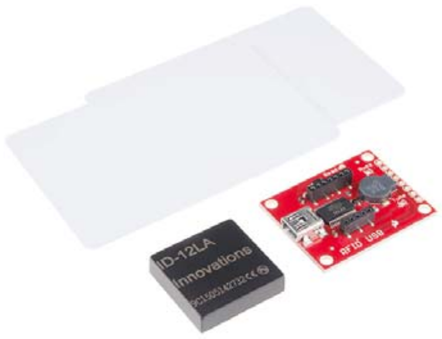
SparkFun RFID Starter Kit Hookup Guide

Introduction

The SparkFun RFID Starter Kit gives you the tools you need to start reading RFID tags with your computer or microcontroller!



Required Materials

The kit contains:

- 1x RFID USB Reader
- 1x ID-12LA RFID Module
- 2x 125KHz RFID cards

You will also need a USB mini-B cable to connect the USB reader to a computer.

Recommended Reading

The ID-12LA module has a serial output. If you've never worked with a serial device or a terminal program before, you might want to take a look at these tutorials first.

- Serial Communication
- Serial Terminal Basics
- Tim's RFID Comparison video - Let Tim from Tech Support walk you through our entire line of RFID options, complete with range tests.
- SparkFun Simple Sketches - RFID Starter Kit video - A simple video demo of the RFID Starter Kit in action.

RFID Overview

But Wait, What is RFID?

If you already know how RFID works (or would just like to continue believing the reader module contains a very small hamster with x-ray eyes), skip ahead to the kit overview section.

How RFID doesn't work

Radio **F**requency **ID**entification uses radio waves to detect the presence of (then read the data stored on) an RFID tag. Tags are embedded in small items like cards, buttons, or tiny capsules.

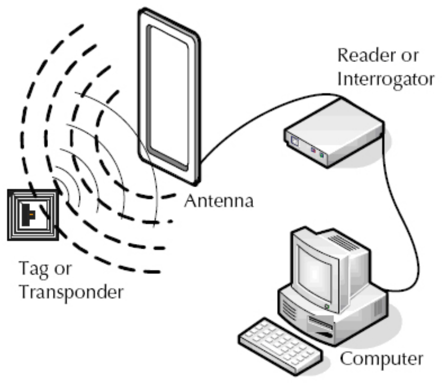


Image courtesy of EPC RFID

Passive tags (like the ones included with this kit) gather electromagnetic energy from the card reader and use that to transmit their unique serial number. More sophisticated tags may have their own internal power supply for increased range.

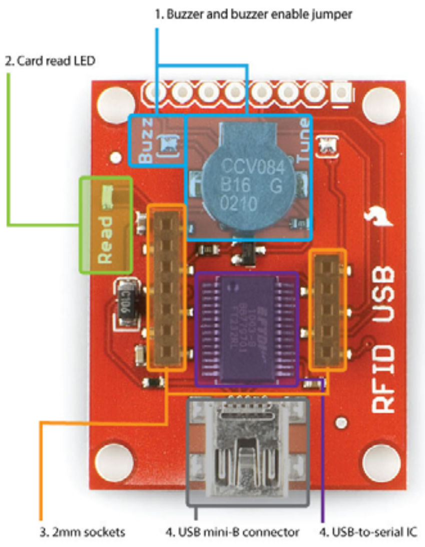
A Few Common RFID Reader Types				
Frequency	**AKA**	**Range**	**Read/write**	**SparkFun product**
120-150 kHz (LF)	"Chips/microchips"(in veterinary applications), prox cards, HID cards (both trade names)	Up to 20 cm	Read only	ID-3LA, ID-12LA, ID-20LA, SparkFun RFID Starter Kit
13.56 MHz (HF)	MiFare, NFC	Up to 1 meter	Read/write	SM130 module and evaluation shield

433 MHz (UHF)	Long-range RFID, powered RFID	Up to 100 meters	Read/write	N/A
Information from Wikipedia: Radio-frequency identification				

Some 125Khz RFID tags (like the ubiquitous "HID ProxCard II" brand ID card and some brands of pet tag) use a **proprietary format**. School IDs and other cards from commercial access control systems may not work with the ID-12LA module. **IDentisource** has a guide to identifying cards.

RFID USB Reader Overview

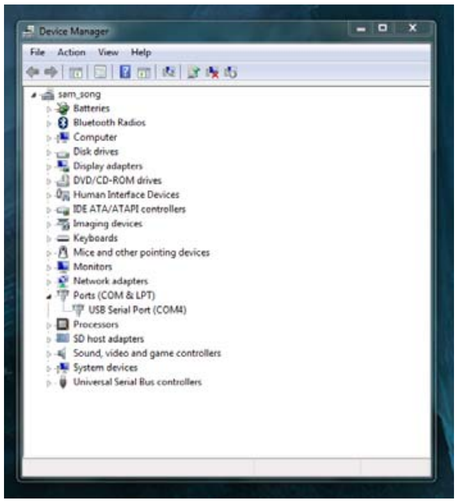
The RFID USB Reader has the following features:



1. A buzzer that sounds when a card is read. If you are using the RFID kit in a stealth application, you can disconnect the buzzer by removing the blob of solder on the **Buzz** jumper.
2. A “card read” LED
3. 2mm-spaced sockets that fit three of SparkFun’s RFID modules (the ID-3LA, the ID-12LA, and the ID-20LA)
4. A USB mini-B connector
5. An FTDI232RL USB-to-serial IC that converts the module’s TTL serial output to USB

Simple Hookup

Place the ID-12 module onto the RFID USB Reader, and plug the base into your computer with a USB mini-B cable. Depending on your operating system, you may need to install FTDI drivers after plugging in the base station.



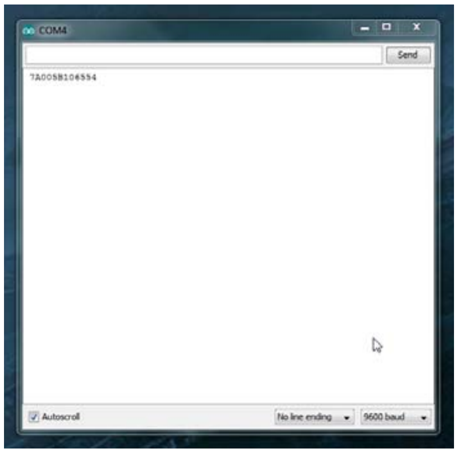
Here's the RFID reader (on COM4) with the FTDI drivers installed.

Open a terminal program of your choice. If you've never used a terminal program before, here's a guide to choosing and using them.

First, use the Arduino IDE's built-in serial monitor:

- Open the Arduino IDE.
- Go to **Tools > Port** and select the RFID reader's port.
- Go to **Tools > Serial Monitor**. The default terminal settings (9600 baud, no line ending) are fine. The monitor should be blank.

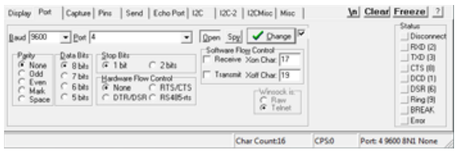
Wave a card over the reader. You should hear a beep and see something like this.



```
7A005B106554
```

Now, we'll do the same thing in RealTerm. (Mac users, you can try this section using CoolTerm)

RealTerm may look like the cockpit of a 747 compared to the Arduino serial monitor, but it has several helpful features. Keep the RealTerm tutorial open if you need it.

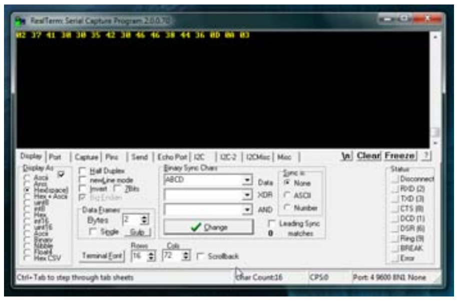


Baud 9600 Port 4

Parity: [✓ None] Odd Even Mark Space
Data Bits: [✓ 8 bits] 7 bits 6 bits 5 bits
Stop Bits: [✓ 1 bit] 2 bits
Hardware Flow Control: [✓ None] RTS/CTS DTR/DSR RS485-rts
Software Flow Control: Receive Xon Char: 17 Transmit Xoff Char: 19
Winsock is: Raw [✓ Telnet]

Char Count:16 CPS:0 Port: 4 9600 8N1 None

*The RealTerm **Port** tab with the port set to* `4` *and the baud rate to* `9600`

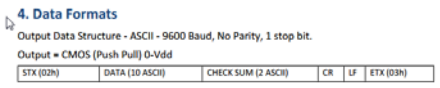


```
02 37 41 30 30 35 42 30 46 46 38 44 36 0D 0A 03
```

*The RealTerm **Display** tab*

With **Display As** set to `Hex[space]` , the card data appears as 16 hex bytes instead of 12 ASCII digits like it did in the Arduino Serial Monitor.

Wait, 16 bytes? Where did the extra four come from?



4. Data Formats

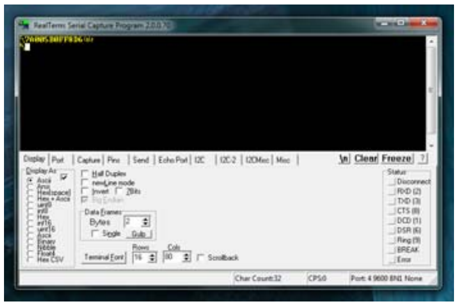
Output Data Structure - ASCII - 9600 Baud, No Parity, 1 stop bit.

Output = CMOS (Push Pull) 0-Vdd

STX (02h)	DATA (10 ASCII)	CHECK SUM (2 ASCII)	CR	LF	ETX (03h)

Here's the “Data format” section from the ID-12 module datasheet. The 12 ASCII characters displayed in the Arduino serial monitor are just the filling in a 16-byte sandwich, with four more non-printing characters (STX or start-of-text, CR/carriage return, LF/linefeed, and ETX/end-of-text) as the bread.

Try switching the **Display As** setting to `ASCII` and scan again:



```
7A005B0FF8D6
```

Now the “bread” is visible! That's cool, right?

Example Project

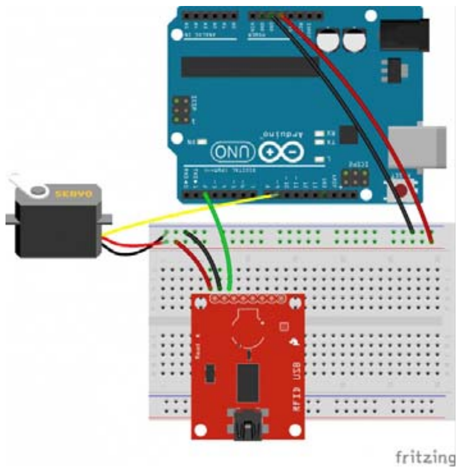
As fun as it is to watch your cards pop up in the serial terminal, you'd probably like to *do* something with all this power.

The example sketch below scans RFID cards and compares them against trusted cards, then moves a servo to unlock the secured* item of your choice.

* *Not suitable for critical applications, e.g. guarding the Hope Diamond.*

In addition to your RFID Reader Kit, you will want:

- A microcontroller like the SparkFun RedBoard or the Arduino Uno
- Jumper Wires - M-to-M
- Breadboard
- A Servo - Larger sizes are suggested for larger locks
- Break-away Male Headers to solder to the board's through-holes. (If you need a soldering refresher, take a look at our through-hole soldering tutorial)



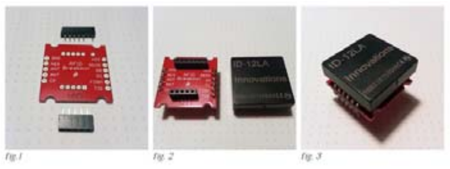
Connect the VCC, GND, and TX pins on the RFID USB Reader to the Arduino's 5V, GND, and D2 pins, and connect the servo to 5V, GND, and D9. Upload the below code, open your Serial Monitor by going to **Tools > Serial Monitor**, and start scanning some cards!

```
/*****************************
     RFID-powered lockbox

This sketch will move a servo when
a trusted tag is read with the
ID-12/ID-20 RFID module

Pinout for SparkFun RFID USB Reader
Arduino ----- RFID module
5V            VCC
GND           GND
D2            TX

Pinout for SparkFun RFID Breakout Board
Arduino ----- RFID module
5V            VCC
GND           GND
D2            D0

Connect the servo's power, ground, and
signal pins to VCC, GND,
and Arduino D9

If using the breakout, you can also
put an LED & 330 ohm resistor between
the RFID module's READ pin and GND for
a "card successfully read" indication

by acavis, 3/31/2015

Inspired by & partially adapted from
http://bildr.org/2011/02/rfid-arduino/

******************************/

#include <SoftwareSerial.h>
#include <Servo.h>

// Choose two pins for SoftwareSerial
SoftwareSerial rSerial(2, 3); // RX, TX

// Make a servo object
Servo lockServo;

// Pick a PWM pin to put the servo on
const int servoPin = 9;

// For SparkFun's tags, we will receive 16 bytes on every
// tag read, but throw four away. The 13th space will always
// be 0, since proper strings in Arduino end with 0

// These constants hold the total tag length (tagLen) and
// the length of the part we want to keep (idLen),
// plus the total number of tags we want to check against (kTa
gs)
const int tagLen = 16;
const int idLen = 13;
const int kTags = 4;

// Put your known tags here!
char knownTags[kTags][idLen] = {
             "111111111111",
             "444444444444",
             "555555555555",
              "7A005B0FF8D6"
};

// Empty array to hold a freshly scanned tag
char newTag[idLen];

void setup() {
  // Starts the hardware and software serial ports
   Serial.begin(9600);
   rSerial.begin(9600);

   // Attaches the servo to the pin
   lockServo.attach(servoPin);

   // Put servo in locked position
   lockServo.write(0);
}

void loop() {
  // Counter for the newTag array
  int i = 0;
  // Variable to hold each byte read from the serial buffer
  int readByte;
  // Flag so we know when a tag is over
  boolean tag = false;

  // This makes sure the whole tag is in the serial buffer bef
ore
  // reading, the Arduino can read faster than the ID module c
an deliver!
  if (rSerial.available() == tagLen) {
    tag = true;
  }

  if (tag == true) {
    while (rSerial.available()) {
      // Take each byte out of the serial buffer, one at a tim
e
      readByte = rSerial.read();

      /* This will skip the first byte (2, STX, start of tex
t) and the last three,
      ASCII 13, CR/carriage return, ASCII 10, LF/linefeed, an
d ASCII 3, ETX/end of
      text, leaving only the unique part of the tag string. I
t puts the byte into
      the first space in the array, then steps ahead one spot
*/
      if (readByte != 2 && readByte!= 13 && readByte != 10 &&
readByte != 3) {
        newTag[i] = readByte;
        i++;
      }

      // If we see ASCII 3, ETX, the tag is over
      if (readByte == 3) {
        tag = false;
      }

    }
  }


  // don't do anything if the newTag array is full of zeroes
  if (strlen(newTag)== 0) {
    return;
  }

  else {
    int total = 0;

    for (int ct=0; ct < kTags; ct++){
        total += checkTag(newTag, knownTags[ct]);
    }

    // If newTag matched any of the tags
    // we checked against, total will be 1
    if (total > 0) {

      // Put the action of your choice here!

      // I'm going to rotate the servo to symbolize unlocking
the lockbox

      Serial.println("Success!");
      lockServo.write(180);
    }

    else {
        // This prints out unknown cards so you can add them t
o your knownTags as needed
        Serial.print("Unknown tag! ");
        Serial.print(newTag);
        Serial.println();
    }
  }

  // Once newTag has been checked, fill it with zeroes
  // to get ready for the next tag read
  for (int c=0; c < idLen; c++) {
    newTag[c] = 0;
  }
}

// This function steps through both newTag and one of the know
n
// tags. If there is a mismatch anywhere in the tag, it will r
eturn 0,
// but if every character in the tag is the same, it returns 1
int checkTag(char nTag[], char oTag[]) {
    for (int i = 0; i < idLen; i++) {
      if (nTag[i] != oTag[i]) {
        return 0;
      }
    }
  return 1;
}
```

Using the RFID Reader Breakout

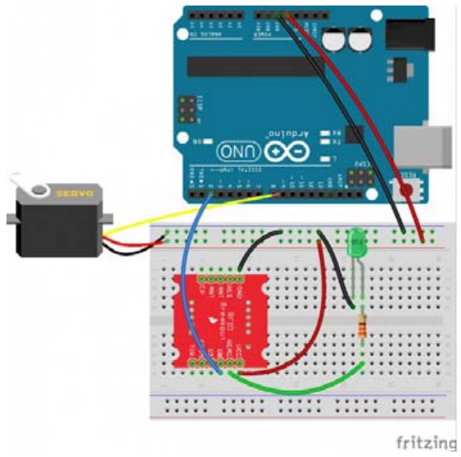
For Arduino projects, you can also use the SparkFun RFID Reader Breakout which gives the module a place to sit and breaks out its odd 2mm-pitch pins to a breadboard-friendly 0.1" spacing.

To keep the module removable (and protect it from accidental damage during soldering), you can trim 2x 2mm Xbee Sockets to size and solder them to the top of the breakout.



- **Fig. 1** Xbee sockets trimmed to size with one pin clipped short
- **Fig. 2** Sockets soldered to the top of the breakout with 0.1" male header pins on the bottom
- **Fig. 3** Module and breakout ready for use on a breadboard

The completed breakout can be used like the larger base station. Here's the same example as above with a green LED and 330 ohm resistor added to the READ pin. **TX** is labeled **D0** on the breakout.



Shown with the module removed so you can read the silk

Resources and Going Further

You can use the SparkFun RFID Reader Kit and an Arduino to control access to just about anything!

- Instead of moving a servo, how about controlling a relay like the PowerSwitch Tail II. No intruder will ever use your soldering iron, toaster, or electric blanket without permission again!

- Rob's NCWP (Non-Crappy Wedding Present) Tutorial – An RFID reader (plus the rest of Rob's parts list) could be all that's separating you from true love.