RFID TECHNOLOGY

What is RFID?

RFID = Radio Frequency IDentification.

An ADC (Automated Data Collection) technology that uses radiofrequency waves to transfer data between a reader and a movable item to identify, categorize, track..
Is fast and does not require physical sight or contact between reader/scanner and the tagged item.Performs the operation using low cost components.Attempts to provide unique identification and backend integration that allows for wide range of applications.
§
Other ADC technologies: Bar codes, OCR.

There are two types of RFID tags:

- Passive or non-battery powered RFID tag
- Active or battery powered RFID tag

A **passive RFID tag** that does not contain a battery will use the interrogator's radio wave energy to relay its stored information back to the interrogator.

- **Disadvantages of a passive RFID tag**:
  - Can be read only at very short distances, typically a few feet at most. This greatly limits many applications.
  - It may not be possible to include sensors that can use electricity for power.
- **Advantages of a passive RFID tag**:
  - Functions without a battery and have a useful life of twenty years +.
  - Less expensive to manufacture
  - Much smaller (size of a grain of rice). These tags have almost unlimited applications in consumer goods and other areas.

An **active RFID tag** is equipped with a battery that can be used as a partial or complete source of power for the tag's circuitry and antenna. Some active tags contain replaceable batteries for years of use; others are sealed units.

- **Disadvantages of an active RFID tag**:
  - Cannot function without battery power, which limits the lifetime of the tag.
  - Typically more expensive.
  - Is physically larger, which may limit applications.
  - Higher cost for maintenance if the batteries are replaced.
  - Battery outages can result in expensive misreads.
- **Advantages of an active RFID tag**:
  - Can be read at distances of one hundred feet or more, greatly improving the utility of the device.
  - May have other sensors that can use electricity for power.

Applications of RFID

- ➢ Manufacturing and Processing
- ➢ Inventory and production process monitoring
- ➢ Warehouse order fulfillment

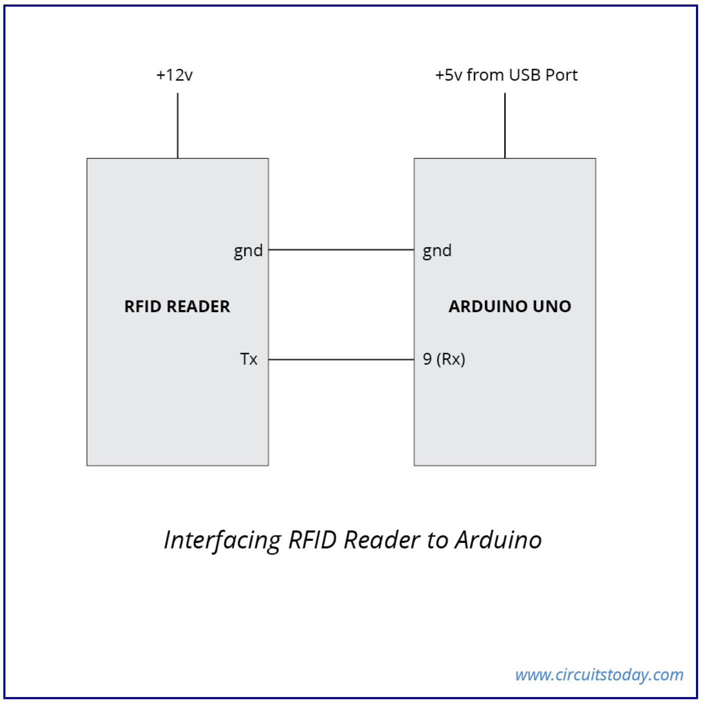
- Supply Chain Management
- Inventory tracking systems
- Logistics management
- Retail
- Inventory control and customer insight
- Auto checkout with reverse logistics
- Security
- Access control
- Counterfeiting and Theft control/prevention
- Location Tracking
- Traffic movement control and parking management
- Wildlife/Livestock monitoring and tracking

How to Interface RFID Reader to Arduino

Note 1:- Power supply requirement of RFID Readers vary from product to product. The RFID reader I used in this tutorial is a 12 Volts one. There are 5 Volts and 9 Volts versions available in the market.

Note 2:- You may ensure the RFID Reader and RFID Tags are frequency compatible. Generally they are supposed to be 125Khz. You may ensure this before purchasing them.

Note 3:- There are two possible outputs from an RFID Reader. One is RS232 compatible output and other one is TTL compatible output. A TTL compatible output pin can be connected directly to Arduino. Whereas an RS232 compatible output must be converted to TTL using an RS232 to TTL converter



Interfacing RFID Reader to Arduino

Make connections as shown. Make sure you connect Ground Pin of RFID reader to Ground Pin of Arduino. I am using the SoftwareSerial Library of Arduino which enables digital pins to be used in serial communication. I have used pin 9 as the Rx of Arduino. (You can also use the hardware Rx pin of Arduino uno – that's pin 0).

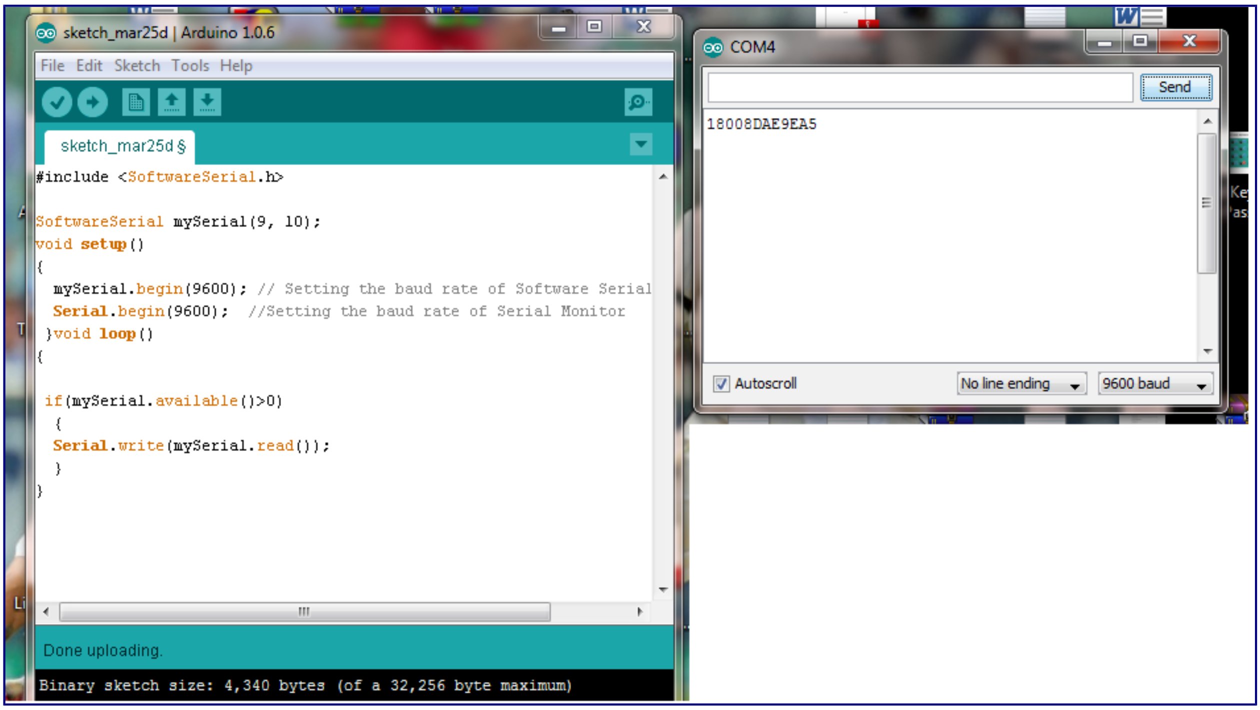
1. Program to read the RFID tags and print the tag ID on the serial monitor

```
 #include <SoftwareSerial.h>

SoftwareSerial mySerial(9, 10);
void setup()
{
  mySerial.begin(9600); // Setting the baud rate of Software Serial Library
  Serial.begin(9600);  //Setting the baud rate of Serial Monitor
 }void loop()
{

 if(mySerial.available()>0)
  {
  Serial.write(mySerial.read());
  }
}
```

mySerial.available() – checks for any data coming from RFID reader module through the SoftwareSerial pin 9. Returns the number of bytes available to read from software serial port. Returns a -1 if no data is available to read.

mySerial.read() – Reads the incoming data through software serial port.

Serial.write() – Prints data to serial monitor of arduino. So the function Serial.write(mySerial.read()) – prints the data collected from software serial port to serial monitor of arduino.



```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(9, 10);
void setup()
{
  mySerial.begin(9600); // Setting the baud rate of Software Serial
  Serial.begin(9600);  //Setting the baud rate of Serial Monitor
 }void loop()
{

 if(mySerial.available()>0)
  {
  Serial.write(mySerial.read());
  }
}
```

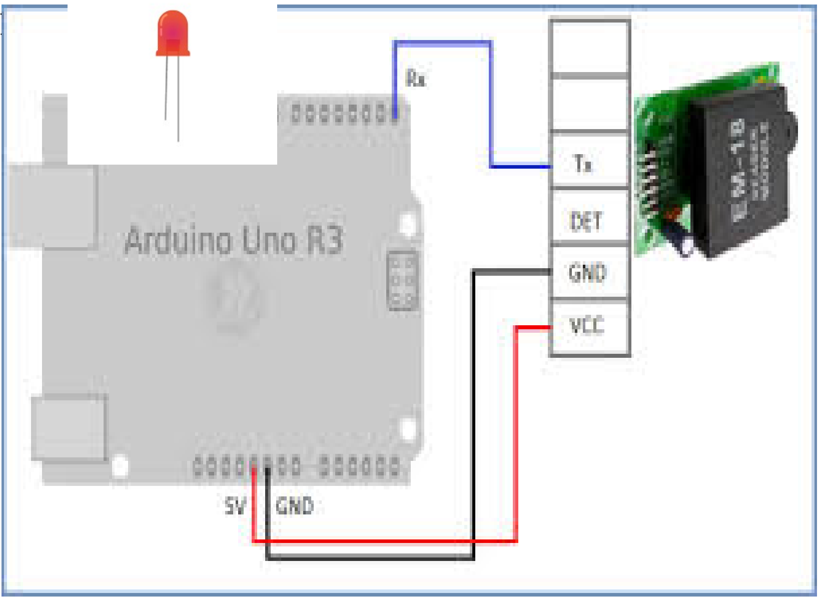
Done uploading.

```
Binary sketch size: 4,340 bytes (of a 32,256 byte maximum)
```

```
18008DAE9EA5
```

2. Program to Provide access base on the Tag.

This program demonstrates the use of R



```
#include<SoftwareSerial.h>

SoftwareSerial mySerial(9, 10);

#define LEDPIN 12

char tag[] ="3C0087D597F9";

 // Replace with your own Tag ID
```

```
char input[12];                          // A variable to store the Tag ID being presented

int count = 0;                           // A counter variable to navigate through the
input[] character array

boolean flag = 0;

// A variable to store the Tag match status

void setup()

{       Serial.begin(9600);        // Initialise Serial Communication with the Serial
Monitor

      mySerial.begin(9600);

        pinMode(LEDPIN,OUTPUT); //WRONG TAG INDICATOR

}

void loop()

{

        if(mySerial.available())// Check if there is incoming data in the RFID Reader Serial
Buffer.

        {

                count = 0; // Reset the counter to zero

        /* Keep reading Byte by Byte from the Buffer till the RFID Reader Buffer is
        empty

                  or till 12 Bytes (the ID size of our Tag) is read */

                while(mySerial.available() && count < 12)

                {

        input[count] = mySerial.read(); // Read 1 Byte of data and store it in the input[]
variable

Serial.write(input[count]);

                        count++; // increment counter

                        delay(5);

                }

                /* When the counter reaches 12 (the size of the ID) we stop and compare
each value

                   of the input[] to the corresponding stored value */

                if(count == 12) //
                {
                        count =0; // reset counter varibale to 0
                        flag = 1;
                        /* Iterate through each value and compare till either the 12 values are
                         all matching or till the first mistmatch occurs */
                        while(count<12 && flag !=0)
                        {
                            if(input[count]==tag[count])
                            flag = 1; // everytime the values match, we set the flag variable
to 1
                            else
                            flag= 0;
                       /* if the ID values don't match, set flag variable to 0 and
                        stop comparing by exiting the while loop */
                            count++; // increment i
                        }
                }
                if(flag == 1) // If flag variable is 1, then it means the tags match
                {
                        Serial.println("Access Allowed!");
                        digitalWrite(LEDPIN,HIGH);
                        delay (2000);
                        digitalWrite (LEDPIN,LOW);
                }
                else
                {
                        Serial.println("Access Denied"); // Incorrect Tag Message
                        digitalWrite(LEDPIN,LOW);
                        delay(2000);
                            }
                /* Fill the input variable array with a fixed value 'F' to overwrite
                all values getting it  empty for the next read cycle */
                for(count=0; count<12; count++)
                {
                        input[count]= 'F';
                }
                count = 0; // Reset counter variable
            }
}
```