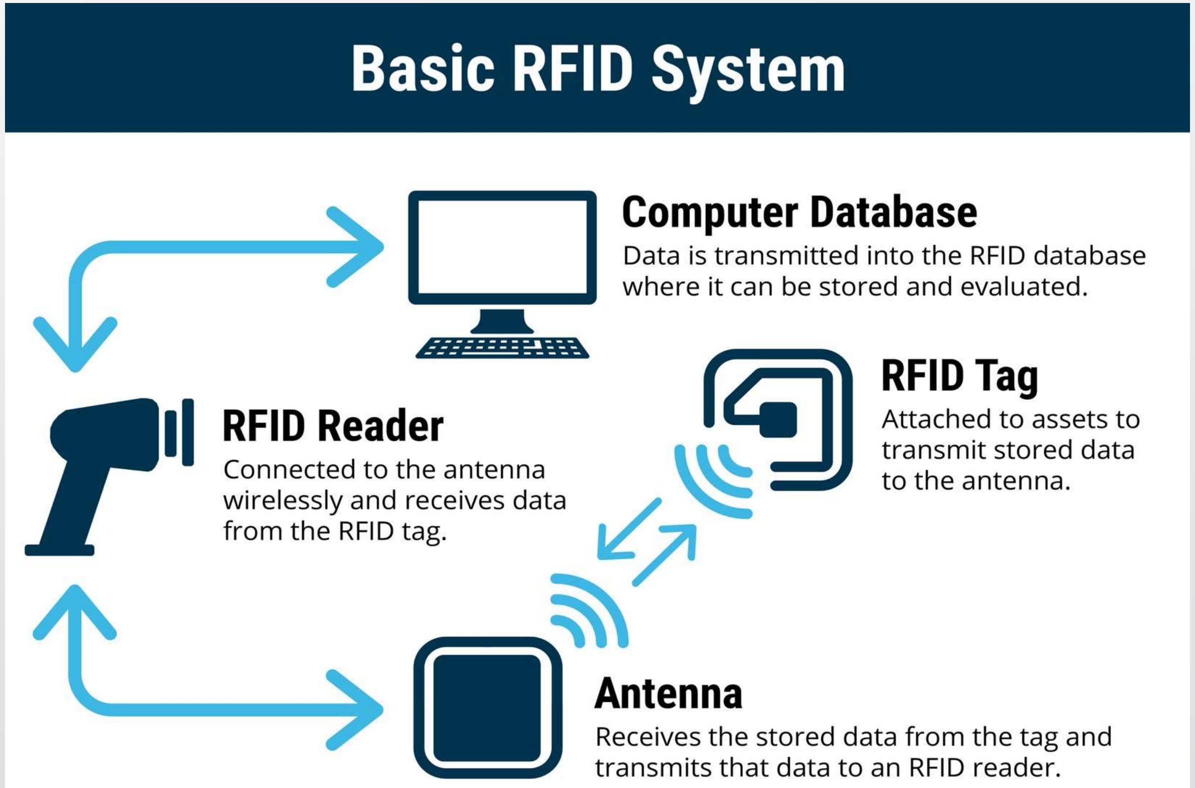
IOT AND CLOUD COMPUTING LAB



Basic RFID System

Computer Database
Data is transmitted into the RFID database where it can be stored and evaluated.

RFID Reader
Connected to the antenna wirelessly and receives data from the RFID tag.

RFID Tag
Attached to assets to transmit stored data to the antenna.

Antenna
Receives the stored data from the tag and transmits that data to an RFID reader.

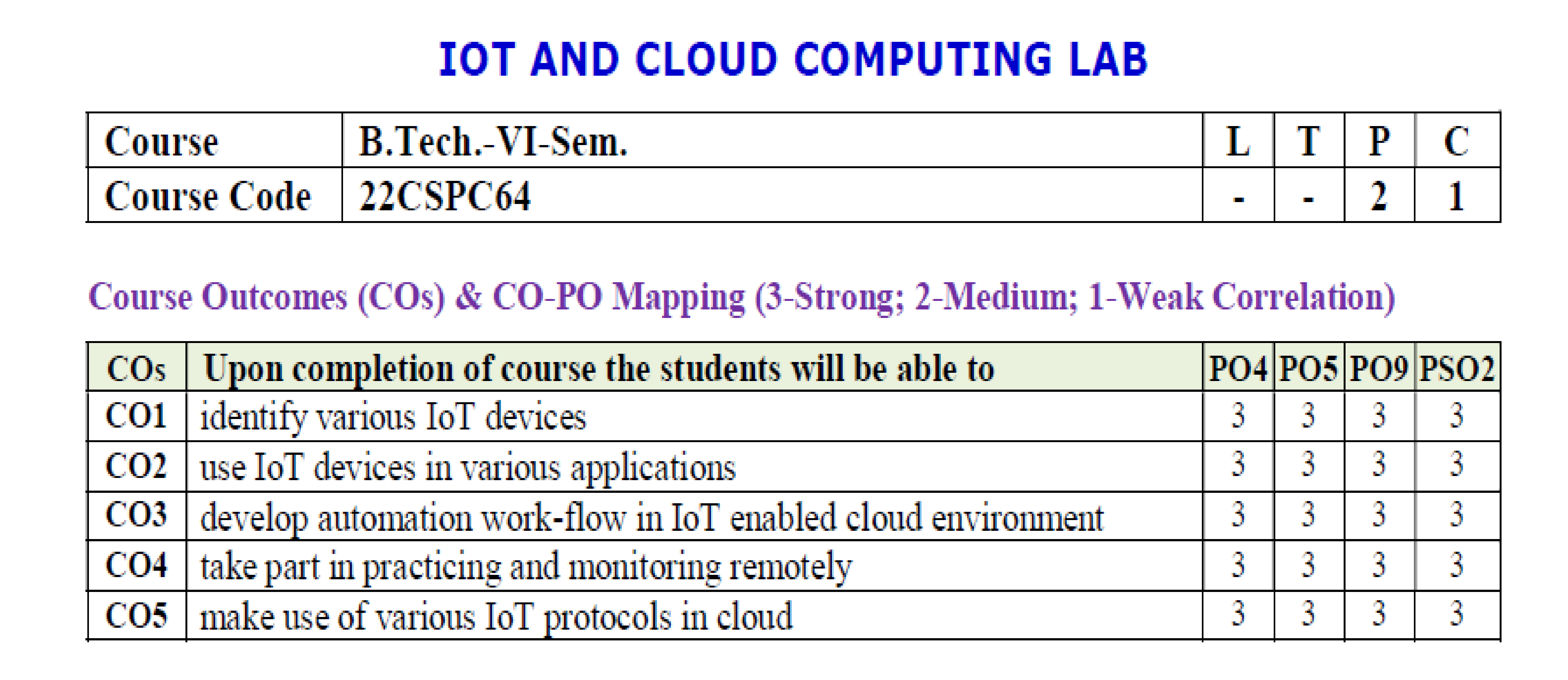
By,

Asst. Prof. S.Asra

M.E(Computer science & Engineering)

B.E(Computer Science & Engineering)

Diploma(Computer Science & Engineering)

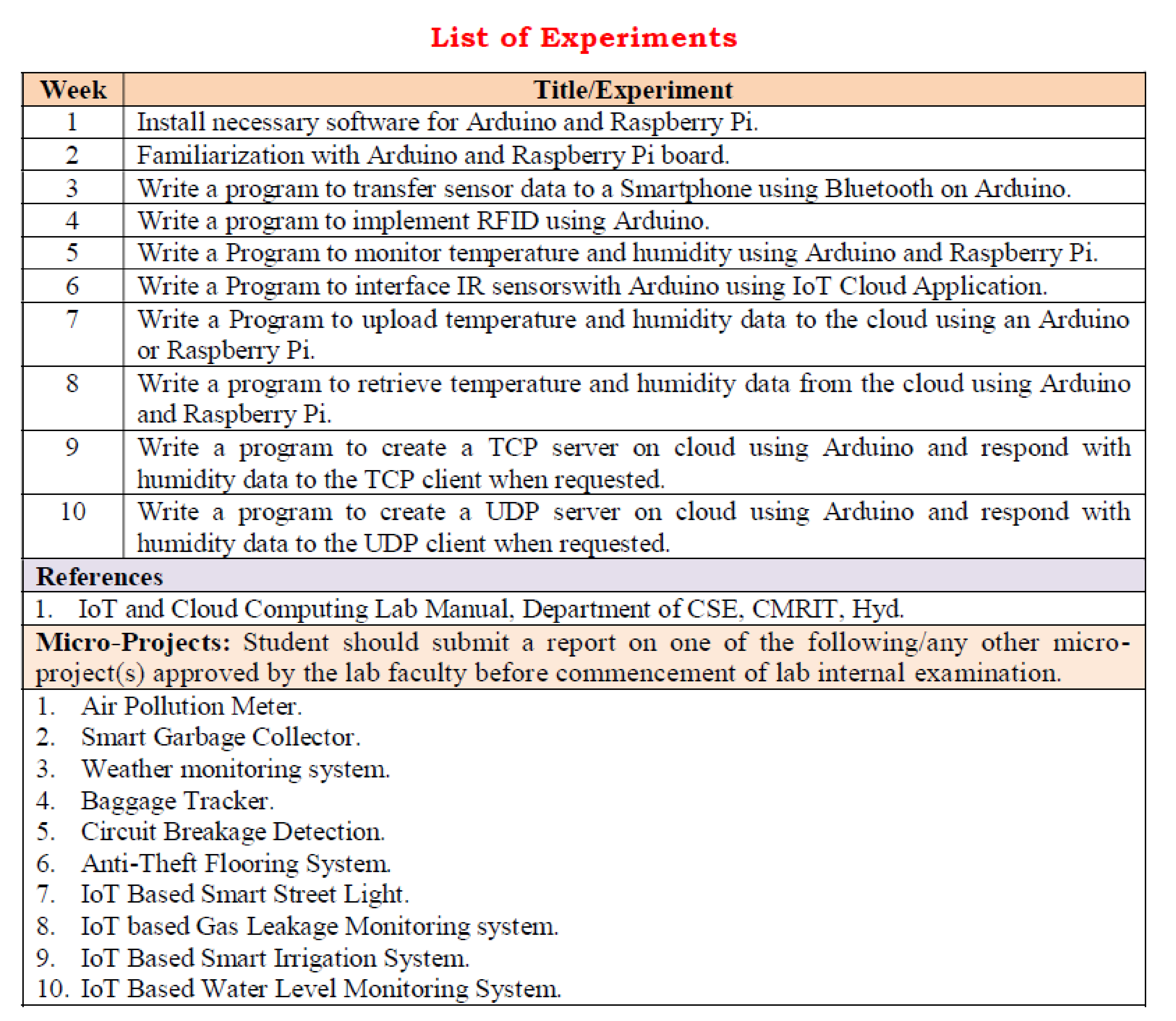


IOT AND CLOUD COMPUTING LAB

Course	B.Tech.-VI-Sem.	L	T	P	C
Course Code	22CSPC64	-	-	2	1

Course Outcomes (COs) & CO-PO Mapping (3-Strong; 2-Medium; 1-Weak Correlation)

COs	Upon completion of course the students will be able to	PO4	PO5	PO9	PSO2
CO1	identify various IoT devices	3	3	3	3
CO2	use IoT devices in various applications	3	3	3	3
CO3	develop automation work-flow in IoT enabled cloud environment	3	3	3	3
CO4	take part in practicing and monitoring remotely	3	3	3	3
CO5	make use of various IoT protocols in cloud	3	3	3	3



List of Experiments

Week	Title/Experiment
1	Install necessary software for Arduino and Raspberry Pi.
2	Familiarization with Arduino and Raspberry Pi board.
3	Write a program to transfer sensor data to a Smartphone using Bluetooth on Arduino.
4	Write a program to implement RFID using Arduino.
5	Write a Program to monitor temperature and humidity using Arduino and Raspberry Pi.
6	Write a Program to interface IR sensorswith Arduino using IoT Cloud Application.
7	Write a Program to upload temperature and humidity data to the cloud using an Arduino or Raspberry Pi.
8	Write a program to retrieve temperature and humidity data from the cloud using Arduino and Raspberry Pi.
9	Write a program to create a TCP server on cloud using Arduino and respond with humidity data to the TCP client when requested.
10	Write a program to create a UDP server on cloud using Arduino and respond with humidity data to the UDP client when requested.

References

1. IoT and Cloud Computing Lab Manual, Department of CSE, CMRIT, Hyd.

Micro-Projects: Student should submit a report on one of the following/any other micro-project(s) approved by the lab faculty before commencement of lab internal examination.

1. Air Pollution Meter.
2. Smart Garbage Collector.
3. Weather monitoring system.
4. Baggage Tracker.
5. Circuit Breakage Detection.
6. Anti-Theft Flooring System.
7. IoT Based Smart Street Light.
8. IoT based Gas Leakage Monitoring system.
9. IoT Based Smart Irrigation System.
10. IoT Based Water Level Monitoring System.

atabase

ed into the RFID database
ored and evaluated.

RFID Tag

Attached to assets to transmit stored data to the antenna.

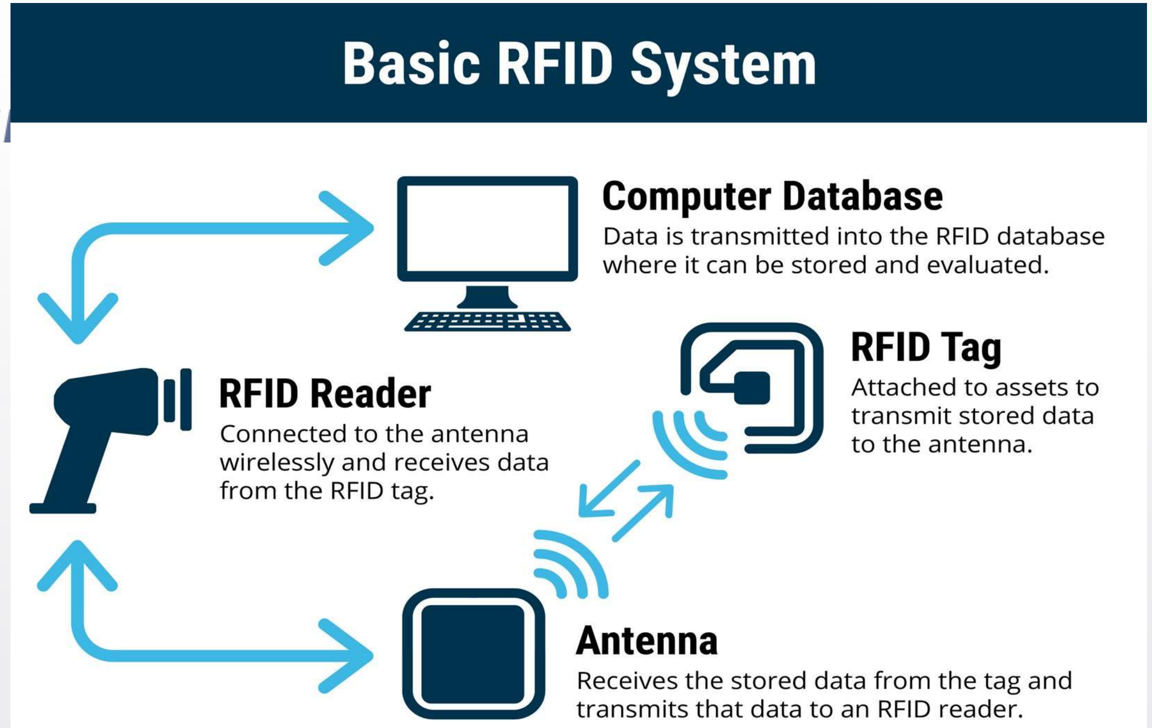
WEEK 4

AIM: Write a program to implement RFID using Arduino.

What is RFID technology and how does it work?

An RFID or radio frequency identification system consists of
two main components,

1) a tag attached to the object to be identified, and
2) a reader that reads the tag.



Basic RFID System

Computer Database
Data is transmitted into the RFID database where it can be stored and evaluated.

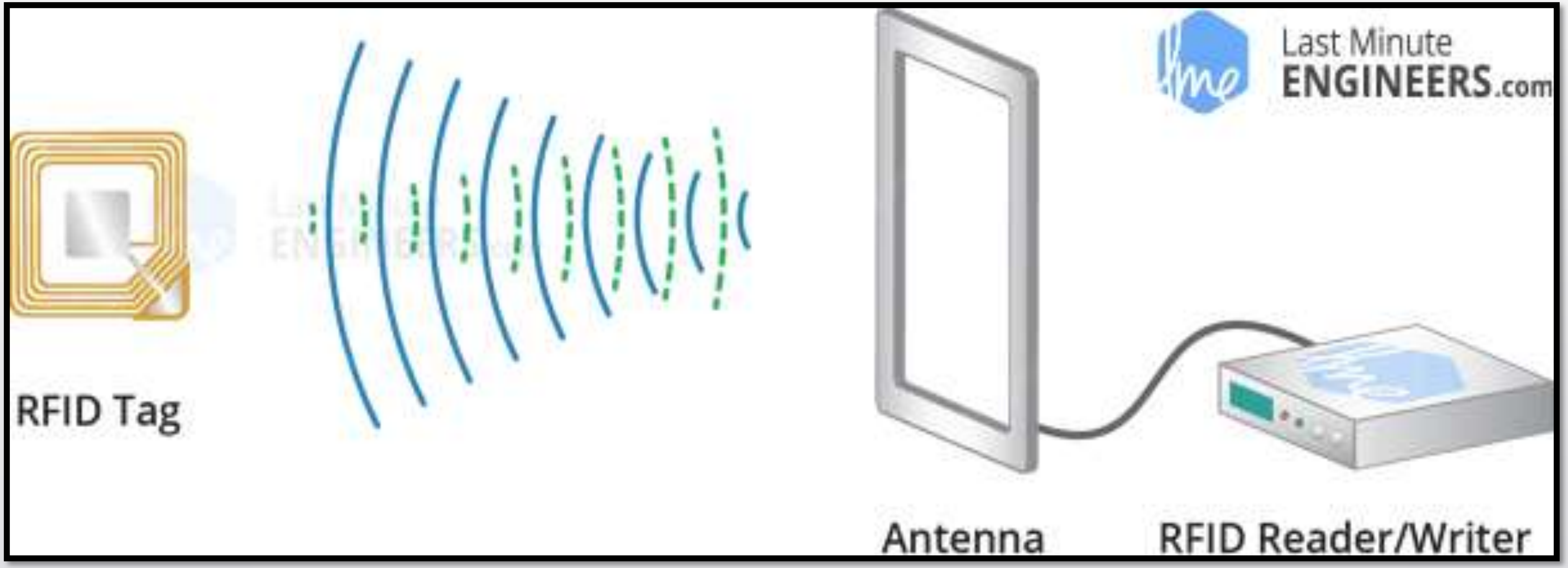
RFID Tag
Attached to assets to transmit stored data to the antenna.

RFID Reader
Connected to the antenna wirelessly and receives data from the RFID tag.

Antenna
Receives the stored data from the tag and transmits that data to an RFID reader.

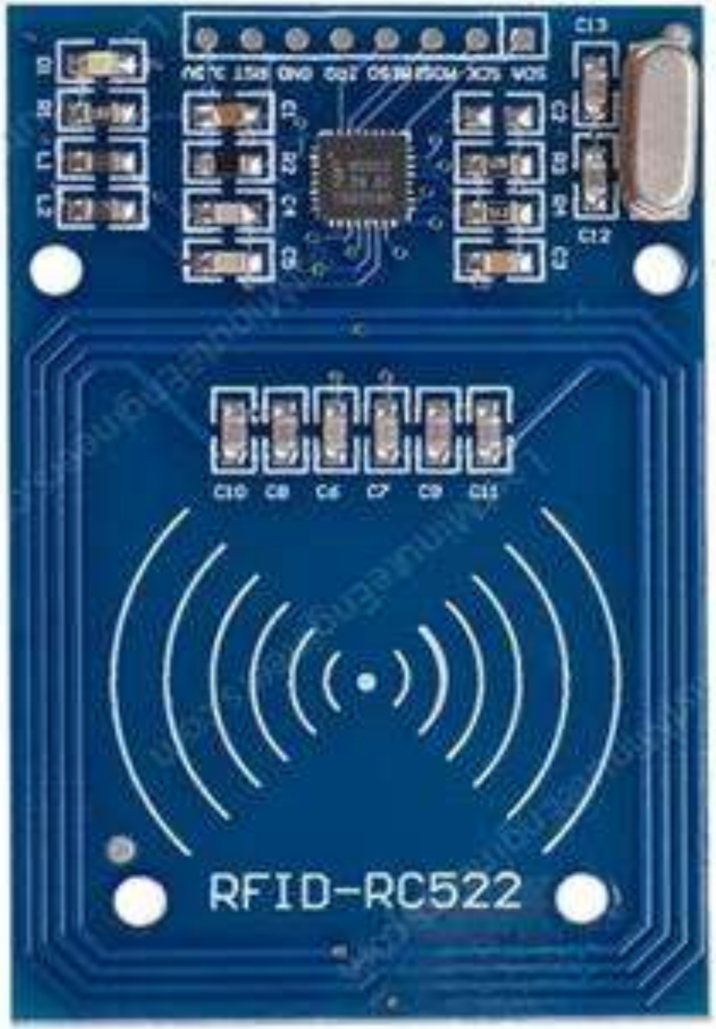
- **A reader consists of a radio frequency module and**
- **an antenna that generates a high frequency electromagnetic field.**
  an antenna for receiving and transmitting a signal.
- Whereas the tag is usually a passive device (it does not have a battery).
  It consists of a microchip that stores and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

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.


Last Minute ENGINEERS.com
RFID Tag
Antenna
RFID Reader/Writer

RFID key FOB tag

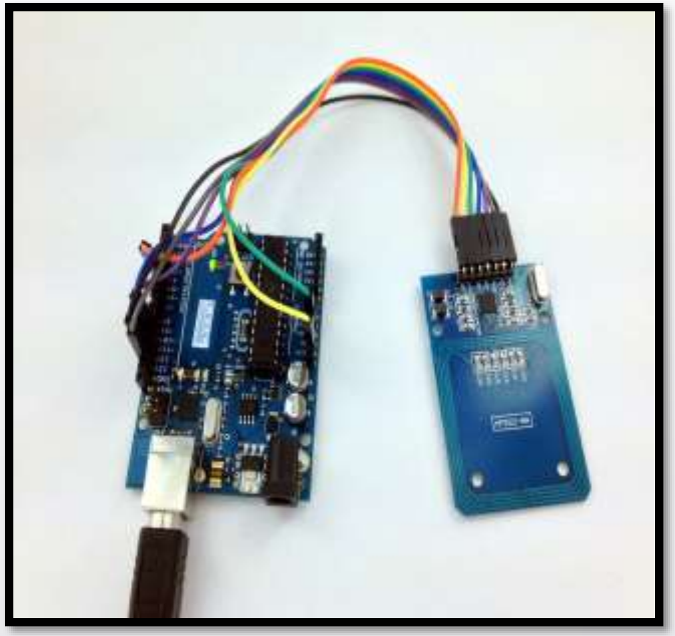
RFID card Tag

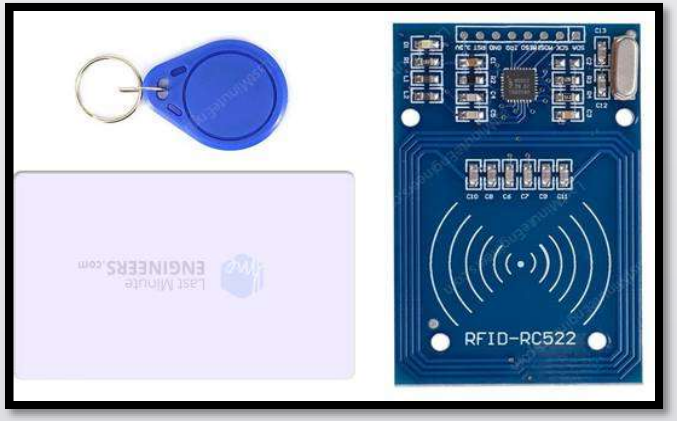


RC522 RFID Module

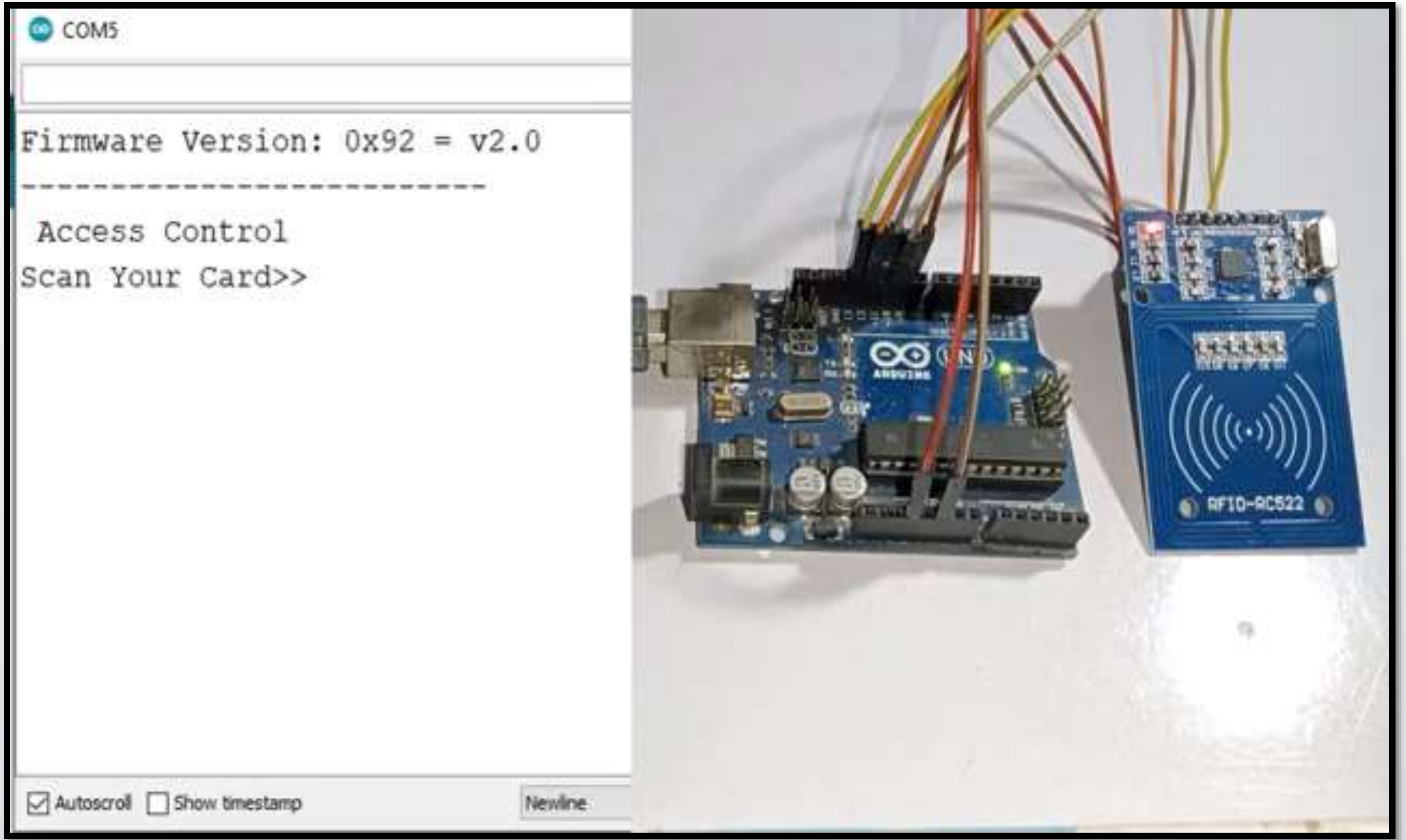
Hardware Overview

- The **RC522 RFID module based on the MFRC522** IC
- **one of the cheapest RFID options you can get online for less than four dollars.**
- **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 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 module's operating voltage ranges from 2.5 to 3.3V, but the good news is that the logic pins are 5-volt tolerant, so we can easily connect it to an Arduino or any 5V logic microcontroller without using a logic level converter.

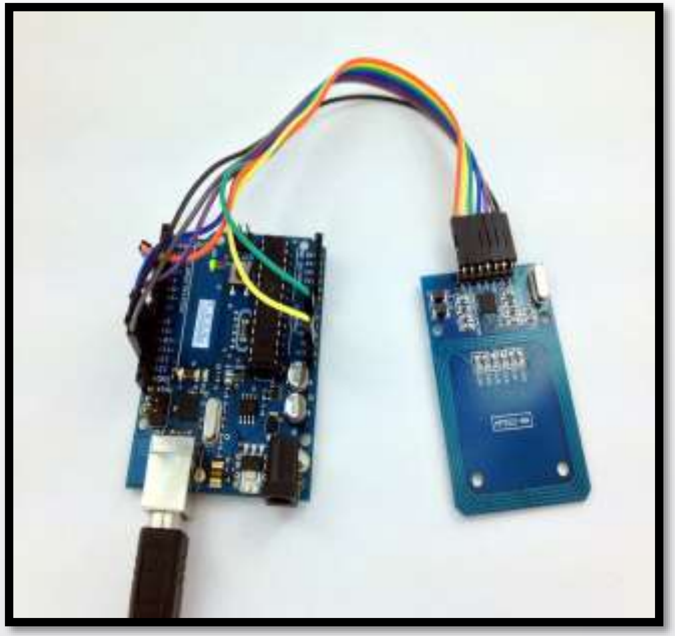


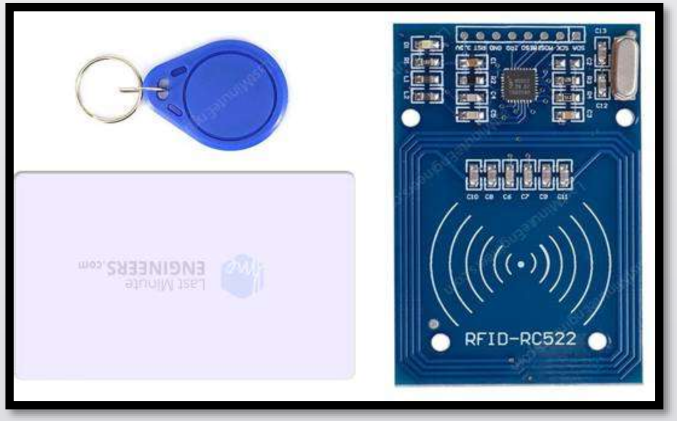


Sample Video

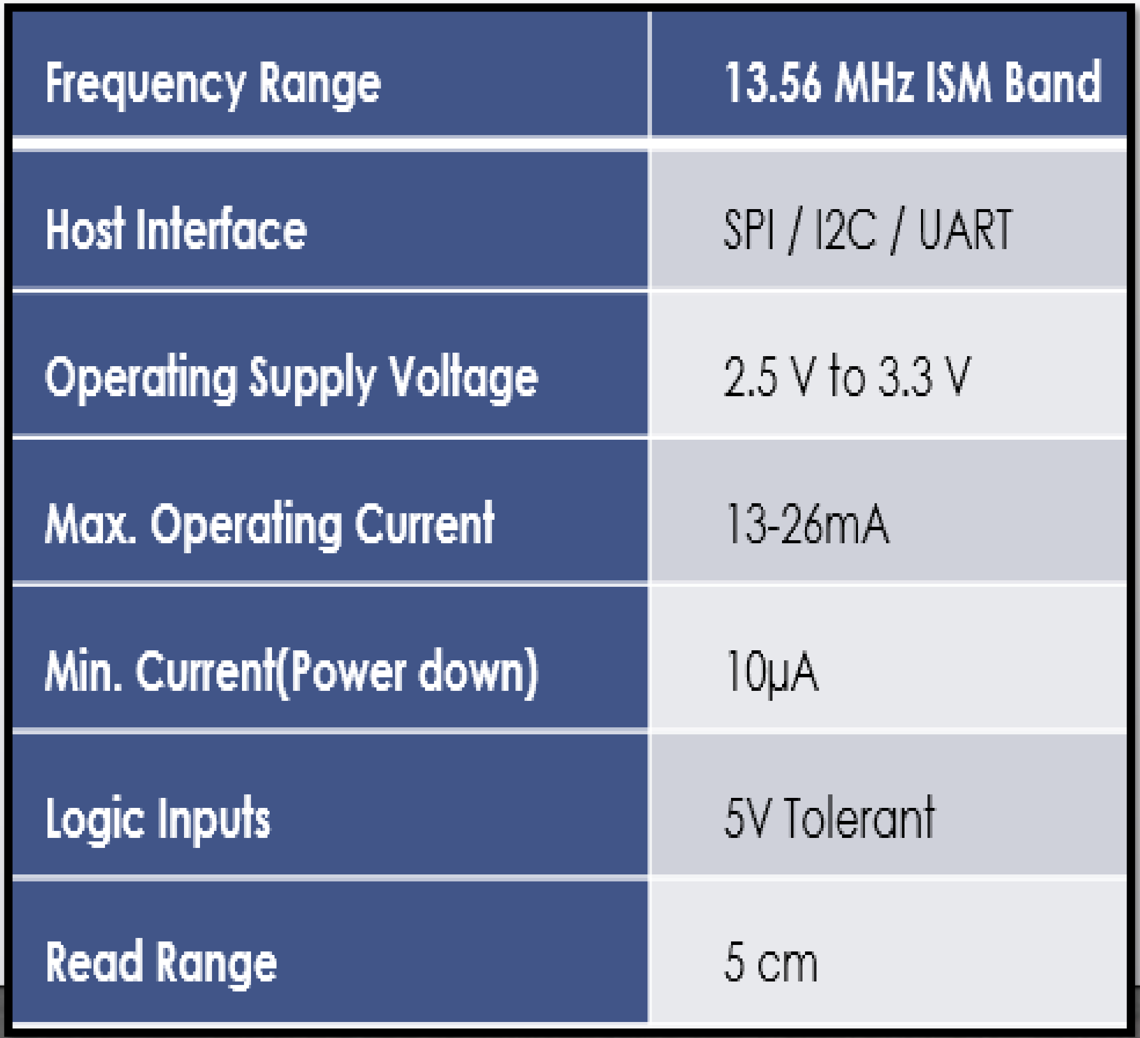


```
Firmware Version: 0x92 = v2.0
---------------------------
 Access Control
Scan Your Card>>
```

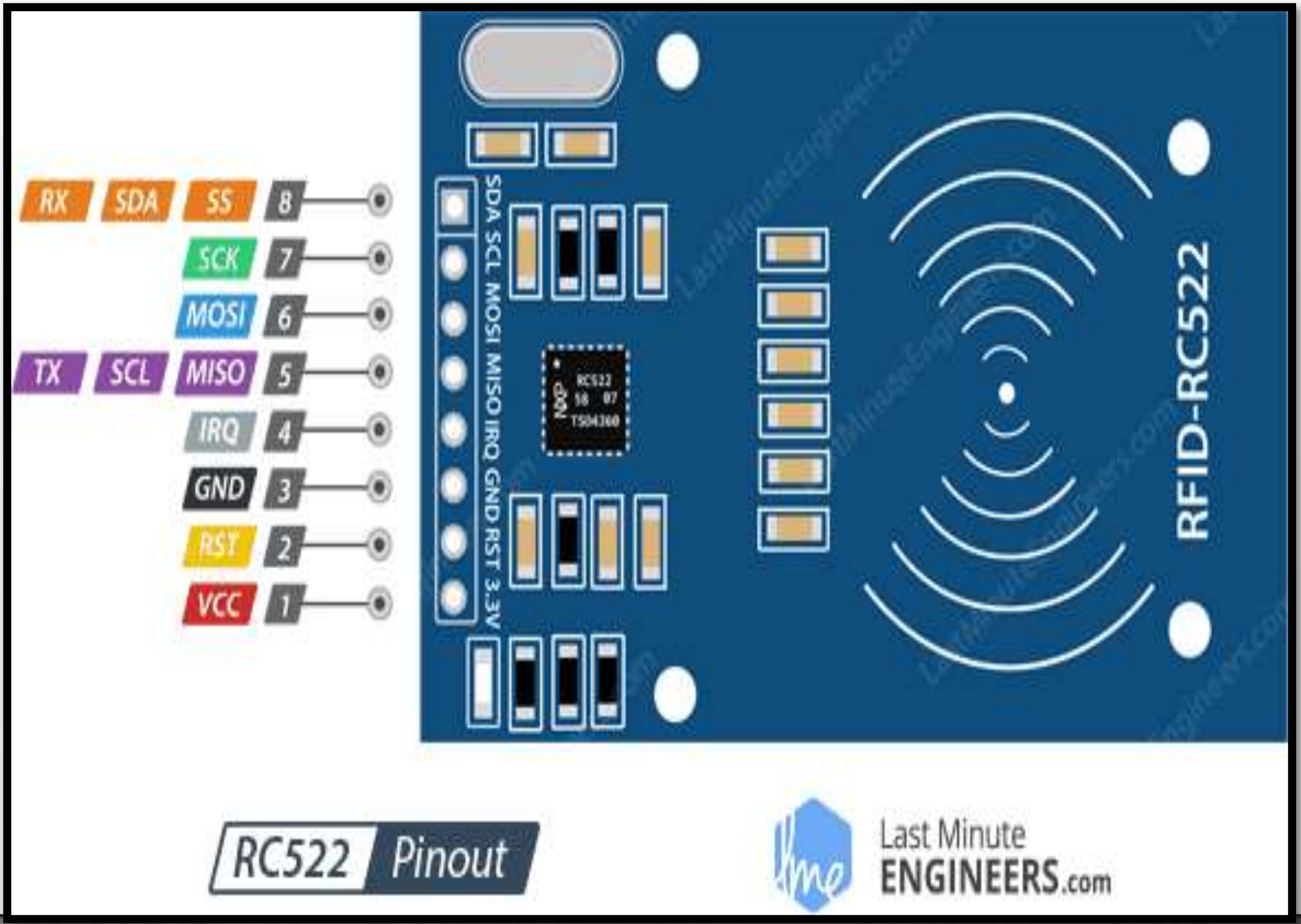




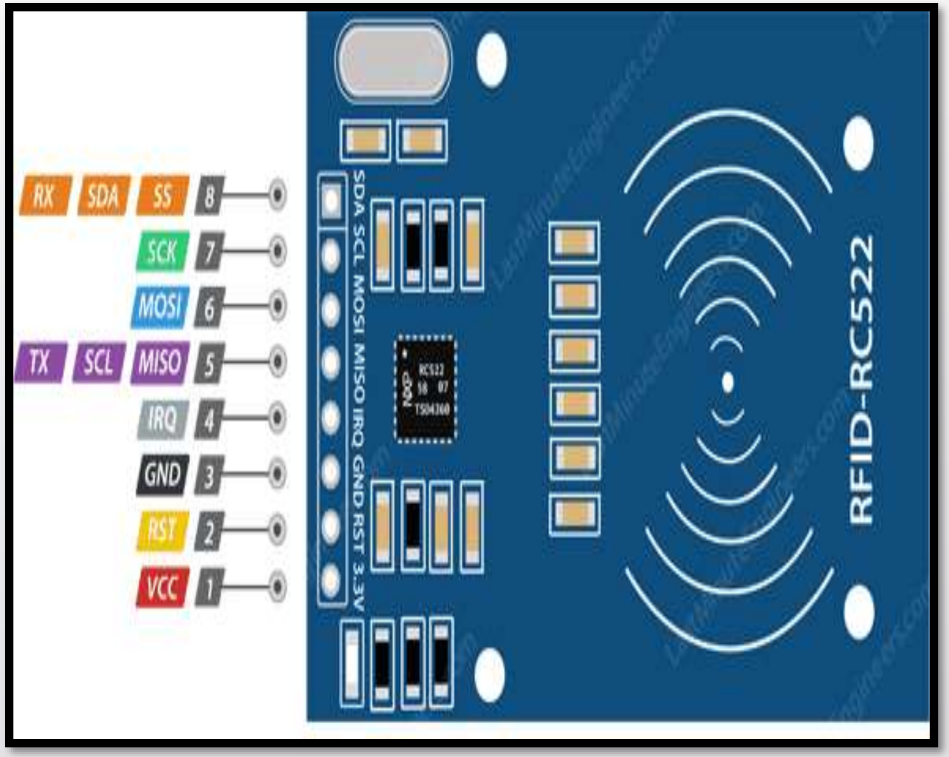
The RC522 module has a total of 8 pins that connect it to the outside world. The connections are as follows:



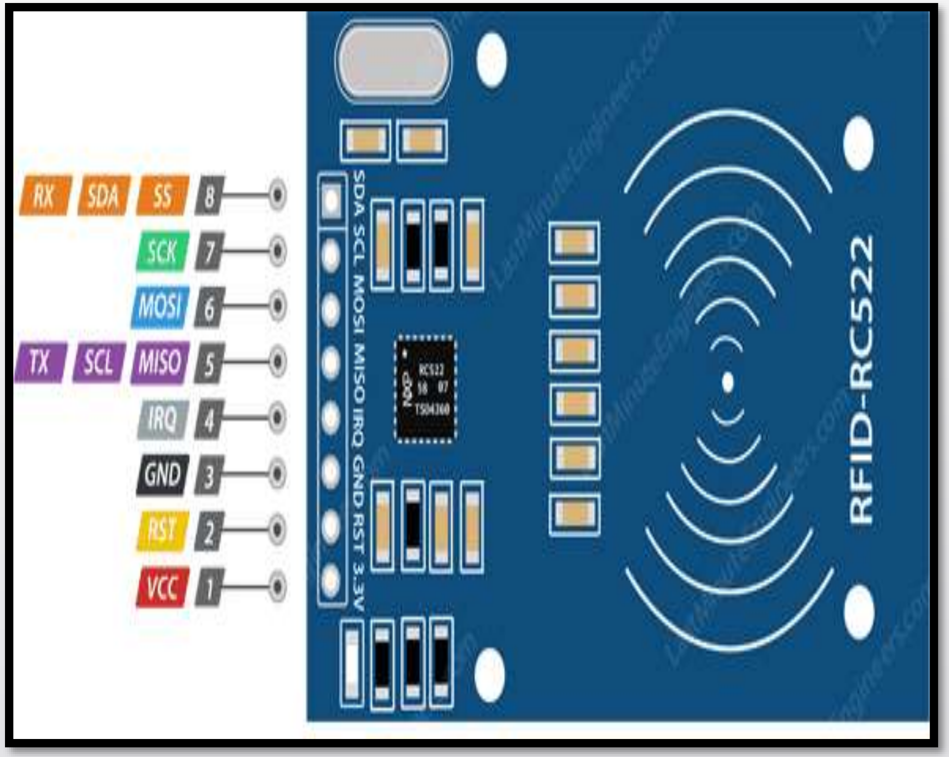
Frequency Range	13.56 MHz ISM Band
Host Interface	SPI / I2C / UART
Operating Supply Voltage	2.5 V to 3.3 V
Max. Operating Current	13-26mA
Min. Current(Power down)	10µA
Logic Inputs	5V Tolerant
Read Range	5 cm



1. VCC(voltage at the common collector) 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 Arduino. But remember that connecting it to the 5V pin will probably destroy your module!
2. RST is an input for reset and power-down. When this pin goes low the module enters power-down mode.
3. GND is the ground pin and needs to be connected to the GND pin on the Arduino.
4. IRQ(interrupt request) is an interrupt pin that alerts the microcontroller when an RFID tag is in the vicinity.



5. **MISO / SCL(serial clock) / Tx** pin acts as master-in-slave-out when SPI(serial peripheral interface) is enabled, as serial clock when I2C(inter-integrated circuit) interface is enabled and as serial data output when the UART (universal asynchronous Recever-Transmitter)interface is enabled.
6. **MOSI (Master Out Slave In)** is the SPI input to the RC522 module.
7. **SCK (Serial Clock)** accepts the clock pulses provided by the SPI bus master i.e. Arduino.
8. **SS(slave select) / SDA(serial data line) / 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.

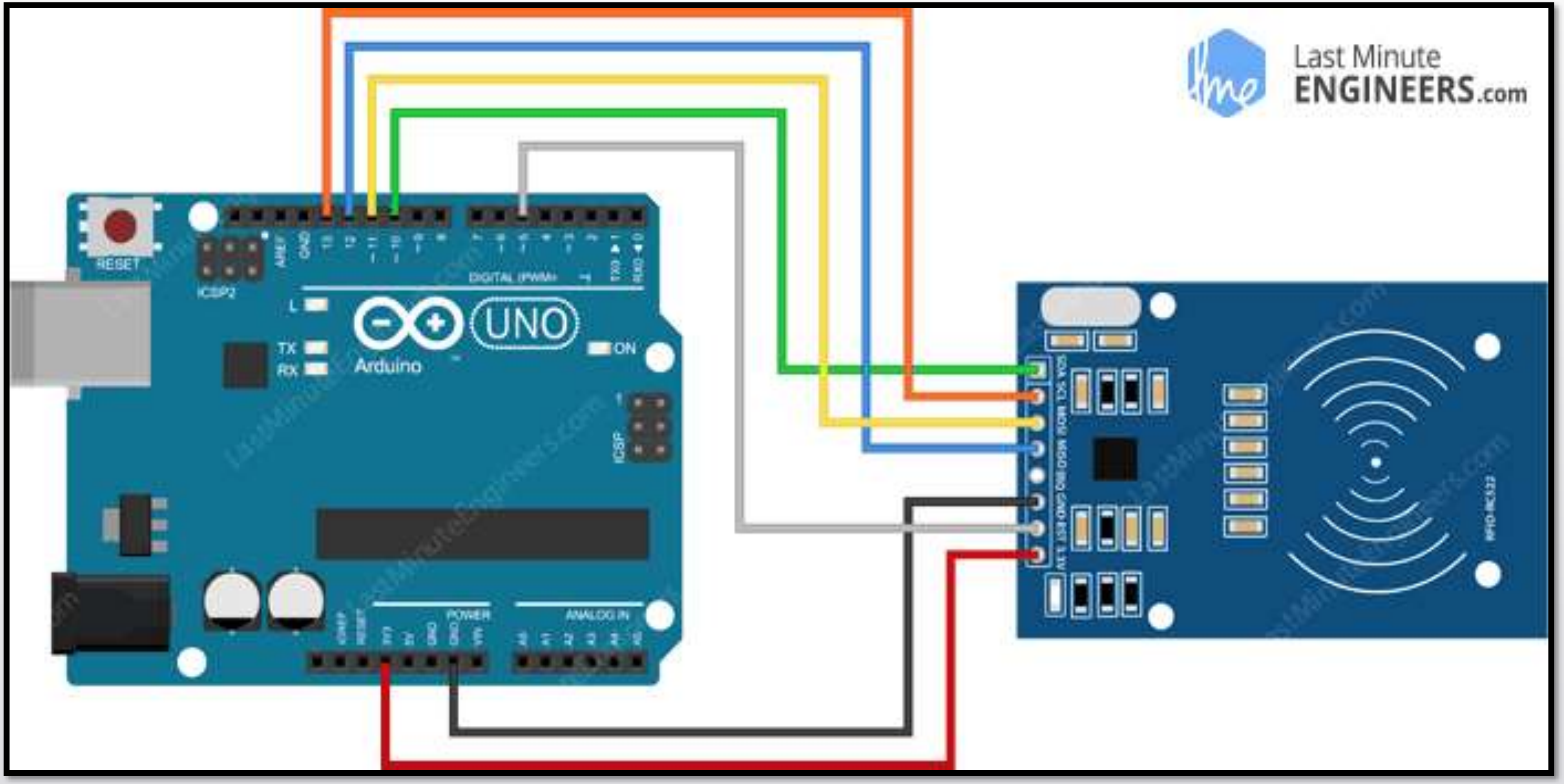


First connect the VCC pin on the module to 3.3V and the GND pin to ground on the Arduino. Pin RST can be connected to any digital pin on the Arduino. In our case, it is connected to digital pin #5. The IRQ pin is left unconnected because the Arduino library we are going to use does not support it.

Now we are left with the pins that are used for SPI communication. Since RC522 modules require a lot of data transfer, they will give the best performance when connected to the hardware SPI pins on the microcontroller.

Note that each Arduino board has different SPI pins that must be connected accordingly. Check the table below for quick understanding.

```
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
*/
```



Wiring RC522 RFID Reader Writer Module with Arduino UNO

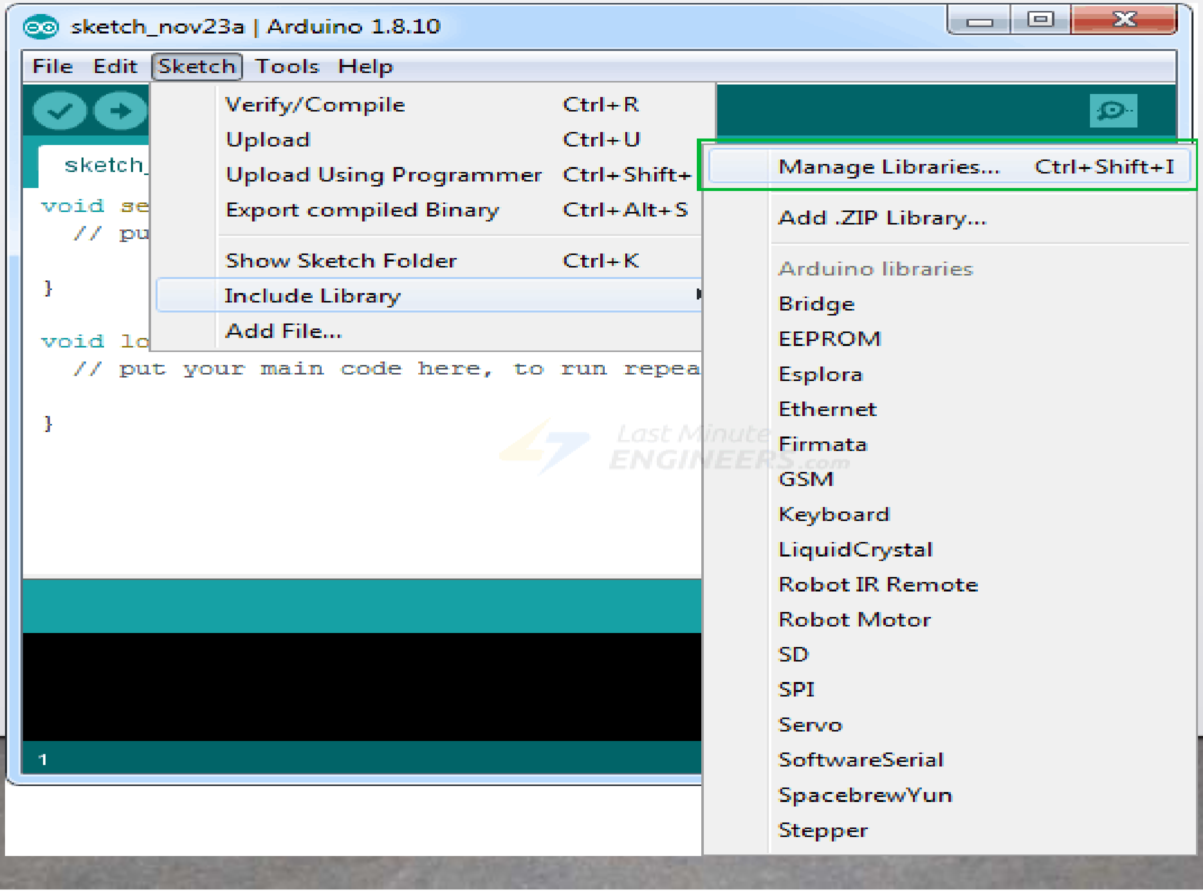
Once you have connected everything you are ready to go!

Library Installation

Communicating with an RC522 RFID module is a lot of work, but luckily for us there is a library called the MFRC522 library that makes reading and writing RFID tags simple.

This library is not included in the Arduino IDE, so you will need to install it first.

To install the library navigate to Sketch > Include Libraries > Manage Libraries… Wait for Library Manager to download the library index and update the list of installed libraries.



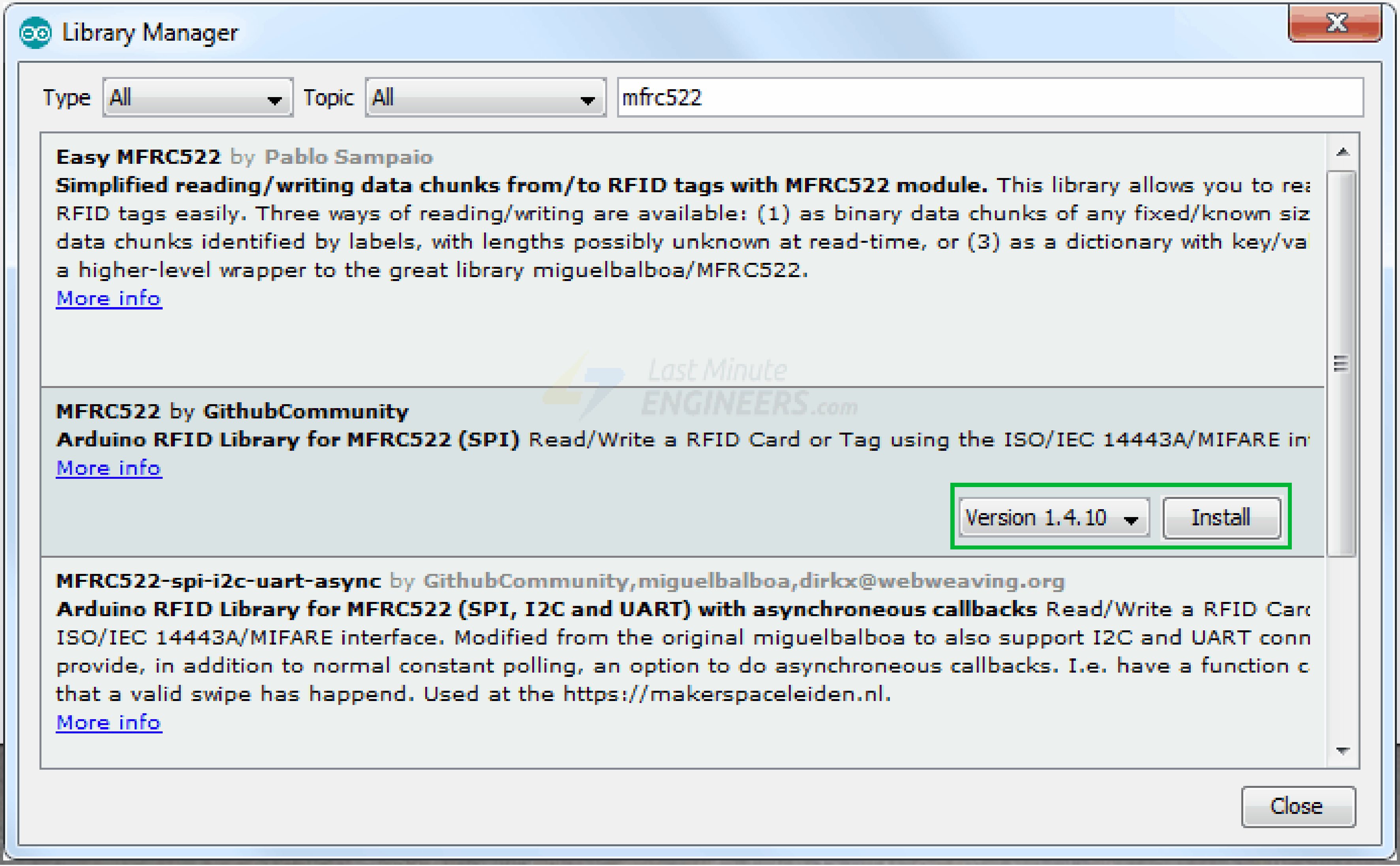
```
void se
  // pu

}

void lo
  // put your main code here, to run repea

}
```

Filter your search by typing ‘mfrc522‘. Look for the library by GithubCommunity. Click on that entry, and then select Install.



mfrc522

Easy MFRC522 by Pablo Sampaio
Simplified reading/writing data chunks from/to RFID tags with MFRC522 module. This library allows you to rea RFID tags easily. Three ways of reading/writing are available: (1) as binary data chunks of any fixed/known siz data chunks identified by labels, with lengths possibly unknown at read-time, or (3) as a dictionary with key/va a higher-level wrapper to the great library miguelbalboa/MFRC522.
More info

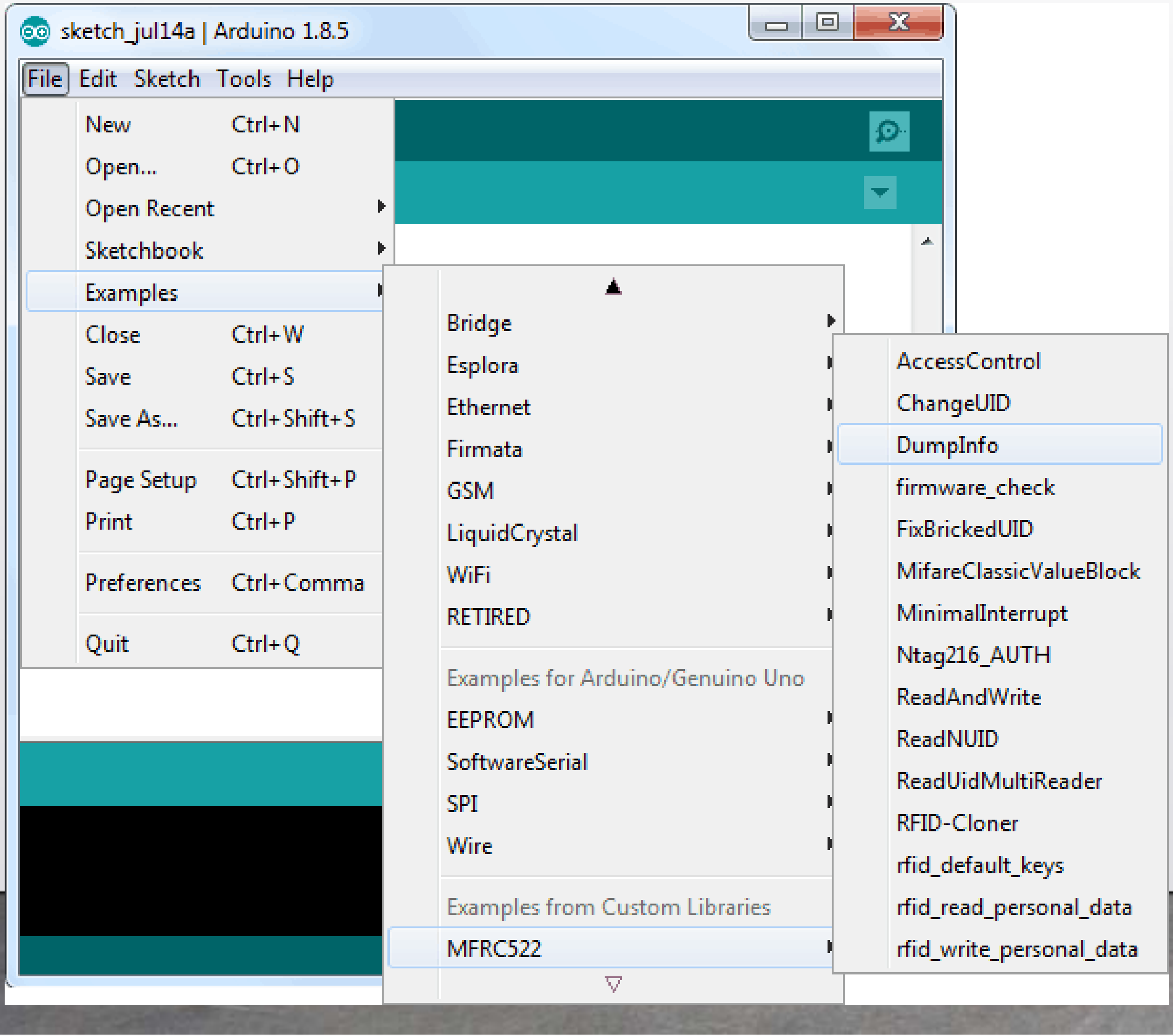
MFRC522 by **GithubCommunity**
Arduino RFID Library for MFRC522 (SPI) Read/Write a RFID Card or Tag using the ISO/IEC 14443A/MIFARE in
More info

Version 1.4.10 Install

MFRC522-spi-i2c-uart-async by GithubCommunity,miguelbalboa,dirkx@webweaving.org
Arduino RFID Library for MFRC522 (SPI, I2C and UART) with asynchroneous callbacks Read/Write a RFID Car ISO/IEC 14443A/MIFARE interface. Modified from the original miguelbalboa to also support I2C and UART conn provide, in addition to normal constant polling, an option to do asynchroneous callbacks. I.e. have a function c that a valid swipe has happend. Used at the https://makerspaceleiden.nl.
More info

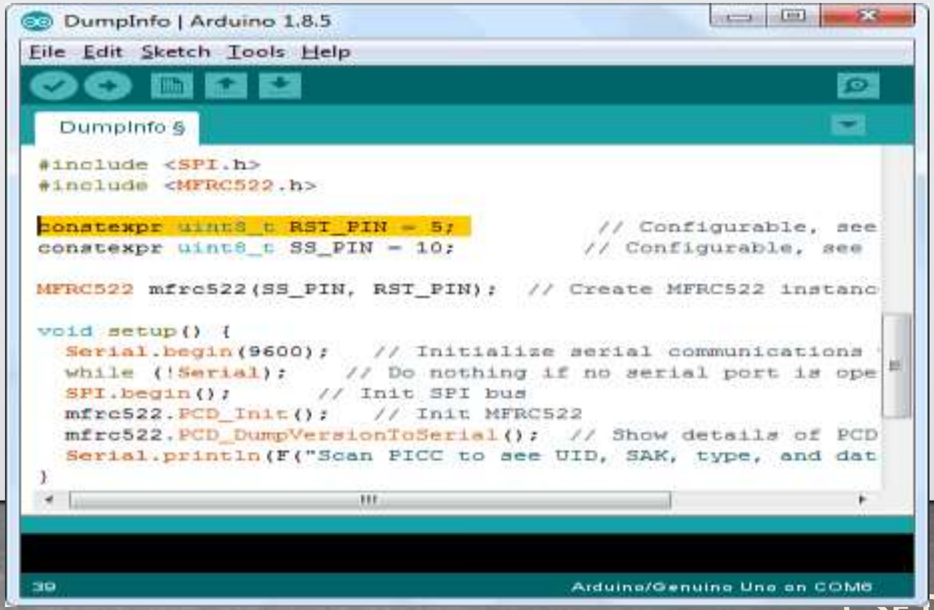
Arduino Code – Reading an RFID Tag

Once you have installed the library, open the Examples submenu and choose MFRC522 > DumpInfo example sketch.



This sketch just reads the tag and displays the information stored in it. This sketch can be very handy before trying out any new tags!

Go to the beginning of the sketch and make sure RST_PIN is initialized correctly, in our case we are using digital pin #5 so change it to 5



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

constexpr uint8_t RST_PIN = 5;           // Configurable, see
constexpr uint8_t SS_PIN = 10;          // Configurable, see

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instanc

void setup() {
  Serial.begin(9600);    // Initialize serial communications
  while (!Serial);     // Do nothing if no serial port is ope
  SPI.begin();       // Init SPI bus
  mfrc522.PCD_Init();    // Init MFRC522
  mfrc522.PCD_DumpVersionToSerial();  // Show details of PCD
  Serial.println(F("Scan PICC to see UID, SAK, type, and dat
}
```

```
#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
	delay(4);				// Optional delay. Some board do need more time after init to be ready, see Readme
	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() {
        // Reset the loop if no new card present on the sensor/reader. This saves the entire process when idle.
        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));
}
```

Now upload the sketch and open Serial Monitor. As you bring the tag closer to the module, you'll get something like the following. Do not move the tag until all the information is displayed.

READ Information Program

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN         9
#define SS_PIN          10
MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup() {
  Serial.begin(9600);
  SPI.begin();

  mfrc522.PCD_Init();

  Serial.println(F("Read personal data"));
}
void loop() {
  MFRC522::MIFARE_Key key;
  for (byte i = 0; i < 6; i++)
  key.keyByte[i] = 0xFF;
  byte block;
  byte len;
  MFRC522::StatusCode status;
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  Serial.println(F("**Card Detected:**"));
  mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));
  Serial.print(F("Name: "));
  block = 4;
```

```
  len = 18;
  byte buffer2[18];
  block = 1;
  status =
mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF
_AUTH_KEY_A, 1, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("Authentication failed:
"));
    Serial.println(mfrc522.GetStatusCodeName(
status));
    return;
  }
  status = mfrc522.MIFARE_Read(block,
buffer2, &len);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("Reading failed: "));
    Serial.println(mfrc522.GetStatusCodeName(
status));
    return;
  }
  for (uint8_t i = 0; i < 16; i++) {
    Serial.write(buffer2[i]);
  }
  Serial.println(F("\n**End Reading**\n"));
  delay(1000); //change value if you want to
read cards faster
  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
}
```

Write Information Program

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN         9
#define SS_PIN          10
MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
  Serial.println(F("Write personal data on a MIFARE PICC "));
}
void loop() {
  MFRC522::MIFARE_Key key;
  for (byte i = 0; i < 6; i++)
  key.keyByte[i] = 0xFF;
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  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(F(" PICC type: "));
  MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
  Serial.println(mfrc522.PICC_GetTypeName(piccType));
  byte buffer[34];
  byte block;
  MFRC522::StatusCode status;
  byte len;
  Serial.setTimeout(20000L) ;
  // Ask personal data: First name
```

```
Serial.println(F("Type First name, ending with #"));
  len = Serial.readBytesUntil('#', (char *) buffer, 20) ;
  for (byte i = len; i < 20; i++) buffer[i] = ' ‘;
  block = 1;
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key,
&(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("PCD_Authenticate() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  status = mfrc522.MIFARE_Write(block, buffer, 16);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Write() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else Serial.println(F("MIFARE_Write() success: "));
  block = 2;
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key,
&(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("PCD_Authenticate() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  status = mfrc522.MIFARE_Write(block, &buffer[16], 16);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Write() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else Serial.println(F("MIFARE_Write() success: "));
  Serial.println(" ");
  mfrc522.PICC_HaltA(); t
  mfrc522.PCD_StopCrypto1();
}
```