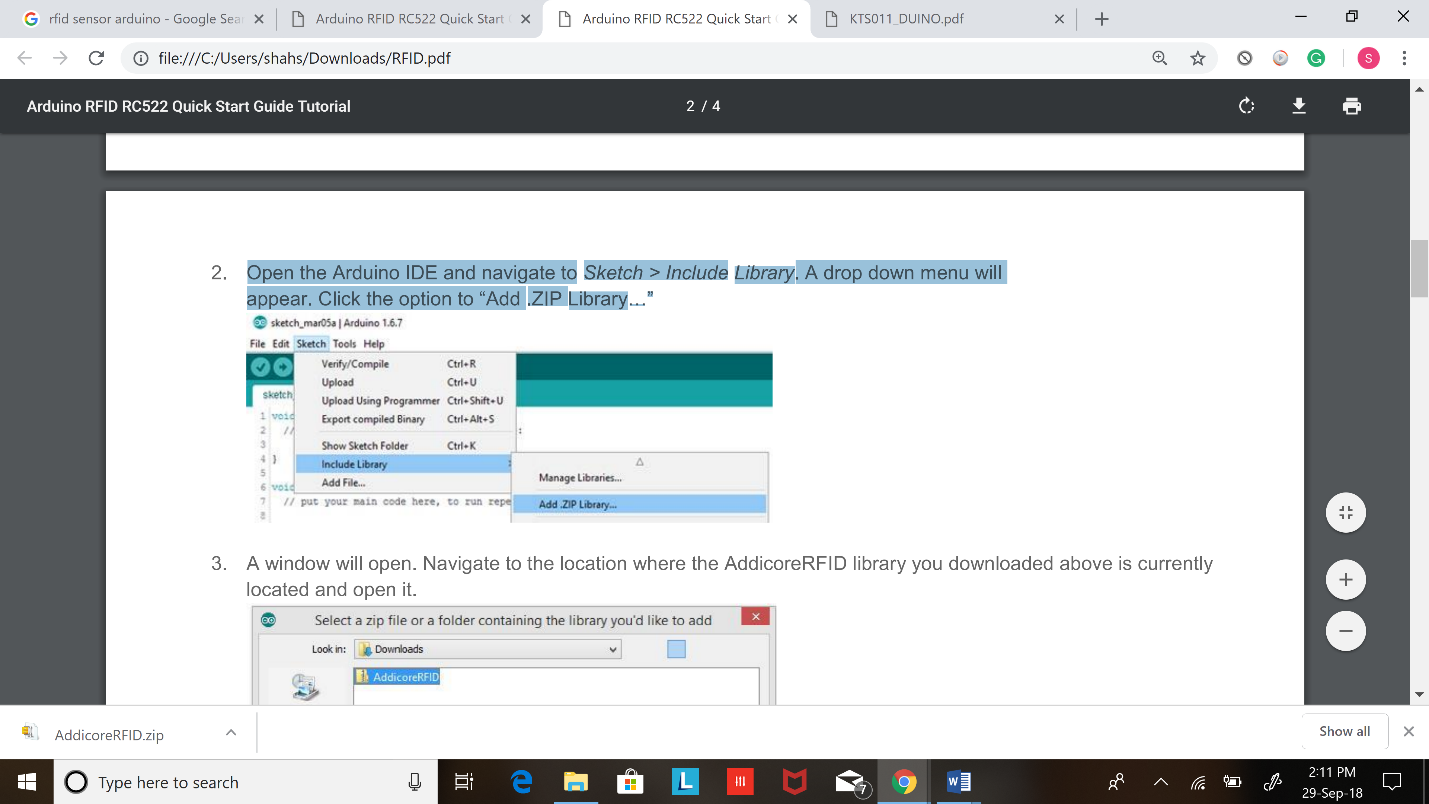
|  |  |  |  |
| --- | --- | --- | --- |
| RFID-RC5 | Arduino Uno | Node Mcu | Arduino Leonardo |
| 1 - SDA | Digital 10 | Digital 4 [GPIO 2] | Digital 10 |
| 2 - SCK | Digital 13 | Digital 5 [GPIO 14] | ICSP-3 |
| 3 - MOSI | Digital 11 | Digital 7 [GPIO 13] | ICSP-4 |
| 4 - MISO | Digital 12 | Digital 6 [GPIO 12] | ICSP-1 |
| 5 - IRQ | unconnected | unconnected | unconnected |
| 6 - GND | Gnd | Gnd | Gnd |
| 7 - RST | Digital 5 | Rst [Flash] | D2 |
| 8 - 3.3V | 3.3v | 3.3v | 3.3v |



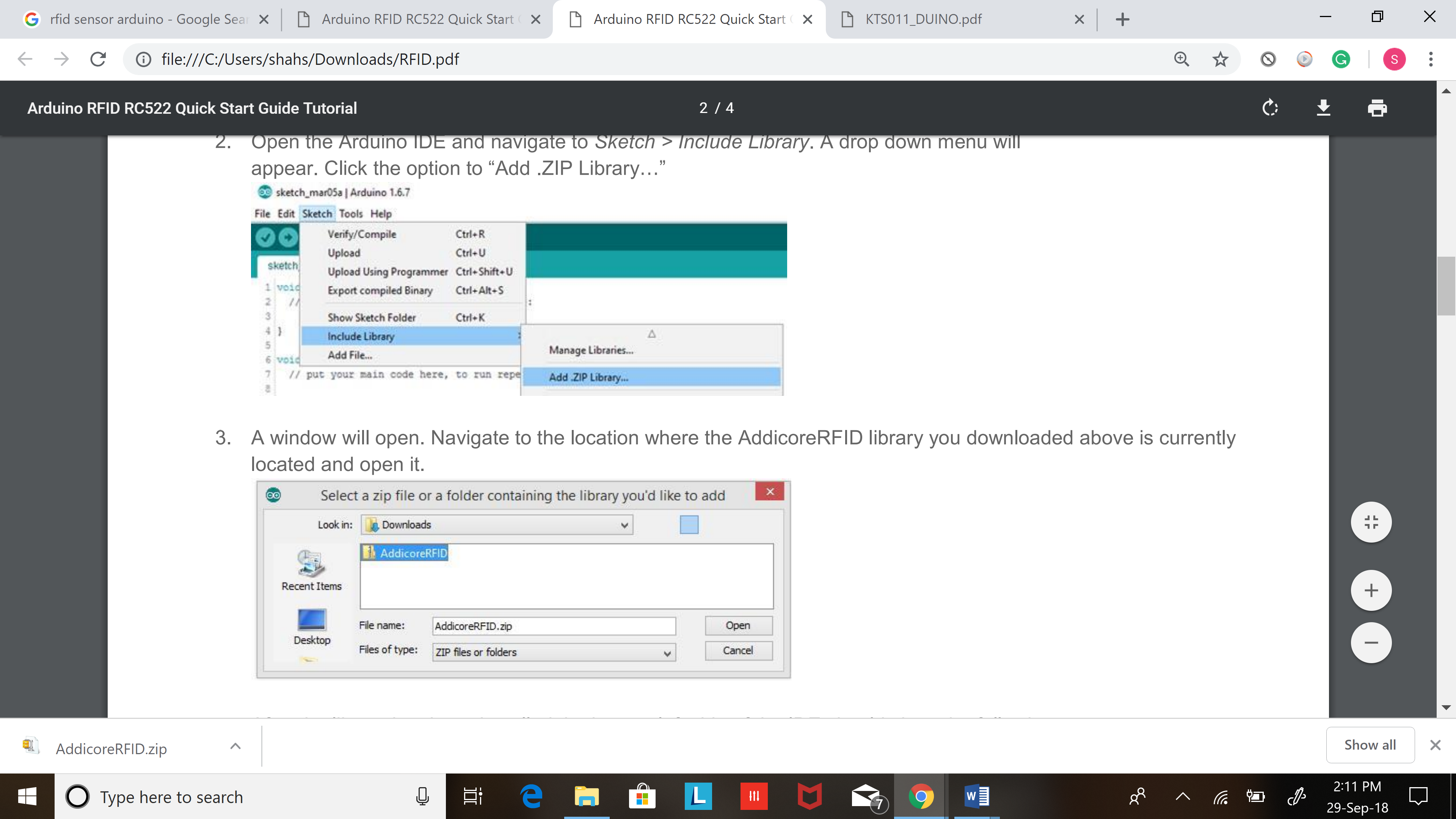
**Adding the Library**

If you haven’t already done so, the AddicoreRFID library needs to be added to your Arduino library depository. 1. Download the [AddicoreRFID](https://www.addicore.com/v/vspfiles/downloadables/Product%20Downloadables/RFID_RC522/AddicoreRFID.zip) library and also the MFRC522 library

Open the Arduino IDE and navigate to Sketch > Include Library. A drop down menu will appear. Click the option to “Add .ZIP Library…”



A window will open. Navigate to the location where the AddicoreRFID library you downloaded above is currently located and open it.



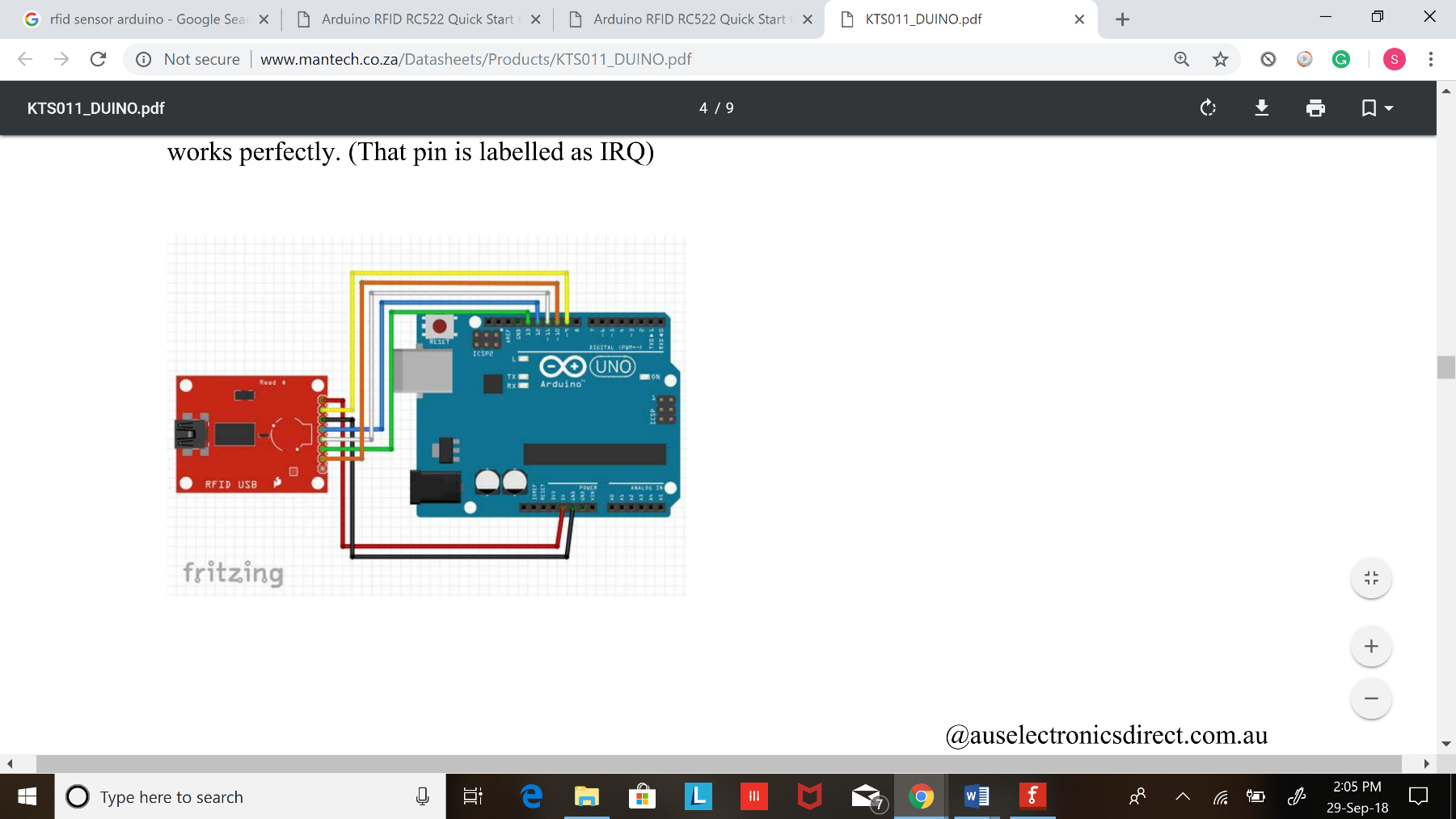
After the library has been installed the bottom left side of the IDE should show the following message: 4. You can confirm that the library has been installed by again navigating to Sketch > Include Library.

The AddicoreRFID library should now show in the list of “Contributed libraries.”, After that is done go to Include Library > Manage Library and search for MFRC522 and go ahead and download it.

Program the Arduino

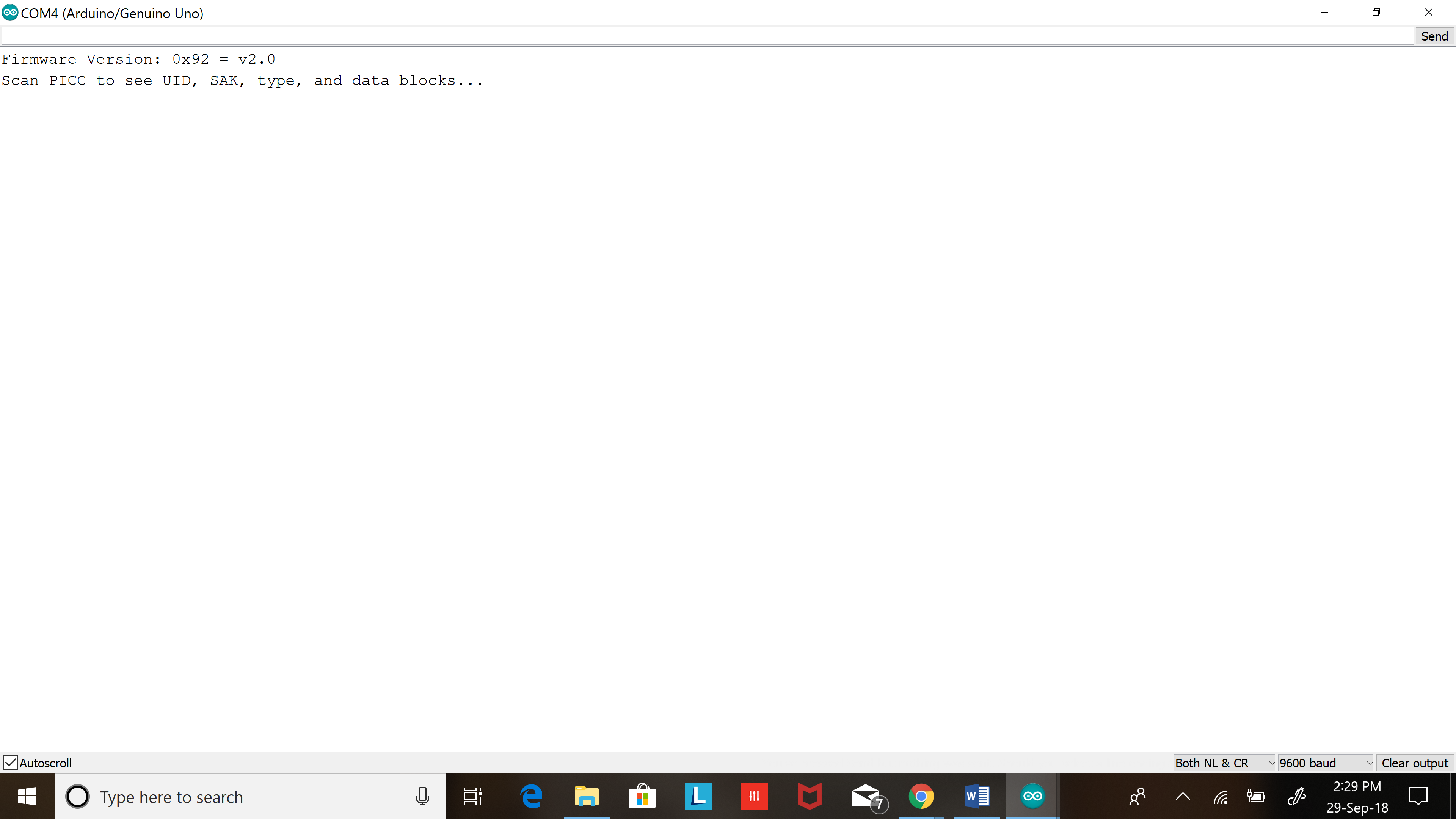
After installing the AddicoreRFID library in the steps above the library will be available to use in sketches but any example sketches included with the library may not be shown in File > Examples until after restarting the Arduino IDE.

1. Restart the Arduino IDE by closing all currently open Arduino IDE windows and then opening the Arduino IDE.
2. When the Arduino IDE opens navigate to File > Examples > AddicoreRFID. Select the “Addicore\_RFID\_Example” sketch. This will open a sketch which we will use with the RFID module that we wired to our Arduino earlier.
3. Now connect your Arduino to your computer and upload the code.

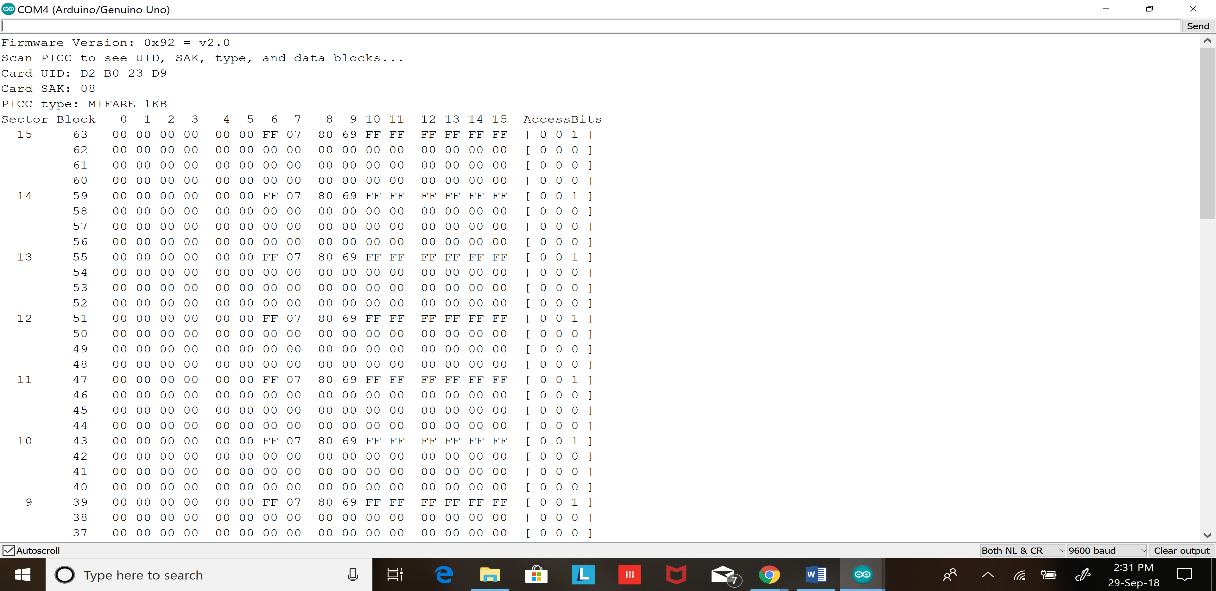
Reading RFID Card

|  |
| --- |
| /\*  Soumil shah  Bachelor in Electronic Engineering  Master’s in electrical engineering  Master’s in computer engineering  \* -----------------------------------------------------------------------------------------  \* MFRC522 Arduino Arduino Arduino Arduino Arduino  \* Reader/PCD Uno/101 Mega Nano v3 Leonardo/Micro Pro Micro  \* Signal Pin Pin Pin Pin Pin Pin  \* -----------------------------------------------------------------------------------------  \* RST/Reset RST 9 5 D9 RESET/ICSP-5 RST  \* SPI SS SDA(SS) 10 53 D10 10 10  \* SPI MOSI MOSI 11 / ICSP-4 51 D11 ICSP-4 16  \* SPI MISO MISO 12 / ICSP-1 50 D12 ICSP-1 14  \* SPI SCK SCK 13 / ICSP-3 52 D13 ICSP-3 15  \*/  #include <SPI.h>  #include <MFRC522.h>  #define RST\_PIN 9 // Configurable, see typical pin layout above  #define SS\_PIN 10 // Configurable, see typical pin layout above  MFRC522 mfrc522(SS\_PIN, RST\_PIN); // Create MFRC522 instance  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..."));  }  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));  } |

This code will dump all information about the card for example the card number data that is in that card and the manufacture of that card



Place card on the reader to check data



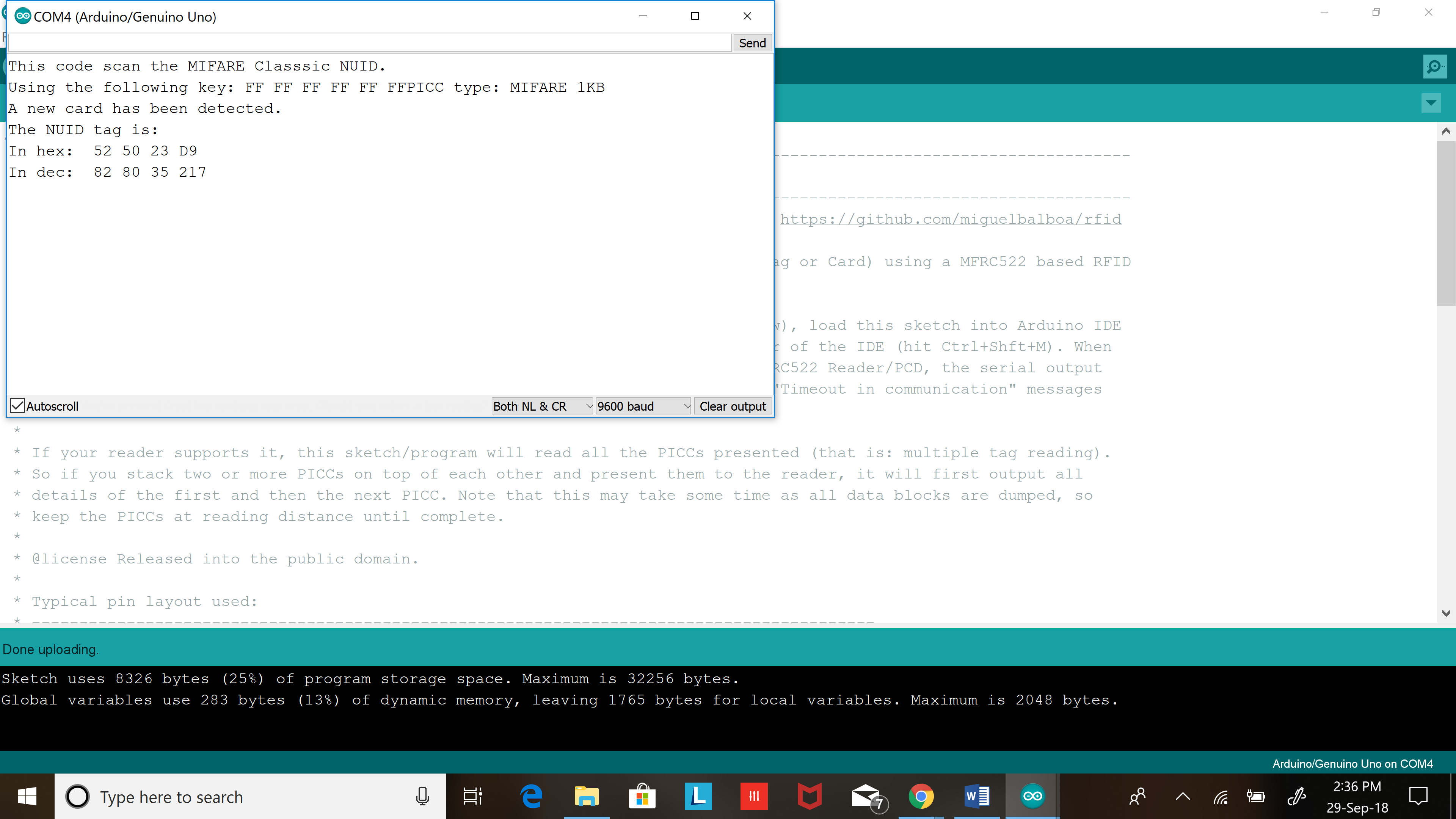
If you see this congrulations you have just read a RFID card

Part II

If you want to read specific value specific data that is NUID go ahead and try this code

|  |
| --- |
| /\*  \*  \* Typical pin layout used:  \* -----------------------------------------------------------------------------------------  \* MFRC522 Arduino Arduino Arduino Arduino Arduino  \* Reader/PCD Uno/101 Mega Nano v3 Leonardo/Micro Pro Micro  \* Signal Pin Pin Pin Pin Pin Pin  \* -----------------------------------------------------------------------------------------  \* RST/Reset RST 9 5 D9 RESET/ICSP-5 RST  \* SPI SS SDA(SS) 10 53 D10 10 10  \* SPI MOSI MOSI 11 / ICSP-4 51 D11 ICSP-4 16  \* SPI MISO MISO 12 / ICSP-1 50 D12 ICSP-1 14  \* SPI SCK SCK 13 / ICSP-3 52 D13 ICSP-3 15  \*/  #include <SPI.h>  #include <MFRC522.h>  #define SS\_PIN 10  #define RST\_PIN 9    MFRC522 rfid(SS\_PIN, RST\_PIN); // Instance of the class  MFRC522::MIFARE\_Key key;  // Init array that will store new NUID  byte nuidPICC[4];  void setup() {  Serial.begin(9600);  SPI.begin(); // Init SPI bus  rfid.PCD\_Init(); // Init MFRC522  for (byte i = 0; i < 6; i++) {  key.keyByte[i] = 0xFF;  }  Serial.println(F("This code scan the MIFARE Classsic NUID."));  Serial.print(F("Using the following key:"));  printHex(key.keyByte, MFRC522::MF\_KEY\_SIZE);  }    void loop() {  // Look for new cards  if ( ! rfid.PICC\_IsNewCardPresent())  return;  // Verify if the NUID has been readed  if ( ! rfid.PICC\_ReadCardSerial())  return;  Serial.print(F("PICC type: "));  MFRC522::PICC\_Type piccType = rfid.PICC\_GetType(rfid.uid.sak);  Serial.println(rfid.PICC\_GetTypeName(piccType));  // Check is the PICC of Classic MIFARE type  if (piccType != MFRC522::PICC\_TYPE\_MIFARE\_MINI &&  piccType != MFRC522::PICC\_TYPE\_MIFARE\_1K &&  piccType != MFRC522::PICC\_TYPE\_MIFARE\_4K) {  Serial.println(F("Your tag is not of type MIFARE Classic."));  return;  }  if (rfid.uid.uidByte[0] != nuidPICC[0] ||  rfid.uid.uidByte[1] != nuidPICC[1] ||  rfid.uid.uidByte[2] != nuidPICC[2] ||  rfid.uid.uidByte[3] != nuidPICC[3] ) {  Serial.println(F("A new card has been detected."));  // Store NUID into nuidPICC array  for (byte i = 0; i < 4; i++) {  nuidPICC[i] = rfid.uid.uidByte[i];  }    Serial.println(F("The NUID tag is:"));  Serial.print(F("In hex: "));  printHex(rfid.uid.uidByte, rfid.uid.size);  Serial.println();  Serial.print(F("In dec: "));  printDec(rfid.uid.uidByte, rfid.uid.size);  Serial.println();  }  else Serial.println(F("Card read previously."));  // Halt PICC  rfid.PICC\_HaltA();  // Stop encryption on PCD  rfid.PCD\_StopCrypto1();  }  /\*\*  \* Helper routine to dump a byte array as hex values to Serial.  \*/  void printHex(byte \*buffer, byte bufferSize) {  for (byte i = 0; i < bufferSize; i++) {  Serial.print(buffer[i] < 0x10 ? " 0" : " ");  Serial.print(buffer[i], HEX);  }  }  /\*\*  \* Helper routine to dump a byte array as dec values to Serial.  \*/  void printDec(byte \*buffer, byte bufferSize) {  for (byte i = 0; i < bufferSize; i++) {  Serial.print(buffer[i] < 0x10 ? " 0" : " ");  Serial.print(buffer[i], DEC);  }  } |

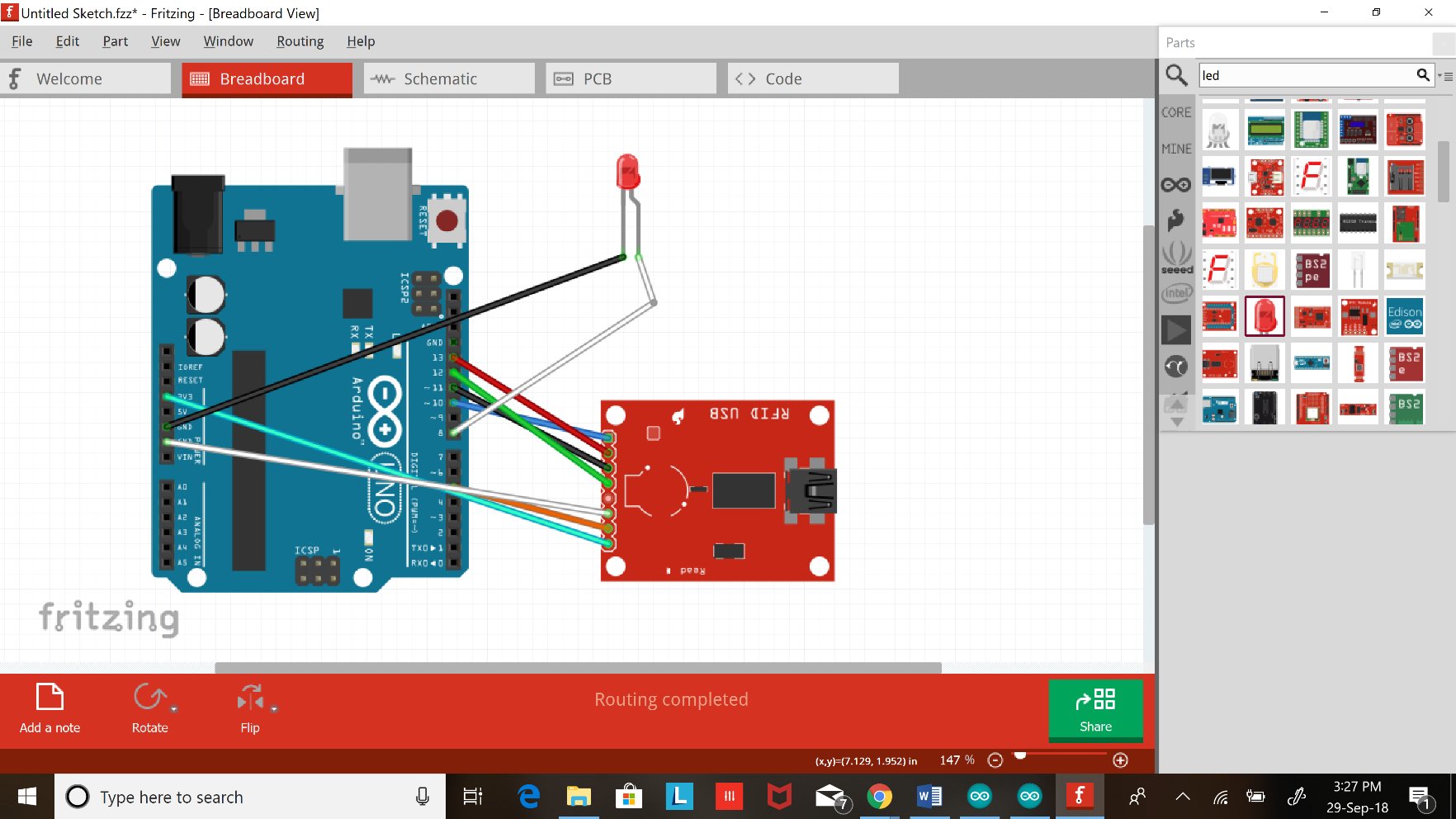
Output of this Code:



Great Job so far and lets move Ahead!

Challenge:

Lets us try to make a system that will turn on 110v AC or lets light up and led when authorized user is given access

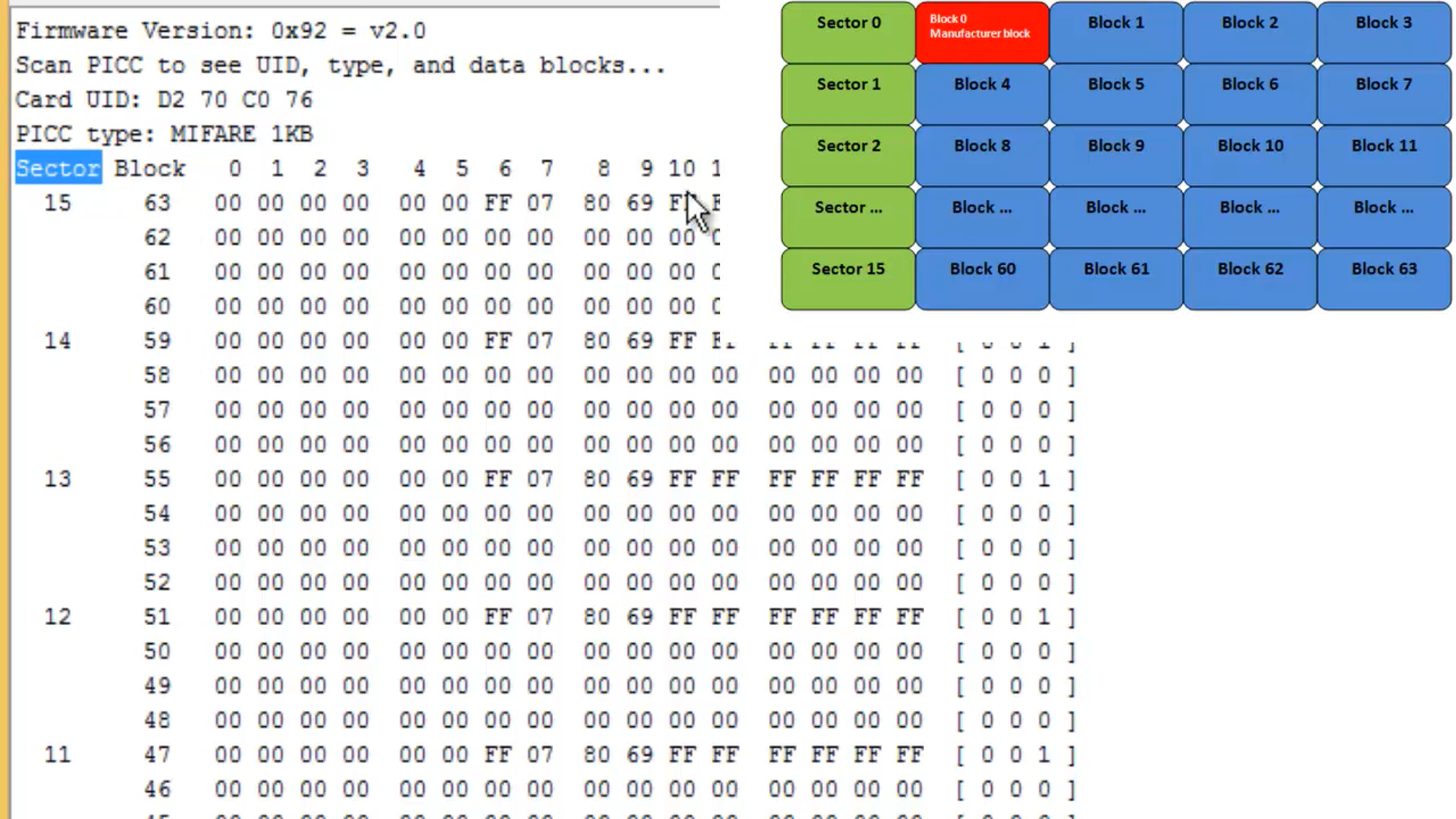


|  |
| --- |
| #include <SPI.h>  #include <MFRC522.h>    #define SS\_PIN 10  #define RST\_PIN 9  MFRC522 mfrc522(SS\_PIN, RST\_PIN); // Create MFRC522 instance.  void setup()  {  Serial.begin(9600); // Initiate a serial communication  SPI.begin(); // Initiate SPI bus  mfrc522.PCD\_Init(); // Initiate MFRC522  Serial.println("Put your card to the reader...");  Serial.println();  pinMode(8,OUTPUT);  }  void loop()  {  // Look for new cards  if ( ! mfrc522.PICC\_IsNewCardPresent())  {  return;  }  // Select one of the cards  if ( ! mfrc522.PICC\_ReadCardSerial())  {  return;  }  //Show UID on serial monitor  Serial.print("UID tag :");  String content= "";  byte letter;  for (byte i = 0; i < mfrc522.uid.size; i++)  {  Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");  Serial.print(mfrc522.uid.uidByte[i], HEX);  content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));  content.concat(String(mfrc522.uid.uidByte[i], HEX));  }  Serial.println();  Serial.print("Message : ");  content.toUpperCase();  //================================================================================================  if (content.substring(1) == "52 50 23 D9") //change here the UID of the card/cards that you want to give access  {  Serial.println("Authorized access");  Serial.println();  delay(600);  digitalWrite(8,HIGH);  }    else {  Serial.println(" Access denied");  delay(3000);  digitalWrite(8,LOW);  }  //============================================================================  } |

**That’s great go ahead and add 3 more and led and turn it on and off using RFID key and also make sure LED turn on and off in a pattern**

**Writing data on RFID**

**IMP Earlier we learned how to read NUID and dump information of the rfid on serial monitor and now lets learn how to write data into particular block. FID has 16 sector which is divided into 4 Block and each block can store 2B OF data.**



|  |
| --- |
| /\*  SOUMIL SHAH  Bachelor in Electronic Engineering  Master’s in electrical engineering  Master in Computer Engineering  \*/  #include <SPI.h>//include the SPI bus library  #include <MFRC522.h>//include the RFID reader library  #define SS\_PIN 10 //slave select pin  #define RST\_PIN 5 //reset pin  MFRC522 mfrc522(SS\_PIN, RST\_PIN); // instatiate a MFRC522 reader object.  MFRC522::MIFARE\_Key key;//create a MIFARE\_Key struct named 'key', which will hold the card information  void setup() {  Serial.begin(9600); // Initialize serial communications with the PC  SPI.begin(); // Init SPI bus  mfrc522.PCD\_Init(); // Init MFRC522 card (in case you wonder what PCD means: proximity coupling device)  Serial.println("Scan a MIFARE Classic card");    // Prepare the security key for the read and write functions - all six key bytes are set to 0xFF at chip delivery from the factory.  // Since the cards in the kit are new and the keys were never defined, they are 0xFF  // if we had a card that was programmed by someone else, we would need to know the key to be able to access it. This key would then need to be stored in 'key' instead.    for (byte i = 0; i < 6; i++) {  key.keyByte[i] = 0xFF;//keyByte is defined in the "MIFARE\_Key" 'struct' definition in the .h file of the library  }  }  //=======================================================================================================  int block=2;//this is the block number we will write into and then read. Do not write into 'sector trailer' block, since this can make the block unusable.    byte blockcontent[16] = {"ID :- 1031686 SHAH"};//an array with 16 bytes to be written into one of the 64 card blocks is defined!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  //byte blockcontent[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};//all zeros. This can be used to delete a block.  //byte readbackblock[18];//This array is used for reading out a block. The MIFARE\_Read method requires a buffer that is at least 18 bytes to hold the 16 bytes of a block.  //========================================================================================================  void loop()  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*establishing contact with a tag/card\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    // Look for new cards (in case you wonder what PICC means: proximity integrated circuit card)  if ( ! mfrc522.PICC\_IsNewCardPresent()) {//if PICC\_IsNewCardPresent returns 1, a new card has been found and we continue  return;//if it did not find a new card is returns a '0' and we return to the start of the loop  }  // Select one of the cards  if ( ! mfrc522.PICC\_ReadCardSerial()) {//if PICC\_ReadCardSerial returns 1, the "uid" struct (see MFRC522.h lines 238-45)) contains the ID of the read card.  return;//if it returns a '0' something went wrong and we return to the start of the loop  }  // Among other things, the PICC\_ReadCardSerial() method reads the UID and the SAK (Select acknowledge) into the mfrc522.uid struct, which is also instantiated  // during this process.  // The UID is needed during the authentication process  //The Uid struct:  //typedef struct {  //byte size; // Number of bytes in the UID. 4, 7 or 10.  //byte uidByte[10]; //the user ID in 10 bytes.  //byte sak; // The SAK (Select acknowledge) byte returned from the PICC after successful selection.  //} Uid;    Serial.println("card selected");    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*writing and reading a block on the card\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    writeBlock(block, blockcontent);//the blockcontent array is written into the card block  //mfrc522.PICC\_DumpToSerial(&(mfrc522.uid));    //The 'PICC\_DumpToSerial' method 'dumps' the entire MIFARE data block into the serial monitor. Very useful while programming a sketch with the RFID reader...  //Notes:  //(1) MIFARE cards conceal key A in all trailer blocks, and shows 0x00 instead of 0xFF. This is a secutiry feature. Key B appears to be public by default.  //(2) The card needs to be on the reader for the entire duration of the dump. If it is removed prematurely, the dump interrupts and an error message will appear.  //(3) The dump takes longer than the time alloted for interaction per pairing between reader and card, i.e. the readBlock function below will produce a timeout if  // the dump is used.  //=============================================================================================================  //mfrc522.PICC\_DumpToSerial(&(mfrc522.uid));//uncomment this if you want to see the entire 1k memory with the block written into it.    //readBlock(block, readbackblock);//read the block back  Serial.print("read block: ");  for (int j=0 ; j<16 ; j++)//print the block contents  {  // Serial.write (readbackblock[j]);//Serial.write() transmits the ASCII numbers as human readable characters to serial monitor  }  Serial.println("");    //====================================================================================================================================================  }  int writeBlock(int blockNumber, byte arrayAddress[])  {  //this makes sure that we only write into data blocks. Every 4th block is a trailer block for the access/security info.  int largestModulo4Number=blockNumber/4\*4;  int trailerBlock=largestModulo4Number+3;//determine trailer block for the sector  if (blockNumber > 2 && (blockNumber+1)%4 == 0){Serial.print(blockNumber);Serial.println(" is a trailer block:");return 2;}//block number is a trailer block (modulo 4); quit and send error code 2  Serial.print(blockNumber);  Serial.println(" is a data block:");    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*authentication of the desired block for access\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  byte status = mfrc522.PCD\_Authenticate(MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A, trailerBlock, &key, &(mfrc522.uid));  //byte PCD\_Authenticate(byte command, byte blockAddr, MIFARE\_Key \*key, Uid \*uid);  //this method is used to authenticate a certain block for writing or reading  //command: See enumerations above -> PICC\_CMD\_MF\_AUTH\_KEY\_A = 0x60 (=1100000), // this command performs authentication with Key A  //blockAddr is the number of the block from 0 to 15.  //MIFARE\_Key \*key is a pointer to the MIFARE\_Key struct defined above, this struct needs to be defined for each block. New cards have all A/B= FF FF FF FF FF FF  //Uid \*uid is a pointer to the UID struct that contains the user ID of the card.  if (status != MFRC522::STATUS\_OK) {  Serial.print("PCD\_Authenticate() failed: ");  Serial.println(mfrc522.GetStatusCodeName(status));  return 3;//return "3" as error message  }  //it appears the authentication needs to be made before every block read/write within a specific sector.  //If a different sector is being authenticated access to the previous one is lost.  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*writing the block\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    status = mfrc522.MIFARE\_Write(blockNumber, arrayAddress, 16);//valueBlockA is the block number, MIFARE\_Write(block number (0-15), byte array containing 16 values, number of bytes in block (=16))  //status = mfrc522.MIFARE\_Write(9, value1Block, 16);  if (status != MFRC522::STATUS\_OK) {  Serial.print("MIFARE\_Write() failed: ");  Serial.println(mfrc522.GetStatusCodeName(status));  return 4;//return "4" as error message  }  Serial.println("block was written");  }  int readBlock(int blockNumber, byte arrayAddress[])  {  int largestModulo4Number=blockNumber/4\*4;  int trailerBlock=largestModulo4Number+3;//determine trailer block for the sector  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*authentication of the desired block for access\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  byte status = mfrc522.PCD\_Authenticate(MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A, trailerBlock, &key, &(mfrc522.uid));  //byte PCD\_Authenticate(byte command, byte blockAddr, MIFARE\_Key \*key, Uid \*uid);  //this method is used to authenticate a certain block for writing or reading  //command: See enumerations above -> PICC\_CMD\_MF\_AUTH\_KEY\_A = 0x60 (=1100000), // this command performs authentication with Key A  //blockAddr is the number of the block from 0 to 15.  //MIFARE\_Key \*key is a pointer to the MIFARE\_Key struct defined above, this struct needs to be defined for each block. New cards have all A/B= FF FF FF FF FF FF  //Uid \*uid is a pointer to the UID struct that contains the user ID of the card.  if (status != MFRC522::STATUS\_OK) {  Serial.print("PCD\_Authenticate() failed (read): ");  Serial.println(mfrc522.GetStatusCodeName(status));  return 3;//return "3" as error message  }  //it appears the authentication needs to be made before every block read/write within a specific sector.  //If a different sector is being authenticated access to the previous one is lost.  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*reading a block\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    byte buffersize = 18;//we need to define a variable with the read buffer size, since the MIFARE\_Read method below needs a pointer to the variable that contains the size...  status = mfrc522.MIFARE\_Read(blockNumber, arrayAddress, &buffersize);//&buffersize is a pointer to the buffersize variable; MIFARE\_Read requires a pointer instead of just a number  if (status != MFRC522::STATUS\_OK) {  Serial.print("MIFARE\_read() failed: ");  Serial.println(mfrc522.GetStatusCodeName(status));  return 4;//return "4" as error message  }  Serial.println("block was read");  } |

Lets read the data that we just written in block 2

|  |
| --- |
| #include <SPI.h>//include the SPI bus library  #include <MFRC522.h>//include the RFID reader library  #define SS\_PIN 10 //slave select pin  #define RST\_PIN 5 //reset pin  MFRC522 mfrc522(SS\_PIN, RST\_PIN); // instatiate a MFRC522 reader object.  MFRC522::MIFARE\_Key key;//create a MIFARE\_Key struct named 'key', which will hold the card information  void setup() {  Serial.begin(9600); // Initialize serial communications with the PC  SPI.begin(); // Init SPI bus  mfrc522.PCD\_Init(); // Init MFRC522 card (in case you wonder what PCD means: proximity coupling device)  Serial.println("Scan a MIFARE Classic card");    // Prepare the security key for the read and write functions - all six key bytes are set to 0xFF at chip delivery from the factory.  // Since the cards in the kit are new and the keys were never defined, they are 0xFF  // if we had a card that was programmed by someone else, we would need to know the key to be able to access it. This key would then need to be stored in 'key' instead.    for (byte i = 0; i < 6; i++) {  key.keyByte[i] = 0xFF;//keyByte is defined in the "MIFARE\_Key" 'struct' definition in the .h file of the library  }  }  //=======================================================================================================  int block=2;//this is the block number we will write into and then read. Do not write into 'sector trailer' block, since this can make the block unusable.    //byte blockcontent[16] = {"ID :- 1031686"};//an array with 16 bytes to be written into one of the 64 card blocks is defined!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  //byte blockcontent[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};//all zeros. This can be used to delete a block.  byte readbackblock[18];//This array is used for reading out a block. The MIFARE\_Read method requires a buffer that is at least 18 bytes to hold the 16 bytes of a block.  //========================================================================================================  void loop()  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*establishing contact with a tag/card\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    // Look for new cards (in case you wonder what PICC means: proximity integrated circuit card)  if ( ! mfrc522.PICC\_IsNewCardPresent()) {//if PICC\_IsNewCardPresent returns 1, a new card has been found and we continue  return;//if it did not find a new card is returns a '0' and we return to the start of the loop  }  // Select one of the cards  if ( ! mfrc522.PICC\_ReadCardSerial()) {//if PICC\_ReadCardSerial returns 1, the "uid" struct (see MFRC522.h lines 238-45)) contains the ID of the read card.  return;//if it returns a '0' something went wrong and we return to the start of the loop  }  // Among other things, the PICC\_ReadCardSerial() method reads the UID and the SAK (Select acknowledge) into the mfrc522.uid struct, which is also instantiated  // during this process.  // The UID is needed during the authentication process  //The Uid struct:  //typedef struct {  //byte size; // Number of bytes in the UID. 4, 7 or 10.  //byte uidByte[10]; //the user ID in 10 bytes.  //byte sak; // The SAK (Select acknowledge) byte returned from the PICC after successful selection.  //} Uid;    Serial.println("card selected");    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*writing and reading a block on the card\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    //writeBlock(block, blockcontent);//the blockcontent array is written into the card block  //mfrc522.PICC\_DumpToSerial(&(mfrc522.uid));    //The 'PICC\_DumpToSerial' method 'dumps' the entire MIFARE data block into the serial monitor. Very useful while programming a sketch with the RFID reader...  //Notes:  //(1) MIFARE cards conceal key A in all trailer blocks, and shows 0x00 instead of 0xFF. This is a secutiry feature. Key B appears to be public by default.  //(2) The card needs to be on the reader for the entire duration of the dump. If it is removed prematurely, the dump interrupts and an error message will appear.  //(3) The dump takes longer than the time alloted for interaction per pairing between reader and card, i.e. the readBlock function below will produce a timeout if  // the dump is used.  //=============================================================================================================  //mfrc522.PICC\_DumpToSerial(&(mfrc522.uid));//uncomment this if you want to see the entire 1k memory with the block written into it.    readBlock(block, readbackblock);//read the block back  Serial.print("read block: ");  for (int j=0 ; j<16 ; j++)//print the block contents  {  Serial.write (readbackblock[j]);//Serial.write() transmits the ASCII numbers as human readable characters to serial monitor  }  Serial.println("");    //====================================================================================================================================================  }  int writeBlock(int blockNumber, byte arrayAddress[])  {  //this makes sure that we only write into data blocks. Every 4th block is a trailer block for the access/security info.  int largestModulo4Number=blockNumber/4\*4;  int trailerBlock=largestModulo4Number+3;//determine trailer block for the sector  if (blockNumber > 2 && (blockNumber+1)%4 == 0){Serial.print(blockNumber);Serial.println(" is a trailer block:");return 2;}//block number is a trailer block (modulo 4); quit and send error code 2  Serial.print(blockNumber);  Serial.println(" is a data block:");    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*authentication of the desired block for access\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  byte status = mfrc522.PCD\_Authenticate(MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A, trailerBlock, &key, &(mfrc522.uid));  //byte PCD\_Authenticate(byte command, byte blockAddr, MIFARE\_Key \*key, Uid \*uid);  //this method is used to authenticate a certain block for writing or reading  //command: See enumerations above -> PICC\_CMD\_MF\_AUTH\_KEY\_A = 0x60 (=1100000), // this command performs authentication with Key A  //blockAddr is the number of the block from 0 to 15.  //MIFARE\_Key \*key is a pointer to the MIFARE\_Key struct defined above, this struct needs to be defined for each block. New cards have all A/B= FF FF FF FF FF FF  //Uid \*uid is a pointer to the UID struct that contains the user ID of the card.  if (status != MFRC522::STATUS\_OK) {  Serial.print("PCD\_Authenticate() failed: ");  Serial.println(mfrc522.GetStatusCodeName(status));  return 3;//return "3" as error message  }  //it appears the authentication needs to be made before every block read/write within a specific sector.  //If a different sector is being authenticated access to the previous one is lost.  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*writing the block\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    status = mfrc522.MIFARE\_Write(blockNumber, arrayAddress, 16);//valueBlockA is the block number, MIFARE\_Write(block number (0-15), byte array containing 16 values, number of bytes in block (=16))  //status = mfrc522.MIFARE\_Write(9, value1Block, 16);  if (status != MFRC522::STATUS\_OK) {  Serial.print("MIFARE\_Write() failed: ");  Serial.println(mfrc522.GetStatusCodeName(status));  return 4;//return "4" as error message  }  Serial.println("block was written");  }  int readBlock(int blockNumber, byte arrayAddress[])  {  int largestModulo4Number=blockNumber/4\*4;  int trailerBlock=largestModulo4Number+3;//determine trailer block for the sector  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*authentication of the desired block for access\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  byte status = mfrc522.PCD\_Authenticate(MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A, trailerBlock, &key, &(mfrc522.uid));  //byte PCD\_Authenticate(byte command, byte blockAddr, MIFARE\_Key \*key, Uid \*uid);  //this method is used to authenticate a certain block for writing or reading  //command: See enumerations above -> PICC\_CMD\_MF\_AUTH\_KEY\_A = 0x60 (=1100000), // this command performs authentication with Key A  //blockAddr is the number of the block from 0 to 15.  //MIFARE\_Key \*key is a pointer to the MIFARE\_Key struct defined above, this struct needs to be defined for each block. New cards have all A/B= FF FF FF FF FF FF  //Uid \*uid is a pointer to the UID struct that contains the user ID of the card.  if (status != MFRC522::STATUS\_OK) {  Serial.print("PCD\_Authenticate() failed (read): ");  Serial.println(mfrc522.GetStatusCodeName(status));  return 3;//return "3" as error message  }  //it appears the authentication needs to be made before every block read/write within a specific sector.  //If a different sector is being authenticated access to the previous one is lost.  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*reading a block\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    byte buffersize = 18;//we need to define a variable with the read buffer size, since the MIFARE\_Read method below needs a pointer to the variable that contains the size...  status = mfrc522.MIFARE\_Read(blockNumber, arrayAddress, &buffersize);//&buffersize is a pointer to the buffersize variable; MIFARE\_Read requires a pointer instead of just a number  if (status != MFRC522::STATUS\_OK) {  Serial.print("MIFARE\_read() failed: ");  Serial.println(mfrc522.GetStatusCodeName(status));  return 4;//return "4" as error message  }  Serial.println("block was read");  } |

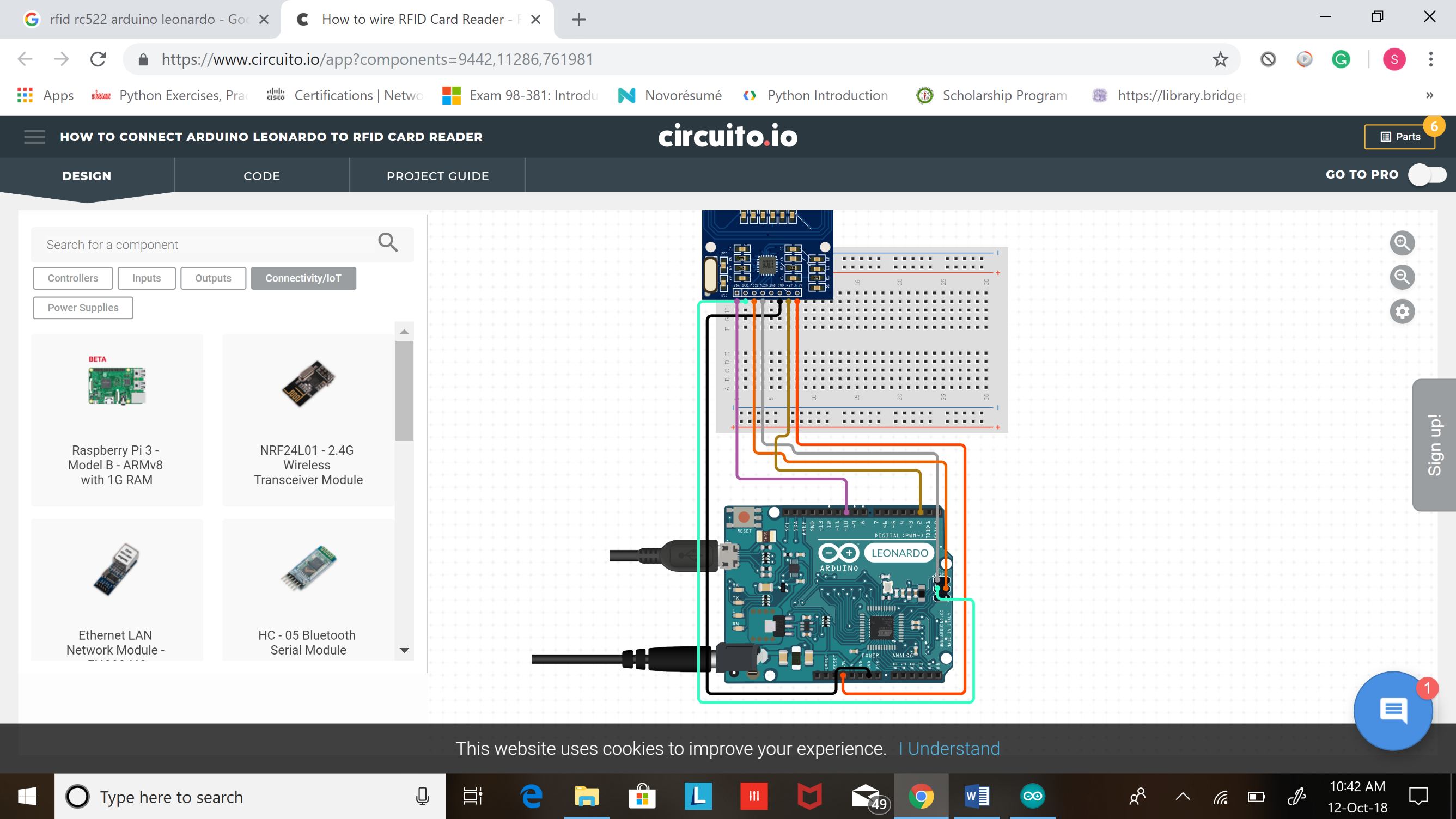
To Delete information on a particular block

|  |
| --- |
| #include <SPI.h>//include the SPI bus library  #include <MFRC522.h>//include the RFID reader library  #define SS\_PIN 10 //slave select pin  #define RST\_PIN 5 //reset pin  MFRC522 mfrc522(SS\_PIN, RST\_PIN); // instatiate a MFRC522 reader object.  MFRC522::MIFARE\_Key key;//create a MIFARE\_Key struct named 'key', which will hold the card information  void setup() {  Serial.begin(9600); // Initialize serial communications with the PC  SPI.begin(); // Init SPI bus  mfrc522.PCD\_Init(); // Init MFRC522 card (in case you wonder what PCD means: proximity coupling device)  Serial.println("Scan a MIFARE Classic card");    // Prepare the security key for the read and write functions - all six key bytes are set to 0xFF at chip delivery from the factory.  // Since the cards in the kit are new and the keys were never defined, they are 0xFF  // if we had a card that was programmed by someone else, we would need to know the key to be able to access it. This key would then need to be stored in 'key' instead.    for (byte i = 0; i < 6; i++) {  key.keyByte[i] = 0xFF;//keyByte is defined in the "MIFARE\_Key" 'struct' definition in the .h file of the library  }  }  //=======================================================================================================  int block=2;//this is the block number we will write into and then read. Do not write into 'sector trailer' block, since this can make the block unusable.    //byte blockcontent[16] = {"ID :- 1031686"};//an array with 16 bytes to be written into one of the 64 card blocks is defined!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  byte blockcontent[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};//all zeros. This can be used to delete a block.  byte readbackblock[18];//This array is used for reading out a block. The MIFARE\_Read method requires a buffer that is at least 18 bytes to hold the 16 bytes of a block.  //========================================================================================================  void loop()  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*establishing contact with a tag/card\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    // Look for new cards (in case you wonder what PICC means: proximity integrated circuit card)  if ( ! mfrc522.PICC\_IsNewCardPresent()) {//if PICC\_IsNewCardPresent returns 1, a new card has been found and we continue  return;//if it did not find a new card is returns a '0' and we return to the start of the loop  }  // Select one of the cards  if ( ! mfrc522.PICC\_ReadCardSerial()) {//if PICC\_ReadCardSerial returns 1, the "uid" struct (see MFRC522.h lines 238-45)) contains the ID of the read card.  return;//if it returns a '0' something went wrong and we return to the start of the loop  }  // Among other things, the PICC\_ReadCardSerial() method reads the UID and the SAK (Select acknowledge) into the mfrc522.uid struct, which is also instantiated  // during this process.  // The UID is needed during the authentication process  //The Uid struct:  //typedef struct {  //byte size; // Number of bytes in the UID. 4, 7 or 10.  //byte uidByte[10]; //the user ID in 10 bytes.  //byte sak; // The SAK (Select acknowledge) byte returned from the PICC after successful selection.  //} Uid;    Serial.println("card selected");    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*writing and reading a block on the card\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    writeBlock(block, blockcontent);//the blockcontent array is written into the card block  //mfrc522.PICC\_DumpToSerial(&(mfrc522.uid));    //The 'PICC\_DumpToSerial' method 'dumps' the entire MIFARE data block into the serial monitor. Very useful while programming a sketch with the RFID reader...  //Notes:  //(1) MIFARE cards conceal key A in all trailer blocks, and shows 0x00 instead of 0xFF. This is a secutiry feature. Key B appears to be public by default.  //(2) The card needs to be on the reader for the entire duration of the dump. If it is removed prematurely, the dump interrupts and an error message will appear.  //(3) The dump takes longer than the time alloted for interaction per pairing between reader and card, i.e. the readBlock function below will produce a timeout if  // the dump is used.  //=============================================================================================================  //mfrc522.PICC\_DumpToSerial(&(mfrc522.uid));//uncomment this if you want to see the entire 1k memory with the block written into it.    readBlock(block, readbackblock);//read the block back  Serial.print("read block: ");  for (int j=0 ; j<16 ; j++)//print the block contents  {  Serial.write (readbackblock[j]);//Serial.write() transmits the ASCII numbers as human readable characters to serial monitor  }  Serial.println("");    //====================================================================================================================================================  }  int writeBlock(int blockNumber, byte arrayAddress[])  {  //this makes sure that we only write into data blocks. Every 4th block is a trailer block for the access/security info.  int largestModulo4Number=blockNumber/4\*4;  int trailerBlock=largestModulo4Number+3;//determine trailer block for the sector  if (blockNumber > 2 && (blockNumber+1)%4 == 0){Serial.print(blockNumber);Serial.println(" is a trailer block:");return 2;}//block number is a trailer block (modulo 4); quit and send error code 2  Serial.print(blockNumber);  Serial.println(" is a data block:");    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*authentication of the desired block for access\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  byte status = mfrc522.PCD\_Authenticate(MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A, trailerBlock, &key, &(mfrc522.uid));  //byte PCD\_Authenticate(byte command, byte blockAddr, MIFARE\_Key \*key, Uid \*uid);  //this method is used to authenticate a certain block for writing or reading  //command: See enumerations above -> PICC\_CMD\_MF\_AUTH\_KEY\_A = 0x60 (=1100000), // this command performs authentication with Key A  //blockAddr is the number of the block from 0 to 15.  //MIFARE\_Key \*key is a pointer to the MIFARE\_Key struct defined above, this struct needs to be defined for each block. New cards have all A/B= FF FF FF FF FF FF  //Uid \*uid is a pointer to the UID struct that contains the user ID of the card.  if (status != MFRC522::STATUS\_OK) {  Serial.print("PCD\_Authenticate() failed: ");  Serial.println(mfrc522.GetStatusCodeName(status));  return 3;//return "3" as error message  }  //it appears the authentication needs to be made before every block read/write within a specific sector.  //If a different sector is being authenticated access to the previous one is lost.  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*writing the block\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    status = mfrc522.MIFARE\_Write(blockNumber, arrayAddress, 16);//valueBlockA is the block number, MIFARE\_Write(block number (0-15), byte array containing 16 values, number of bytes in block (=16))  //status = mfrc522.MIFARE\_Write(9, value1Block, 16);  if (status != MFRC522::STATUS\_OK) {  Serial.print("MIFARE\_Write() failed: ");  Serial.println(mfrc522.GetStatusCodeName(status));  return 4;//return "4" as error message  }  Serial.println("block was written");  }  int readBlock(int blockNumber, byte arrayAddress[])  {  int largestModulo4Number=blockNumber/4\*4;  int trailerBlock=largestModulo4Number+3;//determine trailer block for the sector  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*authentication of the desired block for access\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  byte status = mfrc522.PCD\_Authenticate(MFRC522::PICC\_CMD\_MF\_AUTH\_KEY\_A, trailerBlock, &key, &(mfrc522.uid));  //byte PCD\_Authenticate(byte command, byte blockAddr, MIFARE\_Key \*key, Uid \*uid);  //this method is used to authenticate a certain block for writing or reading  //command: See enumerations above -> PICC\_CMD\_MF\_AUTH\_KEY\_A = 0x60 (=1100000), // this command performs authentication with Key A  //blockAddr is the number of the block from 0 to 15.  //MIFARE\_Key \*key is a pointer to the MIFARE\_Key struct defined above, this struct needs to be defined for each block. New cards have all A/B= FF FF FF FF FF FF  //Uid \*uid is a pointer to the UID struct that contains the user ID of the card.  if (status != MFRC522::STATUS\_OK) {  Serial.print("PCD\_Authenticate() failed (read): ");  Serial.println(mfrc522.GetStatusCodeName(status));  return 3;//return "3" as error message  }  //it appears the authentication needs to be made before every block read/write within a specific sector.  //If a different sector is being authenticated access to the previous one is lost.  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*reading a block\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    byte buffersize = 18;//we need to define a variable with the read buffer size, since the MIFARE\_Read method below needs a pointer to the variable that contains the size...  status = mfrc522.MIFARE\_Read(blockNumber, arrayAddress, &buffersize);//&buffersize is a pointer to the buffersize variable; MIFARE\_Read requires a pointer instead of just a number  if (status != MFRC522::STATUS\_OK) {  Serial.print("MIFARE\_read() failed: ");  Serial.println(mfrc522.GetStatusCodeName(status));  return 4;//return "4" as error message  }  Serial.println("block was read");  } |

Automatic Login into UB portal using RFID Authentication system

|  |
| --- |
| #include <SPI.h>  #include <MFRC522.h>  #include "Keyboard.h"    #define SS\_PIN 10  #define RST\_PIN 2  MFRC522 mfrc522(SS\_PIN, RST\_PIN); // Create MFRC522 instance.    void setup()  {  Serial.begin(9600); // Initiate a serial communication  SPI.begin(); // Initiate SPI bus  mfrc522.PCD\_Init(); // Initiate MFRC522  Serial.println("Put your card to the reader...");  Serial.println();  Keyboard.begin();  }  void loop()  {  // Look for new cards  if ( ! mfrc522.PICC\_IsNewCardPresent())  {  return;  }  // Select one of the cards  if ( ! mfrc522.PICC\_ReadCardSerial())  {  return;  }  //Show UID on serial monitor  Serial.print("UID tag :");  String content= "";  byte letter;  for (byte i = 0; i < mfrc522.uid.size; i++)  {  Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");  Serial.print(mfrc522.uid.uidByte[i], HEX);  content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));  content.concat(String(mfrc522.uid.uidByte[i], HEX));  }  Serial.println();  Serial.print("Message : ");  content.toUpperCase();  //================================================================================================  if (content.substring(1) == "52 50 23 D9") //change here the UID of the card/cards that you want to give access  {  Serial.println("Authorized access");  Serial.println();  delay(600);  Keyboard.print("XXX USERNAME");  Keyboard.write(KEY\_TAB);  Keyboard.print("XXX- PASSWORD XX ----");  Keyboard.write(176);  }    else {  Serial.println(" Access denied");  delay(3000);  }  //============================================================================  } |

**Log into website by scanning RFID**



|  |  |  |  |
| --- | --- | --- | --- |
| RFID-RC5 | Arduino Uno | Node Mcu | Arduino Leonardo |
| 1 - SDA | Digital 10 | Digital 4 [GPIO 2] | Digital 10 |
| 2 - SCK | Digital 13 | Digital 5 [GPIO 14] | ICSP-3 |
| 3 - MOSI | Digital 11 | Digital 7 [GPIO 13] | ICSP-4 |
| 4 - MISO | Digital 12 | Digital 6 [GPIO 12] | ICSP-1 |
| 5 - IRQ | unconnected | unconnected | unconnected |
| 6 - GND | Gnd | Gnd | Gnd |
| 7 - RST | Digital 5 | Rst [Flash] | D2 |
| 8 - 3.3V | 3.3v | 3.3v | 3.3v |

**In this project we shall learn how to log in into UB portal using RFID. No need to enter User name and password through keyboard. This project can be used to protect username and password when using onto a public computer**

**Step 1:**

Go to my Ub portal <https://www.bridgeport.edu/myub/>

**Step 2:** wire the circuit and shown above

**Step 3:** upload code

**Step 4:** again, head to UB portal

**Step 5:** scan your code and you would see user name and password was automatically added and you were logged in

|  |
| --- |
| #include <SPI.h>  #include <MFRC522.h>  #include "Keyboard.h"    #define SS\_PIN 10  #define RST\_PIN 2  MFRC522 mfrc522(SS\_PIN, RST\_PIN); // Create MFRC522 instance.    void setup()  {  Serial.begin(9600); // Initiate a serial communication  SPI.begin(); // Initiate SPI bus  mfrc522.PCD\_Init(); // Initiate MFRC522  Serial.println("Put your card to the reader...");  Serial.println();  Keyboard.begin();  }  void loop()  {  // Look for new cards  if ( ! mfrc522.PICC\_IsNewCardPresent())  {  return;  }  // Select one of the cards  if ( ! mfrc522.PICC\_ReadCardSerial())  {  return;  }  //Show UID on serial monitor  Serial.print("UID tag :");  String content= "";  byte letter;  for (byte i = 0; i < mfrc522.uid.size; i++)  {  Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");  Serial.print(mfrc522.uid.uidByte[i], HEX);  content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));  content.concat(String(mfrc522.uid.uidByte[i], HEX));  }  Serial.println();  Serial.print("Message : ");  content.toUpperCase();  //================================================================================================  if (content.substring(1) == "52 50 23 D9") //change here the UID of the card/cards that you want to give access  {  Serial.println("Authorized access");  Serial.println();  delay(600);  Keyboard.print("USERNAME");  Keyboard.write(KEY\_TAB);  Keyboard.print("PASSWORD");  Keyboard.write(176);  }    else {  Serial.println(" Access denied");  delay(3000);  }  //============================================================================  } |