

A Universal RFID Key

by [drj113](#) on November 16, 2010

Table of Contents

A Universal RFID Key	1
Intro: A Universal RFID Key	2
Step 1: How does RFID work?	2
Step 2: Whats stored on the card?	3
Step 3: How do we emulate a card?	4
File Downloads	5
Step 4: The Software - Entering data into our card	5
File Downloads	5
Step 5: Etching the PCB	6
File Downloads	6
Step 6: Mounting the components	6
Step 7: Programming the Micro	8
Step 8: Testing the project	8
Step 9: Further steps	9
Related Instructables	9
Comments	10



Author:drj113

I have a background in digital electronics, and am very interested in computers. I love things that blink, and am in awe of the physics associated with making blue LEDs.

Intro: A Universal RFID Key

RFID projects have been pretty prominent recently, ranging from projects here in Instructables, to our local Silicon Chip magazine in Australia publishing a RFID door lock project in their November issue. Even I recently purchased a RFID door lock on eBay for \$15 to lock my garage (so my front neighbor could get tools if he wanted to).

We have known that the cheaper RFID technologies were pretty insecure for a number of years. Researchers have demonstrated cloners of all varieties, but simple RFID tags are still being used for access control. Even my current employer uses them.

A while ago, I was looking at Hack A Day, and I saw an amazing project that somebody had made. It was an **RFID card with a keypad** on it. For the next couple of days, I couldn't get the image of the card out of my mind; the project reminded me of how much I wanted to build a RFID spoofer myself. The original author didn't release source code for their project, but they left enough clues that I could follow.

So, in typical fashion, I built my own reader hardware so I could have a look at the data from a card, and created my own version of the Universal RFID key.

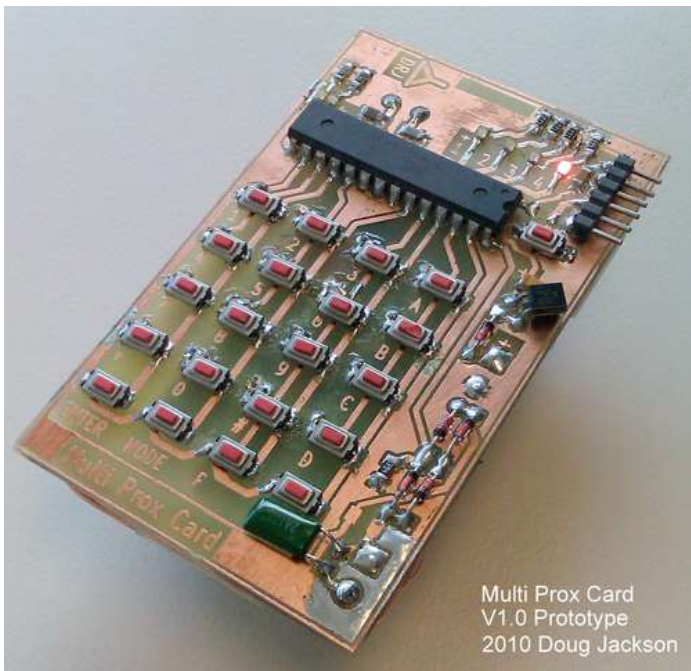
The key I made works beautifully both on my garage door, as well as a number of other RFID readers I have tried!

I have decided to publish this, as more people should be aware of the design flaws that are inherent in older RFID implementations, and to allow others to make their own universal key.

Will this key let you into anybodies RFID protected office? Yes it will, assuming a couple of things are true

- 1) The have to be using 125kHz RFID tags that use the same encoding standard as I have designed this project for, and,
- 2) You have to have access to the number printed on the back of the tag - with that number, you can simply key it into the Universal RFID key, and it will emulate that tag.

So there you go - I hope you enjoy making this project. - And remember, with great power comes great responsibility!



Step 1: How does RFID work?

RFID, or Radio Frequency IDentification is the term used to describe a wide variety of standards that allow data stored within electronic 'tags' to be read by a reader without using wires. There are a number of standards, encoding formats, and frequencies in common use. I will describe the 125 kHz standard that is common for access control mechanisms.

125 kHz RFID tags are commonly encased in a business card sized piece of plastic, or a round disk. The tag consists of a coil of wire, connected to a microchip. When the tag is brought into close proximity to a reader, energy is coupled inductively from the reader to the microchip within the tag.

The energy from the reader has dual use; firstly, it provides power to run the card, and secondly, it provides a communication medium for data to be transmitted. Once powered up, the tag modulates the bit pattern that is programmed into the tag using a signal that the reader can detect. The reader then reads this bit pattern, and passes it onto the door controller. If the bit pattern matches one that is authorised, the door will be unlocked. If the bit pattern does not match an authorised one, then the door won't unlock.

In the RFID system I was playing with, the bit pattern looked like this;

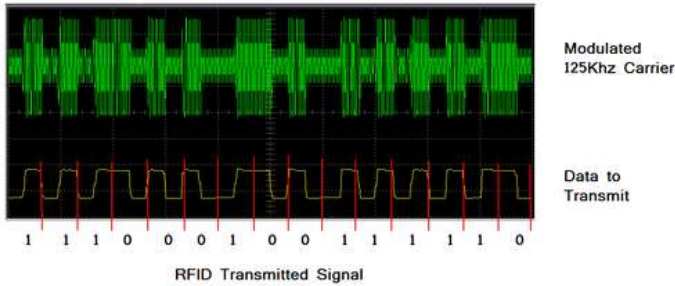
111111111001011100000000000000111110001011110111101001111010000

I will describe what this pattern actually means in the next page.

<http://www.instructables.com/id/A-Universal-RFID-Key/>

One interesting feature of the data transfer between the card and the reader, is that data is encoded using Manchester Encoding, which is a way of encoding data so that it can be transmitted over a single wire ensuring that the clock information is able to be recovered easily. With Manchester encoding, there is always a transition in the middle of a bit. If you want to transmit a 1, the transition would be from low to high, and if you want to transmit a 0, the transition would be from high to low. Because the transitions are in the middle of each bit, you can ensure that you have locked onto valid data. For a detailed description, have a look at this [page](#).

The actual data is transmitted by the card effectively shorting the coil out - this applies an additional load to the transmitter in the reader, which can be detected.



Step 2: Whats stored on the card?

I started by building a RFID card reader (more details in a future article). That showed me the data that was being sent when the card transmitted its information.

The RFID cards that I brought have numbers printed on the back of them. This number says what data the card has included in it.

the card with 0007820706 119,21922 printed on it transmits this pattern:
111111111001011100000000000001111011110101001010101000010101100

The first set of 11111111 bits are the start sequence - it is used to tell the reader that a code is coming - the reader also uses the sequence to lock onto the card data.

Data stored is transmitted in groups of 4 bits, with a parity bit at the end of every group.

The data can be broken up as follows;
00101 11000 00000 00000 01111 01111 01010 01010 10100 00101 0110 0

If we ignore the parity bit at the end of every nibble we have

0010	1100	0000	0000	0111	0111	0101	0101	1010	0010	0110	0
2	C	0	0	7	7	5	5	A	2	CHECKSUM	STOP

This code is 2c 0077 55a2 if we break the code into 3 groups, we have 2c, followed by 0077 (which is 119 in decimal), and finally 55A2, which is 21922 in decimal - this corresponds to the 119,21922.

The same number is also written in another way on these cards 0007820706 (in decimal) is simply the hexadecimal number 7755A2.

WOOT we now understand how the data is stored.

2C is a constant code that is sent with all of the cards. It is simply a facility identifier for this RFID system.

How does the parity and checksum work?

One final piece of data that the card transmits is a checksum word - this is used to ensure that all of the data has been received successfully. Firstly, the parity bit at the end of each nibble of data is Even parity - this means that the transmitter will add a 1 to make sure that each block of data has an 'even' number of '1' bits - So if we look at the '2', which is 0010 in binary - the parity system would detect that there was an odd number of '1' bits, and would add one to compensate. Compare that to the 'C' which is 1100, the parity system would detect that there are an even number of '1' bits, so it would add a zero.

```
00101 2
11000 C
00000 0
00000 0
01111 7
01111 7
01010 5
01010 5
10100 A
00101 2
```

0110 checksum + 0 stop bit

Finally, the checksum is an even parity bit applied to each of the vertical row bits. This way, there is a horizontal and vertical check of every bit sent - everything has to line up, or the reader will simply reject the transmission.

When I decoded the data for my work prox card, it followed a similar sequence here, but (for obvious reasons) I won't actually publish the numbers. Again, part of the sequence was a facility code, and the rest of the sequence held the same number that was printed on the back of the card.



Image Notes

1. This number is all you need to duplicate the card



Image Notes

1. This number is also useful - we can duplicate this card as well

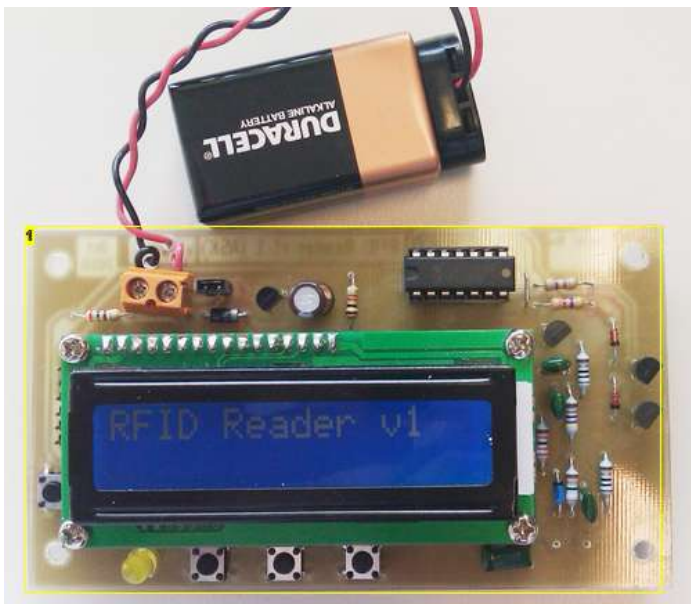


Image Notes

1. My RFID Reader - I will document this on Instructables in the future

Step 3: How do we emulate a card?

So the next step was to identify how to pretend to be a card - I wanted a card that I could type a card number into, so it had to have a microprocessor on it, as well as a keypad to allow the data to be keyed in.

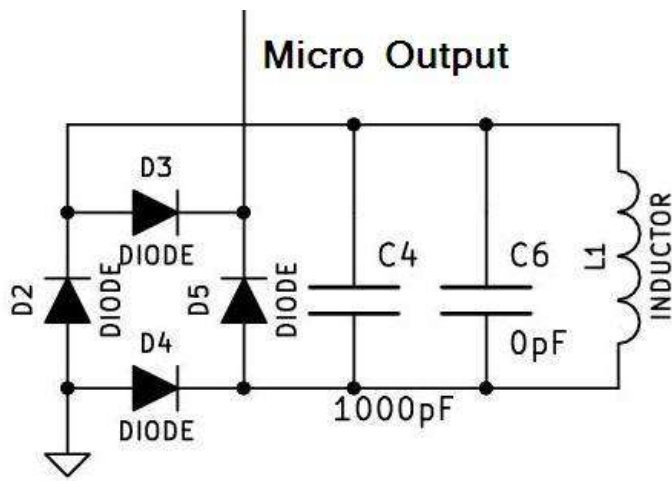
The ATmega manipulates the 125kHz RF field by using a bridge rectifier. When the output of the micro is low, the diodes in the bridge are allowed to be turned on by the current induced in the coil, this effectively short it out. The reader detects the additional load, and a bit transition is detected.

The job of the micro is simply to turn the output on and off in a way that makes sense to our reader. So I created a board that had the micro, a power supply, keypad, and some status LEDs on it.

The attached PDF is the full schematic of the project.

You may notice that c6 is 0pF - That is intentional c6 is a placeholder component allowing me to either use a 1000pF surface mount cap, or a 1000pF through hole cap.

The coil is 100 turns of fine wire wound on an open former that is just smaller than the card border.



File Downloads



RFIDSpoofer-schematic.pdf (54 KB)

[NOTE: When saving, if you see .tmp as the file ext, rename it to 'RFIDSpoofer-schematic.pdf']

Step 4: The Software - Entering data into our card

The software was next. Using the Arduino IDE, I implemented a simple menu system that allowed me to enter the relevant facility and CardID data directly from the keypad. I also provided a way of displaying the data using the LEDs that I mounted on the board.

One problem I came across, was when I was calculating the card data (parity and checksum) on the fly - To be read successfully, the card has to output data in real time (most readers need a number of sequential valid reads), and adding subroutine and calculation delays caused the card to output invalid data as far as the reader was concerned. I worked around this problem by populating an array of bits that gets sent when the card is in transmit more. That way, the calculations are done only once.

When the card is powered up, it waits for the 'mode' button to be pressed. The current mode number is displayed using a set of 4 LEDs. Each press on the 'mode' button increments the current mode. Once the correct mode is displayed, then the 'enter' key starts that function executing.

MODE 1 - Enter low power (sleep) mode

The card enters a low power mode, waiting for the reset button to be pressed to re-awaken it

MODE 2 - Enter a Hex Facility ID

The card waits for 2 digits to be entered signifying the facility code for this system (In this case, it is 2C) - The software defaults to 2C - so this does not need to be entered.

MODE 3 - Decimal Card ID

The card waits for 8 digits to be entered signifying the CardID for the card to be spoofed (In this case, it is 07820706) - This is the long number printed on the back of the card, not the 119,21922 number.

MODE 4 - Dump the facility and Card ID

The Facility and Card ID are Dumped as Hex numbers using the 4 Leds at the top of the card.

MODE 5 - Emulate a card

The card enters emulation mode - all LEDs are turned off. Emulation mode can only be exited by pressing the reset button.

The software relies on Mark Stanley's and Alexander Brevig's Keypad Library <http://www.arduino.cc/playground/Code/Keypad>

File Downloads



RFIDSpoofer_Instructables.pde (14 KB)

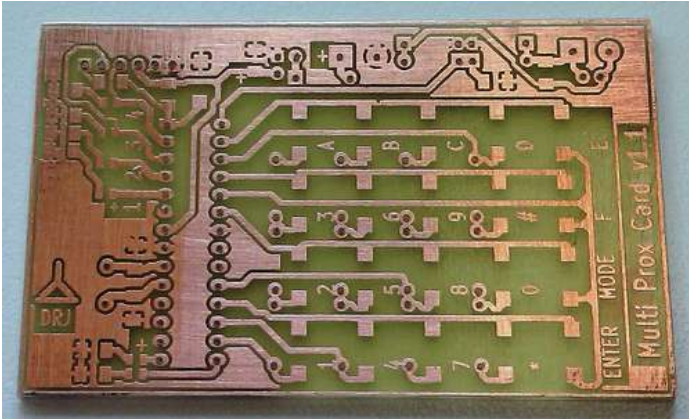
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'RFIDSpoofer_Instructables.pde']

Step 5: Etching the PCB

As per standard, I used toner transfer onto magazine paper to etch a board. If you want to see the details, have a look [here](#).

The etched PCB had its edges cleaned up a bit using a file, and holes were drilled for the IC legs.

Attached are the PDF files that I used for the Toner Transfer.



File Downloads



RFIDSpoofer-Silk.pdf (31 KB)

[NOTE: When saving, if you see .tmp as the file ext, rename it to 'RFIDSpoofer-Silk.pdf']



RFIDSpoofer-PCB.pdf (34 KB)

[NOTE: When saving, if you see .tmp as the file ext, rename it to 'RFIDSpoofer-PCB.pdf']

Step 6: Mounting the components

To keep the project the same size as a normal prox card, I decided to make it on a small PCB that was the same size as a business card.

I decided to use surface mount push buttons that I brought from eBay, so that meant that all of the components must be soldered onto the copper side of the PCB to allow the buttons to be mounted and labeled.

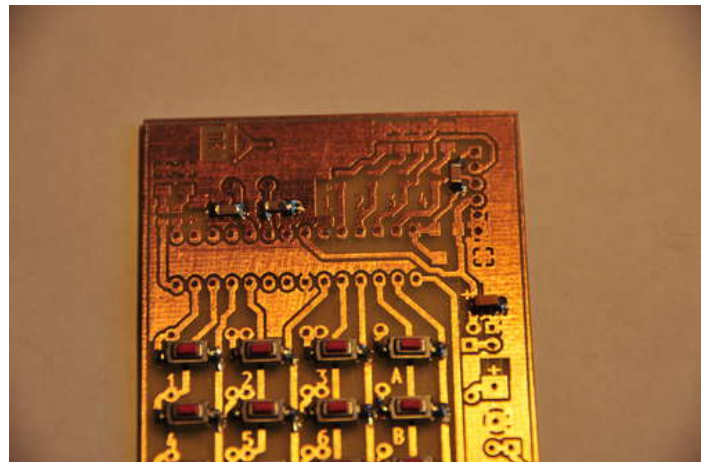
I started by soldering the push buttons, then I mounted the LEDs, resistors and capacitors. I had to install the 16MHz crystal on the bottom of the PCB, as I did not have a surface mount crystal. I also installed 12 jumpers on the back of the card to connect the key columns together.

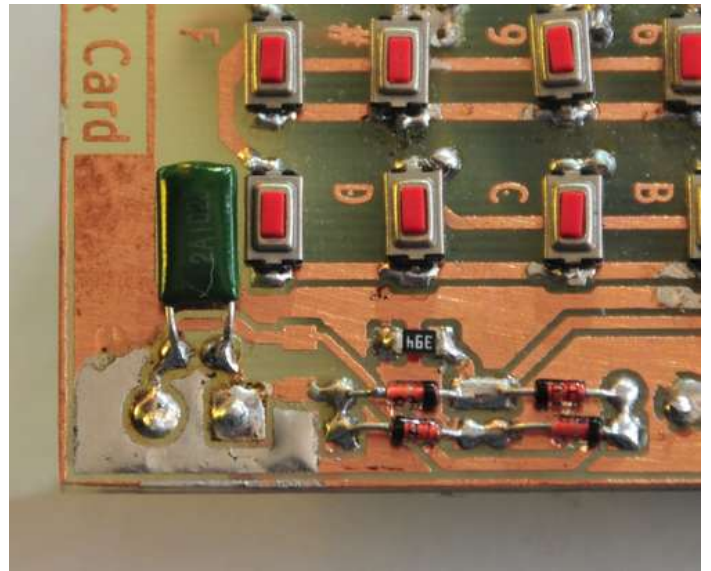
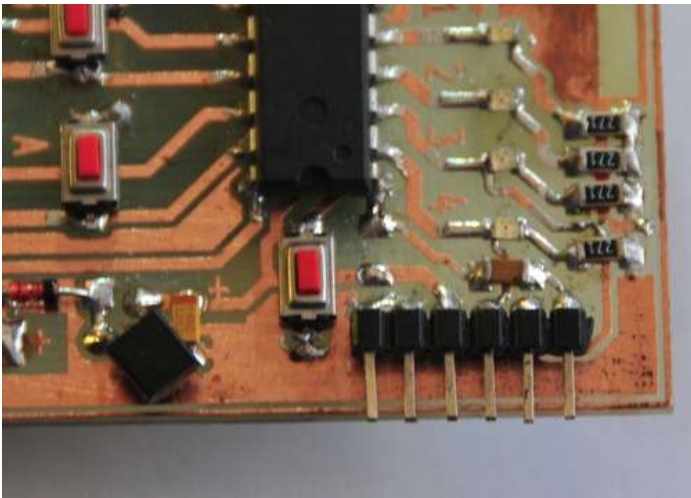
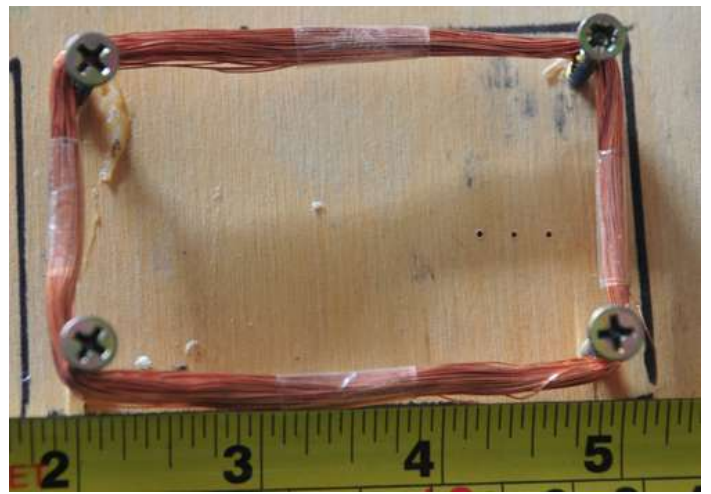
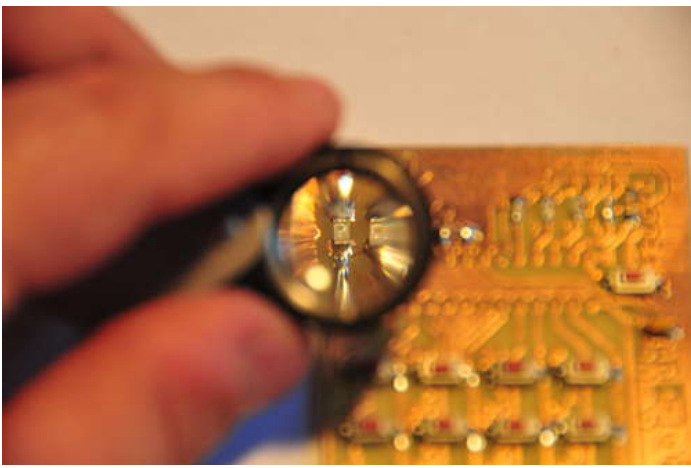
The ATmega168 was mounted next. I did not use a socket, as I wanted to reduce the board thickness.

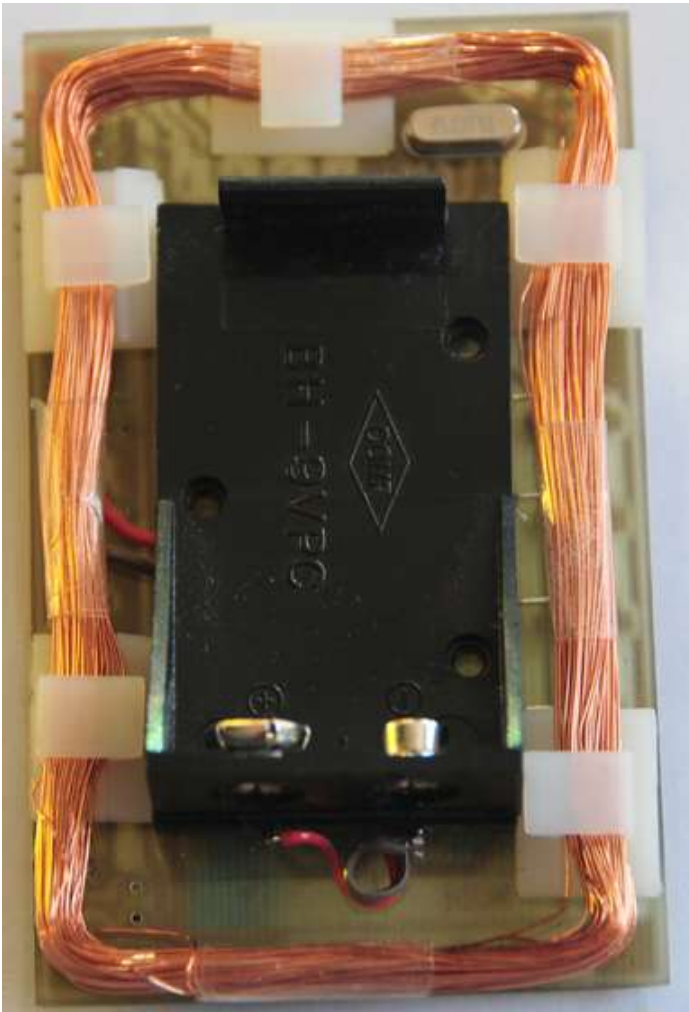
Next, I wound the coil - I used a piece of scrap timber, with 4 screws mounted on it, and counted 100 turns of 0.25mm diameter coil winding wire. Before I removed the coil from the mounts, I wound a small amount of clear tape around each edge to make sure that the coil didn't unwind.

Then, I mounted the coil on the back of the PCB, along with a small battery holder.

I was pretty happy with the result of my handiwork.







Step 7: Programming the Micro

I used a standard 6 pin header mounted on the PCB to allow a FTDI 5V USB-232 cable to be used to program the chip in-situ - this was especially important, as the ATmega chip is soldered directly to the PCB, so it couldn't be removed for insertion into a normal Arduino PCB- This is a small price to pay to have a nice compact project.

The chip was programmed using the .pde Arduino sketch that was supplied in Step 4 - using the normal Arduino IDE.

The .PDE file that I have provided is tailored to the standard cheap eBay RFID systems. It is not the version for the other IFID readers I have access too..... (I just thought I would mention that :-)

Step 8: Testing the project

Testing was a breeze - I typed the relevant code into the keypad, swiped the board against the reader, and was rewarded with a satisfying 'BEEP' indicating that the read was successful.

Testing at the other readers I have access to was just as rewarding, and scored infinite geek points!!!



Step 9: Further steps

This was a 'to prove I could do it' project - I have completed it, so it now sits on my shelf at work to remind others that simple RFID systems are simply not secure.

You are welcome to adapt the project however you would like to, and while you may have the skeleton keys to the kingdom, you still need the little numbers on the back of the access card before you can use the key yourself.

I have considered modifying my card so that it works as all of the compatible RFID tags that I hold. In my job, I need have access to multiple work sites, and it would be great to use the one card, but I don't think that would be a great idea.....

Will this work on all RFID systems?

No it won't. This is a good thing.

The first RFID systems deployed years ago used very simple protocols, based on the intelligence of the chip in the card - They also used a low frequency (125kHz) carrier.

More modern systems use a number of techniques to ensure security, such as one time codes; cryptography; use bi-directional communication; use internal passwords, and use much higher frequencies. So spoofing these systems is a lot more work.

But there are a large number of low tech systems in place now.

What can I do to protect my system?

Firstly, don't equate cards to physical keys - in simple systems they are not equivalent.

Don't give out visitor cards - They are easily duplicated - If you do need Visitor cards, then implement a system where they are only active when they have been issued.

Enable Pass Back systems - If the card system believes you are in a particular room, make sure that the card can't be used in other rooms at the same time.

Remove the numbers from the back of the cards - while they may make it easier to enter card details, but they also make it easy for somebody to use the details for their own purposes.

Finally, look at how to upgrade your access system to a card system that is not trivially spoofed using \$15 worth of parts. And - No, purchasing a new system from eBay for \$15 is not the answer....

Related Instructables



AVR/Arduino RFID Reader with UART Code in C by nevdull



How to connect Arduino and RFID by otaviousp



Interfacing RFID with 8051 Microcontroller (video) by ashoksharma



iPhone RFID Reader by OniDaito



How to block/kill RFID chips by w1n5t0n





RFID Reader Detector and Tilt-Sensitive RFID Tag by nmarquardt


Comments


50 comments [Add Comment](#)


[view all 61 comments](#)


 **jfmateos** says: Feb 25, 2011. 4:12 AM [REPLY](#)
I have replicated your work using a PIC microcontroller and it works properly using white cards like yours, but I cannot understand the meaning of the blue keyfob number... it is also a decimal number?
Thanks and best regards from Madrid


 **jfmateos** says: Feb 26, 2011. 12:28 AM [REPLY](#)
I have proven that it is a decimal number, but you must guess the first to hexadecimal digits. In my blue keyfobs, these two first digits are not 2C, but 04 in some cases and 82 in others


 **tgeorgebogdan** says: Dec 20, 2010. 3:27 AM [REPLY](#)
can you make a hex file for this code? because i don't have an adruino ide?


 **drj113** says: Dec 21, 2010. 2:52 AM [REPLY](#)
I am unable to create a hex file for you - the Arduino IDE is available as a free download from www.arduino.cc under the 'download' tab

 **adi0241** says: Feb 3, 2011. 4:52 AM [REPLY](#)
finally i made a usb to serial communication and instaled the arduino ide but it seems to be an error eaven if i loaded the "StackList" directory in "libraries" directory. the compile error looks like this:
core.a(main.cpp.0): in function 'main'
undefined references to 'setup' and 'loop'
can you help me?


 **tgeorgebogdan** says: Dec 22, 2010. 6:28 AM [REPLY](#)
how do you compile the *pde ? if you compile this file the software dose'nt create any hex file? i've inported the file in avrstudio with gcc plugin instaled and i have one error about the makefile.


 **drj113** says: Dec 22, 2010. 12:19 PM [REPLY](#)
Using the Arduino IDE, it does the compiling in the back end using avrstudio - but it does not appear to keep the temporary files.
What error are you getting?

 **tgeorgebogdan** says: Dec 22, 2010. 3:35 AM [REPLY](#)
how can i write to the microcontroller the software if i don't have an arduino board? i only have a home made usb to serial converter with ftdi232rl. thanks in advance.

 **drj113** says: Dec 22, 2010. 12:21 PM [REPLY](#)
I use an FTDI USB-232 cable from sparkfun. The neat thing about the AtMega chips with the boot loader on them is that you don't have to use an arduino to program them, a FTDI cable plugged into the 6 pin header on the PCB works fine.

 **diwib** says: Jan 28, 2011. 7:22 PM [REPLY](#)
Do you know how would I turn this into a 13.56 Mhz card emulator?

 **drj113** says: Jan 28, 2011. 10:30 PM [REPLY](#)
That would be a very interesting task - Firstly, the tank coil would have to be resonant at 13.56 Mhz - less turns. Then the delays in the code would have to be reduced to allow the card to run faster.... I will have a look at this when I can get hold of a 13.56 Mhz card and reader to play with. It may not be a task for an Atmega CPU - they may be too slow

 **diwib** says: Jan 29, 2011. 3:03 AM [REPLY](#)
I've been using this one together with my arduino: http://www.seeedstudio.com/depot/1356mhz-rfid-module-iosiec-14443-type-a-p-196.html?cPath=144_153&zenid=a6f99b81ba387c32ea4416bf123ff325
It works like a charm. Of course it's not using an Atmega, but maybe the datasheets help you.
I'd love to help but I'm afraid I'm way too new to electronics.

 **rglide** says: Jan 25, 2011. 8:34 AM [REPLY](#)
Do you have the RFID Reader on here?



merter says:

Yes I made it. with 16f628a+LM358 used. are you interesting. please contact me. merter@baltali.net

Jan 27, 2011. 6:48 AM [REPLY](#)



flashmandv says:

Hi All,

what I need to change in the schematics and software to make it work with PSK modulation?
Please help, since many rfid readers are using only PSK :(

Dec 4, 2010. 7:58 AM [REPLY](#)



The Real Elliot says:

Awesome project! I'm going to play around with your antenna / rectifier-as-load idea.

Any hints for tuning the antenna? Or did you just wing it? Seems to me, you'd want to tune it in to 125 kHz for maximum range?

And some comments on the security -- even the unencrypted RFID is better than a traditional key, right? What you showed here is that the RFID is only secure if the card number is secret. This is just the same as a traditional key -- if you tell me the key blank type and the cutting depths, you're compromised.

To RFID's favor, the keyspace is larger, and the keys are _much_ easier / cheaper to revoke. (So I quibble with your "don't give out visitor cards" as well. They're cheap enough that you can give 'em out freely and revoke the code when you don't want that visitor to have access again. I guess if you can't revoke codes, you don't want to be lending your keys out...)

In short, taking the spoofability of RFID (at any frequency or with any encryption scheme) as a given, it's compromising the secret code that breaks security - same as a physical key.

I quibble too much. Rad project and beautiful SMD board fab handiwork!

Dec 1, 2010. 3:43 PM [REPLY](#)



drj113 says:

I tuned the antenna using a signal generator and a CRO - Essentially I selected caps that provided the greatest output at 125Khz.

Unencrypted RFID is better than a physical key, as long as you maintain key control - that's my complaint about visitor cards - most organisations have a set of cards that are active that get re-cycled, They don't understand that once somebody has access to the card, then the card can be spoofed.

Much better to have an environment where visitor cards are either used once, then discarded and disabled, or at the very least, only enabled when they have been actually issued. And certainly not enabled after hours :-)

Thanks for the positive comments - I do enjoy building these projects.

Dec 1, 2010. 5:12 PM [REPLY](#)



kokko says:

sorry!

sorry ... but the capacitor C6 .. mistake or to see and 0pf??

I'm sorry for my ignorance ...

thanks!

Kokko greetings :-)

Dec 1, 2010. 1:06 PM [REPLY](#)



drj113 says:

c6 is 0pF just as a place holder in the schematic editor- I wanted to have EITHER a through hole or a surface mount cap - Just use either c4 or c6

Dec 1, 2010. 2:06 PM [REPLY](#)



Zirgriz758 says:

Sry but you guys aren't understanding what I'm asking. I'm trying to ask how you converted the numbers on the back of the card to the binary numbers.

Could you please explain how you converted the numbers.

Thanks

Nov 29, 2010. 3:29 PM [REPLY](#)



drj113 says:

Firstly, there is the process of converting a number from decimal (Base 10) to Hexadecimal (base 16). In computing, we often use Hexadecimal as a way of representing numbers, as it maps into bits really well.

In Decimal, there are 10 available symbols (0-9) that we can create numbers from - In hexadecimal, there are 16 (0-9 and A-F)
Here is a table of numbers from 0-17 as a start - I am using a 4 bit binary representation

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101

Nov 29, 2010. 4:44 PM [REPLY](#)

14	E	1110
15	F	1111
16	10	10000
17	11	10001

To convert a number from Decimal to Hexadecimal, you have to perform multiple operations - lets look at a simple case of converting the number 20 in Decimal to Hex.

Take 20 - Find out how many 16's fit into that - subtract the number of 16s from it, and use the remainder

$$20 = 1 * 16 + 4 = 14 \text{ in Hex}$$

Same for 254

$$254 = 15 * 16 + 14$$

15 is F in Hex, and 14 is E, so 254 = FE in hex

The numbers on the back are in Decimal - Lets take the number 007820706 as an example

Using a decimal to hex conversion calculator, 7820706 in decimal is 7755A2 in hex.

This hex number (7755A2) can be written 0111 0111 0101 0101 1010 0020. The entire card needs a 10 digit hex number, We already have a facility code of 2C, and we just zero pad the 6 digit number to fit, so in all, our card code needs to be 2 C 0 0 7 7 5 5 A 2.

0010 1100 0000 0000 0111 0111 0101 0101 1010 0010

The actual sequence transmitted needs parity bits. These bits are added to validate that every 4 bits is successfully sent. Even parity is used, so a bit is added to ensure that every nibble has an EVEN number of bits

so to send a 2, we would send 0010 + a parity bit, which would be a 1 to make the number of 1 bits even (two 1 bits in total) --- 00101

To send a zero, we would send 0000 + a parity bit, which would be a 0, still making sure that the number of 1 bits is even, (in this case zero) - 00000

So the sequence

0010 1100 0000 0000 0111 0111 0101 0101 1010 0010

becomes

00101 11000 00000 00000 01111 01111 01010 01010 10100 00101 with the parity bits interspersed.

Then the final column parity is added - The column parity is the same as the row parity, but in this case the first bit of the column parity evens out the number of 1 bits on the first position of each nibble

In this case, the first bits of every nibble are 0 1 0 0 0 0 0 1 0 - so the parity bit would be 0

And the second bits are 0 1 0 0 1 1 1 1 0 0 - so the parity bit would have to be 1

And the third bits are 1 0 0 0 1 1 0 0 1 1 - so the third parity bit would be a 1

and finally, the fourth bits are 0 0 0 0 1 1 1 1 0 0 - so the final parity bit would be 0

This means that the column checksum would be 0110

So our final sequence would look like

00101 11000 00000 00000 01111 01111 01010 01010 10100 00101 followed by the column parity of 0110 and a stop bit of 0

00101 11000 00000 00000 01111 01111 01010 01010 10100 00101 01100

Hopefully that makes sense?



Zirgriz758 says:

Nov 30, 2010. 9:13 PM [REPLY](#)

Thanks a lot for answering my question it really helped and sry for bothering you with all of my questions. But i just have one more. How do you figure out the facility code without a rfid reader can you use the numbers on the card to figure it out?



drj113 says:

Dec 1, 2010. 2:57 AM [REPLY](#)

So far, all of the facility codes for the systems have been constant. This has applied to a variety of eBay RFID systems



dbell says:

Nov 30, 2010. 1:50 PM [REPLY](#)

What part number diodes did you use for the switch?

Dave



drj113 says:

Nov 30, 2010. 7:18 PM [REPLY](#)

Any switching diode is fine - it is not speed critical - I used 1n914



1161858 says:

Nov 30, 2010. 6:09 PM [REPLY](#)

you are currently my hero this is like the ultimate geek fantasy its got the whole james bond sort of feel to it "the key to open any rfid lock" mad props



tagno25 says:

Nov 29, 2010. 4:54 AM [REPLY](#)

Does this work with HID Prox Card II?



drj113 says:

Nov 29, 2010. 12:22 PM [REPLY](#)

It may - I don't have any HID Prox Card II's to read - The encoding of the number on the rear of the card is likely to be different though - That will need some experimentation.



Zirgriz758 says:

Nov 28, 2010. 4:07 PM [REPLY](#)

@askjerry

Thanks that sort of helped. But when I said the numbers on the back I men't the numbers on the back of the card. I was trying to say how he got the first binary code

(11111111001011100000000000001110111101010010101000010101100/ I know the first 9 1's are the starting code) from the numbers on the back of the card (0007820706 119,21922). Is the anyone who can tell me how the got that, like I said before I put the card number in a binary converter and got a different binary code.



drj113 says:

Nov 28, 2010. 5:17 PM [REPLY](#)

Ignoring the start sequence (9 1 bits) the code looks like this:

```
0010111000000000000000111011110101001010101000010101100
```

each 4 bits nibble is sent as a 5 bit sequence (a parity bit is added) so this becomes:

```
00101 11000 00000 00000 01111 01111 01010 01010 10100 00101 01100
```

Lets drop the list bit (parity) this gives us

```
0010 1100 0000 0000 0111 0111 0101 0101 1010 0010 0110
```

The last nibble is just the checksum - lets drop it:

```
0010 1100 0000 0000 0111 0111 0101 0101 1010 0010
2    C    0    0    7    7    5    5    A    2
```

2c is effectively a facility code

007755A2 is the hex version of the card code - which in decimal is 07820706 - which corresponds to the first long number on one of the cards that I have.

does that make sense?

Doug



lbrewer42 says:

Nov 28, 2010. 11:27 AM [REPLY](#)

Flynn Lives!



drj113 says:

Nov 28, 2010. 2:31 PM [REPLY](#)

I have to ask ??????????



lbrewer42 says:

Nov 28, 2010. 3:15 PM [REPLY](#)

The original movie TRON. Flynn used a card that looks very similar to this to enter ENCOM's facility. This is the first thing I thought of when I saw the pic of the card in this 'ible, and as i scrolled down farther and saw a pic of it with an entry keypad (similar also to the movie!), the parallel was just too good not to mention.



lbrewer42 says:

Nov 28, 2010. 3:18 PM [REPLY](#)

BTW - to us geeks old enough to have seen the movie on the silver screen when it was first released, this is a high complement :)



askjerry says:

Nov 28, 2010. 3:47 PM [REPLY](#)

And to us geeks who were waiting for the sequel... it is only days away as of this writing!! But... that's a whole topic in and of it's self.

Flinn Lives!!! (Thumbs up... both of them.)

Jerry



Zirgriz758 says:

Nov 28, 2010. 3:37 PM [REPLY](#)

I don't really understand how you got the binary code with just though's numbers on the back. I've used all the numbers in a decimal/binary calculator but didn't get the same results. Could you please explain how you got the binary code?
Thanks.



askjerry says:

Nov 28, 2010. 3:45 PM [REPLY](#)

Look at each DIGIT of the code... for example, lets just take the first 4 digits from the card in the picture 0007....

0000 = 0
0000 = 0
0000 = 0
0111 = 7
... just keep going...

here are all 16 digits.

0000 = 0
0001 = 1
0010 = 2
0011 = 3
0100 = 4
0101 = 5
0110 = 6
0111 = 7
1000 = 8
1001 = 9
1010 = A (Decimal 10)
1011 = B (Decimal 11)
1100 = C (Decimal 12)
1101 = D (Decimal 13)
1110 = E (Decimal 14)
1111 = F (Decimal 15) Largest number you can make with 4 digits.

Does that help?
Jerry



sswcharlie says:

Nov 28, 2010. 3:40 PM [REPLY](#)

What a cool device.

I want a multi reader that reads tags from different locations for a model railroad. To know where items are. Each antennae at a different location would have its own id.

swchuck aat gmail.com



930913 says:

Nov 27, 2010. 9:24 AM [REPLY](#)

Nice! I saw that magstripe emulator too. Great job



930913 says:

Nov 28, 2010. 3:20 PM [REPLY](#)

I know- im fascinated by RFID. Write up on Adafruit, once again, cool.



gnume says:

Nov 28, 2010. 7:16 AM [REPLY](#)

am. what are you talking about ?



drj113 says:

Nov 28, 2010. 12:25 PM [REPLY](#)

There was a recent swipe card emulator from hack a Day - It was pretty cool. I really liked tha coupling method - that was neat.

This one does not use magnetic swipe technology - it is RFID. Byt thanks for the positive feedback - it is always appreciated :-)



gnume says:

Nov 28, 2010. 1:16 PM [REPLY](#)

can i get a link please ?



skylen says:

Nov 29, 2010. 12:14 PM [REPLY](#)

Here are the links for Craig's magnetic card spoofer project, which includes both the reader for the original card and the spoofer itself which emulates a magnetic card being swiped across a reader.

Spoofing magnetic swipe cards" project: <http://flashingleds.wordpress.com/2010/11/12/magnetic-swipe-card-spoofers>

Hackaday post: <http://hackaday.com/2010/11/27/surprisingly-simple-magnetic-card-spoofers/>



gnume says:
thanks

Nov 29, 2010. 1:07 PM [REPLY](#)



shichae says:
Awesome idea, and excellent implementation!

Nov 28, 2010. 12:36 PM [REPLY](#)



geoff_fellows says:
Interesting project. Can't wait to build it for my class.
Please use <http://physics.nist.gov/cuu/Units/prefixes.html> kHz.

Nov 28, 2010. 12:35 PM [REPLY](#)



infraled says:
Amazing talent you have, thank you for sharing it with the rest of us!

Nov 28, 2010. 11:58 AM [REPLY](#)



gardoglee says:
Correct me if i am wrong, but aren't some of the auto manufacturers using RFID in thier contactless keys? Or to put it another way, with this device is it now easier to steal a 2011 Lincoln than a 1955 Chevy? I never did think RFID would resist hacking all that long.

Nov 28, 2010. 9:13 AM [REPLY](#)

BTW, if I remember correctly many RFID chips also have some amount of memory which can be both read and updated, some of them as much as 8kB. Can your base design be modified to also read this, presuming it is not one of the better quality chips which encrypt the memory in place on the RFID chip?



drj113 says:
Hi,

Nov 28, 2010. 11:48 AM [REPLY](#)

Autos do use RFID technologies in their transponders - but the great thing about them, is that they are secure - They use rolling codes, and crypto. So this card does not assist with stealing cars. The point of this article was to highlite that the older technologies are not as secure as people think.

[view all 61 comments](#)