## **ABSTRACT**

section-heading

This project is developed by using Radio Frequency Identification (RFID) system and student card to get student attendance. Before this project, lecturers needed to use paper to get the student attendance. There were a lot of problems when using the paper as student attendance such as cheating. This project can help lecturers to reduce these problems by the design of an automatic attendance using RFID and student card. In recent years, RFID technology has been widely used in various sectors, such as in-education, transportation, agriculture, store sales and other sectors

RFID utilization in education is student attendance monitoring system by using Internet of Thing’s(IOT) and cloud technology, it willbe produce areal time attendance monitoring system that can be accessed by various parties, such as lecture ,campus, administration and parents. with this monitoring system if there are students who are not present can be immediately discovered and can be taken immediate action and the learningProcess can run smoothly. Taking traditional manual pen-paper attendance, which is very time-consuming, insecure and usually leads to human errors as well as prone to misconduct, as the valuable time and work gets wasted in organizing and structuring the attendance data in registers. Hence to overcome this major hectic problem, we have used relational database management system in real time with appropriate security measures to access, manipulate and represent the data on the basis of the unique RFID tags, which gets fast and easily scanned on the RFID reader.

This system consists of hardware and software with most trending implementation of a lightweight MQTT protocol in IoT technology; designed to take an attendance on the basis of RFID technology with Node MCU firmware. The main objective of this proposed system is to make the effective and efficient computerized attendance on institutes website in excel sheet format with a particular date and students can also see their particular subject attendance on the webpage.

**INTRODUCTION**

This Project is divided into 2 Parts

Part 1 : “Students Data Analysis” that is through IOT

Part 2 : “Digital log book entry”

**STUDENTS DATA ANALYSIS** is mainly focused on on Data gathering and record keeping of Student at dining hall

Each Student will be given an Rfid tag embedded ID card consists of their data such as Name, Registration number, Course, Branch, Contact etc.. They must scan their card every time they come for food. The same Rfid ID card can be used for other purposes such as for IN/OUT Data Logging at Security gate and Attendance in Classroom

**DIGITAL LOG BOOK ENTRY**

One of the factors that support the success of the learning system is the presence of students, because if students are often absent then the material they received becomes less and can’t understand the material that has been taught, one that causes the absence of students is they ditch, so the attendance monitoring system becomes very important because with this process the attendance of students can be monitored properly. Various kinds of presences can be used from manual way to automatic way using electronic equipment, if the monitoring system attendance using electronic equipment so the recording of attendance data becomes easier, fast and accurate.

In this research, attendance monitoring system using RFID technology and with the application of Internet of Things (IoT) and cloud technology will be obtained a presence system that can run and monitored in real time, so that all parties who need information such as lecturers, parents, and energy the administration can immediately find out if there are students who skip classroom, it can immediately be prevented so that the next meeting does not ditch again. Attendance management is one of the most crucial tasks in an educational institution- school, college, or university. Daily student attendance allows teachers to keep a tab on students’ activities.

It also enables parents to understand whether their children are attentive towards studies & attend their class regularly. However, managing the daily attendance of thousands of students is quite a challenging task for teachers. Traditional means of registering daily students’ attendance may result in errors & tremendous manual work. An RFID based attendance system can be a great solution to overcome such challenges as it automates the students’ attendance process & enables teachers and parents to track & monitor students’ activities effortlessly. Besides student attendance management, the RFID attendance system can also be used to track the attendance of faculty or staff to simplify the payroll management procedure.

The attendance system is an essential part of each organization. The time and attendance systems help to monitor the employees and students working and attending time. Attendance and attendance reporting programs can

help an employer monitor their employees’ working hours, early departures, late arrival, and time taken on breaks. Attendance is a crucial part of any organization for several

kinds of stuff. The student’s attendance is another significant factor used

for several purposes. By using the student attendance, educational institutions decide the examinations, scholarships, memberships, and much more. However, universities and other education sectors still using the traditional paper-based system. The problems in the traditional system are time-consuming, higher cost,interruption of pedagogical activities, and insecurity—almost a teacher spending more than 30 minutes for getting and storing student’s attendance. It is delaying and ruin the time of pedagogical activities.

The need of personal identification of a person for whom the book is being issued has gained its preference in the field of Library Management System (LMS). So, thus many libraries across the world have started to adopt and develop the RFID technology to identify and verify the person and book that being issued from library. Amongst all the other library management techniques, the RFID technologies have gained into importance. Almost all other techniques that have been proposed t

An automated library with the support of

RFID technology would be a “self service station” that demands least intervention by the

library personnel. Efforts are being made to introduce self-service “check-in” and “checkout”

that avoid long delay in the delivery of library material and also for achieving better

efficiency in operations.

It is used in libraries primarily to automate the book handling process including checkout,

inventory control, check-in, shelf management and anti-theft. When combined with

computer assisted sorting equipment, RFID facilitates and

**COMPONENTS REQURIED FOR THE SYSTEM**

**HARDWARE COMPONENTS**

**🡪**NODE MCU ESP8266

🡪RFID READER

🡪JUMPER WIRES

🡪USB CABLE

🡪BUZZER

**SOFTWARE COMPONENTS**

**🡪**AURDINO IDE

🡪GOOGLE SHEETS

🡪GOOGLE APP SCRIPTS

## **RFID based Digital Logbook**

RFID based digital logbook is an advanced attendance management system that has an RFID reader, RFID Tag, LCD display, a microcontroller unit that allows the wireless communications to establish the identity of students, faculty, or any other staff.

RFID attendance system is used to take attendance for student in school, college, and university. By placing their ID cards on the reader, students or workers can immediately verify their attendance.

RFID attendance system provides wireless identification of stakeholders when they fall in the radiofrequency range of the RFID attendance reader. To mark the attendance automatically, the students or staff need to carry the RFID tag that contains unique information about them such as name/ID number/class/section. The receiver/reader of the RFID attendance system automatically registers the attendance & saves the attendance data in the Microsoft Excel sheet of ERP system. The administrator can monitor lively and anytime they can extract the data to get a summary of student attendance history & keep a tab on them as well as faculty attendance for salary & payroll management.

## **BLOCK DIAGRAM**

section-heading

NodeMCU

LCD Display

RFID cards

Buzzer

RFID reader

Google SpreadSheet

# **NodeMCU ESP8266**

NodeMCU is an open-source Lua based firmware and **development board** specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.



🡪The ESP8266 is also hard to access and use. You must solder wires, with the appropriate analog voltage, to its pins for the simplest tasks such as powering it on or sending a keystroke to the “computer” on the chip.

## 🡪NodeMCU is an open-source Lua based firmware and **development board** specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

🡪 NodeMCU is an open-source LUA based firmware developed for the ESP8266 wifi chip. By exploring functionality with the ESP8266 chip, NodeMCU firmware comes with the ESP8266 Development board. However, as a chip, the ESP8266 is also hard to access and use.

🡪 You must solder wires, with the appropriate analog voltage, to its pins for the simplest tasks such as powering it on or sending a keystroke to the “computer” on the chip. You also have to program it in low-level machine instructions that can be interpreted by the chip hardware.

🡪 This level of integration is not a problem using the ESP8266 as an embedded controller chip in mass-produced electronics. It is a huge burden for hobbyists, hackers, or students who want to experiment with it in their own IoT projects.

## **Brief About NodeMCU ESP8266**

The **NodeMCU ESP8266 development board** comes with the ESP-12E module containing the ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects. NodeMCU can be powered using a Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.



|  |  |  |
| --- | --- | --- |
| **Pin Category** | **Name** | **Description** |
| Power | Micro-USB, 3.3V, GND, Vin | **Micro-USB:** NodeMCU can be powered through the USB port  **3.3V:** Regulated 3.3V can be supplied to this pin to power the board  **GND:** Ground pins  **Vin:**External Power Supply |
| Control Pins | **EN, RST** | The pin and the button resets the microcontroller |
| Analog Pin | A0 | Used to measure analog voltage in the range of 0-3.3V |
| GPIO Pins | GPIO1 to GPIO16 | NodeMCU has 16 general purpose input-output pins on its board |
| SPI Pins | SD1, CMD, SD0, CLK | NodeMCU has four pins available for SPI communication. |
| UART Pins | TXD0,RXD0, TXD2, RXD2 | NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program. |
| I2C Pins |  | NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C. |

### **NodeMCU ESP8266 Specifications & Features**

* Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
* Operating Voltage: 3.3V
* Input Voltage: 7-12V
* Digital I/O Pins (DIO): 16
* Analog Input Pins (ADC): 1
* UARTs: 1
* SPIs: 1
* I2Cs: 1
* Flash Memory: 4 MB
* SRAM: 64 KB
* Clock Speed: 80 MHz
* USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
* PCB Antenna
* Small Sized module to fit smartly inside your IoT projects

### **Applications**

* Prototyping of IoT devices
* Low power battery operated applications
* Network projects

Projects requiring multiple I/O interfaces with Wi-Fi and Bluetooth functionalities

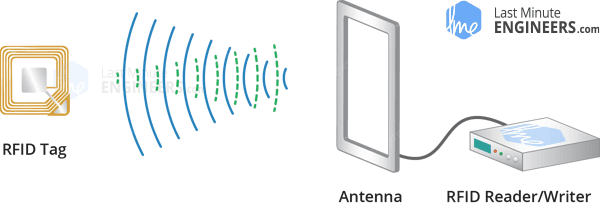
## **RFID TECHNOLOGY**

section-heading

## **What is RFID technology and how does it work?**

An [RFID](https://en.wikipedia.org/wiki/Radio-frequency_identification) or radio frequency identification system consists of two main components, a tag attached to the object to be identified, and a reader that reads the tag.

A reader consists of a radio frequency module and an antenna that generates a high frequency electromagnetic field. Whereas the tag is usually a passive device (it does not have a battery). It consists of a microchip that stores and processes information, and an antenna for receiving and transmitting a signal.



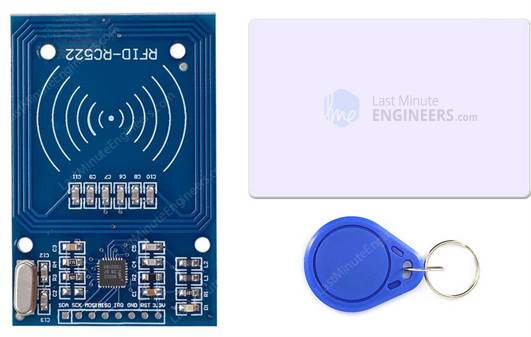
When the tag is brought close to the reader, the reader generates an electromagnetic field. This causes electrons to move through the tag’s antenna and subsequently powers the chip.

The chip then responds by sending its stored information back to the reader in the form of another radio signal. This is called a backscatter. The reader detects and interprets this backscatter and sends the data to a computer or microcontroller.

The RFID reader is a network-connected device that can be portable or permanently attached. It **uses radio waves to transmit signals that activate the tag.** **Once activated, the tag sends a wave back to the antenna, where it is translated into data**. The transponder is in the RFID tag itself.

## **Hardware Overview**

The RC522 RFID module based on the MFRC522 IC from NXP is one of the cheapest RFID options you can get. It usually comes with an RFID card tag and a key fob tag with 1KB of memory. And the best part is that it can write a tag that means you can store any message in it.



The RC522 RFID reader module is designed to create a 13.56MHz electromagnetic field and communicate with RFID tags (ISO 14443A standard tags).

The reader can communicate with a microcontroller over a 4-pin SPI with a maximum data rate of 10 Mbps. It also supports communication over I2C and UART protocols.

The RC522 RFID module can be programmed to generate an interrupt, allowing the module to alert us when a tag approaches it, instead of constantly asking the module “Is there a card nearby?”.

The RC522 RFID module is a smart card reader which, among other things, allows to activate a mechanism when the right card is presented to the reader.

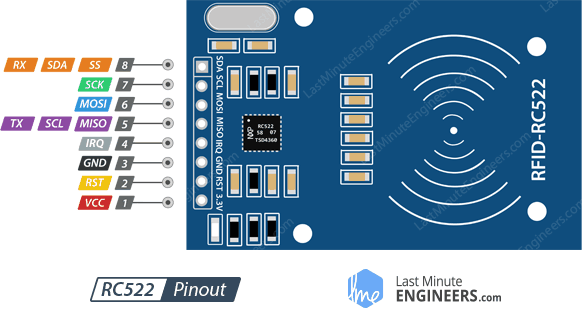
### **Technical Specifications**

Here are the specifications:

|  |  |
| --- | --- |
| Frequency Range | 13.56 MHz ISM Band |
| Host Interface | SPI / I2C / UART |
| Operating Supply Voltage | 2.5 V to 3.3 V |
| Max. Operating Current | 13-26mA |
| Min. Current(Power down) | 10µA |
| Logic Inputs | 5V Tolerant |
| Read Range | 5 cm |

## **RC522 RFID Module Pinout**

The RC522 module has a total of 8 pins that connect it to the outside world. The connections are as follows:



VCC supplies power to the module. This can be anywhere from 2.5 to 3.3 volts. You can connect it to the 3.3V output from your NodeMCU.

RST is an input for reset and power-down. When this pin goes low the module enters power-down mode. In which the oscillator is turned off and the input pins are disconnected from the outside world. Whereas the module is reset on the rising edge of the signal.

GND is the ground pin and needs to be connected to the G pin on the NodeMCU.

IRQ is an interrupt pin that alerts the microcontroller when an RFID tag is in the vicinity.

MISO / SCL / Tx pin acts as master-in-slave-out when SPI interface is enabled, as serial clock when I2C interface is enabled and as serial data output when the UART interface is enabled.

MOSI (Master Out Slave In) is the SPI input to the RC522 module.

SCK (Serial Clock) accepts the clock pulses provided by the SPI bus master i.e. Arduino.

SS / SDA / Rx pin acts as a signal input when the SPI interface is enabled, as serial data when the I2C interface is enabled and as a serial data input when the UART interface is enabled.

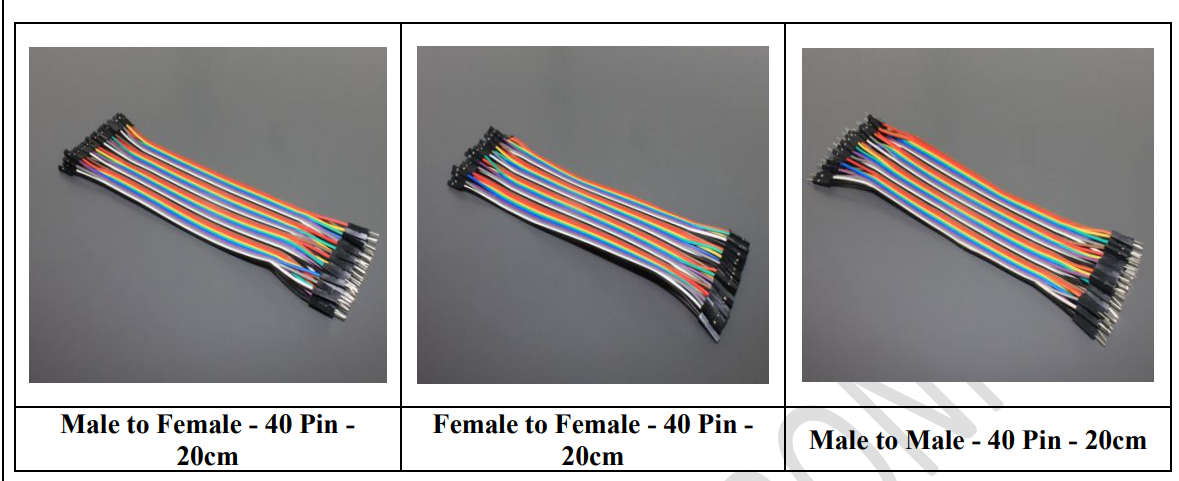
This pin is usually marked by encasing the pin in a square so that it can be used as a reference to identify other pins.

The RC522 RFID module uses the SPI protocol to communicate with the ESP8266. The SPI communication uses specific boxes on this type of microcontroller.

SS / SDA / Rx pin **acts as Signal input when SPI interface is enabled, acts as serial data when I2C interface is enabled and acts as serial data input when UART interface is enabled**

**JUMPER WIRES**

These premium jumper wires are good to wire harness or jumpering between headers on PCB's. These premium jumper wires are 8" (20cm) long and come in a strip of 40 wires (with 4 pieces of each of ten colors). They have 0.1" male header contacts on either end and fit cleanly next to each other on standard-pitch 0.1" (2.54mm) header. The wires come in a 40-pin ribbon cable which you can either keep them together to neatly organize, or pull the ribbon wires off to make individual jumpers. For best results, when plugging these in a line, have the sides with the 'silver latch bit' sticking out since that side is a tiny bit wider than 0.1"



FEATURES

• Female connectors are good for microcontrollers, arduino boards and sensor boards

• Male connectors for breadboard

• 40 different wires with 4 pieces of each of 10 different rainbow colors

• Standard 0.1" (2.54mm) pitch

• Each wire is 28 AWG

• Types : o Premium Female to Female Jumper Wires - 40 x 8" (200mm) They have 0.1" male header contacts on either end, and fit cleanly next to each other on standard-pitch 0.1" (2.54mm) header. o Premium Male to Male Jumper Wires - 40 x 8" (200mm) They have 0.1" male header contacts on either end and fit cleanly next to each other on standard-pitch 0.1" (2.54mm) header. o Premium Male to Female Jumper Wires - 40 x 8" (200mm) They have 0.1" male header contacts on one end and female header on the other end, and fit clea nly next to each other on standard-pitch 0.1" (2.54mm)

**Buzzer**

This buzzer also has an important role in this module. Whenever the user will scan his/her card then this buzzer beeps for a few seconds so that the one who scanned the card will know that his card is scanned properly. Without a buzzer, one can only assume the card has been detected but he/she will not be sure so the buzzer is important here



**USB CABLE:**

It is mainly used to connect nodemcu and the source code present in laptop. One meter long USB cable. It has the regular sized A plug on one end and a micro plug on the other. Ideal for new Arduino Uno or boards and node mcu, charging almost every new mobile phone except those from Apple, and more. Has the full five-wire configuration in a neat flat cable style.

****

either for data transmission or phone charging, this 1M Noodle Flat Wire Data Line V8 Micro USB to USB Charging Cable for Android Phone will bring you much convenience and help in daily life

**WIRING/CONNECTION**

The pinout is as follows (left side RC522, right side ESP8266):

Vcc <-> 3V3 (or Vin(5V) depending on the module version)

RST (Reset) <-> D3

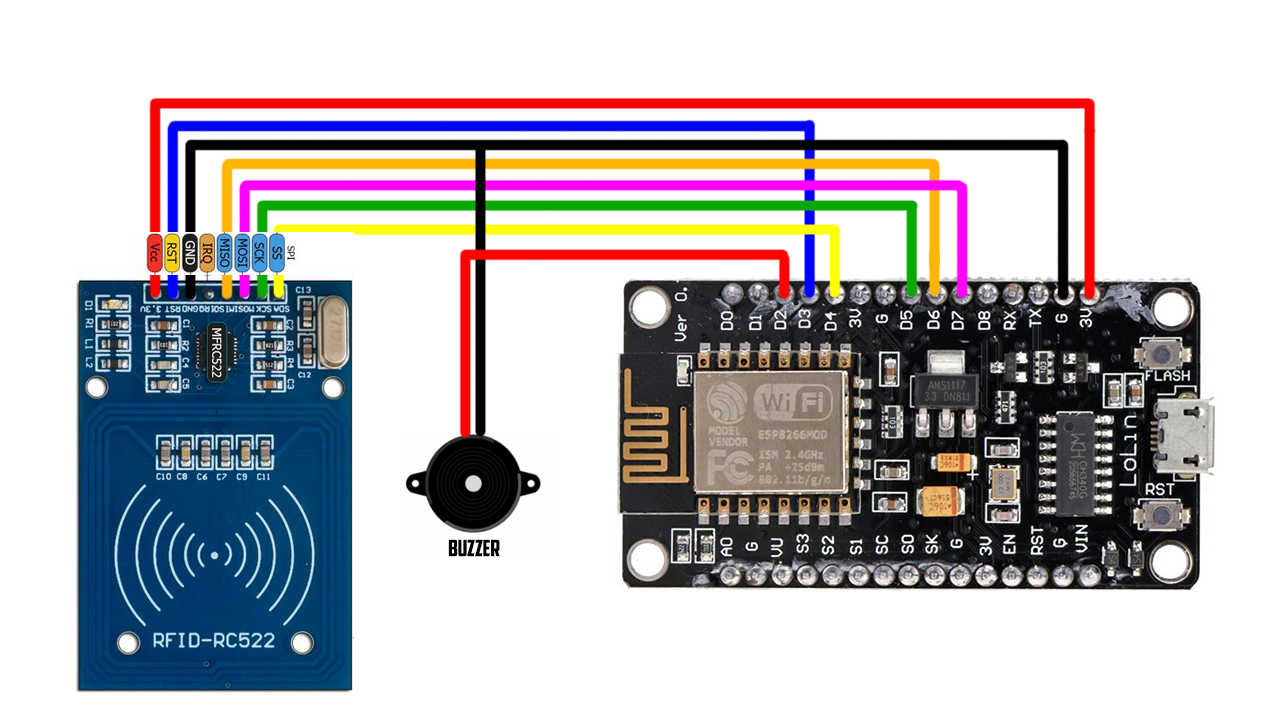
GND (Ground) <-> GND

MISO (Master Input Slave Output) <-> D6

MOSI (Master Output Slave Input) <-> D7

SCK (Serial Clock) <-> D5

SS/SDA (Slave select) <-> D4



**IMPLEMENTATION**

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on the Arduino board.

In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

**Step 1** − First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

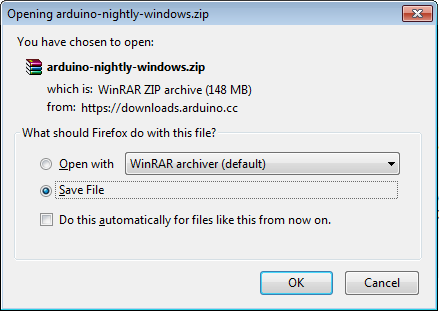


In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.



**Step 2 − Download Arduino IDE Software.**

You can get different versions of Arduino IDE from the [Download page](https://www.arduino.cc/en/Main/Software) on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



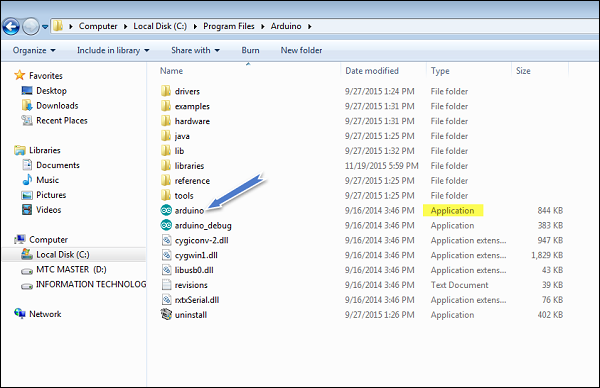
**Step 3 − Power up your board.**

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

**step 4 − Launch Arduino IDE.**

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

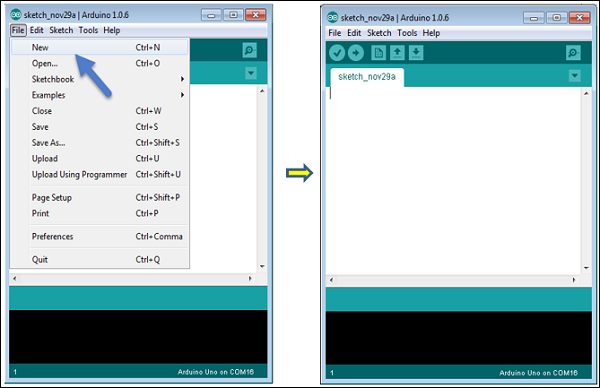


**Step 5 − Open your first project.**

Once the software starts, you have two options −

* Create a new project.
* Open an existing project example.

To create a new project, select File → **New**.



To open an existing project example, select File → Example → Basics → Blink.

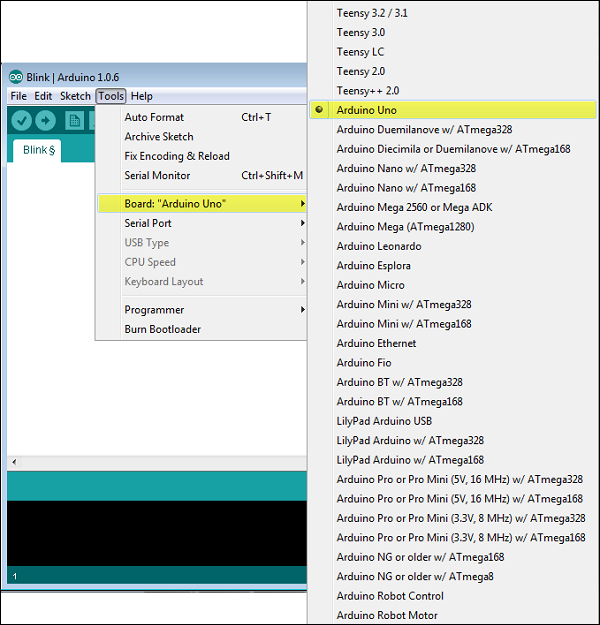
Open Project

Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

**Step 6 − Select your Arduino board.**

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

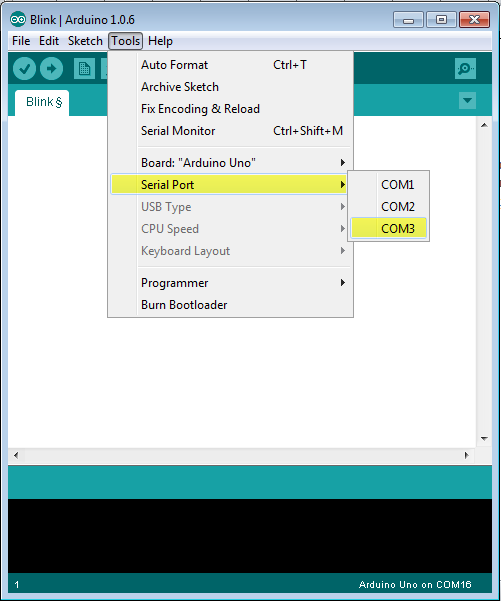
Go to Tools → Board and select your board.



Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

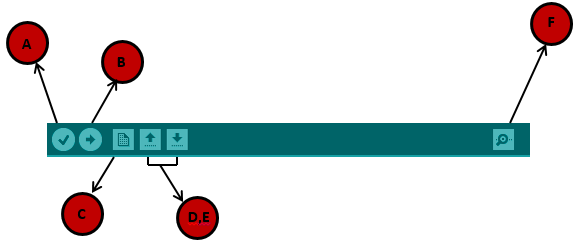
**Step 7 − Select your serial port.**

Select the serial device of the Arduino board. Go to **Tools → Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



**Step 8 − Upload the program to your board.**

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



**A** − Used to check if there is any compilation error.

**B** − Used to upload a program to the Arduino board.

**C** − Shortcut used to create a new sketch.

**D** − Used to directly open one of the example sketch.

**E** − Used to save your sketch.

**Writing Data to RFID Card using RC522 RFID and Arduino**

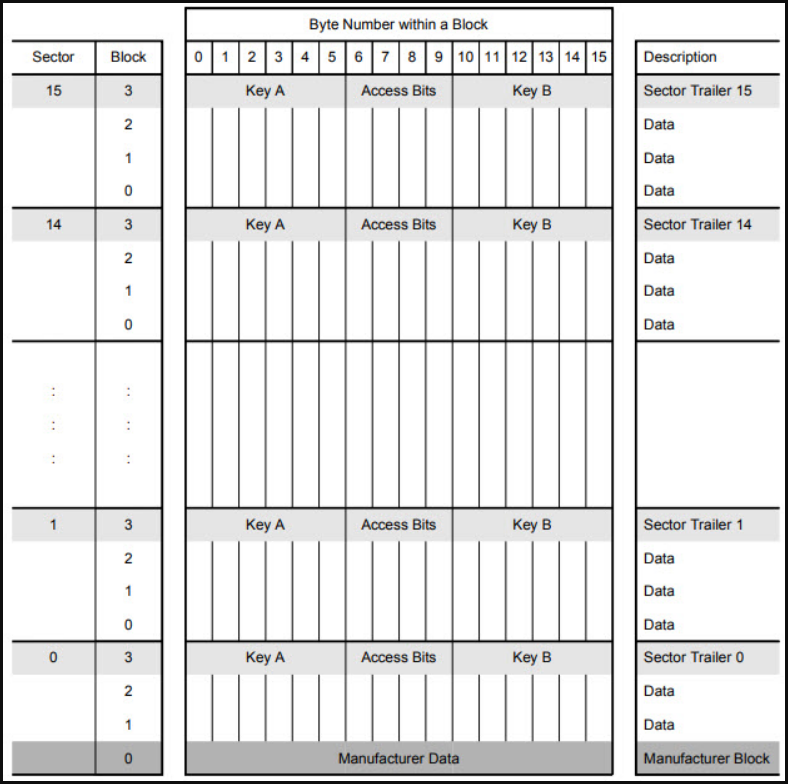
A typical MIFARE 1K RFID tag has 1K Byte of memory organized into 16 Sectors (Sector 0 to Sector 15). Each Sector consists of 4 Blocks.

**Understanding memory Map of MIFARE 1K Tag**

For example, Sector 0 has Blocks 0, 1, 2 and 3. Sector 1 has Blocks 4, 5, 6 and 7 and so on and finally Sector 15 has Blocks 60, 61, 62 and 63. Each Block can store 16 Bytes of data.

**NOTE:** This numbering is just to understand the memory layout.

So, 16 Sectors \* 4 Blocks \* 16 Bytes = 1024 Bytes = 1K



Block 0 of Sector 0 is reserved for storing Manufacturer Data. Usually, this Block contains 4 Byte UID (Unique ID) in case of MIFARE 1K Tags (and also MIFARE 4K, MIFARE Mini tags from NXP). Advanced Tags like MIFARE Plus, MIFARE Ultralight, MIFARE DESFire consists of a 7 Byte UID.

Each Sector consists of three Data Blocks, which can be used for storing user data. The last Block of each Sector i.e., Block 3 in case of Sector 0, Block 7 in case of Sector 1 and so on is known as Sector Trailer.

As there are 16 Sectors, there are 16 Sector Trailers. Each Sector Trailer consists of the following information:

* A mandatory 6 Byte Key A.
* 4 Bytes for Access Bits.
* Optional 6 Byte Key B (if not used, data can be stored).

## RC522-DumpInfo-Serial-4 **PROGRAM FOR DATA WRITING**

## **PROGRAM FOR DATA WRITING**

#include <SPI.h>

#include <MFRC522.h>

//--------------------------------------------------

//GPIO 0 --> D3

//GPIO 2 --> D4

const uint8\_t RST\_PIN = 0;

const uint8\_t SS\_PIN = 2;

//--------------------------------------------------

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

MFRC522::MIFARE\_Key key;

//--------------------------------------------------

/\* Be aware of Sector Trailer Blocks \*/

int blockNum = 4;

/\* Create array to read data from Block \*/

/\* Length of buffer should be 4 Bytes

more than the size of Block (16 Bytes) \*/

byte bufferLen = 18;

byte readBlockData[18];

//--------------------------------------------------

MFRC522::StatusCode status;

//--------------------------------------------------

void setup()

{

//------------------------------------------------------

//Initialize serial communications with PC

Serial.begin(9600);

//------------------------------------------------------

//Initialize SPI bus

SPI.begin();

//------------------------------------------------------

//Initialize MFRC522 Module

mfrc522.PCD\_Init();

Serial.println("Scan a MIFARE 1K Tag to write data...");

//------------------------------------------------------

}

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

\* loop() function

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

void loop()

{

//------------------------------------------------------------------------------

/\* Prepare the ksy for authentication \*/

/\* All keys are set to FFFFFFFFFFFFh at chip delivery from the factory \*/

for (byte i = 0; i < 6; i++){

key.keyByte[i] = 0xFF;

}

//------------------------------------------------------------------------------

/\* Look for new cards \*/

/\* Reset the loop if no new card is present on RC522 Reader \*/

if ( ! mfrc522.PICC\_IsNewCardPresent()){return;}

//------------------------------------------------------------------------------

/\* Select one of the cards \*/

if ( ! mfrc522.PICC\_ReadCardSerial()) {return;}

//------------------------------------------------------------------------------

Serial.print("\n");

Serial.println("\*\*Card Detected\*\*");

/\* Print UID of the Card \*/

Serial.print(F("Card 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);

}

Serial.print("\n");

/\* Print type of card (for example, MIFARE 1K) \*/

Serial.print(F("PICC type: "));

MFRC522::PICC\_Type piccType = mfrc522.PICC\_GetType(mfrc522.uid.sak);

Serial.println(mfrc522.PICC\_GetTypeName(piccType));

//------------------------------------------------------------------------------

byte buffer[18];

byte len;

//wait until 20 seconds for input from serial

Serial.setTimeout(20000L);

//MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM

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

Serial.println(F("Enter Student ID, ending with #"));

len = Serial.readBytesUntil('#', (char \*) buffer, 16);

//add empty spaces to the remaining bytes of buffer

for (byte i = len; i < 16; i++) buffer[i] = ' ';

blockNum = 4;

WriteDataToBlock(blockNum, buffer);

ReadDataFromBlock(blockNum, readBlockData);

dumpSerial(blockNum, readBlockData);

//MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM

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

Serial.println(F("Enter Full Name, ending with #"));

len = Serial.readBytesUntil('#', (char \*) buffer, 16);

for (byte i = len; i < 16; i++) buffer[i] = ' ';

blockNum = 5;

WriteDataToBlock(blockNum, buffer);

ReadDataFromBlock(blockNum, readBlockData);

dumpSerial(blockNum, readBlockData);

//MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM

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

Serial.println(F("Enter Course, ending with #"));

len = Serial.readBytesUntil('#', (char \*) buffer, 16);

for (byte i = len; i < 16; i++) buffer[i] = ' ';

blockNum = 6;

WriteDataToBlock(blockNum, buffer);

ReadDataFromBlock(blockNum, readBlockData);

dumpSerial(blockNum, readBlockData);

//MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM

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

Serial.println(F("Enter Branch, ending with #"));

len = Serial.readBytesUntil('#', (char \*) buffer, 16);

for (byte i = len; i < 16; i++) buffer[i] = ' ';

blockNum = 8;

WriteDataToBlock(blockNum, buffer);

ReadDataFromBlock(blockNum, readBlockData);

dumpSerial(blockNum, readBlockData);

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

Serial.println(F("Enter Phone Number, ending with #"));

len = Serial.readBytesUntil('#', (char \*) buffer, 16);

for (byte i = len; i < 16; i++) buffer[i] = ' ';

blockNum = 9;

WriteDataToBlock(blockNum, buffer);

ReadDataFromBlock(blockNum, readBlockData);

dumpSerial(blockNum, readBlockData);

}

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

\* Writ() function

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

void WriteDataToBlock(int blockNum, byte blockData[])

{

//Serial.print("Writing data on block ");

//Serial.println(blockNum);

//------------------------------------------------------------------------------

/\* Authenticating the desired data block for write access using Key A \*/

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

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

Serial.print("Authentication failed for Write: ");

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

return;

}

//------------------------------------------------------------------------------

else {

//Serial.print("Authentication OK - ");

}

//------------------------------------------------------------------------------

/\* Write data to the block \*/

status = mfrc522.MIFARE\_Write(blockNum, blockData, 16);

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

Serial.print("Writing to Block failed: ");

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

return;

}

else {

//Serial.println("Write OK");

}

//------------------------------------------------------------------------------

}

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

\* ReadDataFromBlock() function

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

void ReadDataFromBlock(int blockNum, byte readBlockData[])

{

//Serial.print("Reading data from block ");

//Serial.println(blockNum);

//----------------------------------------------------------------------------

/\* Prepare the ksy for authentication \*/

/\* All keys are set to FFFFFFFFFFFFh at chip delivery from the factory \*/

for (byte i = 0; i < 6; i++) {

key.keyByte[i] = 0xFF;

}

//------------------------------------------------------------------------------

/\* Authenticating the desired data block for Read access using Key A \*/

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

//------------------------------------------------------------------------------

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

Serial.print("Authentication failed for Read: ");

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

return;

}

else {

//Serial.print("Authentication OK - ");

}

//------------------------------------------------------------------------------

/\* Reading data from the Block \*/

status = mfrc522.MIFARE\_Read(blockNum, readBlockData, &bufferLen);

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

Serial.print("Reading failed: ");

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

return;

}

else {

//readBlockData[16] = ' ';

//readBlockData[17] = ' ';

//Serial.println("Read OK");

}

//------------------------------------------------------------------------------

}

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

\* dumpSerial() function

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

void dumpSerial(int blockNum, byte blockData[])

{

Serial.print("\n");

Serial.print("Data saved on block");

Serial.print(blockNum);

Serial.print(": ");

for (int j=0 ; j<16 ; j++){

Serial.write(readBlockData[j]);

}

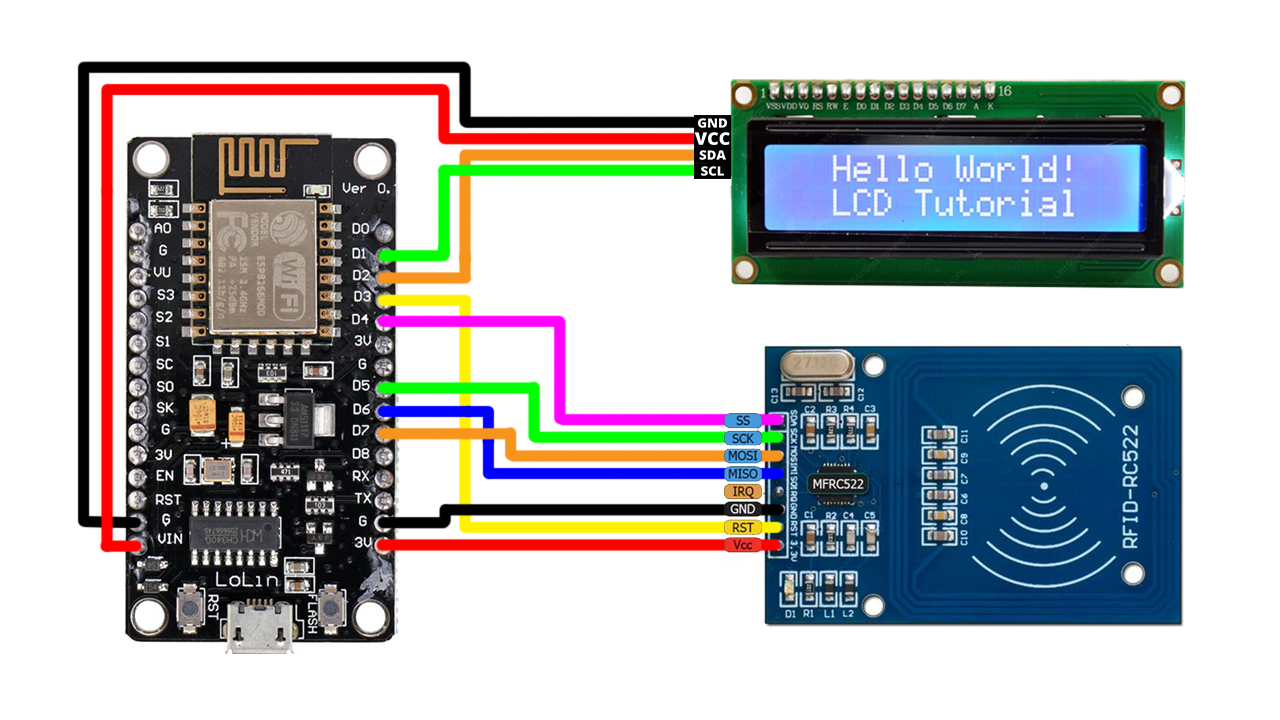
Serial.print("\n");

//Empty readBlockData array

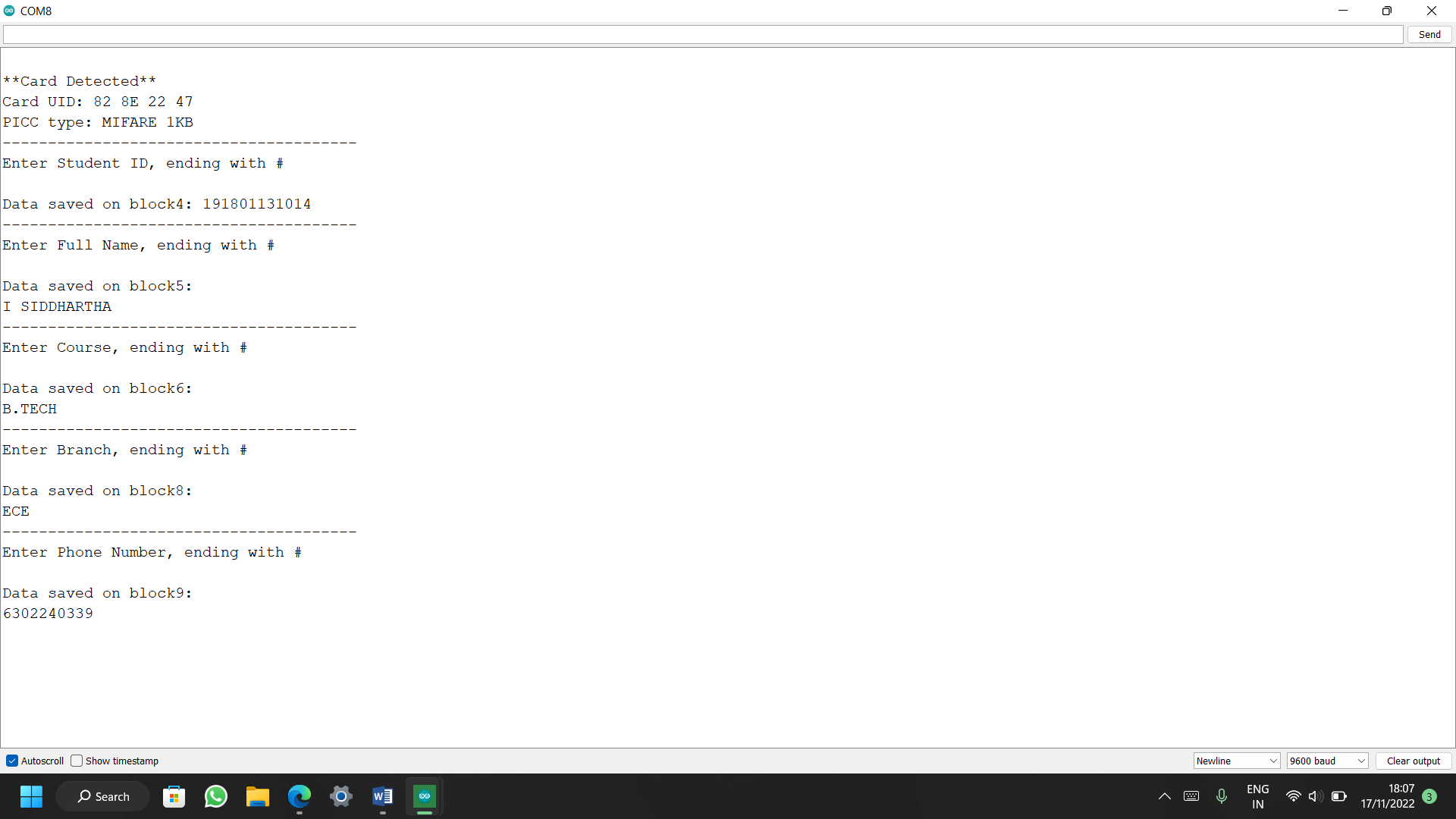
for( int i = 0; i < sizeof(readBlockData); ++i )

readBlockData[i] = (char)0; //empty space

}



## **RESULT**



Student ID has been stored in Block 4

Name has been stored in Block 5

Course has been stored in Block 6

Branch has been stored in Block 8

Phone Number has been stored in Block 9

## **Google Apps Script**

Google Apps Script is a cloud-based scripting language for extending the functionality of Google Apps and building lightweight cloud-based applications.

1. Create a New Spreadsheet in Google Sheets
2. Open its Apps Script
3. Update Script with the following & deploy

Script

// Enter Spreadsheet ID here

var SS = SpreadsheetApp.openById('1LIc24k9BhwzL6qGBuGE6qs1RLx4DzmgrsB4MU8rofwY');

var timezone = "Asia/Kolkata"

var hours = 5.30;

var str = "";

function doPost(e) {

  var parsedData;

  var result = {};

  try {

    parsedData = JSON.parse(e.postData.contents);

  }

  catch(f){

    return ContentService.createTextOutput("Error in parsing request body: " + f.message);

  }

  if (parsedData !== undefined){

    var flag = parsedData.format;

    if (flag === undefined){

      flag = 0;

    }

    var sheet = SS.getSheetByName(parsedData.sheet\_name); // sheet name to publish data to is specified in Arduino code

    var dataArr = parsedData.values.split(","); // creates an array of the values to publish

    //var date\_now = Utilities.formatDate(new Date(), "CST", "yyyy/MM/dd"); // gets the current date

    //var time\_now = Utilities.formatDate(new Date(), "CST", "hh:mm:ss a"); // gets the current time

    var Curr\_Date = new Date(new Date().setHours(new Date().getHours() + hours));

    var Curr\_Time = Utilities.formatDate(Curr\_Date, timezone, 'HH:mm:ss');

    var value0 = dataArr [0]; // value0 from Arduino code - Student ID

    var value1 = dataArr [1]; // value1 from Arduino code - First Name

    var value2 = dataArr [2]; // value2 from Arduino code - Last Name

    var value3 = dataArr [3]; // value0 from Arduino code - Phone Number

    var value4 = dataArr [4]; // value1 from Arduino code - Address

    // read and execute command from the "payload\_base" string specified in Arduino code

    switch (parsedData.command) {

      case "insert\_row":

         sheet.insertRows(2); // insert full row directly below header text

         //var range = sheet.getRange("A2:D2");              // use this to insert cells just above the existing data instead of inserting an entire row

         //range.insertCells(SpreadsheetApp.Dimension.ROWS); // use this to insert cells just above the existing data instead of inserting an entire row

         sheet.getRange('A2').setValue(Curr\_Date); // publish current date to cell A2

         sheet.getRange('B2').setValue(Curr\_Time); // publish current time to cell B2

         sheet.getRange('C2').setValue(value0);   // publish Student ID from Arduino code to cell C2

         sheet.getRange('D2').setValue(value1);   // publish First Name from Arduino code to cell D2

         sheet.getRange('E2').setValue(value2);   // publish Last Name from Arduino code to cell E2

         sheet.getRange('F2').setValue(value3);   // publish Phone Number from Arduino code to cell F2

         sheet.getRange('G2').setValue(value4);   // publish Address from Arduino code to cell G2

         str = "Success"; // string to return back to Arduino serial console

         SpreadsheetApp.flush();

         break;

      case "append\_row":

         var publish\_array = new Array(); // create a new array

         publish\_array [0] = date\_now; // add current date to position 0 in publish\_array

         publish\_array [1] = time\_now; // add current time to position 1 in publish\_array

         publish\_array [2] = value0;   // add value0 from Arduino code to position 2 in publish\_array

         publish\_array [3] = value1;   // add value1 from Arduino code to position 3 in publish\_array

         publish\_array [4] = value2;   // add value2 from Arduino code to position 4 in publish\_array

         sheet.appendRow(publish\_array); // publish data in publish\_array after the last row of data in the sheet

         str = "Success"; // string to return back to Arduino serial console

         SpreadsheetApp.flush();

         break;

    }

    return ContentService.createTextOutput(str);

  } // endif (parsedData !== undefined)

  else {

    return ContentService.createTextOutput("Error! Request body empty or in incorrect format.");

  }

}

## **SOURCE CODE**

#include <Arduino.h>

#include <ESP8266WiFi.h>

#include <SPI.h>

#include <MFRC522.h>

#include <HTTPSRedirect.h>

#include<Wire.h>

//LCD RS = 10; EN = 9; 2,0,4,5

//#include<LiquidCrystal\_I2C.h>

//LiquidCrystal\_I2C lcd(0x27, 16, 2);

//---------------------------------------------------------------------------------------------------------

// Enter Google Script Deployment ID:

const char \*GScriptId = "AKfycbwiLf4vHqiuO7gdJH9I07oKzEsZHBIYOEQBphB5\_6IY2YGhbTpA1D-hj9FZ0waQfkHT2Q";

//---------------------------------------------------------------------------------------------------------

// Enter network credentials:

const char\* ssid = "Siddhu";

const char\* password = "cheppanupoo";

//---------------------------------------------------------------------------------------------------------

// Enter command (in sert\_row or append\_row) and your Google Sheets sheet name (default is Sheet1):

String payload\_base = "{\"command\": \"insert\_row\", \"sheet\_name\": \"Sheet1\", \"values\": ";

String payload = "";

//---------------------------------------------------------------------------------------------------------

// Google Sheets setup (do not edit)

const char\* host = "script.google.com";

const int httpsPort = 443;a

const char\* fingerprint = "";

String url = String("/macros/s/") + GScriptId + "/exec";

HTTPSRedirect\* client = nullptr;

//------------------------------------------------------------

// Declare variables that will be published to Google Sheets

String student\_id;

//------------------------------------------------------------

int blocks[] = {4,5,6,8,9,10};

#define total\_blocks (sizeof(blocks) / sizeof(blocks[0]))

//------------------------------------------------------------

#define RST\_PIN 0 //D3

#define SS\_PIN 2 //D4

//#define BUZZER 4 //D2

//------------------------------------------------------------

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

MFRC522::MIFARE\_Key key;

MFRC522::StatusCode status;

//------------------------------------------------------------

/\* Be aware of Sector Trailer Blocks \*/

int blockNum = 2;

/\* Create another array to read data from Block \*/

/\* Legthn of buffer should be 2 Bytes more than the size of Block (16 Bytes) \*/

byte bufferLen = 18;

byte readBlockData[18];

//------------------------------------------------------------

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

\* setup Function

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

void setup() {

//----------------------------------------------------------

Serial.begin(9600);

delay(10);

Serial.println('\n');

//----------------------------------------------------------

SPI.begin();

//----------------------------------------------------------

//initialize lcd screen

// lcd.init();

// turn on the backlight

// lcd.backlight();

//lcd.clear();

//lcd.setCursor(0,0); //col=0 row=0

//lcd.print("Connecting to");

//lcd.setCursor(0,1); //col=0 row=0

//lcd.print("WiFi...");

//----------------------------------------------------------

// Connect to WiFi

WiFi.begin(ssid, password);

Serial.print("Connecting to ");

Serial.print(ssid); Serial.println(" ...");

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.print(".");

}

Serial.println('\n');

Serial.println("Connection established!");

Serial.print("IP address:\t");

Serial.println(WiFi.localIP());

//----------------------------------------------------------

// Use HTTPSRedirect class to create a new TLS connection

client = new HTTPSRedirect(httpsPort);

client->setInsecure();

client->setPrintResponseBody(true);

client->setContentTypeHeader("application/json");

//----------------------------------------------------------

//lcd.clear();

//lcd.setCursor(0,0); //col=0 row=0

//lcd.print("Connecting to");

//lcd.setCursor(0,1); //col=0 row=0

//lcd.print("Google ");

//delay(5000);

//----------------------------------------------------------

Serial.print("Connecting to ");

Serial.println(host);

//----------------------------------------------------------

// Try to connect for a maximum of 5 times

bool flag = false;

for(int i=0; i<5; i++){

int retval = client->connect(host, httpsPort);

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

if (retval == 1){

flag = true;

String msg = "Connected. OK";

Serial.println(msg);

// lcd.clear();

//lcd.setCursor(0,0); //col=0 row=0

//lcd.print(msg);

//delay(2000);

//break;

}

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

else

Serial.println("Connection failed. Retrying...");

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

}

//----------------------------------------------------------

if (!flag){

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//lcd.clear();

//lcd.setCursor(0,0); //col=0 row=0

//lcd.print("Connection fail");

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Serial.print("Could not connect to server: ");

Serial.println(host);

delay(5000);

return;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

//----------------------------------------------------------

delete client; // delete HTTPSRedirect object

client = nullptr; // delete HTTPSRedirect object

//----------------------------------------------------------

}

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

\* loop Function

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

void loop() {

//----------------------------------------------------------------

static bool flag = false;

if (!flag){

client = new HTTPSRedirect(httpsPort);

client->setInsecure();

flag = true;

client->setPrintResponseBody(true);

client->setContentTypeHeader("application/json");

}

if (client != nullptr){

if (!client->connected())

{client->connect(host, httpsPort);}

}

else{Serial.println("Error creating client object!");}

//----------------------------------------------------------------

//lcd.clear();

//lcd.setCursor(0,0); //col=0 row=0

//lcd.print("Scan your Tag");

/\* Initialize MFRC522 Module \*/

mfrc522.PCD\_Init();

/\* Look for new cards \*/

/\* Reset the loop if no new card is present on RC522 Reader \*/

if ( ! mfrc522.PICC\_IsNewCardPresent()) {return;}

/\* Select one of the cards \*/

if ( ! mfrc522.PICC\_ReadCardSerial()) {return;}

/\* Read data from the same block \*/

Serial.println();

Serial.println(F("Reading last data from RFID..."));

//----------------------------------------------------------------

String values = "", data;

/\*

//creating payload - method 1

//----------------------------------------------------------------

ReadDataFromBlock(blocks[0], readBlockData); //student id

data = String((char\*)readBlockData); data.trim();

student\_id = data;

//----------------------------------------------------------------

ReadDataFromBlock(blocks[1], readBlockData); //first name

data = String((char\*)readBlockData); data.trim();

first\_name = data;

//----------------------------------------------------------------

ReadDataFromBlock(blocks[2], readBlockData); //last name

data = String((char\*)readBlockData); data.trim();

last\_name = data;

//----------------------------------------------------------------

ReadDataFromBlock(blocks[3], readBlockData); //phone number

data = String((char\*)readBlockData); data.trim();

phone\_number = data;

//----------------------------------------------------------------

ReadDataFromBlock(blocks[4], readBlockData); //address

data = String((char\*)readBlockData); data.trim();

address = data; data = "";

//----------------------------------------------------------------

values = "\"" + student\_id + ",";

values += first\_name + ",";

values += last\_name + ",";

values += phone\_number + ",";

values += address + "\"}";

//----------------------------------------------------------------\*/

//creating payload - method 2 - More efficient

for (byte i = 0; i < total\_blocks; i++) {

ReadDataFromBlock(blocks[i], readBlockData);

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

if(i == 0){

data = String((char\*)readBlockData);

data.trim();

student\_id = data;

values = "\"" + data + ",";

}

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

else if(i == total\_blocks-1){

data = String((char\*)readBlockData);

data.trim();

values += data + "\"}";

}

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

else{

data = String((char\*)readBlockData);

data.trim();

values += data + ",";

}

}

//----------------------------------------------------------------

// Create json object string to send to Google Sheets

// values = "\"" + value0 + "," + value1 + "," + value2 + "\"}"

payload = payload\_base + values;

//----------------------------------------------------------------

// lcd.clear();

// lcd.setCursor(0,0); //col=0 row=0

//lcd.print("Publishing Data");

// lcd.setCursor(0,1); //col=0 row=0

//lcd.print("Please Wait...");

//----------------------------------------------------------------

// Publish data to Google Sheets

Serial.println("Publishing data...");

Serial.println(payload);

if(client->POST(url, host, payload)){

// do stuff here if publish was successful

// lcd.clear();

//lcd.setCursor(0,0); //col=0 row=0

//lcd.print("Student ID: "+student\_id);

//lcd.setCursor(0,1); //col=0 row=0

//lcd.print("Present");

}

//----------------------------------------------------------------

else{

// do stuff here if publish was not successful

Serial.println("Error while connecting");

//lcd.clear();

//lcd.setCursor(0,0); //col=0 row=0

//lcd.print("Failed.");

//lcd.setCursor(0,1); //col=0 row=0

//lcd.print("Try Again");

}

//----------------------------------------------------------------

// a delay of several seconds is required before publishing again

delay(5000);

}

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

\* ReadDataFromBlock() function

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

void ReadDataFromBlock(int blockNum, byte readBlockData[])

{

//----------------------------------------------------------------------------

/\* Prepare the ksy for authentication \*/

/\* All keys are set to FFFFFFFFFFFFh at chip delivery from the factory \*/

for (byte i = 0; i < 6; i++) {

key.keyByte[i] = 0xFF;

}

//----------------------------------------------------------------------------

/\* Authenticating the desired data block for Read access using Key A \*/

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

//----------------------------------------------------------------------------s

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

Serial.print("Authentication failed for Read: ");

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

return;

}

//----------------------------------------------------------------------------

else {

Serial.println("Authentication success");

}

//----------------------------------------------------------------------------

/\* Reading data from the Block \*/

status = mfrc522.MIFARE\_Read(blockNum, readBlockData, &bufferLen);

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

Serial.print("Reading failed: ");

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

return;

}

//----------------------------------------------------------------------------

else {

readBlockData[16] = ' ';

readBlockData[17] = ' ';

Serial.println("Block was read successfully");

}

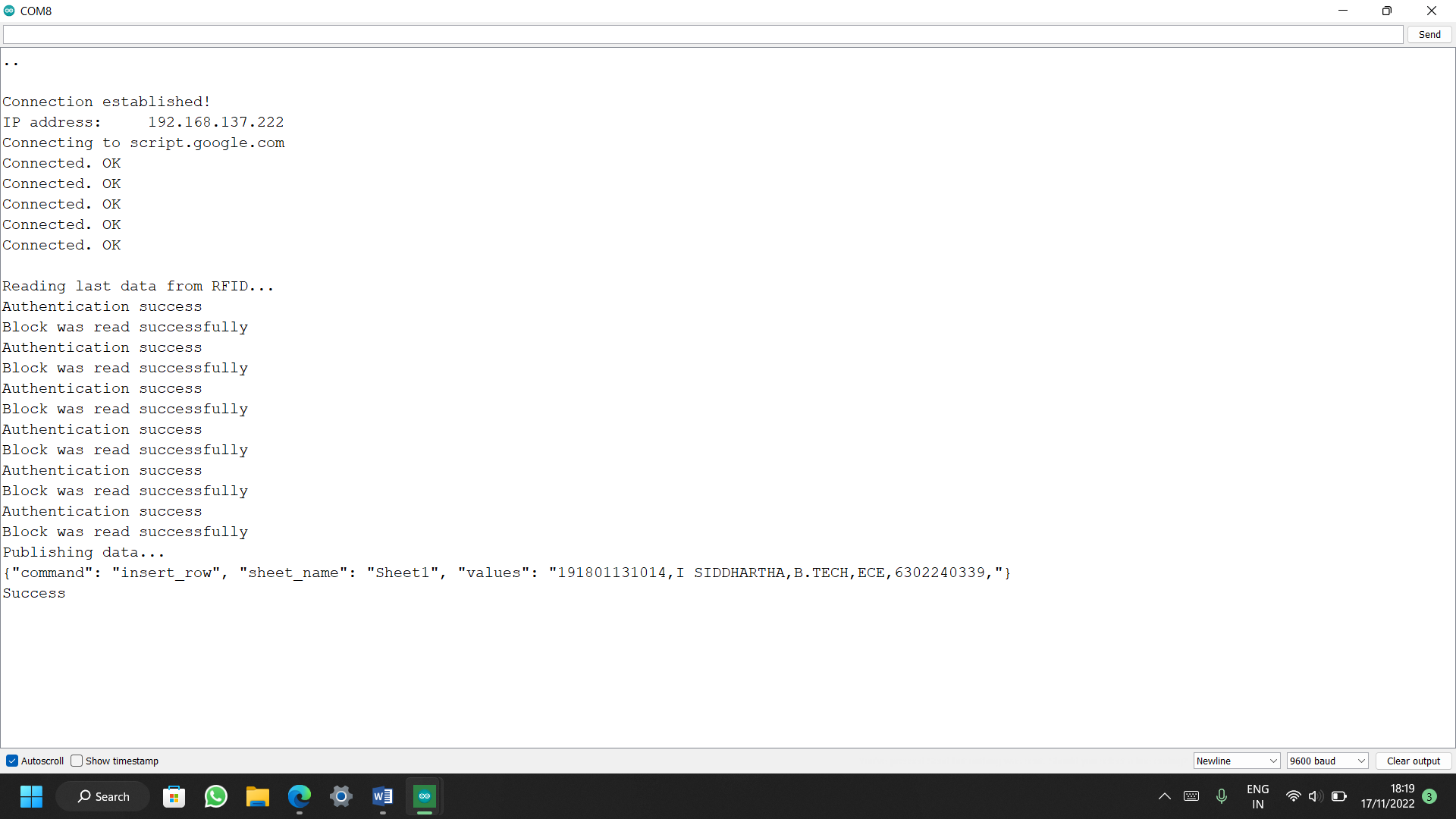
//----------------------------------------------------------------------------

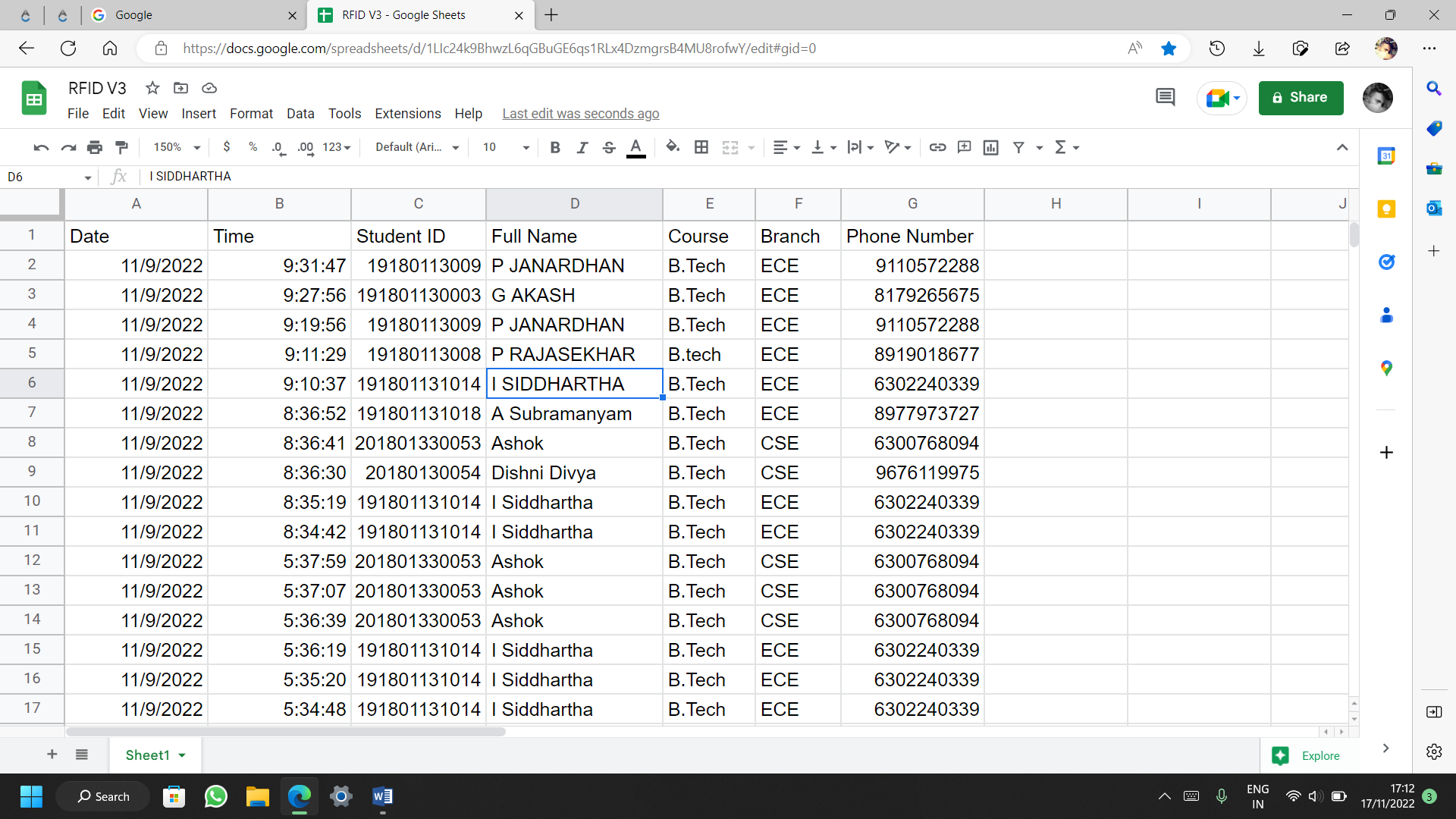
}

## **BILL OF MATERIALS**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Name | Specification | Quantity |
| 1 | microcontroller with board | NodeMCU | 1 |
| 2 | RFID Module | RC522 | 1 |
| 3 | RFID Cards | RC522 compatible | Depends on no of items |
| 4 | USB A – Micro B Cable | NodeMCU  Compatible | 1 |
| 5 | Liquid Crystal Display | 16x2 | 1 |
| 6 | Buzzer | Piezo 5V | 1 |
| 7 | Jumper wires | Female to Male | 30 |

## **RESULT**





The above Google Sheet can be downloaded and need to be manipulated accordingly for using as dataframe for Machine Learning. Label : **ds** is Date & **y** is No of Students scanned their RFID cards on that particular day

