Arduino Mega 2560 Crazy kit Manual

B2CQSHOP

2011 - 10 - 30

Contents

The Kit Contents

- 1 X Arduino Mega 2560 board (High quality , 100% clone board)
- 1 X 128 * 64 LCD
- 1 X RFID module
- 1 X Rectangle tag
- 1 X Round tag
- 1 X Mega IO Sensor Shield
- 1 X bundles Breadboard jumper wires
- 1 X High quality Breadboard
- 1 X 4*4 Matrix Keypad
- 1 X SD card Module
- 1 X 8*8 Matrix LEDs
- 1 X IR controller
- 1 X IR receiver
- 1 X IR sender
- 1 X AT24C64 EEPROM Chip
- 1 X 74HC595N
- 1 X DS1307 RTC Chip
- 1 X 32.768 KHz watch crystal
- 1 X CR2032 battery socket
- 1 X 5V Relay
- 1 X 5V Buzzer
- 1 X Photoresistor
- 10 X Green 3mm LEDs
- 10 X Yellow 3mm LEDs
- 10 X Red 3mm LEDs
- 10 X 220 ohm Resistors
- 10 X 1K ohm Resistors
- 10 X 10K ohm Resistors
- 1 X 9V power adapter
- 1 X 7seg-4digit LED display
- 1 X 10K Rotary Potentiometer
- 1 X 3V-6V motor
- 5 X 2N2222 transistors
- 5 X 1N4001 Diode
- 2 X 100nF capacitor
- 1 X 20 pin Straight male header

Introduction

Thank you for purchasing this Arduino Mage 2560 Kit. You are now well on your way inyour journey into the wonderful world of the Arduino and microcontroller electronics.
This book will guide you, step by step, through using the Starter Kit to learn about the Arduino hardware, software and general electronics theory. Through the use of electronic projects we will take you from the level of complete beginner through to having an intermediate set of skills in using the Arduino.

The purpose of this book and the kit is to give you a gentle introduction to the Arduino, electronics and programming in C and to set you up with the necessary skills needed to progress beyond the book and the kit into the world of the Arduino and microcontroller electronics.

The booklet has been written presuming that you have no prior knowledge of electronics, the Arduino hardware, software environment or of computer programming. At no time will we get too deep into electronics or programming in C. There are many other resources available for free that will enable you to learn a lot more about this subject if you wish to go further. The best possible way to learn the Arduino, after using this kit of course, is to join the Arduino Forum on the Arduino website and to check out the code and hardware examples in the 'Playground' section of the Arduino website too.
We hope you enjoy using the kit and get satisfaction from creating the projects and seeing your creations come to life.

How to use it

The book starts off with an introduction to the Arduino, how to set up the hardware, install the software, etc.

We then explain the Arduino IDE and how to use it before we dive right into some projects progressing from very basic stuff through to advanced topics. Each project will start off with a description of how to set up the hardware and what code is needed to get it working. We will then describe separately the code and the hardware and explain in some detail how it works.

Everything will be explained in clear and easy to follow steps. The book contains a lot of diagrams and photographs to make it as easy as possible to check that you are following along with the project correctly.

What you will need

Firstly, you will need access to the internet to be able to download the Arduino IDE (Integrated Development Environment) and to also download the Code Samples within this book (if you don't want to type them out yourself) and also any code libraries that may be necessary to get your project working.

You will need a well lit table or other flat surface to lay out your components and this will need to be next to your desktop or laptop PC to enable you to upload the code to the

Arduino. Remember that you are working with electricity (although low voltage DC) and therefore a metal table or surface will first need to be covered in a non-conductive material (e.g. tablecloth, paper, etc.) before laying out your materials.

Also of some benefit, although not essential, may be a pair of wire cutters, a pair of long nosed pliers and a wire stripper. A notepad and pen will also come in handy for drawing out rough schematics, working out concepts and designs, etc.

Finally, the most important thing you will need is enthusiasm and a willingness to learn. The Arduino is designed as a simple and cheap way to get involved in microcontroller electronics and nothing is too hard to learn if you are willing to at least 'give it a go'. This Arduino Kit will help you on that journey and introduce you to this exciting and creative hobby.

(Thank you very much for the book of " ***Mike McRoberts*** " , book title is " **Arduino_Starter_Kit_Manual-Mar2010** "

Thank you very much for the book of "***Michael.Margolis*** ",book title is "**Arduino Cookbook 2011**".

And Thank you very much the team of **B2CQSHOP** , too)

B2CQSHOP

Webstite : ***www.b2cqshop.com***

Email : **Qshopb2c@gmail.com**

MSN : **B2cqsohp@hotmail.com**

Skype : **evanjoo**

What exactly is an Arduino?

an Arduino is a tiny computer that you can program to process inputs and outputs going to and from the chip.

The Arduino is what is known as a Physical or Embedded Computing platform, which means that it is an interactive system, that through the use of hardware and software can interact with it's environment.

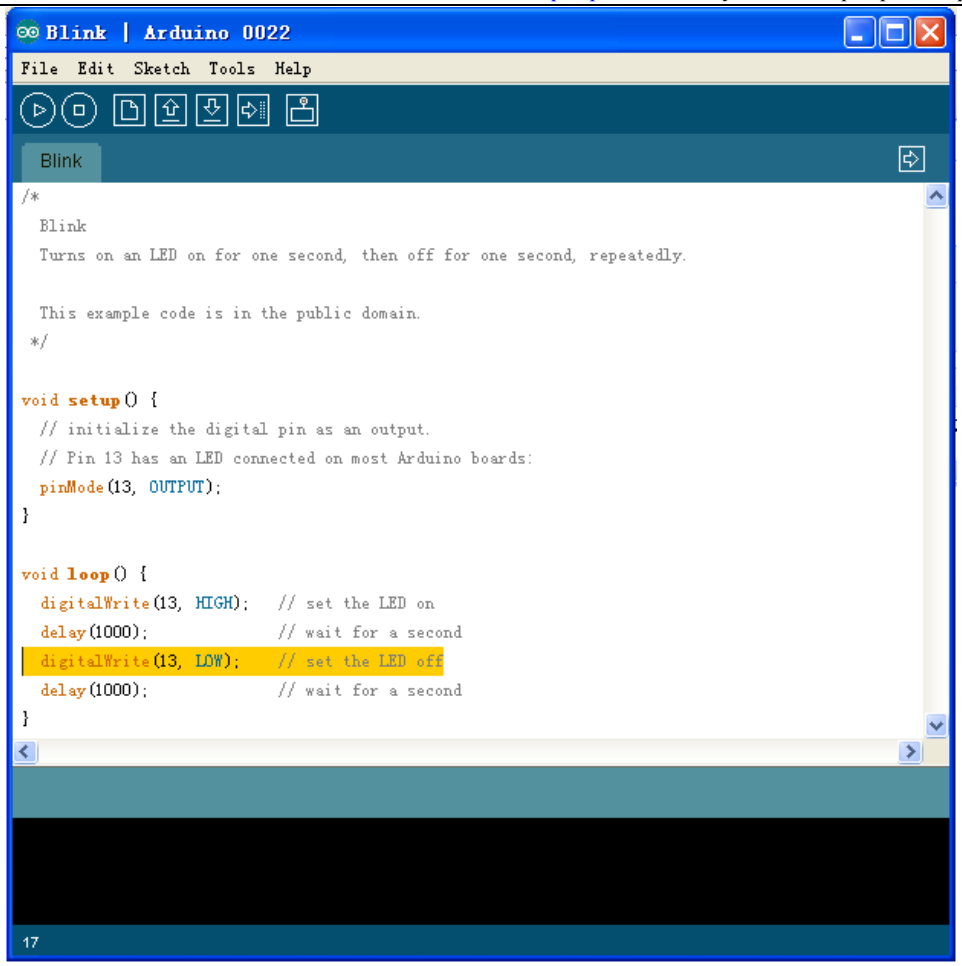
For example, a simple use of the Arduino would be to turn a light on for a set period of time, let's say 30 seconds, after a button has been pressed (we will build this very same project later in the book). In this example, the Arduino would have a lamp connected to it as well as a button. The Arduino would sit patiently waiting for the button to be pressed. When you press the button it would then turn the lamp on and start counting. Once it had counted 30 seconds it would then turn the lamp off and then carry on sitting there waiting for another button press. You could use this set-up to control a lamp in an under-stairs cupboard for example. You could extend this example to sense when the cupboard door was opened and automatically turn the light on, turning it off after a set period of time.

The Arduino can be used to develop stand-alone interactive objects or it can be connected to a computer to retrieve or send data to the Arduino and then act on that data (e.g. Send sensor data out to the internet).

The Arduino can be connected to LED displays, LED's. Dot Matrix displays, buttons, switches, motors, temperature sensors, pressure sensors, distance sensors, SD card, webcams, printers, GPS receivers, ethernet modules, and so on .

The Arduino board is made of an an Atmel AVR Microprocessor, a crystal or oscillator (basically a crude clock that sends time pulses to the microcontroller to enable it to operate at the correct speed) and a 5-volt linear regulator. Depending on what type of Arduino you have, you may also have a USB connector to enable it to be connected to a PC or Mac to upload or retrieve data. The board exposes the microcontroller's I/O (Input/Output) pins to enable you to connect those pins to other circuits or to sensors, etc.

To program the Arduino (make it do what you want it to) you also use the Arduino IDE (Integrated Development Environment), which is a piece of free software, that enables you to program in the language that the Arduino understands. In the case of the Arduino the language is C. The IDE enables you to write a computer program, which is a set of step-bystep instructions that you then upload to the Arduino. Then your Arduino will carry out those instructions and interact with the world outside. In the Arduino world, programs are known as ' Sketches ' .

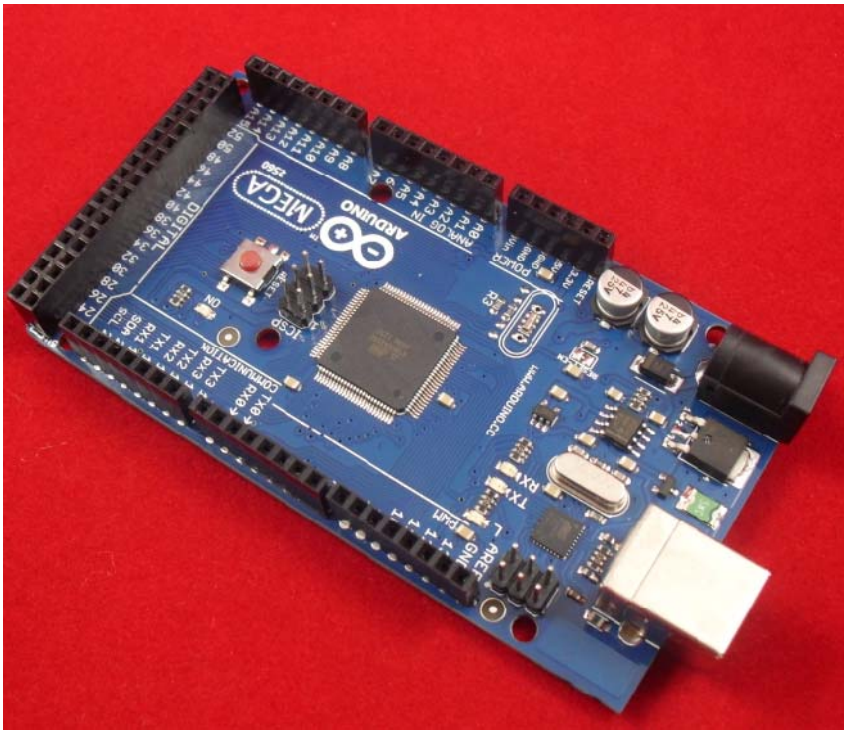


The Arduino hardware and software are both Open Source, which means the code, the schematics, design, etc. are all open for anyone to take freely and do what they like with it.

This means there is nothing stopping anyone from taking the schematics and PCB designs of the Arduino and making their own and selling them. This is perfectly legal, and indeed the whole purpose of Open Source, and indeed the Freeduino that comes with the Earthshine Design Arduino Starter Kit is a perfect example of where someone has taken the Arduino PCB design, made their own and are selling it under the Freeduino name. You could even make your own Arduino, with just a few cheap components, on a breadboard.

The only stipulation that the Arduino development team put on outside developers is that the Arduino name can only be used exclusively by them on their own products and hence the clone boards have names such as Dfrobot, Freeduino, Boarduino, Roboduino, etc.

As the designs are open source, any clone board, such as the Freeduino is 100% compatible with the Arduino and therefore any software, hardware, shields, etc. will all be 100% compatible with a genuine Arduino.



The Arduino can also be extended with the use of ‘Shields’ which are circuit boards containing other devices (e.g. GPS receivers, LCD Displays, Ethernet connections, etc.) that you can simply slot into the top of your Arduino to get extra functionality. You don’t have to use a shield if you don’t want to as you can make the exact same circuitry using a breadboard, some veroboard or even by making your own PCB’s.

There are many different variants of the Arduino available. The most common one is the Diecimila or the Duemilanove. You can also get Mini, Nano and Bluetooth Arduino’s.

New to the product line is the new Arduino Mega 2560 and Mega 1280 with increased memory and number of I/O pins.

Probably the most versatile Arduino, and hence the reason it is the most popular, is the Duemilanove. This is because it uses a standard 28 pin chip, attached to an IC Socket. The beauty of this systems is that if you make something neat with the Arduino and then want to turn it into something permanent (e.g. Or understairs cupboard light), then instead of using the relatively expensive Arduino board, you can simply use the Arduino to develop your device, then pop the chip out of the board and place it into your own circuit board in your custom device. You would then have made a custom embedded device, which is really cool. Then, for a couple of quid or bucks you can replace the AVR chip in your Arduino with a new one. The chip must be pre-programmed with the Arduino Bootloader to enable it to work with the Arduino IDE, but you can either burn the bootloader yourself if you purchase an AVR Programmer, or you can buy these preprogrammed from many suppliers around the world. Of course, Earthshine Design provide preprogrammed Arduino chips in it’ store for a very reasonable price.

If you do a search on the Internet by simply typing ‘Arduino’ into the search box of your favourite search engine, you will be amazed at the huge amount of websites dedicated to the Arduino. You can find a mind boggling amount of information on projects

made with the Arduino and if you have a project in mind, will easily find information that will help you to get your project up and running easily.

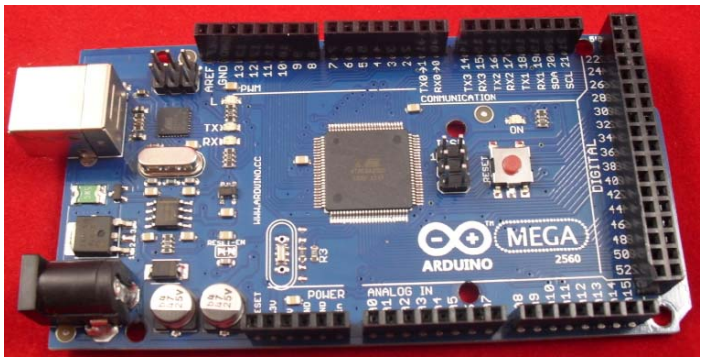
The Arduino is an amazing device and will enable you to make anything from interactive works of art to robots. With a little enthusiasm to learn how to program the Arduino and make it interact with other components a well as a bit of imagination, you can build anything you want.

This book and the kit will give you the necessary skills needed to get started in this exciting and creative hobby.

So, now you know what an Arduino is and what you can do with it, let's open up the starter kit and dive right in.

Getting Started

This section will presume you have a PC running Windows or a Mac running OSX , Linux (on the playground wiki).



Get an Arduino Mega 2560 board and USB cable

Firstly, get your Arduino board and lay it on the table in front of you. Take the USB cableand plug the B plug (the fatter squarer end) into the USB socket on the Arduino Mega 2560 baord .

At this stage do NOT connect the Arduino Mega 2560 board to your PC or Mac yet.

Download the Arduino IDE

Get the latest version from the download page .

When the download finishes, unzip the downloaded file. Make sure to preserve the folder structure. Double-click the folder to open it. There should be a few files and sub-folders inside.

If you double-click the folder, you will see a few files and sub-folders inside.

Connect the 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're using an Arduino Diecimila, you'll need 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's on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labelled PWR) should go on.

Install the USB Drivers

The Installing drivers for the Mega 2560 is nearly the same as Arduino Uno,

Please check the following step.

(1) Installing drivers for the Arduino Uno with Windows7, Vista, or XP:

- Plug in your board and wait for Windows to begin it's driver installation process. After a few moments, the process will fail, despite its best efforts
- Click on the Start Menu, and open up the Control Panel.
- While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.
- Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)"
- Right click on the "Arduino UNO (COmxx)" port and choose the "Update Driver Software" option.
- Next, choose the "Browse my computer for Driver software" option.
- Finally, navigate to and select the Uno's driver file, named **"ArduinoUNO.inf"**, located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory).
- Windows will finish up the driver installation from there.

(2) Installing drivers for the Arduino Duemilanove, Nano, or Diecimila with Windows7, Vista, or XP:

When you connect the board, Windows should initiate the driver installation process (if you haven't used the computer with an Arduino board before).

On Windows Vista, the driver should be automatically downloaded and installed. (Really, it works!)

On Windows XP, the Add New Hardware wizard will open:

- When asked **Can Windows connect to Windows Update to search for software?** select **No, not this time**. Click next.
- Select **Install from a list or specified location (Advanced)** and click next.
- Make sure that **Search for the best driver in these locations** is checked; uncheck **Search removable media**; check **Include this location in the search** and browse to the **drivers/FTDI USB Drivers** directory of the Arduino distribution. (The latest version of the drivers can be found on the FTDI website.) Click next.
- The wizard will search for the driver and then tell you that a "USB Serial Converter" was found. Click finish.
- The new hardware wizard will appear again. Go through the same steps and select the same options and location to search. This time, a "USB Serial Port" will be found.

You can check that the drivers have been installed by opening the Windows Device Mananger (in the Hardware tab of System control panel). Look for a "USB Serial Port" in the Ports section; that's the Arduino board.

Upload your first Sketch

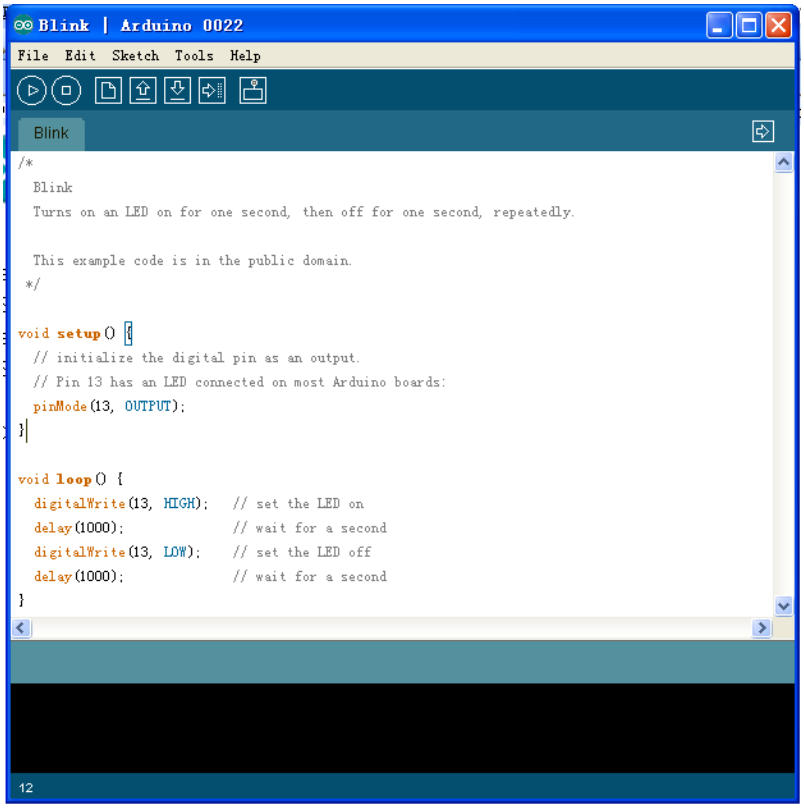
Now that your Arduino Mage 2560 board has been connected and the drivers for the USB chip have been installed, we are now ready to try out the Arduino for the first time and upload your first Sketch.

Navigate to your newly unzipped Arduino folder and look for the Arduino IDE icon, which looks something like this....

Double click the ICON to open up the IDE. You will then be presented with a blue and white screen with a default sketch loaded inside.

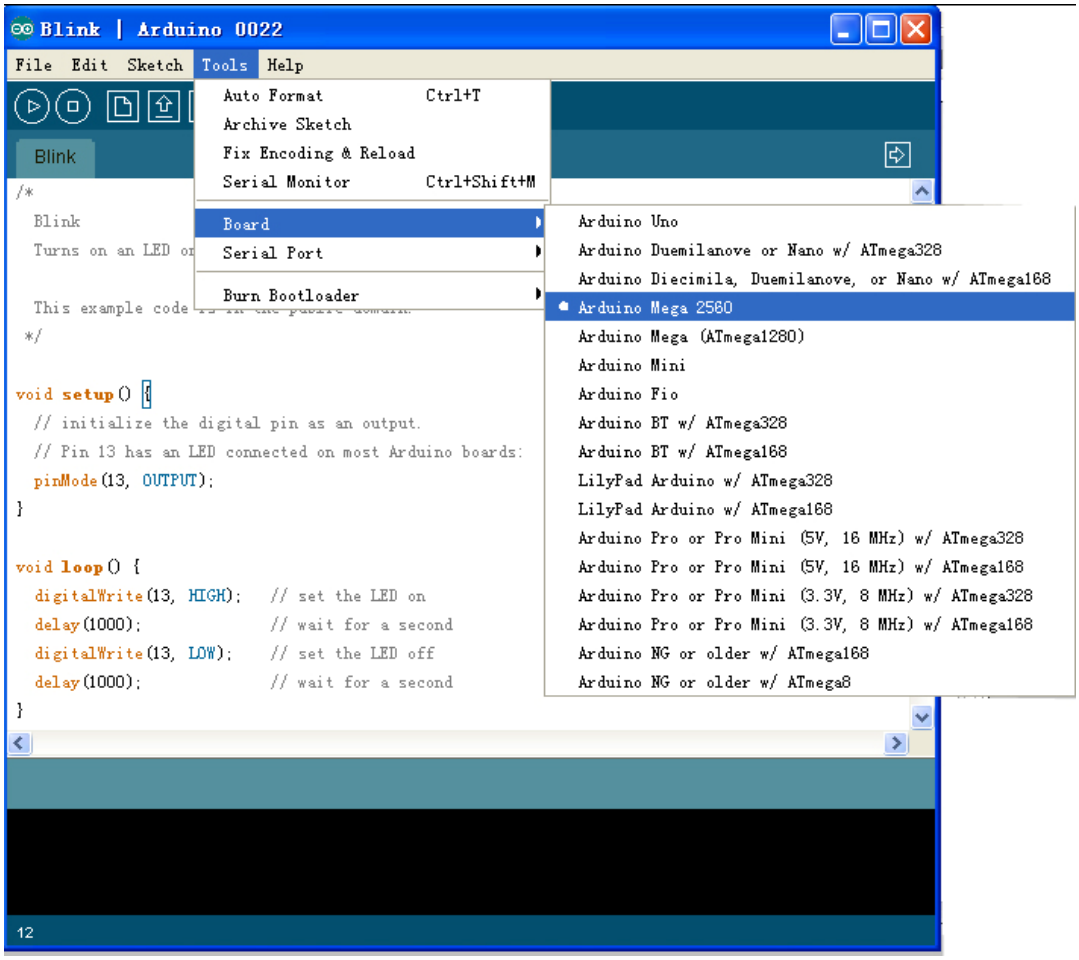
This is the Arduino IDE (Integrated Development Environment) and is where you will write your Sketches (programs) to upload to your Arduino board .

1) Open the LED blink example sketch: **File > Examples > 1.Basics > Blink**.



2) And then you should select your board .

You'll need to select the entry in the **Tools > Board** menu that corresponds to your Arduino.



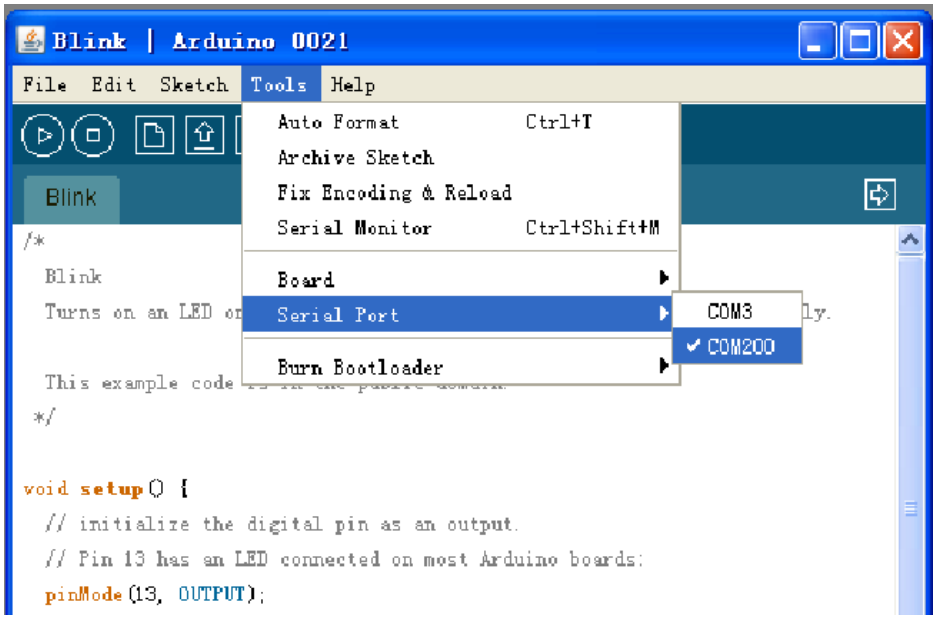
Selecting an Arduino Mega 2560

For Duemilanove Arduino boards with an ATmega328 (check the text on the chip on the board), select Arduino Duemilanove or Nano w/ ATmega328. Previously, Arduino boards came with an ATmega168; for those, select Arduino Diecimila, Duemilanove, or Nano w/ ATmega168. (Details of the board menu entries are available on the environment page.)

For Mega 1280 , select Arduino Mega (Atmega1280)

3) Select your serial port

Select the serial device of the Arduino board from the 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 the Arduino board. Reconnect the board and select that serial port.



4) Upload the program

Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar. (*Note:* If you have an Arduino Mini, NG, or other board, you'll need to physically present the reset button on the board immediately before pressing the upload button.)

A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in orange). If it does, congratulations! You've gotten Arduino up-and-running.

Tips :

Verify/ Compile **Stop** **New** **Open** **Save** **Upload** **Serial Monitor**

The Toolbar buttons are listed above. The functions of each button are as follows :

● **Verify/Compile**	Checks the code for errors
● **Stop**	Stops the serial monitor, or un-highlights other buttons
● **New**	Creates a new blank Sketch
● **Open**	Shows a list of Sketches in your sketchbook
● **Save**	Saves the current Sketch
● **Upload**	Uploads the current Sketch to the Arduino
● **Serial Monitor**	Displays serial data being sent from the Arduino

The **Verify/Compile** button is used to check that yourcode is correct, before you upload it to your Arduino.

The **Stop** button will stop the Serial Monitor from operating. It will also un-highlight other selected buttons. Whilst the Serial Monitor is operating you may wish to press the Stop button to obtain a 'snapshot' of the serial data so far to examine it. This is particularly useful if you are sending data out to the Serial Monitor quicker than you can read it.

The **New** button will create a completely new and blank Sketch read for you to enter code into. The IDE will ask you to enter a name and a location for your Sketch (try to use the default location if possible) and will then give you a blank Sketch ready to be coded. The tab at the top of the Sketch will now contain the name you have given to your new sketch.

The **Open** button will present you with a list of Sketches stored within your sketchbook as well as a list of Example sketches you can try out with various peripherals once connected.

The **Save** button will save the code within the sketch window to your sketch file. Once complete you will get a 'Done Saving message at the bottom of the code window.

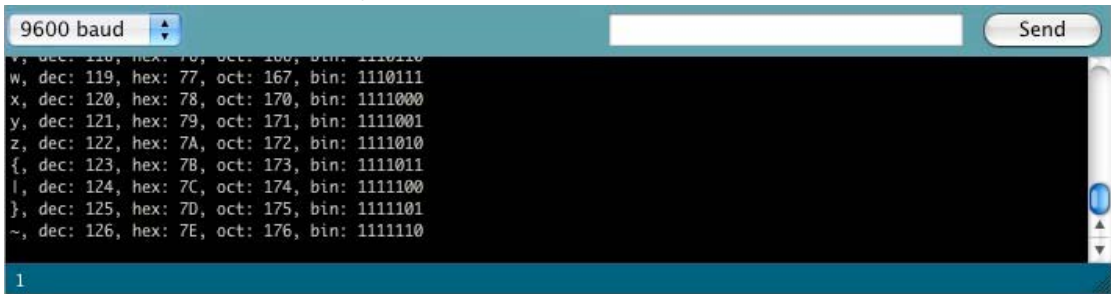
The **Upload to I/O Board** button will upload the code within the current sketch window to your Arduino. You need to make sure that you have the correct board and port selected (in the Tools menu) before uploading. It is essential that you Save your sketch before you upload it to your board in case a strange error causes your system to hang or the IDE to crash.

It is also advisable to Verify/Compile the code before you upload to ensure there are no errors that need to be debugged first.

The **Serial Monitor** is a very useful tool, especially for debugging your code. The

monitor displays serial data being sent out from your Arduino (USB or Serial board). You can also send serial data back to the Arduino using the Serial Monitor. If you click the Serial Monitor button you will be presented with an image like the one above.

On the left hand side you can select the Baud Rate that the serial data is to be sent to/from the Arduino. The Baud Rate is the rate, per second, that state changes or bits (data) are sent to/from the board. The default setting is 9600 baud, which means that if you were to send a text novel over the serial communications line (in this case your USB cable) then 9600 letters, or symbols, of the novel, would be sent per second.



To the right of this is a blank text box for you to enter text to send back to the Arduino and a **Send** button to send the text within that field. Note that no serial data can be received by the Serial Monitor unless you have set up the code inside your sketch to do so. Similarly, the Arduino will not receive any data sent unless you have coded it to do so.

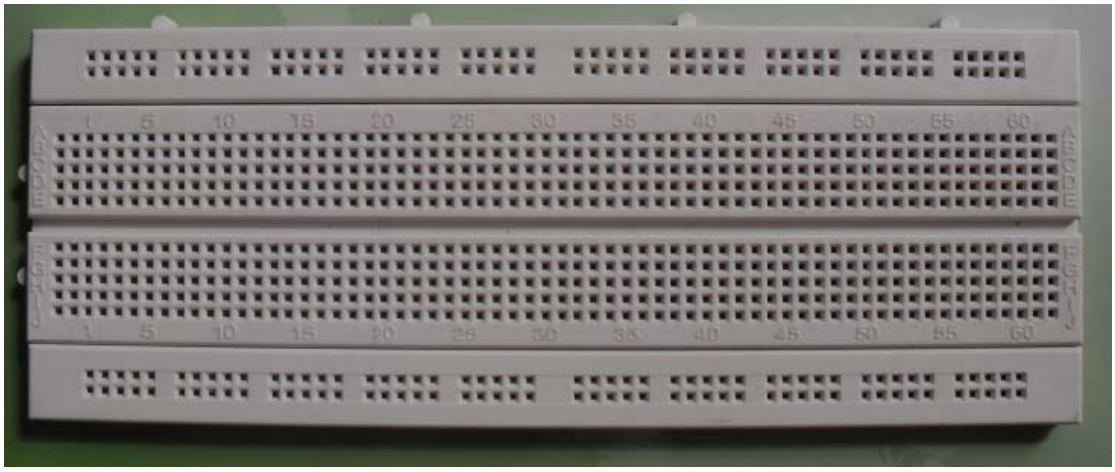
Finally, the black area is where your serial data will be displayed. In the image above, the Arduino is running the ASCIITable sketch, that can be found in the Communications examples. This program outputs ASCII characters, from the Arduino via serial (the USB cable) to the PC where the Serial monitor then displays them.

To start the Serial Monitor press the Serial Monitor button and to stop it press the Stop button. On a Mac or in Linux, Arduino board will reset itself (rerun the code from the beginning) when you click the Serial Monitor button.

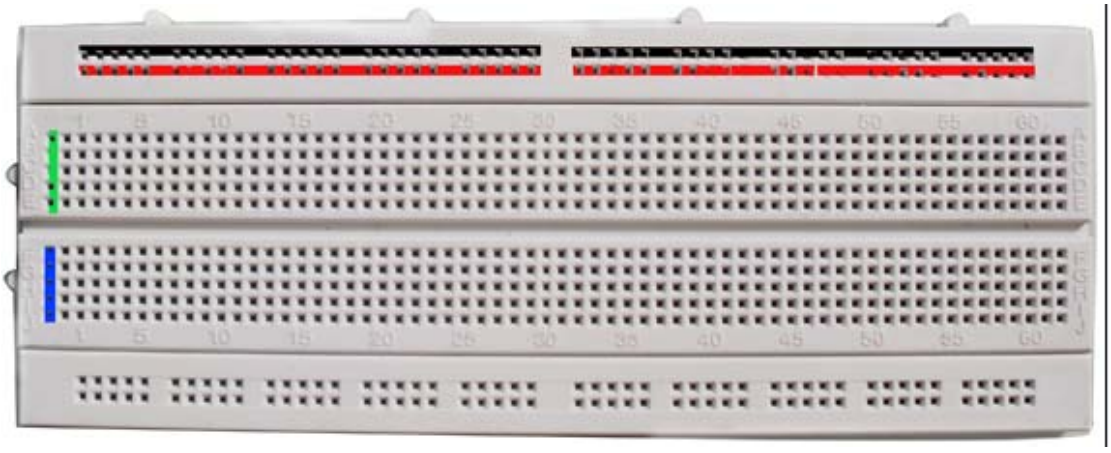
Once you are proficient at communicating via serial to and from the Arduino you can use other programs such as Processing, Flash, MaxMSP, etc. To communicate between the Arduino and your PC.

Solderless Breadboard

The breadboard has 840 Tie points. It is Compatible with all kind of ProtoShield. Completely reusable,Reusable for fast build a prototype of an electronic circuit. With Twin adhesive back, it could be fix and remove to any position easily. Its Dimension: 171mm (L) X 64mm (W) X 10mm (H).



In the middle points of the breadboard , the top and bottom Five row columns are separate columns, with no internal connections made between them inside the breadboard. The only five internal connections inside the breadboard are to any five vertical consecutive holes . So the five point which are in one column is connected internally.



Project 1 - LED Flasher

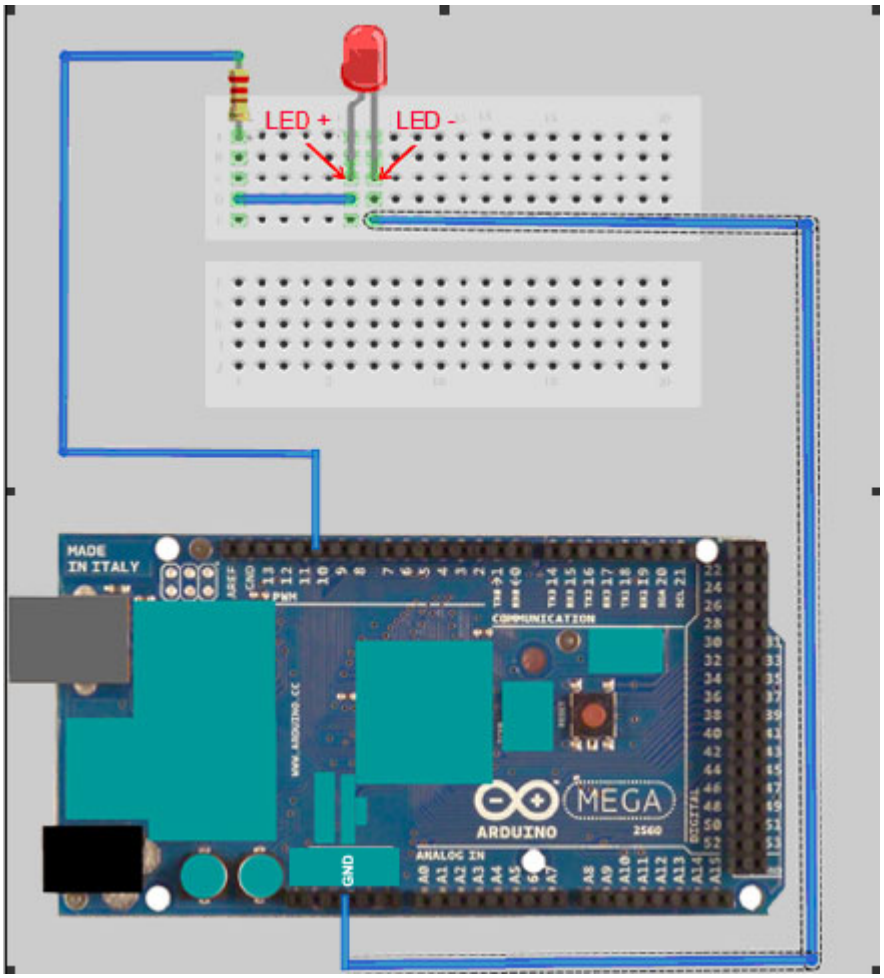
In this project we are going to repeat what we did in setting up and testing the Arduino, that is to blink an LED. However, this time we are going to use one of the LED's in the kit and you will also learn about some electronics and coding in C along the way.

What you will need

Breadboard
Red LED
220 Ω Resistor
Jumper Wires

Connect it up

Now, first make sure that your Arduino is powered off. You can do this either by unplugging the USB cable or by taking out the Power Selector Jumper on the Arduino board. Then connect everything up like this :



It doesn't matter if you use different coloured wires or use different holes on the

breadboard as long as the components and wires are connected in the same order as the picture. Be careful when insterting components into the Breadboard. The Breadboard is brand new and the grips in the holes will be stiff to begin with. Failure to insert components carefully could result in damage. Make sure that your LED is connected the right way with the longer leg connected to Digital Pin 10. The long led is the Anode of the LED and always must go to the +5v supply (in this case coming out of Digital Pin 10) and the short leg is the Cathode and must go to Gnd (Ground).

When you are happy that everything is connected up correctly, power up your Arduino and connect the USB cable.

Enter the code

Now, open up the Arduino IDE and type in the following code :-

```
// Project 1 - LED Flasher

int ledPin = 10;

void setup() {
      pinMode(ledPin, OUTPUT);
}

void loop() {
      digitalWrite(ledPin, HIGH);
      delay(1000);
      digitalWrite(ledPin, LOW);
      delay(1000);
}
```

Now press the Verify/Compile button at the top of the IDE to make sure there are no errors in your code. If this is successful you can now click the Upload button to upload the code to your Arduino.

If you have done everything right you should now see the Red LED on the breadboard flashing on and off every second. Now let's take a look at the code and the hardware and find out how they both work.

Now let's take a look at the code and the hardware and find out how they both work.

Project 1 - Code verview

```
// Project 1 - LED Flasher
int ledPin = 10;
void setup() {
! pinMode(ledPin, OUTPUT);
}
void loop() {
! digitalWrite(ledPin, HIGH);
! delay(1000);
! digitalWrite(ledPin, LOW);
! delay(1000);
}
```

So let's take a look at the code for this project. Our first line is

// Project 1 - LED Flasher

This is simply a comment in your code and is ignored by the compiler (the part of the IDE that turns your code into instructions the Arduino can understand before uploading it). Any text entered behind a // command will be ignored by the compiler and is simply there for you, or anyone else that reads your code. Comments are essential in your code to help you understand what is going on and how your code works. Comments can also be put after commands as in the next line of the program.

Later on as your projects get more complex and your code expands into hundreds or maybe thousands of lines, comments will be vital in making it easy for you to see how it works. You may come up with an amazing piece of code, but if you go back and look at that code days, weeks or months alter, you may forget how it all works. Comments will help you understand it easily. Also, if your code is meant to be seen by other people (and as the whole ethos of the Arduino, and indeed the whole Open Source community is to share code and schematics. We hope when you start making your own cool stuff with the Arduino you will be willing to share it with the world) then comments will enable that person to understand what is going on in your code.

You can also put comments into a block statement by using the /* and */ commands. E.g.

/* All of the text within the slash and the asterisks is a comment and will be ignored by the compiler */

The IDE will automatically turn the colour of any commented text to grey.

The next line of the program is:

```
int ledPin = 10;
```

This is what is know as a variable. A variable is a place to store data. In this case you are setting up a variable of type int or integer. An integer is a number within the range of -32,768 to 32,767. Next you have assigned that integer the name of ledPin and have given it a value of 10. We didn't have to call it ledPin, we could have called it anything we

wanted to. But, as we want our variable name to be descriptive we call it ledPin to show that the use of this variable is to set which pin on the Arduino we are going to use to connect our LED. In this case we are using Digital Pin 10. At the end of this statement is a semi-colon. This is a symbol to tell the compiler that this statement is now complete.

Although we can call our variables anything we want, every variable name in C must start with a letter, the rest of the name can consist of letters, numbers and underscore characters. C recognises upper and lower case characters as being different. Finally, you cannot use any of C's keywords like main, while, switch etc as variable names. Keywords are constants, variables and function names that are defined as part of the Arduino language. Don't use a variable name that is the same as a keyword. All keywords within the sketch will appear in red.

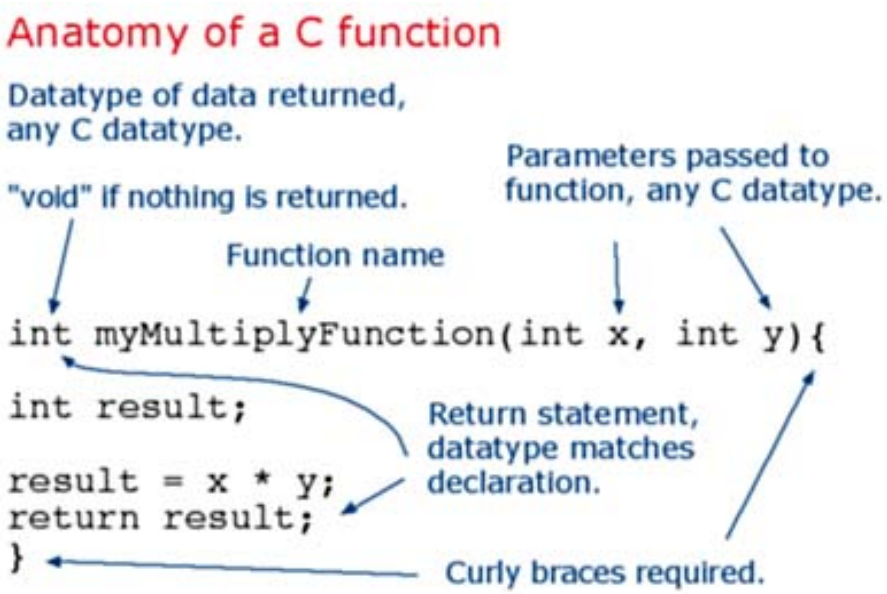
So, you have set up an area in memory to store a number of type integer and have stored in that area the number 10. Imagine a variable as a small box where you can keep things. A variable is called a variable because you can change it. Later on we will carryout mathematical calculations on variables to make our program do more advanced stuff.

Next we have our setup() function :

```
void setup() {
! pinMode(ledPin, OUTPUT);
}
```

An Arduino sketch must have a setup() and loop() function otherwise it will not work. The setup() function is run once and once only at the start of the program and is where you will issue general instructions to prepare the program before the main loop runs, such as setting up pin modes, setting serial baud rates, etc.

Basically a function is a block of code assembled into one convenient block. For example, if we created our own function to carry out a whole series of complicated mathematics that had many lines of code, we could run that code as many times as we liked simply by calling the function name instead of writing



out the code again each time. Later on we will go into functions in more detail when we start to create our own.

In the case of our program the setup() function only has one statement to carry out. The function starts with

void setup()

and here we are telling the compiler that our function is called setup, that it returns no data (void) and that we pass no parameters to it (empty parenthesis). If our function returned an integer value and we also had integer values to pass to it (e.g. for the function to process) then it would look something like this :

```
int myFunc(int x, int y)
```

In this case we have created a function (or a block of code) called myFunc. This function has been passed two integers called X and Y. Once the function has finished it will then return an integer value to the point after where our function was called in the program (hence int before the function name).

All of the code within the function is contained within the curly braces. A { symbol starts the block of code and a } symbol ends the block. Anything in between those two symbols is code that belongs to the function.

We will go into greater detail about functions later on so don't worry about them for now. All you need to know is that in this program, we have two functions, the first function is called setup and it's purpose is to setup anything necessary for our program to work before the main program loop runs.

```
void setup() {
  pinMode(ledPin, OUTPUT);
}
```

Our setup function only has one statement and that is pinMode. Here we are telling the Arduino that we want to set the mode of one of our digital pins to be Output mode, rather than Input. Within the parenthesis we put the pin number and the mode (OUTPUT or INPUT). Our pin number is ledPin, which has been previously set to the value 10 in our program. Therefore, this statement is simply telling the Arduino that the Digital Pin 10 is to be set to OUTPUT mode. As the setup() function runs only once, we now move onto the main function loop.

```
void loop() {
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
}
```

The loop() function is the main program function and runs continuously as long as our Arduino is turned on.Every statement within the loop() function (within the curly braces) is carried out, one by one, step by step, until the bottom of the function is reached, then the loop starts again at the top of the function, and so on forever or until you turn the Arduino off or press the Reset switch.

In this project we want the LED to turn on, stay on for one second, turn off and remain off for one second, and then repeat. Therefore, the commands to tell the Arduino to do that are contained within the loop() function as we wish them to repeat over and over.

The first statement is

digitalWrite(ledPin, HIGH);

and this writes a HIGH or a LOW value to the digital pin within the statement (in this case ledPin, which is Digital Pin 10). When you set a digital pin to HIGH you are sending out 5 volts to that pin. When you set it to LOW the pin becomes 0 volts, or Ground. This statement therefore sends out 5v to digital pin 10 and turns the LED on.

After that is

delay(1000);

and this statement simply tells the Arduino to wait for 1000 milliseconds (to 1 second as there are 1000 milliseconds in a second) before carrying out the next statement which is

digitalWrite(ledPin, LOW);

which will turn off the power going to digital pin 10 and therefore turn the LED off. There is then another delay statement for another 1000 milliseconds and then the function ends. However, as this is our main loop() function, the function will now start again at the beginning. By following the program structure step by step again we can see that it is very simple.

```
// Project 1 - LED Flasher
int ledPin = 10;
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

We start off by assigning a variable called ledPin, giving that variable a value of 10.

Then we move onto the setup() function where we simply set the mode for digital pin 10 as an output.

In the main program loop we set Digital Pin 10 to high, sending out 5v. Then we wait for a second and then turn off the 5v to Pin 10, before waiting another second. The loop then starts again at the beginning and the LED will therefore turn on and off continuously for as long as the Arduino has power.

Now that you know this you can modify the code to turn the LED on for a different period of time and also turn it off for a different time period.

seconds, then go off for half a second we could do

this:-

```
void loop() {
    digitalWrite(ledPin, HIGH);
```

```
    delay(2000);
    digitalWrite(ledPin, LOW);
    delay(500);
}
```

or maybe you would like the LED to stay off for 5 seconds and then flash briefly (250ms), like the LED indicator on a car alarm then you could do this :

```
void loop() {
    digitalWrite(ledPin, HIGH);
    delay(250);
    digitalWrite(ledPin, LOW);
    delay(5000);
}
```

or make the LED flash on and off very fast

```
void loop() {
    digitalWrite(ledPin, HIGH);
    delay(50);
    digitalWrite(ledPin, LOW);
    delay(50);
}
```

By varying the on and off times of the LED you create any effect you want. Well, within the bounds of a single LED going on and off that is.

Before we move onto something a little more exciting let's take a look at the hardware and see how it works.

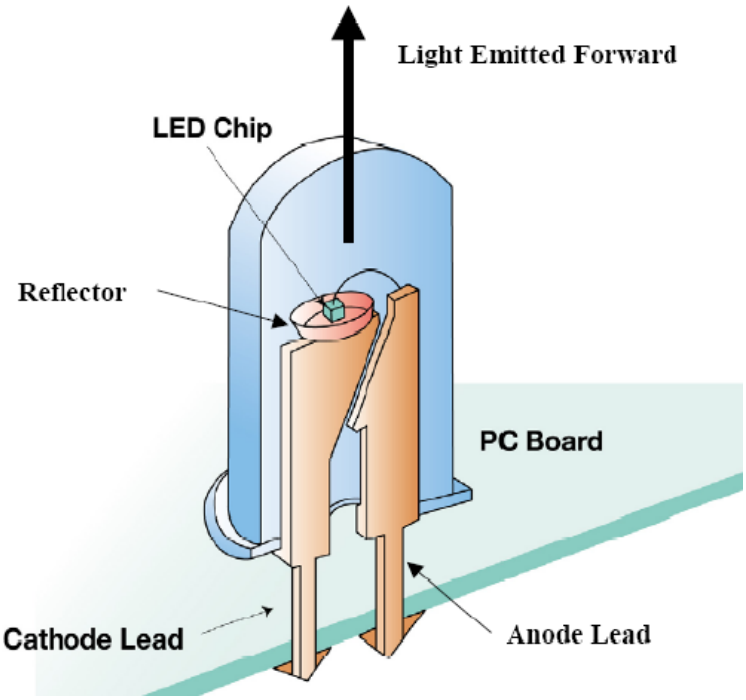
Project 1 - Hardware Overview

The hardware used for this project was :-

Breadboard
Red LED
220Ω Resistor
Jumper Wires

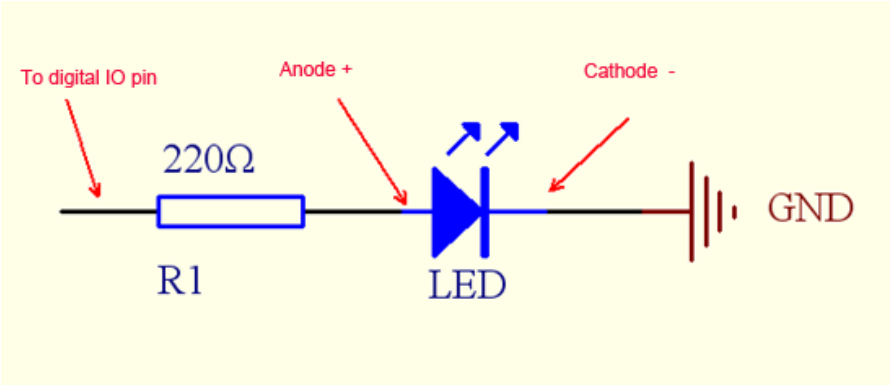
LED

The internal structure Of LED.



The longer one lead is Anode (“+”), the another one is Cathode (“ --”) .

The wiring of LED



The next component we have is a Resistor. A resistor is a device designed to cause ‘resistance’ to an electric current and therefore cause a drop in voltage across it’s terminals. If you imagine a resistor to be like a water pipe that is a lot thinner than the pipe connected to it. As the water (the electric current) comes into the resistor, the pipe gets thinner and the current coming out of the other end is therefore reduced. We use resistors to decrease voltage or current to other devices. The value of resistance is known as an Ohm and it’s symbol is a greek Omega symbol **Ω**.

In this case Digital Pin 10 is outputting 5 volts DC at (according to the Atmega datasheet) 40mA (milliamps) and our LED’s require (according to their datasheet) a voltage of 2v and a current of 20mA. We therefore need to put in a resistor that will reduce the 5v to 2v and the current from 40mA to 20mA if we want to display the LED at it’s maximum brightness. If we want the LED to be dimmer we could use a higher value of resistance.

To work out what resistor we need to do this we use what is called Ohm’s law which is I = V/R where I is current, V is voltage and R is resistance. So to work out the resistance we arrange the formula to be R = V/I which is R = 3/0.014 which is 220 Ohms. V is 3 because we need the Voltage Drop, which is the supply voltage (5v) minus the Forward Voltage (2v) of the LED (found in the LED datasheet) which is 3v. We therefore need to find a 220Ω resistor. So how do we do that?

A resistor is too small to put writing onto that could be readable by most people so instead resistors use a colour code. Around the resistor you will typically find 4 coloured bands and by using the colour code in the chart on the next page you can find out the value of a resistor or what colour codes a particular resistance will be.

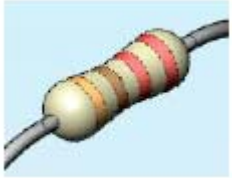
WARNING:
Always put a resistor (commonly known as a current limiting resistor) in series with an LED. If you fail to do this you will supply too much current to the LED and it could blow or damage your circuit.

Colour	1st Band	2nd Band	3rd Band (multiplier)	4th Band (tolerance)
Black	0	0	$x10^0$	
Brown	1	1	$x10^1$	±1%
Red	2	2	$x10^2$	±2%
Orange	3	3	$x10^3$	
Yellow	4	4	$x10^4$	
Green	5	5	$x10^5$	±0.5%
Blue	6	6	$x10^6$	±0.25%
Violet	7	7	$x10^7$	±0.1%
Grey	8	8	$x10^8$	±0.05%
White	9	9	$x10^9$	
Gold			$x10^{-1}$	±5%
Silver			$x10^{-2}$	±10%
None				±20%

For example , If We need a 150Ω resistor, so if we look at the colour table we see that we need 1 in the first band, which is Brown, followed by a 5 in the next band which is Green and we then need to multiply this by 101 (in other words add 1 zero) which is Brown in the 3rd
band. The final band is irrelevant for our purposes as this is the tolerance. Our resistor has a gold band and therefore has a tolerance of ±5% which means the actual value of the resistor can vary between 142.5Ω and 157.5Ω. We therefore need a resistor with a Brown, Green, Brown, Gold colour band combination

which looks like this:-

If we needed a 1K (or 1 kilo-ohm) resistor we would need a Brown, Black, Red combination (1, 0, +2 zeros). If we needed a 570K resistor the colours would be Green, Violet and Yellow.



In the same way, if you found a resistor and wanted to know

what value it is you would do the same in reverse. So if you found this resistor and wanted to find out what value it was so you could store it away in your nicely labelled resistor storage box, we could look at the table to see it has a value of 220Ω.

Our final component is an LED (I'm sure you can figure out what the jumper wires do for yourself), which stands for Light Emitting Diode. A Diode is a device that permits current to flow in only one direction. So, it is just like a valve in a water system, but in this case it is letting electrical current to go in one direction, but if the current tried to reverse and go back in the opposite direction the diode would stop it from doing so. Diodes can be useful to prevent someone from accidently connecting the Power and Ground to the wrong terminals in a circuit and damaging the components.

An LED is the same thing, but it also emits light. LED's come in all kinds of different colours and brightnesses and can also emit light in the ultraviolet and infrared part of the spectrum (like in the LED's in your TV remote control).

If you look carefully at the LED you will notice two things. One is that the legs are of different lengths and also that on one side of the LED, instead of it being cylindrical, it is flattened. These are indicators to show you which leg is the Anode (Positive) and which is the Cathode (Negative). The longer leg gets connected to the Positive Supply (3.3v) and the leg with the flattened side goes to Ground.

If you connect the LED the wrong way, it will not damage it (unless you put very high currents through it) and indeed you can make use of that 'feature' as we will see later on.

It is essential that you always put a resistor in series with the LED to ensure that the correct current gets to the LED. You can permanently damage the LED if you fail to do this.

As well as single colour resistors you can also obtain bi-colour and tricolour LED's. These will have several legs coming out of them with one of them being common (i.e. Common anode or common cathode).

Supplied with your kit is an RGB LED, which is 3 LED's in a single package. An RGB LED has a Red, Green and a Blue (hence RGB) LED in one package.

The LED has 4 legs, one will be a common anode or cathode, common to all 3 LED's and the other 3 will then go to the anode or cathode of the individual Red, Green and Blue LED's. By adjusting the brightness values of the R, G and B channels of the RGB LED you can get any colour you want. The same effect can be obtained if you used 3 separate red, green and blue LED's.

Now that you know how the components work and how the code in this project works, let's try something a bit more interesting.

Project 2 - Pulsating Lamp

We are now going to delve further into a more advanced method of controlling LED's. So far we have simply turned the LED on or off. How about being able to adjust it's brightness too? Can we do that with an Arduino? Yes we can.

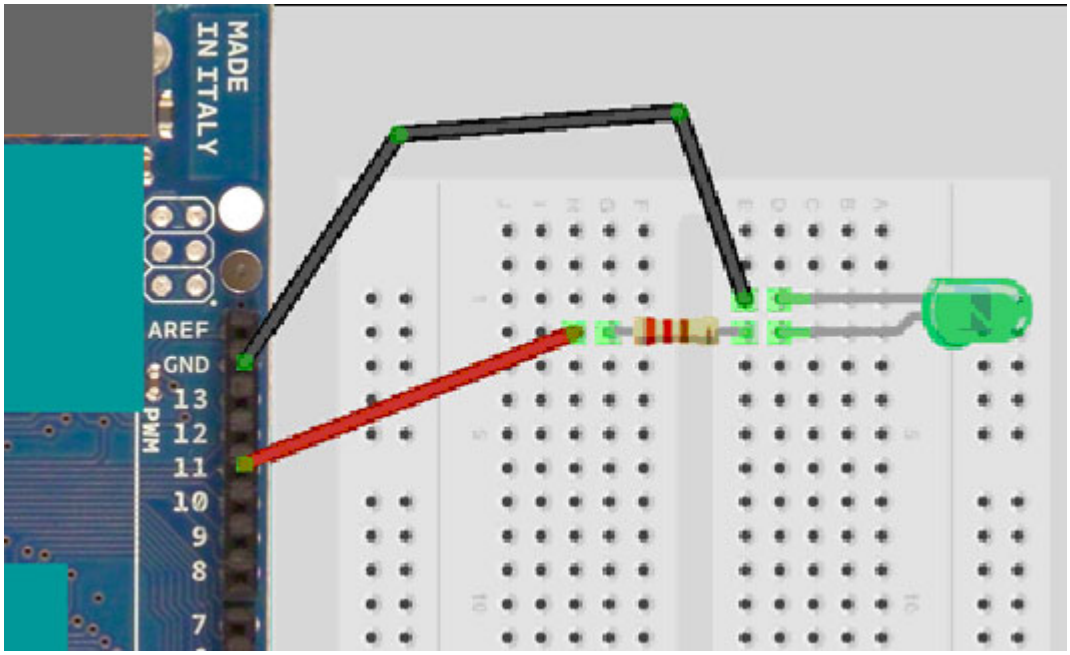
Time to go back to basics.

What you will need

Green Diffused LED

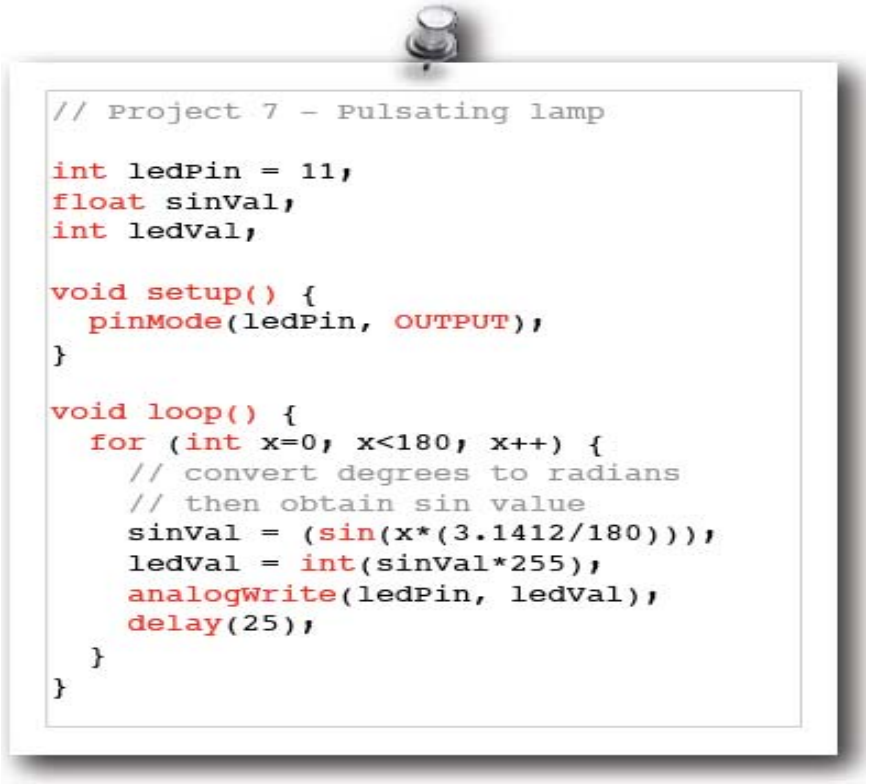
220Ω Resistor

Connect it up



Enter the code

Enter this simple program.



```
// Project 7 - Pulsating lamp

int ledPin = 11;
float sinVal;
int ledVal;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  for (int x=0; x<180; x++) {
    // convert degrees to radians
    // then obtain sin value
    sinVal = (sin(x*(3.1412/180)));
    ledVal = int(sinVal*255);
    analogWrite(ledPin, ledVal);
    delay(25);
  }
}
```

Verify and upload. You will now see your LED pulsate on and off steadily. Instead of a simple on/off state we are now adjusting it's brightness. Let's find out how this works.

Project 2 - Code Overview

The code for this project is very simple, but requires some explanation.

```
// Project 7 - Pulsating lamp
int ledPin = 11;
float sinVal;
int ledVal;
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  for (int x=0; x<180; x++) {
  // convert degrees to radians
  // then obtain sin value
      sinVal = (sin(x*(3.1412/180)));
      ledVal = int(sinVal*255);
      analogWrite(ledPin, ledVal);
      delay(25);
  }
}
```

We first set up the variables for the LED Pin, a float (floating point data type) for a sine wave value and ledVal which will hold the integer value to send out to Pin 11.

The concept here is that we are creating a sine wave and having the brightness of the LED follow the path of that wave. This is what makes the light pulsate in that way instead of just fade up to full brightness and back down again.

We use the **sin()** function, which is a mathematical function to work out the sine of an angle. We need to give the function the degree in radians. We have a for loop that goes from 0 to 179, we don't want to go past halfway as this will take us into negative values and the brightness value we need to put out to Pin 11 needs to be from 0 to 255 only.

The **sin()** function requires the angle to be in radians and not degrees so the equation of x*(3.1412/180) will convert the degree angle into radians. We then transfer the result to ledVal, multiplying it by 255 to give us our value. The result from the sin() function will be a number between -1 and 1 so we need to multiply that by 255 to give us our maximum brightness. We 'cast' the floating point value of sinVal into an integer by the use of int() in the statement

```
ledVal = int(sinVal*255);
```

Then we send that value out to Digital Pin 11 using the statement

```
analogWrite(ledPin, ledVal);
```

But, how can we send an analog value to a digital pin? Well, if we take a look at our

Arduino and look at the Digital Pins you can see that 6 of those pins (3, 5, 6, 9, 10 & 11) have PWM written next to them. Those pins differ from the remaining digital pins in that they are able to send out a PWM signal.

PWM stands for Pulse Width Modulation. PWM is a technique for getting analog results from digital means. On these pins the Arduino sends out a square wave by switching the pin on and off very fast. The pattern of on/offs can simulate a varying voltage between 0 and 5v. It does this by changing the amount of time that the output remains high (on) versus off (low). The duration of the on time is known as the 'Pulse Width'.

For example, if you were to send the value 0 out to Pin 11 using analogWrite() the ON period would be zero, or it would have a 0% Duty Cycle. If you were to send a value of 64 (25% of the maximum of 255) the pin would be ON for 25% of the time and OFF for 75% of the time. The value of 191 would have a Duty Cycle of 75% and a value of 255 would have a duty cycle of 100%. The pulses run at a speed of approx. 500Hz or 2 milliseconds each.

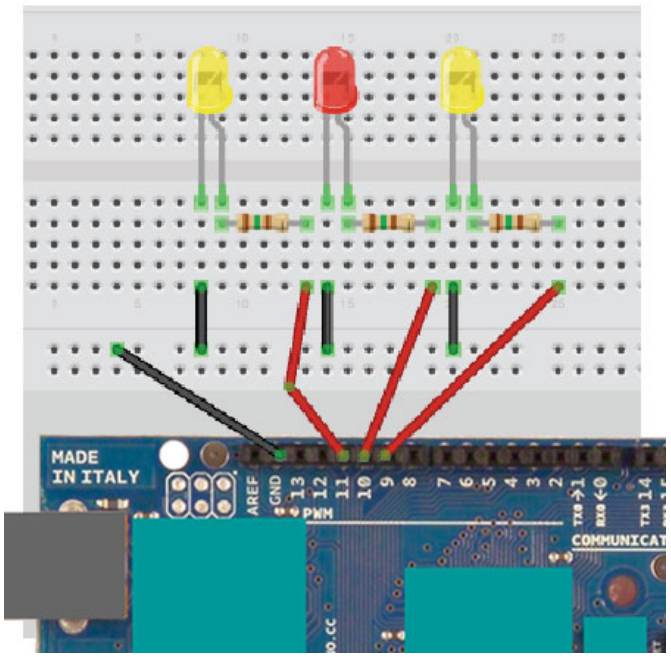
So, from this we can see in our sketch that the LED is being turned on and off very fast. If the Duty Cycle was 50% (a value of 127) then the LED would pulse on and off at 500Hz and would display at half the maximum brightness. It is basically an illusion that we can use to our advantage by allowing us to use the digital pins to output a simulated analog value to our LED's.

Note that even though only 6 of the pins have the PWM function, you can easily write software to give a PWM output from all of the digital pins if you wish.

Later on we will revisit PWM as we can utilise it to create audible tones using a piezo sounder.

Project 3 - Serial Controlled Mood Lamp

Please check the follow picture circuit, This project we will delve into the world of serial communications and control our lamp by sending commands from the PC to the Arduino using the Serial Monitor in the Arduino IDE.



This project also introduces how we manipulate text strings. So leave the hardware set up the same as before and enter the new code.

Enter the code

```
// Project 10 - Serial controlled RGB Lamp
char buffer[18];
int red, green, yellow;
int RedPin = 11;
int GreenPin = 10;
int YellowPin = 9;
void setup()
{
  Serial.begin(9600);
  Serial.flush();
  pinMode(RedPin, OUTPUT);
  pinMode(GreenPin, OUTPUT);
  pinMode(YellowPin, OUTPUT);
}
void loop()
{
  if (Serial.available() > 0) {
    int index=0;
    delay(100); // let the buffer fill up
    int numChar = Serial.available();
    if (numChar>15) {
        numChar=15;
    }
    while (numChar--) {
        buffer[index++] = Serial.read();
    }
    splitString(buffer);
  }
```

```
}
void splitString(char* data) {
  Serial.print("Data entered: ");
  Serial.println(data);
  char* parameter;
  parameter = strtok (data, " ,");
  while (parameter != NULL) {
      setLED(parameter);
      parameter = strtok (NULL, " ,");
  }
  // Clear the text and serial buffers
  for (int x=0; x<16; x++) {
      buffer[x]='\0';
  }
  Serial.flush();
}
void setLED(char* data) {
  if ((data[0] == 'r') || (data[0] == 'R')) {
      int Ans = strtol(data+1, NULL, 10);
      Ans = constrain(Ans,0,255);
      analogWrite(RedPin, Ans);
      Serial.print("Red is set to: ");
      Serial.println(Ans);
  }
  if ((data[0] == 'g') || (data[0] == 'G')) {
      int Ans = strtol(data+1, NULL, 10);
      Ans = constrain(Ans,0,255);
      analogWrite(GreenPin, Ans);
      Serial.print("Green is set to: ");
      Serial.println(Ans);
   }
  if ((data[0] == 'y') || (data[0] == 'Y')) {
      int Ans = strtol(data+1, NULL, 10);
      Ans = constrain(Ans,0,255);
      analogWrite(YellowPin, Ans);
      Serial.print("Yellow is set to: ");
      Serial.println(Ans);
  }
}
```

Once you've verified the code, upload it to your Arduino.

Now when you upload the program nothing seems to happen. This is because the program is waiting for your input. Start the Serial Monitor by clicking it's icon in the Arduino

IDE taskbar.

In the Serial Monitor text window you can now enter the R, G and Y values for each of the 3 LED's manually and the LED's will change to the colour you have input.

E.g. If you enter R255 the Red LED will display at full brightness.

If you enter R255, G255, then both the red and green LED's will display at full brightness.

Now enter R127, G100, Y255 and you will get a nice purplish colour.

If you type, r0, g0, y0 all the LED's will turn off.

The input text is designed to accept both a lowercase or upper-case R, G and B and then a value from 0 to 255. Any values over 255 will be dropped down to 255 maximum. You can enter a comma or a space in between parameters and you can enter 1, 2 or 3 LED values at any one time.

E.g.
r255 y100
r127 y127 g127
G255, Y0
Y127, R0, G255
Etc.

Project 3 - Code Overview

This project introduces a whole bunch of new concepts, including serial communication, pointers and string manipulation. So, hold on to your hats this will take a lot of explaining.

First we set up an array of char (characters) to hold our text string. We have made it 18 characters long, whichis longer than the maximum of 16 we will allow to ensure we don't get "buffer overflow" errors.

```
char buffer[18];
```

We then set up the integers to hold the red, green and blue values as well as the values for the digital pins.

```
int red, green, blue;
int RedPin = 11;
int GreenPin = 10;
int YellowPin = 9;
```

In our setup function we set the 3 digital pins to be outputs. But, before that we have the Serial.begin command.

```
void setup()
{
Serial.begin(9600);
Serial.flush();
pinMode(RedPin, OUTPUT);
pinMode(GreenPin, OUTPUT);
pinMode(BluePin, OUTPUT);
}
```

Serial.begin tells the Arduino to start serial communications and the number within the
parenthesis, in this case 9600, sets the baud rate (characters per second) that the serial line will communicate at.

The **Serial.flush** command will flush out any characters that happen to be in the serial line so that it is empty and ready for input/output.

The serial communications line is simply a way for the Arduino to communicate with the outside world, in this case to and from the PC and the Arduino IDE's Serial Monitor.

In the main loop we have an if statement. The condition it is checking for is

```
if (Serial.available() > 0) {
```

The Serial.available command checks to see if any characters have been sent down the serial line. If any characters have been received then the condition is met and the code within the if statements code block is now executed.

```
if (Serial.available() > 0) {
  int index=0;
  delay(100); // let the buffer fill up
```

```
    int numChar = Serial.available();
    if (numChar>15) {
        numChar=15;
    }
    while (numChar--) {
        buffer[index++] = Serial.read();
    }
    splitString(buffer);
}
```

An integer called index is declared and initialised as zero. This integer will hold the position of a pointer to the characters within the char array.

We then set a delay of 100. The purpose of this is to ensure that the serial buffer (the place in memory where the serial data that is received is stored prior to processing) is full before we carry on and process the data. If we don't do that, it is possible that the function will execute and start to process the text string, before we have received all of the data. The serial communications line is very slow compared to the speed the rest of the code is executing at. When you send a string of characters the Serial.available function will immediately have a value higher than zero and the if function will start to execute. If we didn't have the delay(100) statement in there it could start to execute the code within the if statement before all of the text string had been received and the serial data may only be the first few characters of the line of text entered.

After we have waited for 100ms for the serial buffer to fill up with the data sent, we then declare and initialise the numChar integer to be the number of characters within the text string.

E.g. If we sent this text in the Serial Monitor:

R255, G255, Y255

Then the value of numChar would be 17. It is 17 and ot 16 as at the end of each line of text there is an invisible character called a NULL character. This is a 'nothing' symbol and simply tells the Arduino that the end of the line of text has been reached.

The next if statement checks if the value of numChar is greater than 15 or not and if so it sets it to be 15. This ensures that we don't overflow the array char buffer[18];

After this comes a while command. This is something

we haven't come across before so let me explain. We have already used the for loop, which will loop a set number of times. The while statement is also a loop, but one that executes only while a condition is true.

The syntax is

```
while(expression) {
! // statement(s)
}
```

In our code the while loop is

```
while (numChar--) {
```

```
buffer[index++] = Serial.read();
}
```

The condition it is checking is simply numChar, so in other words it is checking that the value stored in the integer numChar is not zero. numChar has -- after it.

This is what is known as a post-decrement. In other words, the value is decremented AFTER it is used. If we had used --numChar the value in numChar would be decremented (have one subtracted from it) before

it was evaluated. In our case, the while loop checks the value of numChar and then subtracts one from it.

If the value of numChar was not zero before the decrement, it then carries out the code within its code block.

numChar is set to the length of the text string that we have entered into the Serial Monitor window. So, the code within the while loop will execute that many times.

The code within the while loop is

```
buffer[index++] = Serial.read();
```

Which sets each element of the buffer array to each character read in from the Serial line. In other words, t fills up the buffer array with the letters we have entered into the Serial Monitor's text window.

The **Serial.read()** command reads incoming serial data, one byte at a time.

So now that our character array has been filled with the characters we entered in the Serial Monitor the while loop will end once numChar reaches zero (i.e. The length of the string).

After the while loop we have

```
splitString(buffer);
```

Which is a call to one of the two functions we have created and called splitString(). The function looks like this:

```
void splitString(char* data) {
  Serial.print("Data entered: ");
  Serial.println(data);
  char* parameter;
  parameter = strtok (data, " ,");
  while (parameter != NULL) {
    setLED(parameter);
    parameter = strtok (NULL, " ,");
  }
  // Clear the text and serial buffers
  for (int x=0; x<16; x++) {
    buffer[x]='\0';
  }
  Serial.flush();
}
```

We can see that the function returns no data, hence it's data type has been set to void. We pass the function one parameter and that is a char data type that we have called data.

However, in the C and C++ programming languages you are not allowed to send a character array to a function. We have got around that by using a pointer. We know we have used a pointer as an asterisk '*' has been added to the variable name *data. Pointers are quite an advanced subject in C so we won't go into too much detail about them. All you need to know for now is that by declaring 'data' as a pointer it is simply a variable that points to another variable.

You can either point it to the address that the variable is stored within memory by using the & symbol, or in our case, to the value stored at that memory address using the * symbol. We have used it to 'cheat' the system, as we are not allowed to send a character array to a function. However we are allowed to send a pointer to a character array to our function. So, we have declared a variable of data type Char and called it data, but the * symbol before it means that it is 'pointing to' the value stored within the 'buffer' variable.

When we called splitString we sent it the contents of 'buffer' (actually a pointer to it as we saw above).

splitString(buffer);

So we have called the function and passed it the entire contents of the buffer character array.

The first command is

```
Serial.print("Data entered: ");
```

and this is our way of sending data back from the Arduino to the PC. In this case the print command sends whatever is within the parenthesis to the PC, via the USB cable, where we can read it in the Serial Monitor window. In this case we have sent the words "Data entered: ". Text must be enclosed within quotes "". The next line is similar

```
Serial.println(data);
```

and again we have sent data back to the PC, this time we send the char variable called data. The Char type variable we have called 'data' is a copy of the contents of the 'buffer' character array that we passed to the function. So, if our text string entered was

R255 G127 Y56

Then the

```
Serial.println(data);
```

Command will send that text string back to the PC and print it out in the Serial Monitor window (make sure you have enabled the Serial Monitor window first).

This time the print command has ln on the end to make it println. This simply means 'print' with a 'linefeed'.

When we print using the print command, the cursor (the point at where the next symbol will appear) remains at the end of whatever we have printed.

When we use the println command a linefeed command is issued or in other words the text prints and then the cursor drops down to the next line.

```
Serial.print("Data entered: ");
Serial.println(data);
```

If we look at our two print commands, the first one prints out "Data entered: " and then the cursor remains at the end of that text. The next print command will print 'data', or in other words the contents of the array called 'buffer' and then issue a linefeed, or drop the

cursor down to the next line.

This means that if we issue another print or println statement after this whatever is printed in the Serial Monitor window will appear on the next line underneath the last.

We then create a new char data type called parameter

Char* parameter;

and as we are going to use this variable to access elements of the 'data' array it must be the same type, hence the * symbol. You cannot pass data from one data type type variable to another as the data must be converted first. This variable is another example of one that has 'local scope'. It can be 'seen' only by the code within this function. If you try to access the parameter variable outside of the splitString function you will get an error.

We then use a strtok command, which is a very useful command to enable us to manipulate text strings. Strtok gets it's name from String and Token as it's purpose is to split a string using tokens. In our case the token it is looking for is a space or a comma. It is used to split text strings into smaller strings.

We pass the 'data' array to the strtok command as the first argument and the tokens (enclosed within quotes) as the second argument. Hence

parameter = strtok (data, " ,");

And it splits the string at that point. So we are using it to set 'parameter' to be the part of the string up to a space or a comma.

So, if our text string was

R127 G56 Y98

Then after this statement the value of 'parameter' will be

R127

as the strtok command would have split the string up to the first occurrence of a space of a comma.

After we have set the variable 'parameter' to the part of the text string we want to strip out (i.e. The bit up to the first space or comma) we then enter a while loop whose condition is that parameter is not empty (i.e. We haven't reached the end of the string) using

while (parameter != NULL) {

Within the loop we call our second function

setLED(parameter);

Which we will look at later on. Then it sets the variable 'parameter' to the next part of the string up to the next space or comma. We do this by passing to strtok a NULL parameter

parameter = strtok (NULL, " ,");

This tells the strtok command to carry on where it last left off.

So this whole part of the function

```
char* parameter;
    parameter = strtok (data, " ,");
    while (parameter != NULL) {
        setLED(parameter);
        parameter = strtok (NULL, " ,");
```

```
}
```

is simply stripping out each part of the text string that is separated by spaces or commas and sending that part of the string to the next function called setLED().

The final part of this function simply fills the buffer array with NULL character, which is done with the /0 symbol and then flushes the Serial data out of the Serial buffer ready for the next set of data to be entered.

```
// Clear the text and serial buffers
for (int x=0; x<16; x++) {
    buffer[x]='\0';
}
Serial.flush();
```

The setLED function is going to take each part of the text string and set the corresponding LED to the colour we have chosen. So, if the text string we enter is

$ G125 Y55

Then the splitString() function splits that into the two separate components

$ G125

$ Y55

and send that shortened text string onto the setLED() function, which will read it, decide what LED we have chosen and set it to the corresponding brightness value.

So let's take a look at the second function called setLED().

```
void setLED(char* data) {
  if ((data[0] == 'r') || (data[0] == 'R'))
  {
  int Ans = strtol(data+1, NULL, 10);
  Ans = constrain(Ans,0,255);
  analogWrite(RedPin, Ans);
  Serial.print("Red is set to: ");
  Serial.println(Ans);
  }
  if ((data[0] == 'g') || (data[0] == 'G'))
  {
  int Ans = strtol(data+1, NULL, 10);
  Ans = constrain(Ans,0,255);
  analogWrite(GreenPin, Ans);
  Serial.print("Green is set to: ");
  Serial.println(Ans);
  }
  if ((data[0] == 'y') || (data[0] == 'Y'))
  {
  int Ans = strtol(data+1, NULL, 10);
  Ans = constrain(Ans,0,255);
  analogWrite(YellowPin, Ans);
  Serial.print("Yellow is set to: ");
```

```
    Serial.println(Ans);
    }
  }
```

We can see that this function contains 3 very similar if statements. We will therefore take a look at just one of them as the other 2 are almost identical.

```
if ((data[0] == 'r') || (data[0] == 'R')) {
  int Ans = strtol(data+1, NULL, 10);
  Ans = constrain(Ans,0,255);
  analogWrite(RedPin, Ans);
  Serial.print("Red is set to: ");
  Serial.println(Ans);
}
```

The if statement checks that the first character in the string data[0] is either the letter r or R (upper case and lower case characters are totally different as far as C is concerned. We use the logical OR command whose symbol is || to check if the letter is an r OR an R as either will do.

If it is an r or an R then the if statement knows we wish to change the brightness of the Red LED and so the code within executes. First we declare an integer called Ans (which has scope local to the setLED function only) and use the strtol (String to long integer) command to convert the characters after the letter R to an integer. The strtol command takes 3 parameters and these are the string we are passing it, a pointer to the character after the integer (which we don't use as we have already stripped the string using the strtok command and hence pass a NULL character) and then the 'base', which in our case is base 10 as we are using normal decimal numbers (as opposed to binary, octal or hexadecimal which would be base 2, 8 and 16 respectively). So in other words we declare an integer and set it to the value of the text string after the letter R (or the number bit).

Next we use the constrain command to make sure that Ans goes from 0 to 255 and no more. We then carry out an analogWrite command to the red pin and send it the value of Ans. The code then sends out "Red is set to: " followed by the value of Ans back to the Serial Monitor. The other two if statements do exactly the same but for the Green and the Blue LED's.

We have covered a lot of ground and a lot of new concepts in this project. To make sure you understand exactly what is going on in this code I am going to set the project code side by side with pseudo-code (an fake computer language that is essentially the computer language translated into a language humans can understand).

The C Programming Language	**Pseudo-Code**
`// Project 10 - Serial controlled RGB Lamp`	
`char buffer[18];`	A comment with the project number and name Declare a
`int red, green, blue;`	character array of 18 letters Declare 3 integers called red, green

int RedPin = 11;	and blue An integer for which pin to use for Red LED
int GreenPin = 10;	" " Green
int YellowPin = 9;	" " Blue
void setup()	The setup function
{	
Serial.begin(9600);	Set serial comms to run at 9600 chars per second
Serial.flush();	Flush the serial line
pinMode(RedPin, OUTPUT);	Set the red led pin to be an output pin
pinMode(GreenPin, OUTPUT);	Same for green
pinMode(YellowPin, OUTPUT);	And yellow
}	
void loop()	The main program loop
{	
if (Serial.available() > 0) {	If data is sent down the serial line...
int index=0;	Declare integer called index and set to 0
delay(100); // let the buffer fill up	Wait 100 millseconds
int numChar = Serial.available();	Set numChar to the incoming data from serial
if (numChar>15) {	If numchar is greater than 15 characters...
numChar=15;	Make it 15 and no more
}	
while (numChar--) {	While numChar is not zero (subtract 1 from it) Set
buffer[index++] = Serial.read();	element[index] to value read in (add 1) Call splitString
}	function and send it data in buffer
splitString(buffer);	The splitstring function references buffer data
}	
}	
void splitString(char* data) {	
Serial.print("Data entered: ");	Print "Data entered: "
Serial.println(data);	Print value of data and then drop down a line
char* parameter;	Declare char data type parameter
parameter = strtok (data, " ,");	Set it to text up to the first space or comma
while (parameter != NULL) {	While contents of parameter are not empty..
setLED(parameter);	! Call the setLED function
parameter = strtok (NULL, " ,");	Set parameter to next part of text string
}	
// Clear the text and serial buffers	Another comment
for (int x=0; x<16; x++) {	We will do the next line 16 times
buffer[x]='\0';	Set each element of buffer to NULL (empty)
}	
Serial.flush();	Flush the serial comms
}	
void setLED(char* data) {	A function called setLED is passed buffer

```
if ((data[0] == 'r') || (data[0] == 'R'))   If first letter is r or R...
{
int Ans = strtol(data+1, NULL, 10);         Set integer Ans to number in next part of text
Ans = constrain(Ans,0,255);                 Make sure it is between o and 255
analogWrite(RedPin, Ans);                   Write that value out to the red pin
Serial.print("Red is set to: ");            Print out "Red is set to: "
Serial.println(Ans);                        And then the value of Ans
}
if ((data[0] == 'g') || (data[0] == 'G'))   If first letter is g or G...
{
int Ans = strtol(data+1, NULL, 10);         Set integer Ans to number in next part of text
Ans = constrain(Ans,0,255);                 Make sure it is between o and 255
analogWrite(GreenPin, Ans);                 Write that value out to the green pin
Serial.print("Green is set to: ");          Print out "Green is set to: "
Serial.println(Ans);                        And then the value of Ans
}
if ((data[0] == 'y') || (data[0] == 'Y'))   If first letter is y or Y...
{
int Ans = strtol(data+1, NULL, 10);         Set integer Ans to number in next part of text
Ans = constrain(Ans,0,255);                 Make sure it is between o and 255
analogWrite(YellowPin, Ans);                Write that value out to the yellow pin
Serial.print(" Yellow is set to: ");        Print out "Yellow is set to: "
Serial.println(Ans);                        And then the value of Ans
}
}
```

Hopefully you can use this 'pseudo-code' to make sure you understand exactly what is going on in this projects code.

We are now going to leave LED's behind for a little while and look at how to control a DC Motor.

Project 4 - Piezo Sounder Player

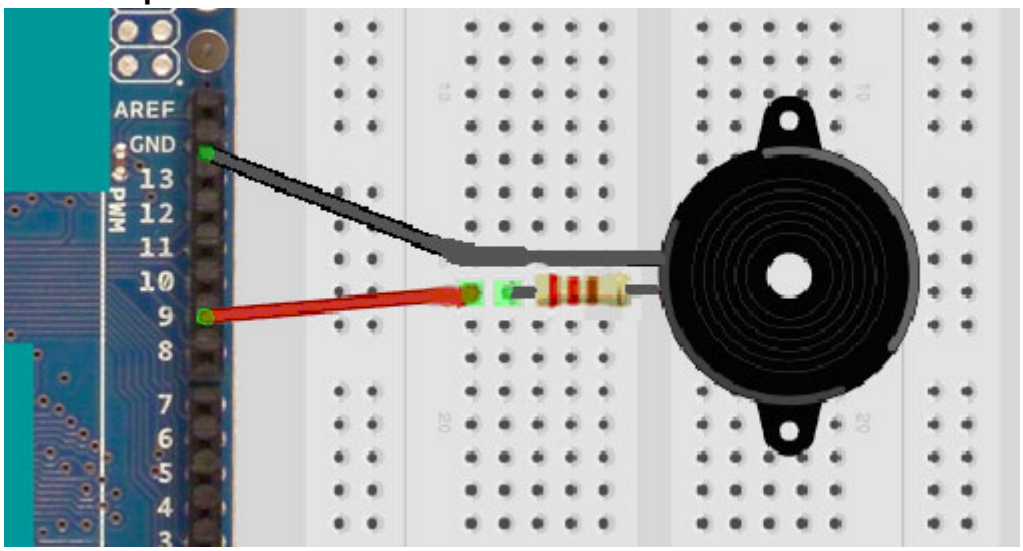
In this project we are going to use a super simple circuit to produce sounds from our Arduino using a Piezo Sounder.

What you will need

Piezo Disc

220 ohm resistor

Connect it up



```
//(courtesy of
http://www.arduino.cc/en/Tutorial/Melody )
// Project 12 - Melody Player
int speakerPin = 9;
int length = 15; // the number of notes
char notes[] = "ccggaagffeeddc "; // a space
represents a rest
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1,
1, 1, 1, 1, 2, 4 };
int tempo = 300;
void playTone(int tone, int duration) {
for (long i = 0; i < duration * 1000L; i +=
tone * 2) {
digitalWrite(speakerPin, HIGH);
delayMicroseconds(tone);
digitalWrite(speakerPin, LOW);
delayMicroseconds(tone);
}
}
void playNote(char note, int duration) {
char names[] = { 'c', 'd', 'e', 'f', 'g', 'a',
'b', 'C' };
int tones[] = { 1915, 1700, 1519, 1432, 1275,
1136, 1014, 956 };


// play the tone corresponding to the note
name
for (int i = 0; i < 8; i++) {
if (names[i] == note) {
playTone(tones[i], duration);
}
}
}
void setup() {
pinMode(speakerPin, OUTPUT);
}
void loop() {
for (int i = 0; i < length; i++) {
if (notes[i] == ' ') {
delay(beats[i] * tempo); // rest
} else {
playNote(notes[i], beats[i] * tempo);
}
// pause between notes
delay(tempo / 2);
}
}
```

When you run this code the Arduino will play a very nice (yeah ok it's terrible) rendition of 'Twinkle Twinkle Little Star'. Sounding very similar to those annoying birthday cards you can buy that play a tune when you open it up.

Let's take a look at this code and see how it works and find out what a piezo disc is.

Project 4 - Code Overview

In this project we are making sounds using a piezo disc. A piezo disc can do nothing more than make a click when we apply a voltage to it. So to get the tones we can hear out of it we need to make it click many times a second fast enough that it becomes a recognisable note.

The program starts off by setting up the variables we need. The piezo sounders positive (red) cable is attached to Pin 9.

```
int speakerPin = 9;
```

The tune we are going to play is made up of 15 notes.

```
int length = 15; // the number of notes
```

The notes of the tune are stored in a character array as a text string.

```
char notes[] = "ccggaagffeeddc ";
```

Another array, this time of integers, is set up to store the length of each note.

```
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
```

And finally we set a tempo for the tune to be played at,

```
int tempo = 300;
```

Next you will notice that we declare two functions before our setup() and loop() functions. It doesn't matter if we put our own functions before or after setup() and loop(). When the program runs, the code within these two functions will not run before setup() runs as we have not called those functions yet.

Let's look at the setup and loop functions before we look at the playTone and playNote functions. All that happens in setup() is we assign the speaker pin (9) as an output.

```
void setup() {
pinMode(speakerPin, OUTPUT);
}
```

In the main program loop we have an if/else statement inside a for loop.

```
for (int i = 0; i < length; i++) {
if (notes[i] == ' ') {
delay(beats[i] * tempo); // rest
} else {
playNote(notes[i], beats[i] * tempo);
}
```

As you can see, the first if statement has as it's condition, that the array element [i] that the element contains a space character.

```
if (notes[i] == ' ')
```

If this is TRUE then the code within it's block is executed.

```
delay(beats[i] * tempo); // rest
```

and this simply works out the value of `eats[i] * empo` nd causes a delay of that length to cause a rest in the notes. We then have an else statement.

```
else {
```

```
playNote(notes[i], beats[i] * tempo);
}
```

After an if statement we can extend it with an else statement. An else statements is carried out if the condition within the if statement is false. So, for example. Let's say we had an integer called test and it's value was 10 and this if/else statement:

```
if (test == 10) {
digitalWrite(ledPin, HIGH)
} else {
digitalWrite(ledPin, LOW)
}
```

Then if 'test' had a value of 10 (which it does) the ledPin would be set to HIGH. If the value of test was anything other than 10, the code within the else statement would be carried out instead and the ledPin would be set to LOW.

The else statement calls a function called playNote and passes two parameters. The first parameter is the value of notes[i] and the second is the value calculated from beats[i] * tempo.

```
playNote(notes[i], beats[i] * tempo);
```

After if/else statement has been carried out, there is a delay whose value is calculated by dividing tempo by 2.

```
delay(tempo / 2);
```

Let us now take a look at the two functions we have created for this project.

The first function that is called from the main program loop is playNote.

```
void playNote(char note, int duration) {
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a',
  'b', 'C' };
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
// play the tone corresponding to the note name
  for (int i = 0; i < 8; i++) {
    if (names[i] == note) {
    playTone(tones[i], duration);
   }
}
```

Two parameters have been passed to the function and within the function these have been given the names note (character) and duration (integer).

The function sets up a local variable array of data type char called □names□. This variable has local scope so is only visible to this function and not outside of it.

This array stores the names of the notes from middle C to high C.

We then create another array of data type integer and this array stores numbers that correspond to the frequency of the tones, in Kilohertz, of each of the notes in the names[] array.

```
int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
```

After setting up the two arrays there is a for loop that looks through the 8 notes in the

names[] array and compares it to the note sent to the function.

```
for (int i = 0; i < 8; i++) {
  if (names[i] == note) {
  playTone(tones[i], duration);
  }
}
```

The tune that is sent to this function is 'ccggaagffeeddc' so the first note will be a middle C.

The for loop compares that note with the notes in the names[] array and if there is a match, calls up the second function, called play Tone, to play thecorresponding tone using in the tones[] array using a note length of 'duration'.

The second function is called playTone.

```
void playTone(int tone, int duration) {
    for (long i = 0; i < duration * 1000L; i +=tone * 2) {
      digitalWrite(speakerPin, HIGH);
      delayMicroseconds(tone);
      digitalWrite(speakerPin, LOW);
      delayMicroseconds(tone);
    }
}
```

Two parameters are passed to this function. The first is the tone (in kilohertz) that we want the piezo speaker to reproduce and the second is the duration (made up by calculating `beats[i] * tempo`.

The function starts a for loop

```
for (long i = 0; i < duration * 1000L; i += tone
    * 2)
```

As each for loop must be of a different length to make each note the same length (as the delay differs between clicks to produce the desired frequency) the for loop will run to duration multiplied by 1000 and the increment of the loop is the value of 'tone' multiplied by 2.

Inside the for loop we simply make the pin connected to the piezo speaker go high, wait a short period of time, then go low, then wait another short period of time, then repeat.

```
digitalWrite(speakerPin, HIGH);
delayMicroseconds(tone);
digitalWrite(speakerPin, LOW);
delayMicroseconds(tone);
```

These repetitive clicks, of different lengths and with different pauses (of only microseconds in length) in between clicks, makes the piezo produce a tone of varying frequencies.

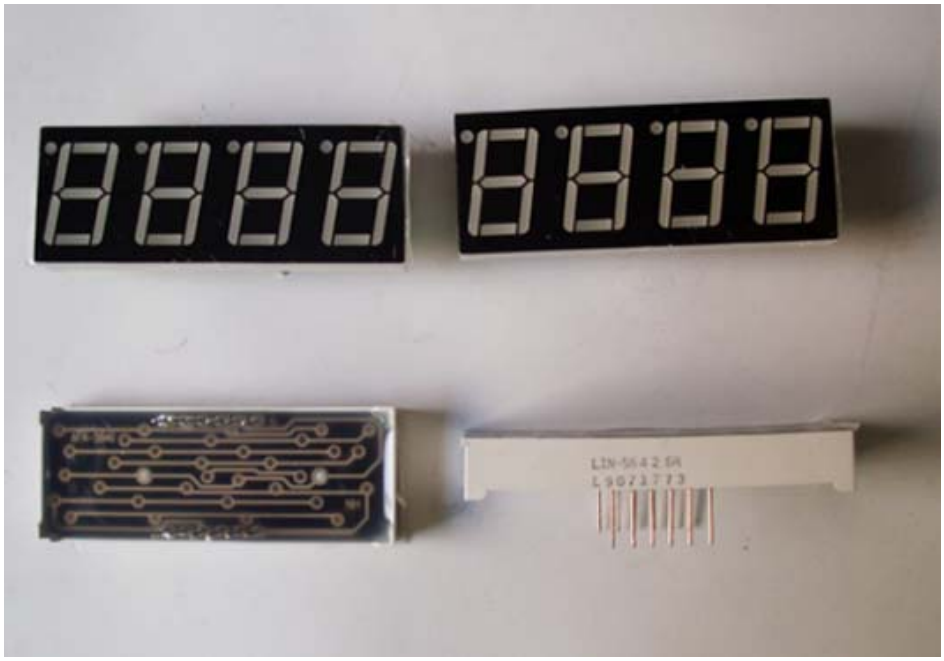
Exercise

1. Change the notes and beats to make other tunes such as 'Happy Birthday' or 'Merry Christmas'.

2. Write a program to make a rising and falling tone from the piezo, similar to a car alarm or police siren.

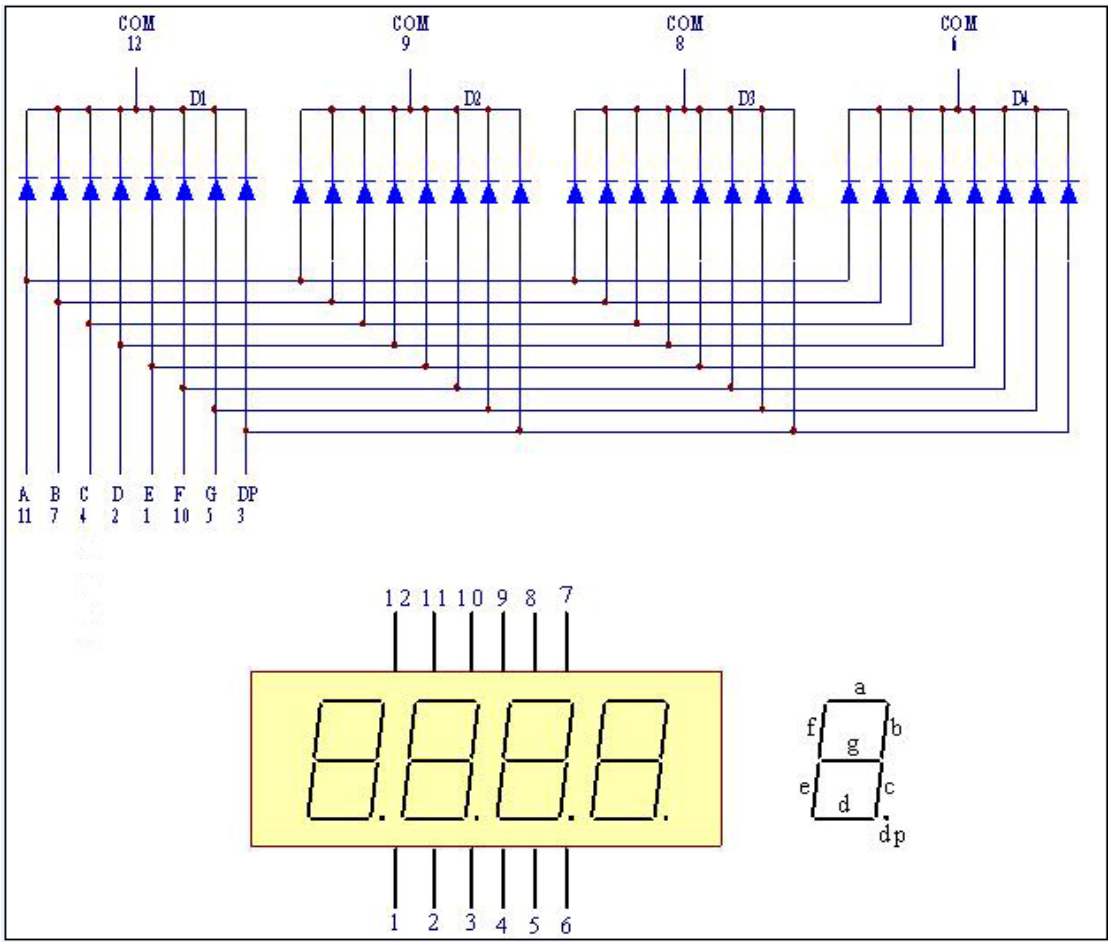
Project 5 – 7seg-4digit LED Display

1. What is an 7seg-4digit LED Display?



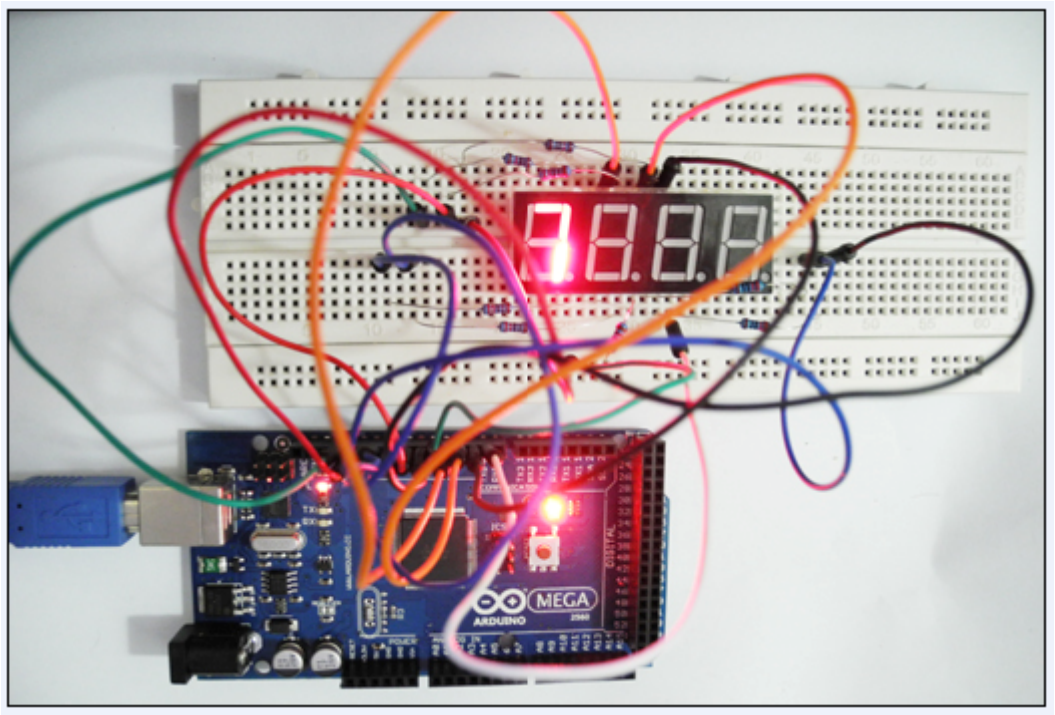
In the following picture, there are other LED displays.

The internal structure Of LED display. This is the **Common cathode** 7seg-4digit LED Display that we will use in this Experiment.



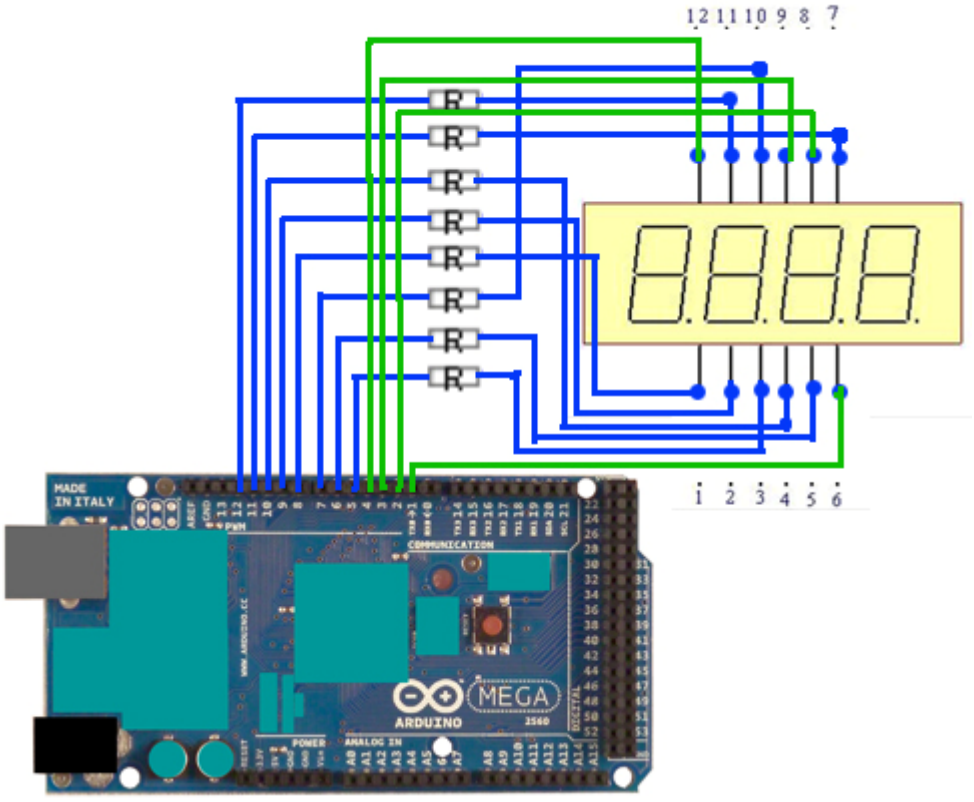
First Experiment:

Prepare: 1 X 7seg-4digit LED Display , 8 X 220-ohm resistor , some Jumper cable.



To build the circuit, attach the LED display's (through a 220-ohm resistor) Pin 11, 7, 4, 2, 1, 10, 5, 3 to your Arduino Mega 2560 board's digital Pin 12, 11, 10, 9, 8, 7, 6, 5 .

Attach the LED display's 12, 9, 8, 6 to the Arduino Mega 2560 board's digital Pin 4, 3, 2, 1 .



Code : open the folder Project_5 , and open the Project_5.pde

In the program, It will loop to show the number from 0 to 9

```
//设置控制各段的数字 IO 脚
int a = 12;
int b = 11;
int c = 10;
int d = 9;
int e = 8;
int f = 7;
int g = 6;
int dp = 5;
int ls1 = 1; //最右边第一位数码管
int ls2 = 2; //第二位数码管
int ls3 = 3; //第三位数码管
int ls4 = 4; //第四位数码管


//显示数字 0 ， 11111100
void digital_0(void)
{
  unsigned char j;
  digitalWrite(dp,LOW);//给数字 5 引脚低电平，熄灭小数点 DP 段
  digitalWrite(g,LOW);//熄灭 g 段
  for(j=7;j<=12;j++)//点亮其余段
    digitalWrite(j,HIGH);
}
//显示数字 1 , 01100000
void digital_1(void)
{
  unsigned char j;
  digitalWrite(a,LOW);
  for(j=5;j<=9;j++)
    digitalWrite(j,LOW);
  digitalWrite(b,HIGH);
  digitalWrite(c,HIGH);
}
//显示数字 2
void digital_2(void)
{
  unsigned char j;
  digitalWrite(c,LOW);
```

```
    digitalWrite(f,LOW);
    digitalWrite(dp,LOW);
    for(j=11;j<=12;j++)
      digitalWrite(j,HIGH);
    for(j=8;j<=9;j++)
      digitalWrite(j,HIGH);
    digitalWrite(g,HIGH);
}
//显示数字 3 22
void digital_3(void)
{
    unsigned char j;
    digitalWrite(e,LOW);
    digitalWrite(f,LOW);
    digitalWrite(dp,LOW);
    for(j=9;j<=13;j++)
      digitalWrite(j,HIGH);
    digitalWrite(g,HIGH);
}
//显示数字 4
void digital_4(void)
{
    digitalWrite(a,LOW);
    digitalWrite(d,LOW);
    digitalWrite(e,LOW);
    digitalWrite(dp,LOW);
    digitalWrite(b,HIGH);
    digitalWrite(c,HIGH);
    digitalWrite(f,HIGH);
    digitalWrite(g,HIGH);
}
//显示数字 5
void digital_5(void)
{
    unsigned char j;
    digitalWrite(b,LOW);
    digitalWrite(e,LOW);
    digitalWrite(dp,LOW);

    digitalWrite(a,HIGH);
    for(j=6;j<=7;j++)
      digitalWrite(j,HIGH);
    for(j=9;j<=10;j++)
```

```
        digitalWrite(j,HIGH);
}
//显示数字 6
void digital_6(void)
{
   unsigned char j;
   digitalWrite(b,LOW);
   digitalWrite(dp,LOW);

   digitalWrite(a,HIGH);
   for(j=6;j<=10;j++)
      digitalWrite(j,HIGH);
}
//显示数字 7
void digital_7(void)
{
   unsigned char j;
   for(j=5;j<=9;j++)
      digitalWrite(j,LOW);
   for(j=10;j<=12;j++)
      digitalWrite(j,HIGH);
}
//显示数字 8
void digital_8(void)
{
   unsigned char j;
   digitalWrite(dp,LOW);
   for(j=6;j<=12;j++)
      digitalWrite(j,HIGH);
}
//显示数字 9
void digital_9(void)
{
   unsigned char j;
   digitalWrite(e,LOW);
   digitalWrite(dp,LOW);

   for(j=9;j<=12;j++)
      digitalWrite(j,HIGH);
   for(j=6;j<=7;j++)
      digitalWrite(j,HIGH);
}
```

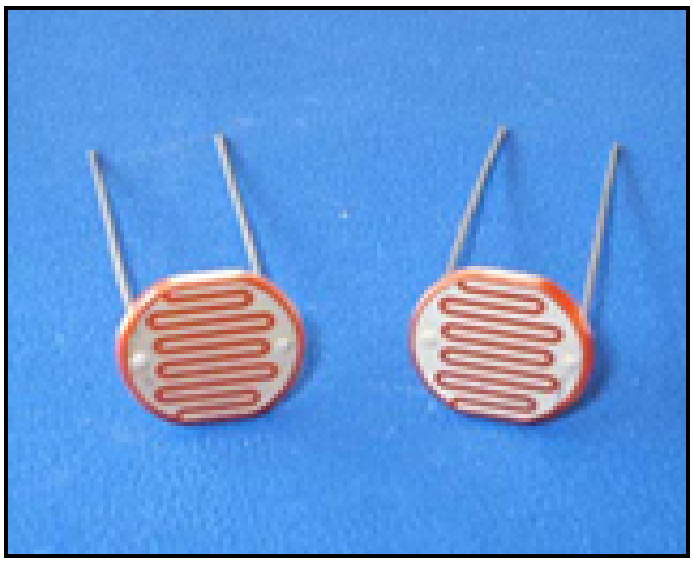
```
//显示----------------------------
void Display(unsigned char mun)
{
  switch (mun) {
    case 0:
      digital_0();//显示数字 0
      break;
    case 1:
      digital_1();
      break;
    case 2:
      digital_2();
      break;
    case 3:
      digital_3();
      break;
    case 4:
      digital_4();
      break;
    case 5:
      digital_5();
      break;
    case 6:
      digital_6();
      break;
    case 7:
      digital_7();
      break;
   case 8:
      digital_8();
      break;
   case 9:
      digital_9();
      break;
    default: return;
      // if nothing else matches, do the default
      // default is optional
  }
}

void setup()
{
  int i;//定义变量
```

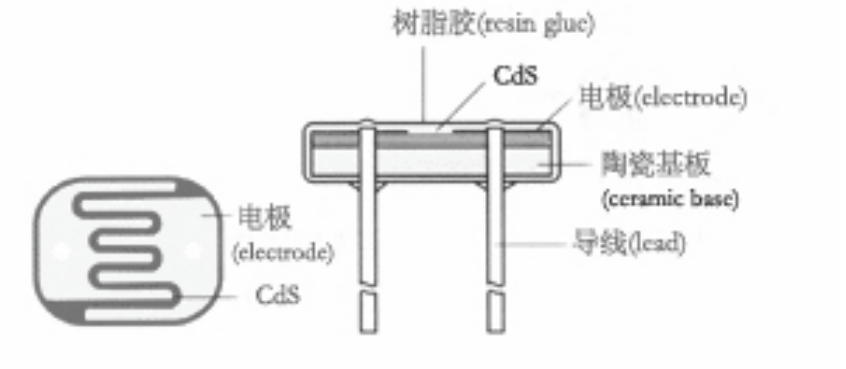
```
  for(i=1;i<=12;i++)
    pinMode(i,OUTPUT);//设置 1～12 引脚为输出模式
}
void loop()
{
  unsigned char i, j, t;
  while(1)
  {
    t = 1;
    for(j=0; j<4 ; j++)
    {
       digitalWrite(j,LOW); //熄灭数码管使能端
    }

    for(i=0; i<=9 ; i++)
    {
      if(t==1) //第一位数码管显示数字
      {
        digitalWrite(ls4,HIGH);
        digitalWrite(ls3,HIGH);
        digitalWrite(ls2,HIGH);
        digitalWrite(ls1,LOW);
      }
      if(t==2) //第二位数码管显示数字
      {
        digitalWrite(ls4,HIGH);
        digitalWrite(ls3,HIGH);
        digitalWrite(ls1,HIGH);
        digitalWrite(ls2,LOW);
      }
      if(t==3) //第三位数码管显示数字
      {
        digitalWrite(ls4,HIGH);
        digitalWrite(ls2,HIGH);
        digitalWrite(ls1,HIGH);
        digitalWrite(ls3,LOW);
      }
      if(t==4) //第四位数码管显示数字
      {
        t = 0;
        digitalWrite(ls3,HIGH);
        digitalWrite(ls2,HIGH);
        digitalWrite(ls1,HIGH);
```

```
            digitalWrite(ls4,LOW);
          }
          Display(i);
          delay(2000);//延时 2s
          t++;

          for(j=0; j<4 ; j++)
          {
            digitalWrite(ls1,LOW); //熄灭数码管使能端
          }
        }
      }
    }
```

Lesson 6 -- Ambient Light sensor

1. What is an Ambient Light sensor ?

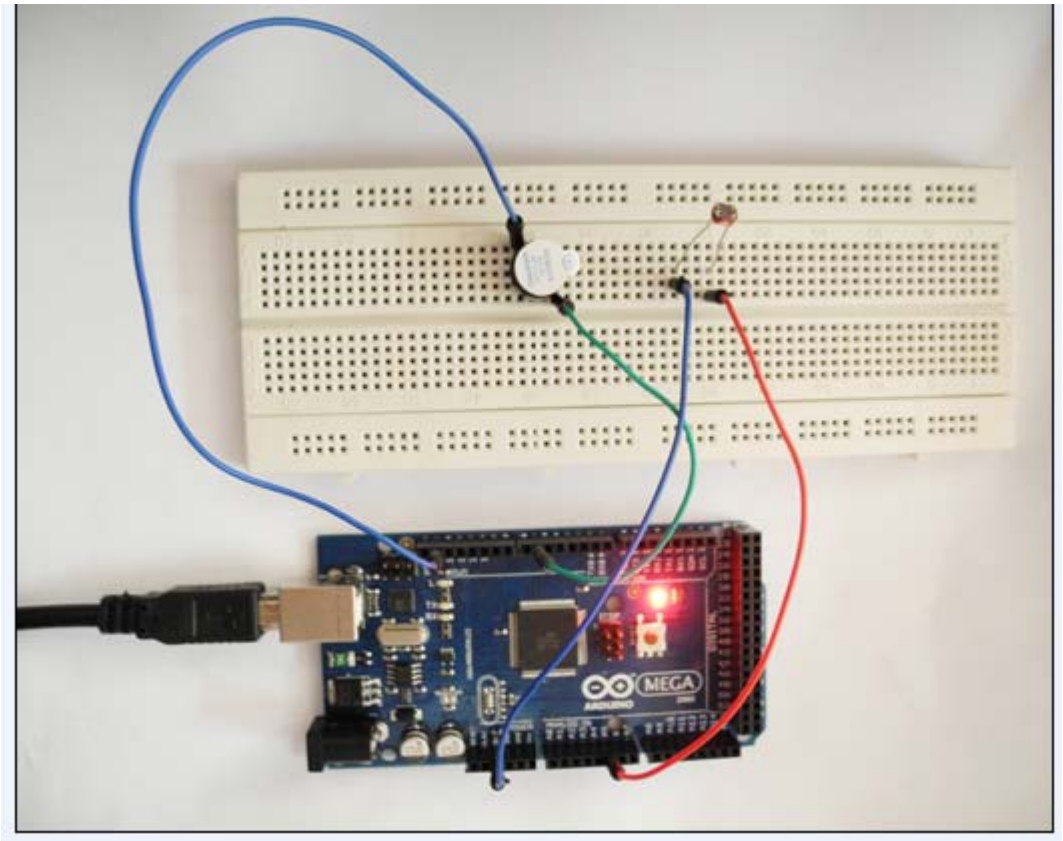




In the light intensity is strong , it's resistance decreases; however , when the light is weak, this sensor's resistance increases.

First Experiment:

Prepare: 1 X Ambient Light sensor , 1 X buzzer , some Jumper cable.



To build the circuit, attach buzzer's long, positive leg (called the anode) to pin 6 . Connect the **cathode** (the shorter, negative leg) directly to ground.

Attach the Ambient Light sensor pin to the 5 volt supply, the other side to Analog Pin 5 .

Code : open the folder Project_6 , and open the Project_6.pde

In the program, when the Light is stronger, the resistance of sensor will be smaller, and then the buzzer speak louder.

```
/*
Created By: Samleong
Created On: 2011-9-17
Website / More Infomation: http://www.b2cqshop.com
Email: b2cqshop@gmail.com
*/
void setup()
{
    pinMode(6,OUTPUT);
}
void loop()
{
```

```
  int val = analogRead(5);
  char i,j;
  if(val>1000)
  {
        for(i=0;i<80;i++)       //Output a frequency of sound
         {
         digitalWrite(6,HIGH);
         delay(1);
         digitalWrite(6,LOW);
         delay(1);
         }
  }
  else if(val>800)
  {
         for(i=0;i<100;i++)     //Output another frequency of sound
         {
         digitalWrite(6,HIGH);
         delay(4);
         digitalWrite(6,LOW);
         delay(4);
         }
  }
}
```

Project 7 – EEPROM

In the project, you will use I2C bus .

An EEPROM is an Electrically Erasable Programmable Read-Only Memory. It is a form of non-volatile memory that can remember things with the power being turned off. The beauty of this kind of memory is that we can store data generated within a sketch on a more permanent basis. Sometimes you might need to store a lot of reference data for use in calculations during a sketch, such as a mathematical table; or perhaps numerical representations of maps or location data; or create your own interpreter within a sketch that takes instruction from data stored in an array.

In other words, an EEPROM can be used to store data of a more permanent use, ideal for when your main microcontroller doesn't haven enough memory for you to store the data in the program code. However, EEPROMs are not really designed for random-access or constant read/write operations – they have a finite lifespan. But their use is quite simple, so we can take advantage of them.

What sort of data can be stored in EEPROM?

Anything that can be represented as bytes of data. One byte of data is made up of eight bits of data. A bit can be either on (value 1) or off (value 0), and are perfect for representing numbers in binary form. In other words, a binary number can only uses zeros and ones to represent a value. Thus binary is also known as "base-2", as it can only use two digits.

How can a binary number with only the use of two digits represent a larger number?

It uses a lot of ones and zeros. Let's examine a binary number, say 10101010. As this is a base-2 number, each digit represents 2 to the power of x, from x=0 onwards:

	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
Binary	1	0	1	0	1	0	1	0
Base-10	1	0	4	0	16	0	64	0

See how each digit of the binary number can represent a base-10 number. So the binary number above represents 85 in base-10 – the value 85 is the sum of the base-10 values. Another example – 11111111 in binary equals 255 in base 10.

	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
Binary	1	1	1	1	1	1	1	1
Base-10	1	2	4	8	16	32	64	128

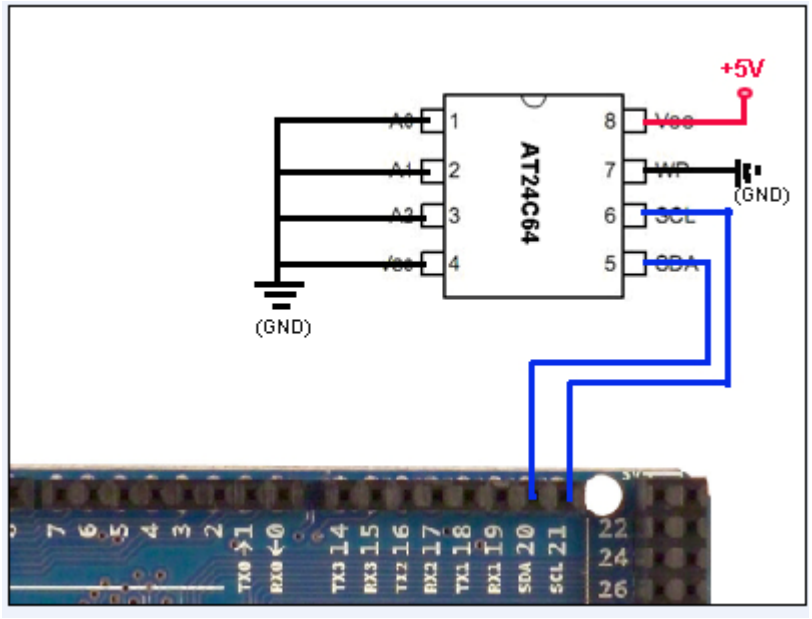
Now each digit in that binary number uses one 'bit' of memory, and eight bits make a byte. Due to internal limitations of the microcontrollers in our Arduino boards, we can only

store 8-bit numbers (one byte) in the EEPROM. This limits the decimal value of the number to fall between zero and 255. It is then up to you to decide how your data can be represented with that number range. Don't let that put you off – numbers arranged in the correct way can represent almost anything!

What you will need

1 X EEPROM chip AT24C64

Connect it up



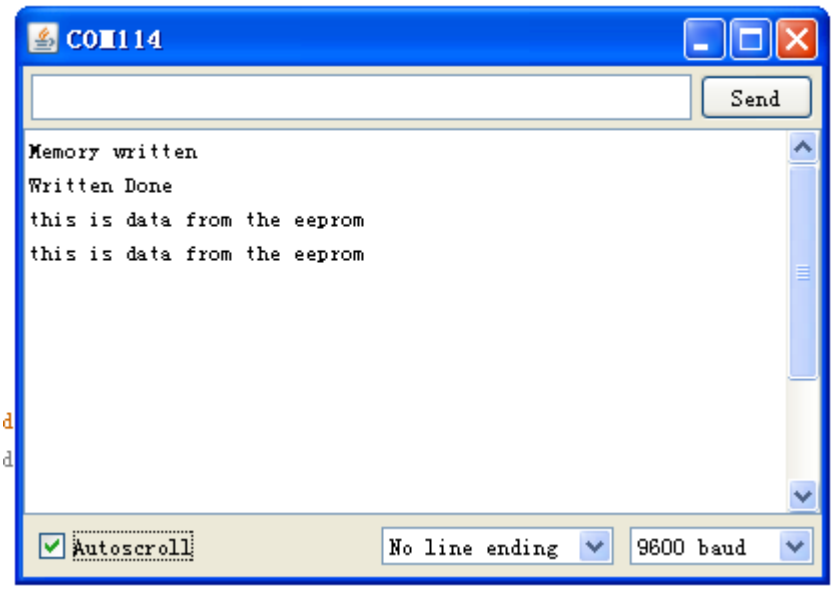
Code :

```
/*
  *  Use the I2C bus with EEPROM 24C64
  *  Sketch:    eeprom.pde
  *  WWW.B2CQSHOP.COM
  *  Author: Evan
  *  Date: 21/10/2011
  *
  */

  #include <Wire.h> //I2C library

  void i2c_eeprom_write_byte( int deviceaddress, unsigned int eeaddress, byte data ) {
    // writes a byte of data 'data' to the chip at I2C address 'deviceaddress', in memory location
```

```
'eeaddress'
      int rdata = data;
      Wire.beginTransmission(deviceaddress);
      Wire.send((int)(eeaddress >> 8)); // MSB
      Wire.send((int)(eeaddress & 0xFF)); // LSB
      Wire.send(rdata);
      Wire.endTransmission();
   }

   // WARNING: address is a page address, 6-bit end will wrap around
   // also, data can be maximum of about 30 bytes, because the Wire library has a buffer of 32 bytes
   void i2c_eeprom_write_page( int deviceaddress, unsigned int eeaddresspage, byte* data, byte length ) {
      Wire.beginTransmission(deviceaddress);
      Wire.send((int)(eeaddresspage >> 8)); // MSB
      Wire.send((int)(eeaddresspage & 0xFF)); // LSB
      byte c;
      for ( c = 0; c < length; c++)
         Wire.send(data[c]);
      Wire.endTransmission();
   }

   byte i2c_eeprom_read_byte( int deviceaddress, unsigned int eeaddress ) {
      // reads a byte of data from memory location 'eeaddress' in chip at I2C address 'deviceaddress'
      byte rdata = 0xFF; // returned value
      Wire.beginTransmission(deviceaddress);    //    these three lines set the pointer position in the EEPROM
      Wire.send((int)(eeaddress >> 8)); // MSB
      Wire.send((int)(eeaddress & 0xFF)); // LSB
      Wire.endTransmission();
      Wire.requestFrom(deviceaddress,1);// now get the byte of data...
      if (Wire.available()) rdata = Wire.receive();
      return rdata;
   }

   // maybe let's not read more than 30 or 32 bytes at a time!
   void i2c_eeprom_read_buffer( int deviceaddress, unsigned int eeaddress, byte *buffer, int length ) {
      Wire.beginTransmission(deviceaddress);
      Wire.send((int)(eeaddress >> 8)); // MSB
      Wire.send((int)(eeaddress & 0xFF)); // LSB
      Wire.endTransmission();
```

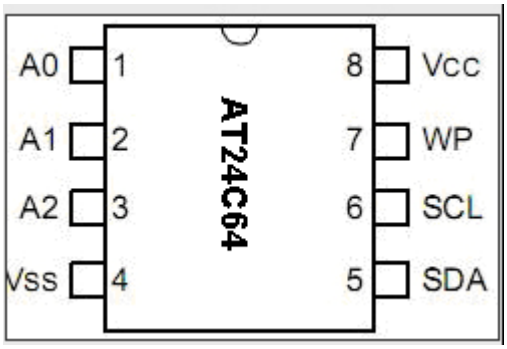
```
    Wire.requestFrom(deviceaddress,length);
    int c = 0;
    for ( c = 0; c < length; c++ )
      if (Wire.available()) buffer[c] = Wire.receive();
}




void setup()
{
     Serial.begin(9600);
      Serial.println("Memory written");
    char somedata[] = "this is data from the eeprom"; // data to write
    Wire.begin(); // initialise the connection

    i2c_eeprom_write_page(0x50, 0, (byte *)somedata, sizeof(somedata)); // write to EEPROM

    delay(10); //add a small delay

    Serial.println("Written Done");
}

void loop()
{
    int addr=0; //first address
    byte b = i2c_eeprom_read_byte(0x50, 0); // access the first address from the memory

    while (b!=0)
    {
      Serial.print((char)b); //print content to serial port
      addr++; //increase address
      b = i2c_eeprom_read_byte(0x50, addr); //access an address from the memory
    }
    Serial.println(" ");
    delay(2000);

}
```

The result of the project is :

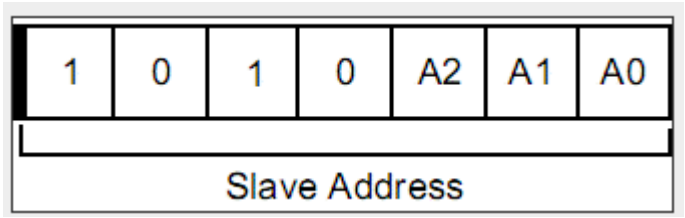

COM114
Send
Memory written
Written Done
this is data from the eeprom
this is data from the eeprom
Autoscroll
No line ending
9600 baud

Project 7 - Hardware Overview

EEPROMS, like anything else come in many shapes and sizes. The model we will examine today is the Microchip 24LC64 . It can hold 64 kilobits of data (that's 8 kilobytes). If you are using an Arduino LilyPad or Pro Mini 3.3V, etc., this EEPROM can also run on 3.3V. This model also has selectable device addresses using three pins, so we can use up to eight at once on the same bus. An example:



Pin 7 (WP) is "write protect" – set this low for read/write or high for read only. You could also control this in software if necessary. Once again we need to create a slave I2C device address using pins 1, 2 and 3 – these correlate to A2, A2 and A0 in the following table:



So if you were just using one 24LC64, the easiest solution would be to set A0~A2 to GND – which makes your slave address 1010000 or 0×50 in hexadecimal. There are several things to understand when it comes to reading and writing our bytes of data. As this IC has 8 kilobytes of storage, we need to be able to reference each byte in order to read or write to it. There is a slight catch in that you need more than one byte to reference 32767 (as in binary 32767 is 11111111 0100100 [16 bits]).

So when it comes time to send read and write requests, we need to send two bytes down the bus – one representing the higher end of the address (the first 8 bits from left to right), and the next one representing the lower end of the address (the final 8 bits from left to right)

An example – we need to reference byte number 25000. In binary, 25000 is 0110000110101000. So we split that up into 01100001 and 10101000, then covert the binary values to numerical bytes with which to send using the Wire.send(). Thankfully there are two operators to help us with this. This first is >>, known as bitshift right. This will take the higher end

of the byte and drop off the lower end, leaving us with the first 8 bits. To isolate the lower end of the address, we use another operator &, known as bitwise and. This unassuming character, when used with 0XFF can separate the lower bits for us. This may seem odd, but will work in the examples below.

Writing data to the 24LC256

Writing data is quite easy. But first remember that a byte of data is 11111111 in binary, or 255 in decimal. First we wake up the I2C bus with

Wire.beginTransmission(0x50); // if pins A0~A2 are set to GND

then send down some data. The first data are the two bytes representing the address (**25000**) of the byte (**12**) we want to write to the memory.

```
Wire.send(25000 >> 8);    // send the left-hand side of the address down
Wire.send(25000 & 0xFF); // send the right-hand side of the address down
```

And finally, we send the byte of data to store at address 25000, then finish the connection:

```
Wire.send(12);
Wire.endTransmission();
```

There we have it. Now for getting it back…

Reading data from the 24LC256

Reading is quite similar. First we need to start things up and move the pointer to the data we want to read:

```
Wire.beginTransmission(0x50); // if pins A0~A2 are set to GND
Wire.send(25000 >> 8);    // send the left-hand side of the address down
Wire.send(25000 & 0xFF); // send the right-hand side of the address down
Wire.endTransmission();
```

Then, ask for the byte(s) of data starting at the current address:

```
Wire.beginTransmission(0x50); // if pins A0~A2 are set to GND
Wire.requestFrom(0x50,1);
Wire.receive(incomingbyte);
```

In this example, incomingbyte is a byte variable used to store the data we retrieved from the IC.

Project 8 - Infrared Remote Control

In this project we will learn how to control the Infrared Remote .

To know more about how to control IR Remote Control. You should have a knowledge of the **NEC Protocol** . Just do this project first, Go aheard .

(**Please note that, connect the Infrared Receiver correctly .)**

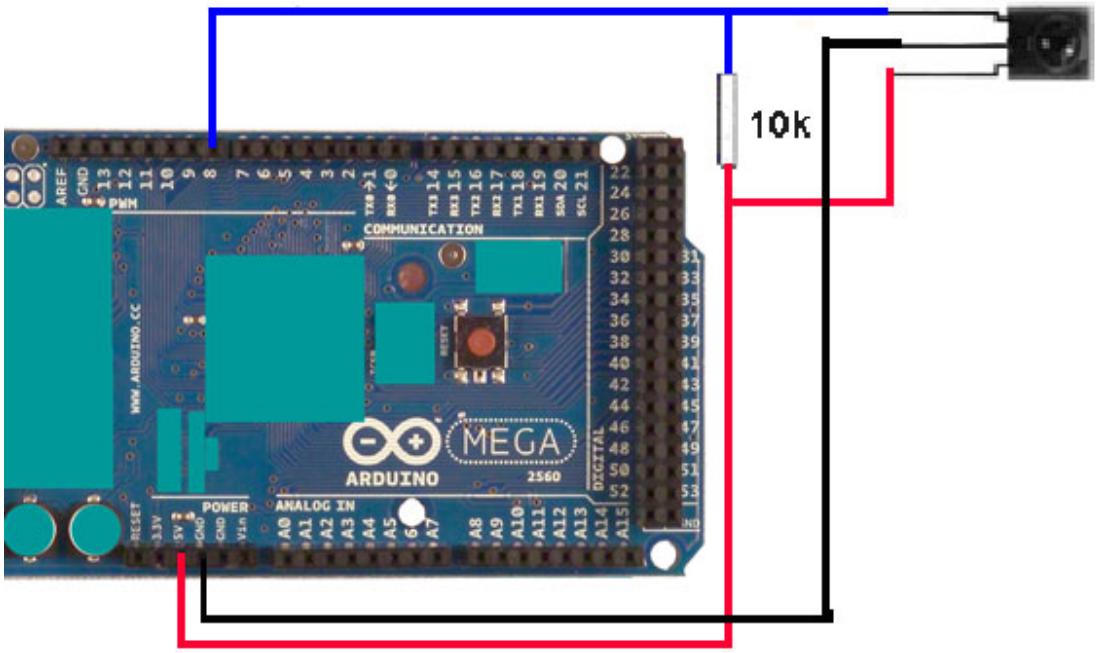
What you will need

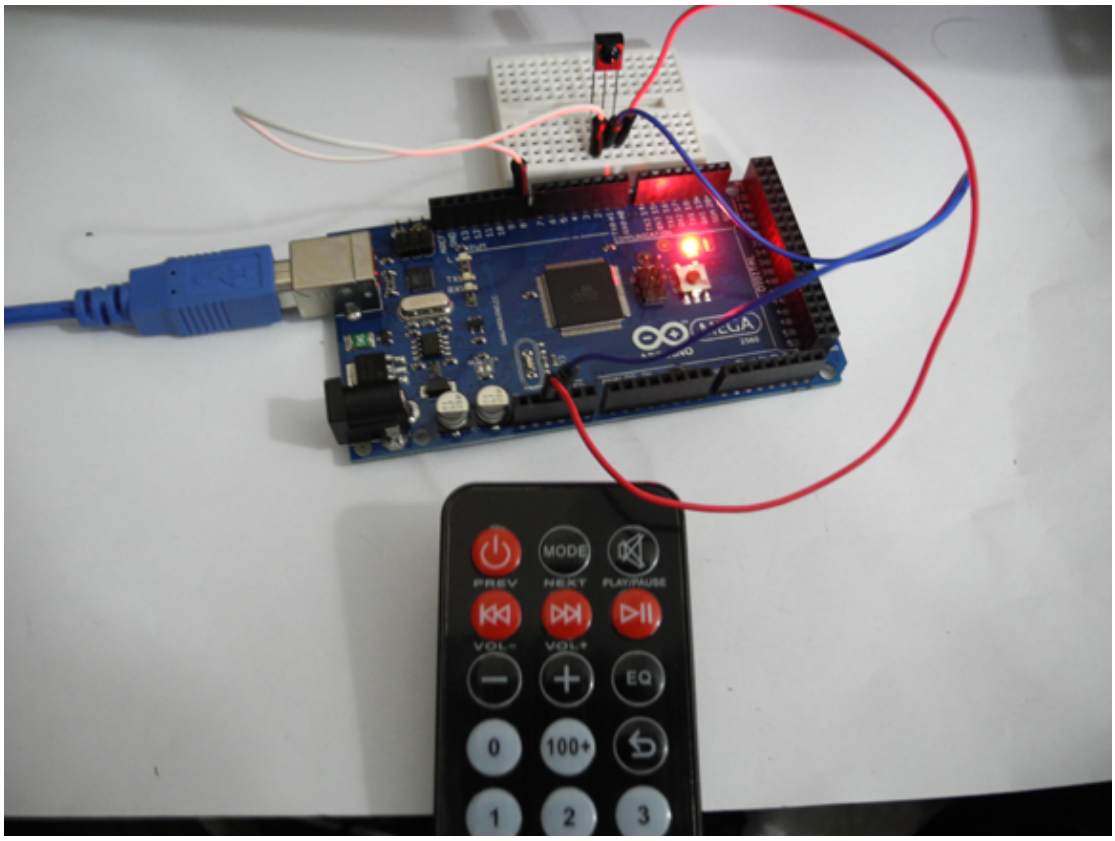
Infrared Receiver (IRM_3638)

IR Remote Control

10 kΩ Resistors

Connect it up





Enter the code

```
/*
Created By: Samleong
Created On: 2011-9-17
Website / More Infomation: http://www.b2cqshop.com
Email: b2cqshop@gmail.com
*/
#define IR_IN   8   //Infrared receiver Pin

int Pulse_Width=0;//Storage width
int adr_code=0x00;// User-coded values
char comL_code=0x00;//Command code
char comH_code=0x00;//Anti-code command

void timer1_init(void)//Timer initialization function
{
  TCCR1A = 0X00;
  TCCR1B = 0X05;//To the timer clock source
  TCCR1C = 0X00;
  TCNT1 = 0X00;
  TIMSK1 = 0X00;   //Disable timer overflow interrupt
```

```
}
void remote_deal(void)//The results of the implementation of decoding function
{
        //Show Data
        if(adr_code!=0XFF)
        {
          Serial.print("the Address Code is : ");
          Serial.println(adr_code, HEX);//Hexadecimal display
          Serial.print("the Command code is : ");
          Serial.println(comL_code, HEX);//Hexadecimal display
        }
}
char logic_value()//Determine the logic value "0" and "1" Functions
{
  TCNT1 = 0X00;
  while(!(digitalRead(IR_IN))); //Low wait
  Pulse_Width=TCNT1;
  TCNT1=0;
  if(Pulse_Width>=7&&Pulse_Width<=10)//Low 560us
  {
    while(digitalRead(IR_IN));//Is waiting for another job
    Pulse_Width=TCNT1;
    TCNT1=0;
    if(Pulse_Width>=7&&Pulse_Width<=10)//Then high 560us
      return 0;
    else if(Pulse_Width>=25&&Pulse_Width<=27) //Then high 1.7ms
      return 1;
  }
  return -1;
}
void pulse_deal()//Receiver address code and command code pulse function
{
  int i;
  int j;
  adr_code=0x00;// Clear
  comL_code=0x00;// Clear
  comH_code=0x00;// Clear

  //Parsing remote code value in the user code
  for(i = 0 ; i < 16; i++)
  {
    if(logic_value() == 1) //if 1
        adr_code |= (1<<i);//Save value
  }
```

```
  //Parsing code in the remote control command codes
  for(i = 0 ; i < 8; i++)
  {
    if(logic_value() == 1) //if 1
      comL_code |= (1<<i);//Save value
  }
  //Parsing code in the remote control command codes counter code
  for(j = 0 ; j < 8; j++)
  {
    if(logic_value() == 1) //if 1
        comH_code |= (1<<j);//Save value
  }
}
void remote_decode(void)//Decoding function
{
  TCNT1=0X00;
  while(digitalRead(IR_IN))//if high then waiting
  {
    if(TCNT1>=1563)  //When the high lasts longer than 100ms, that no button is pressed at this time
    {
      adr_code=0x00ff;// User-coded values
      comL_code=0x00;//Key code value of the previous byte
      comH_code=0x00;//After a byte key code value
      return;
    }
  }

  //If the high does not last more than 100ms
  TCNT1=0X00;

  while(!(digitalRead(IR_IN))); //Low wait
  Pulse_Width=TCNT1;
  TCNT1=0;
  if(Pulse_Width>=140&&Pulse_Width<=142)//9ms
  {

    while(digitalRead(IR_IN));//high wait
    Pulse_Width=TCNT1;
    TCNT1=0;
    if(Pulse_Width>=68&&Pulse_Width<=72)//4.5ms
    {
      pulse_deal();
      return;
    }
```

```
    else if(Pulse_Width>=34&&Pulse_Width<=36)//2.25ms
    {
      while(!(digitalRead(IR_IN)));//low wait
      Pulse_Width=TCNT1;
      TCNT1=0;
      if(Pulse_Width>=7&&Pulse_Width<=10)//560us
      {
        return;
      }
    }
  }
}
void setup()
{
  unsigned char i;
  pinMode(IR_IN,INPUT);//Set the infrared receiver input pin
  // start serial port at 9600 bps:
  Serial.begin(9600);
}
void loop()
{
  timer1_init();//Timer initialization
  while(1)
  {
    remote_decode();    //Decoding
    remote_deal();      //Perform decoding results
  }
}
```

Press the key of the remote controller, user the **Serial Monitor** tool , the receiver will receive the command code .

Project 8 - Hardware Overview

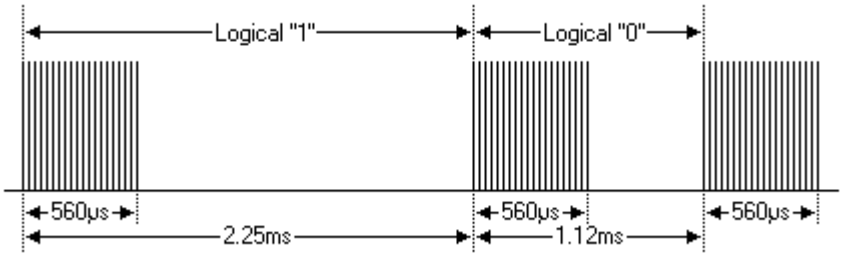
The picture is the IRM_3638 **Infrared Receiver .** The Pin of it is showing in the picture, when you wire this project, please check it carefully .

About the NEC protocol

NEC Protocol Features : (http://www.sbprojects.com/knowledge/ir/nec.htm)

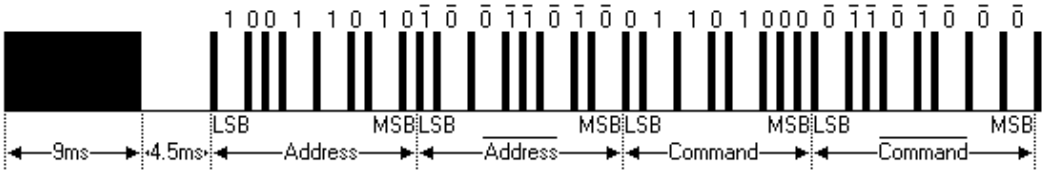
- 8 bit address and 8 bit command length
- Address and command are transmitted twice for reliability
- Pulse distance modulation
- Carrier frequency of 38kHz
- Bit time of 1.125ms or 2.25ms

Modulation:

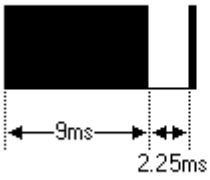


The NEC protocol uses pulse distance encoding of the bits. Each pulse is a 560µs long 38kHz carrier burst (about 21 cycles). A logical "1" takes 2.25ms to transmit, while a logical "0" is only half of that, being 1.125ms. The recommended carrier duty-cycle is 1/4 or 1/3.

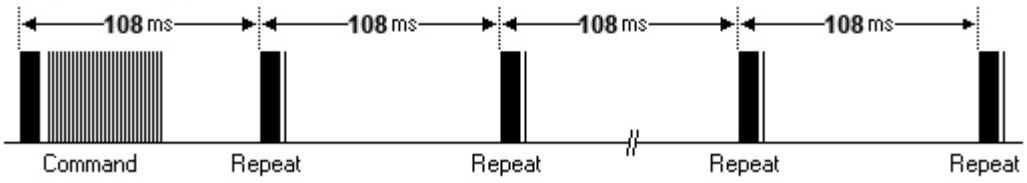
Protocol :



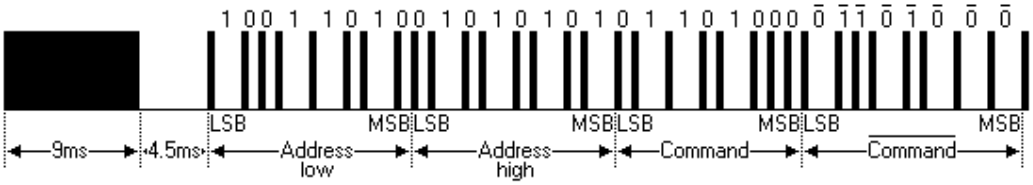
The picture above shows a pulse train of the NEC protocol. With this protocol the LSB is transmitted first. In this case Address $59 and Command $16 is transmitted. A message is started by a 9ms AGC burst, which was used to set the gain of the earlier IR receivers. This AGC burst is then followed by a 4.5ms space, which is then followed by the Address and Command. Address and Command are transmitted twice. The second time all bits are inverted and can be used for verification of the received message. The total transmission time is constant because every bit is repeated with its inverted length. If you're not interested in this reliability you can ignore the inverted values, or you can expand the Address and Command to 16 bits each!



A command is transmitted only once, even when the key on the remote control remains pressed. Every 110ms a repeat code is transmitted for as long as the key remains down. This repeat code is simply a 9ms AGC pulse followed by a 2.25ms space and a 560µs burst.



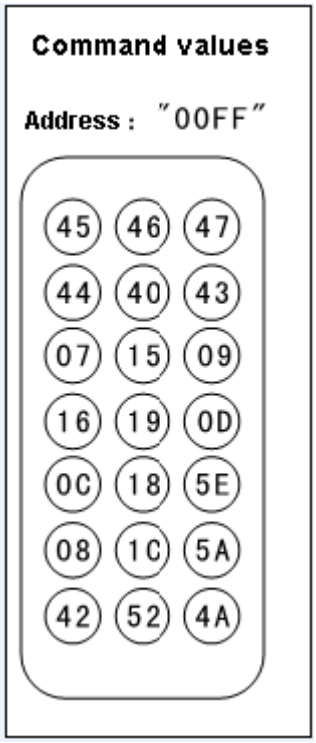
In this Experiment, we will use a little different IR Remote Control , which use WD6122 chip . (Extended NEC protocol)



Pay attention: **When there is no infrared signals, the receiver's output is High ; while, it receives an signal, its output is LOW . We can check the received pulse through the Oscilloscope, analyse the program according to the waveform.**

About the IR Remote controller.

The Command Values of the IR Remote Control's key like the following picture.



Project 9 - 8x8 LED Display

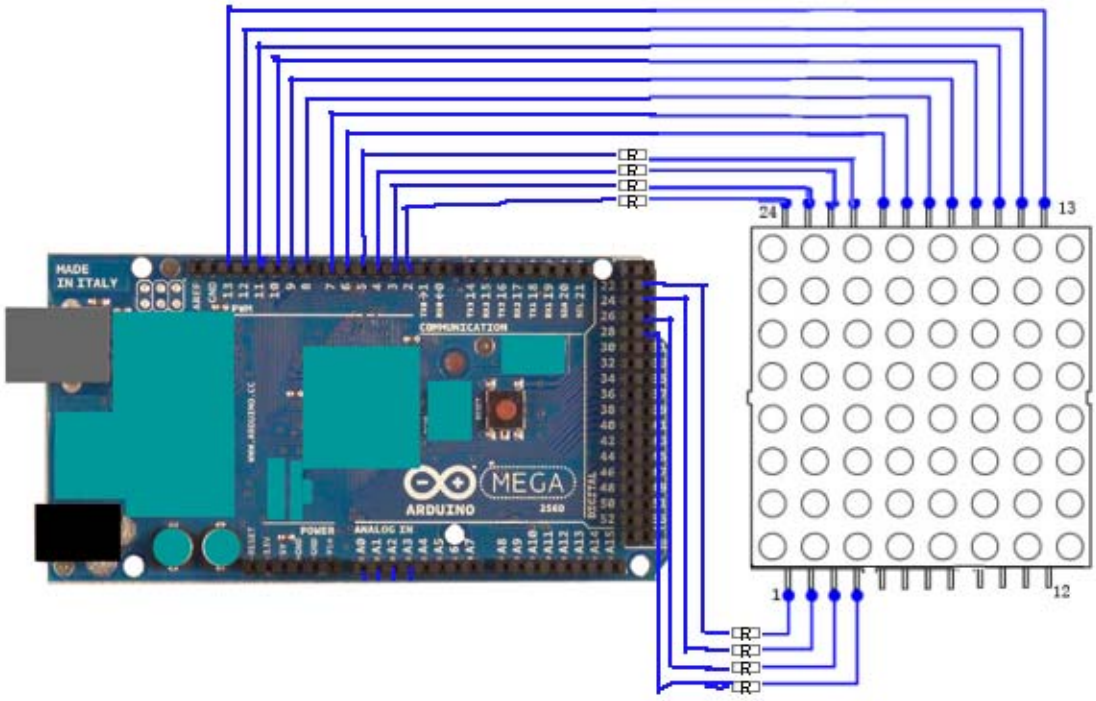
In this project we are going to control the LED Matrix 8x8 Dot LED Display .

What you will need

8x8 Dot LED Display

220 ohm resistor

Connect it up



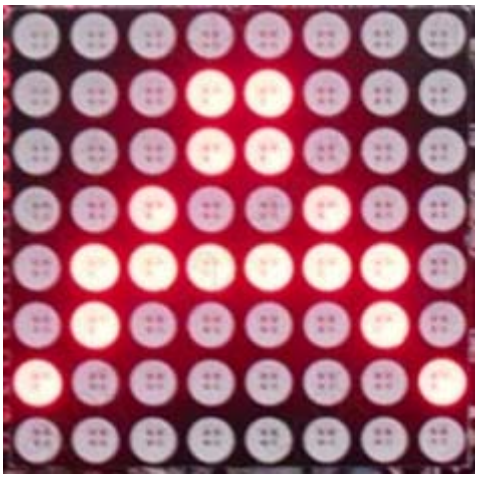
Enter the Code :

```
//www.b2cqshop.com

// display array size
#define display_array_size 8
// ascii 8x8 dot font
#define data_null 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 // null char
#define data_ascii_A   0x02,0x0C,0x18,0x68,0x68,0x18,0x0C,0x02   /*"A",0*/
#define data_ascii_B   0x00,0x7E,0x52,0x52,0x52,0x52,0x2C,0x00   /*"B",1*/
#define data_ascii_C   0x00,0x3C,0x66,0x42,0x42,0x42,0x2C,0x00   /*"C",2*/
#define data_ascii_D   0x00,0x7E,0x42,0x42,0x42,0x66,0x3C,0x00   /*"D",3*/
#define data_ascii_E   0x00,0x7E,0x52,0x52,0x52,0x52,0x52,0x42   /*"E",4*/
#define data_ascii_F   0x00,0x7E,0x50,0x50,0x50,0x50,0x50,0x40   /*"F",5*/
```

```
#define data_ascii_G   0x00,0x3C,0x66,0x42,0x42,0x52,0x16,0x1E   /*"G",6*/
#define data_ascii_H   0x00,0x7E,0x10,0x10,0x10,0x10,0x7E,0x00   /*"H",7*/
#define data_ascii_I   0x00,0x00,0x00,0x7E,0x00,0x00,0x00,0x00   /*"I",8*/
// display array
byte data_ascii[][display_array_size] = {
                                                    data_null,
                                                    data_ascii_A,
                                                    data_ascii_B,
                                                    data_ascii_C,
                                                    data_ascii_D,
                                                    data_ascii_E,
                                                    data_ascii_F,
                                                    data_ascii_G,
                                                    data_ascii_H,
                                                    data_ascii_I,
                                                    };

     //the pin to control ROW
   const int row1 =   2;          // the number of the row pin 24
   const int row2 =   3;          // the number of the row pin 23
   const int row3 =   4;          // the number of the row pin 22
   const int row4 =   5;          // the number of the row pin 21
   const int row5 =   29;           // the number of the row pin 4
   const int row6 =   27;           // the number of the row pin 3
   const int row7 =   25;           // the number of the row pin 2
   const int row8 = 23;           // the number of the row pin 1
 //the pin to control COl
   const int col1 =   6;        // the number of the col pin 20
   const int col2 =   7;        // the number of the col pin 19
   const int col3 =   8;        // the number of the col pin 18
   const int col4 =   9;       // the number of the col pin 17
   const int col5 =   10;       // the number of the col pin 16
   const int col6 =   11;       // the number of the    col    pin 15
   const int col7 =   12;       // the number of the    col    pin 14
   const int col8 = 13;          // the number of the    col    pin 13

void displayNum(byte rowNum,int colNum)
{
   int j;
   byte temp = rowNum;
   for(j=2;j<6;j++)
      {
      digitalWrite(j, LOW);
      }
```

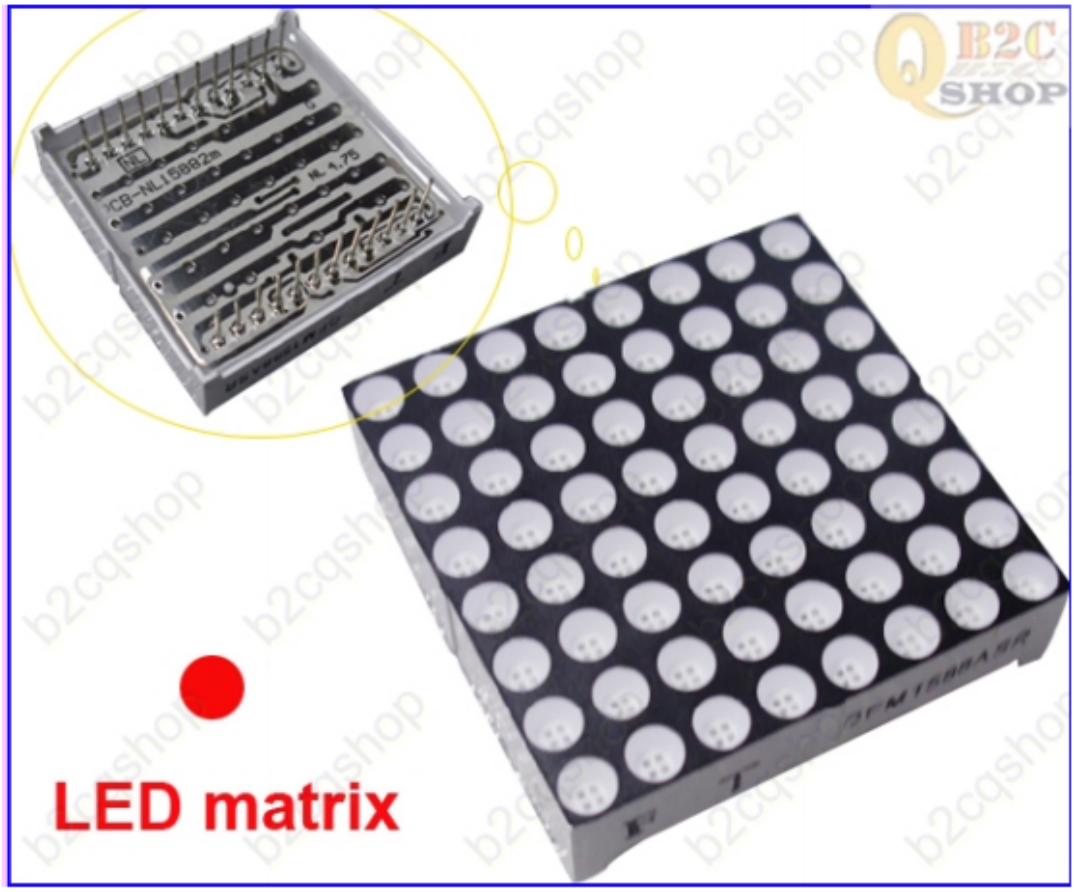
```
    digitalWrite(row5, LOW);
    digitalWrite(row6, LOW);
    digitalWrite(row7, LOW);
    digitalWrite(row8, LOW);

  for(j=6;j<14;j++)
    {
    digitalWrite(j, HIGH);            }
    switch(colNum)
    {
      case 1: digitalWrite(col1, LOW); break
      case 2: digitalWrite(col2, LOW); break;
      case 3: digitalWrite(col3, LOW); break;
      case 4: digitalWrite(col4, LOW); break;
      case 5: digitalWrite(col5, LOW); break;
      case 6: digitalWrite(col6, LOW); break;
      case 7: digitalWrite(col7, LOW); break;
      case 8: digitalWrite(col8, LOW); break;
    default: break;
    }
 for(j = 1 ;j < 9; j++)
    {
      temp = (0x80)&(temp) ;
      if(temp==0)
            {
                  temp =      rowNum<<j;
                  continue;
            }
            switch(j)
                {
          case 1: digitalWrite(row1, HIGH); break;
          case 2: digitalWrite(row2, HIGH); break;
          case 3: digitalWrite(row3, HIGH); break;
          case 4: digitalWrite(row4, HIGH); break;
          case 5: digitalWrite(row5, HIGH); break;
          case 6: digitalWrite(row6, HIGH); break;
          case 7: digitalWrite(row7, HIGH); break;
          case 8: digitalWrite(row8, HIGH); break;
          default: break;
                }
            temp =      rowNum<<j;
    }
}
```

```
void setup(){
   int i = 0 ;
   for(i=2;i<14;i++)
      {
      pinMode(i, OUTPUT);
      }
      pinMode(row5, OUTPUT);
      pinMode(row6, OUTPUT);
      pinMode(row7, OUTPUT);
      pinMode(row8, OUTPUT);
   for(i=2;i<14;i++)        {
      digitalWrite(i, LOW);
      }
      digitalWrite(row5, LOW);
      digitalWrite(row6, LOW);
      digitalWrite(row7, LOW);
      digitalWrite(row8, LOW);
}
void loop(){
   int t1;
   int l;
   int arrage;
   for(arrage=0;arrage<10;arrage++)

      {
         for(l=0;l<512;l++)
                {
                     for(t1=0;t1<8;t1++)
                     {
                              displayNum(data_ascii[arrage][t1],(t1+1));
                     }
                }

      }
}
```



Project 9 - Hardware Overview

----- Common Anode 24 Pin



Project 10 – SD card module Control

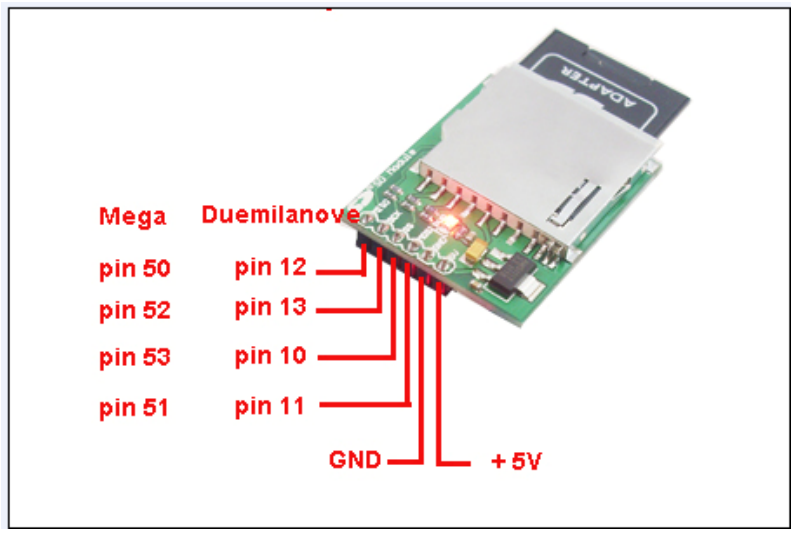
In this project , we will use the **SD card library** of the arduino-0022 Version . It will shows how to read and write data to and from an SD card .

What you will need

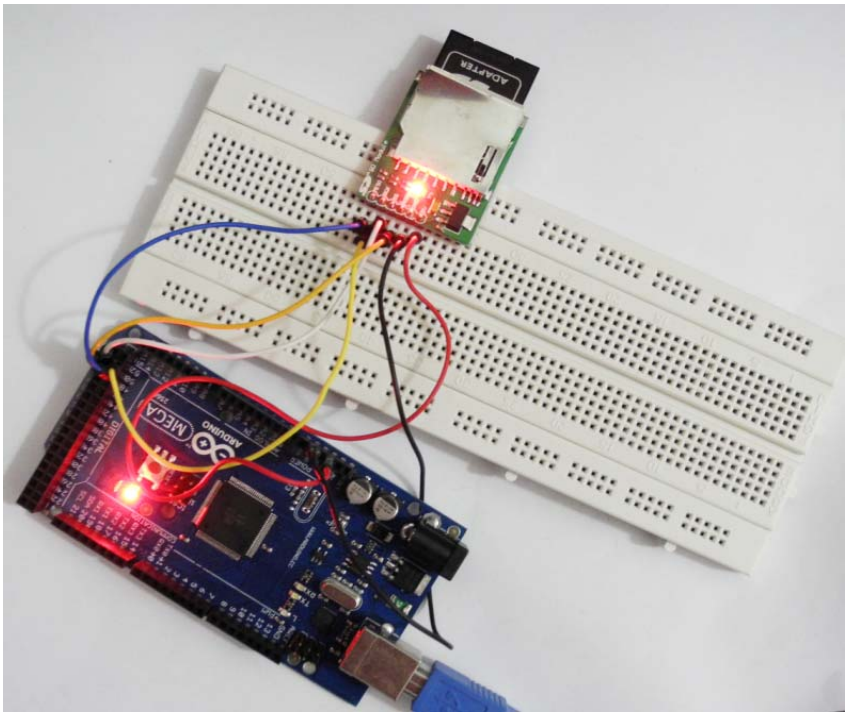
SD card module

Breadboard

Connect it up



For the Mega 2560, pin 50 to the MISO , pin 52 to the SCK , pin 53 to the SS , pin 51 to the MOSI .



Before you use do this project, it must format the SD card, and the SD-Card must be formatted with FAT-16 Filesystem with only a single partition on it.

unzip the file "**SdFatV2Demo20110108.rar**" , and copy all of the file "**SdFat**" to your arduino software file " **arduino-0022\libraries** "

Enter the code

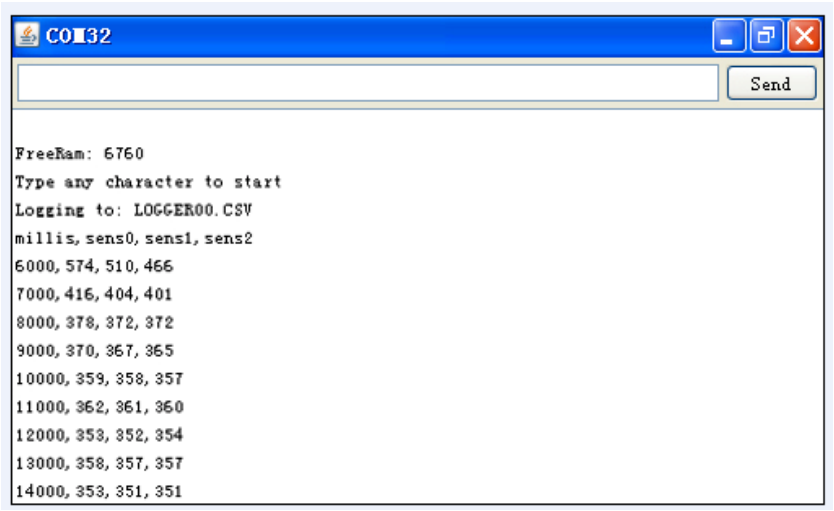
```
// A simple data logger for the Arduino analog pins
#define LOG_INTERVAL  1000  // mills between entries
#define SENSOR_COUNT  3  // number of analog pins to log
#define ECHO_TO_SERIAL  1  // echo data to serial port
#define WAIT_TO_START  1  // Wait for serial input in setup()
#define SYNC_INTERVAL 1000  // mills between calls to sync()
uint32_t syncTime = 0;  // time of last sync()

#include <SdFat.h>
#include <SdFatUtil.h>

// file system object
SdFat sd;

// binary file object
SdFile file;

// Serial print stream
ArduinoOutStream cout(Serial);
```

```
// buffer to format data - makes it eaiser to echo to Serial
char buf[50];
//------------------------------------------------------------------------------
#if SENSOR_COUNT > 6
#error SENSOR_COUNT too large
#endif  // SENSOR_COUNT
//------------------------------------------------------------------------------
// store error strings in flash to save RAM
#define error(s) sd.errorHalt_P(PSTR(s))
//------------------------------------------------------------------------------
void setup() {
  Serial.begin(9600);

  // pstr stores strings in flash to save RAM
  cout << endl << pstr("FreeRam: ") << FreeRam() << endl;

#if WAIT_TO_START
  cout << pstr("Type any character to start\n");
  while (!Serial.available());
#endif  // WAIT_TO_START

  // initialize the SD card at SPI_HALF_SPEED to avoid bus errors with
  // breadboards.  use SPI_FULL_SPEED for better performance.
  // if SD chip select is not SS, the second argument to init is CS pin number
  if (!sd.init(SPI_HALF_SPEED)) sd.initErrorHalt();

  // create a new file in root, the current working directory
  char name[] = "LOGGER00.CSV";
  for (uint8_t i = 0; i < 100; i++) {
    name[6] = i/10 + '0';
    name[7] = i%10 + '0';
    if (file.open(name, O_CREAT | O_EXCL | O_WRITE)) break;
  }
  if (!file.isOpen()) error("file.open");

  cout << pstr("Logging to: ") << name << endl;

  //format header in buffer
  obufstream bout(buf, sizeof(buf));

  bout << pstr("millis");
  for (uint8_t i = 0; i < SENSOR_COUNT; i++) {
    bout << ",sens" << int(i);
```

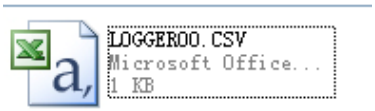
```
    }

#if ECHO_TO_SERIAL
    cout << buf << endl;
#endif  // ECHO_TO_SERIAL

    // add CR/LF for Windows style file (file.write is binary, not text)
    bout << "\r\n";
    file.write(buf);

    // force write of header to SD card
    file.sync();

    if (file.writeError) error("write header failed");
}
//------------------------------------------------------------------------------
void loop() {
    uint32_t m;

    // wait for time to be a multiple of interval
    do {
        m = millis();
    } while(m % LOG_INTERVAL);

    // use buffer stream to format line
    obufstream bout(buf, sizeof(buf));

    // start with time in millis
    bout << m;
    // read analog pins and format data
    for (uint8_t ia = 0; ia < SENSOR_COUNT; ia++) {
        bout << ',' << analogRead(ia);
    }
#if ECHO_TO_SERIAL
    cout << buf << endl;
#endif  // ECHO_TO_SERIAL

    // add CR/LF for Windows style file (file.write is binary, not text)
    bout << "\r\n";
    file.write(buf);

    // don't sync too often - requires 2048 bytes of I/O to SD card
    if ((millis() - syncTime) >= SYNC_INTERVAL) {
```

```
      file.sync();
      syncTime = millis();
    }
    // check for errors
    if (file.writeError) error("write data failed");

    // don't log two points in the same millis
    if (m == millis()) delay(1);
  }
```

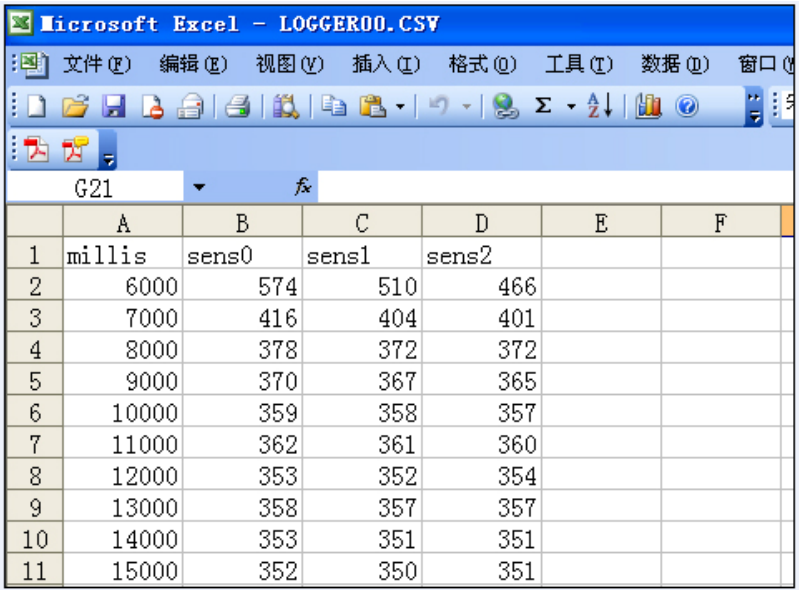
Type any character to the **Serial Monitor** tool ,Then the Mega 2560 will write some data in the sd card . And when you Use the sd reader to read the sd card it will show like the following pictures.



---->>>>>>



----->>>>



Microsoft Excel - LOGGER00.CSV

文件(F) 编辑(E) 视图(V) 插入(I) 格式(O) 工具(T) 数据(D) 窗口(W)

G21

	A	B	C	D	E	F
1	millis	sens0	sens1	sens2		
2	6000	574	510	466		
3	7000	416	404	401		
4	8000	378	372	372		
5	9000	370	367	365		
6	10000	359	358	357		
7	11000	362	361	360		
8	12000	353	352	354		
9	13000	358	357	357		
10	14000	353	351	351		
11	15000	352	350	351		

Project 10 - Hardware Overview

To know more about the The Arduino SdFat Library , Please read the SdFatV2Demo20110108.rar (which is in this project file)

Project 11 – Show in 128*64 LCD

In this project, we will learn how to control the 128*64 LCD .

First , unzip the file "**glcd-20110423.zip**" , and copy all of the file "glcd" to your arduino software file " **arduino-0022\libraries** "

Second, wire as the following picture, to connect your arduino mega to the 128*64 LCD .

Third , upload the simple Sketches to your mega board .

What you will need

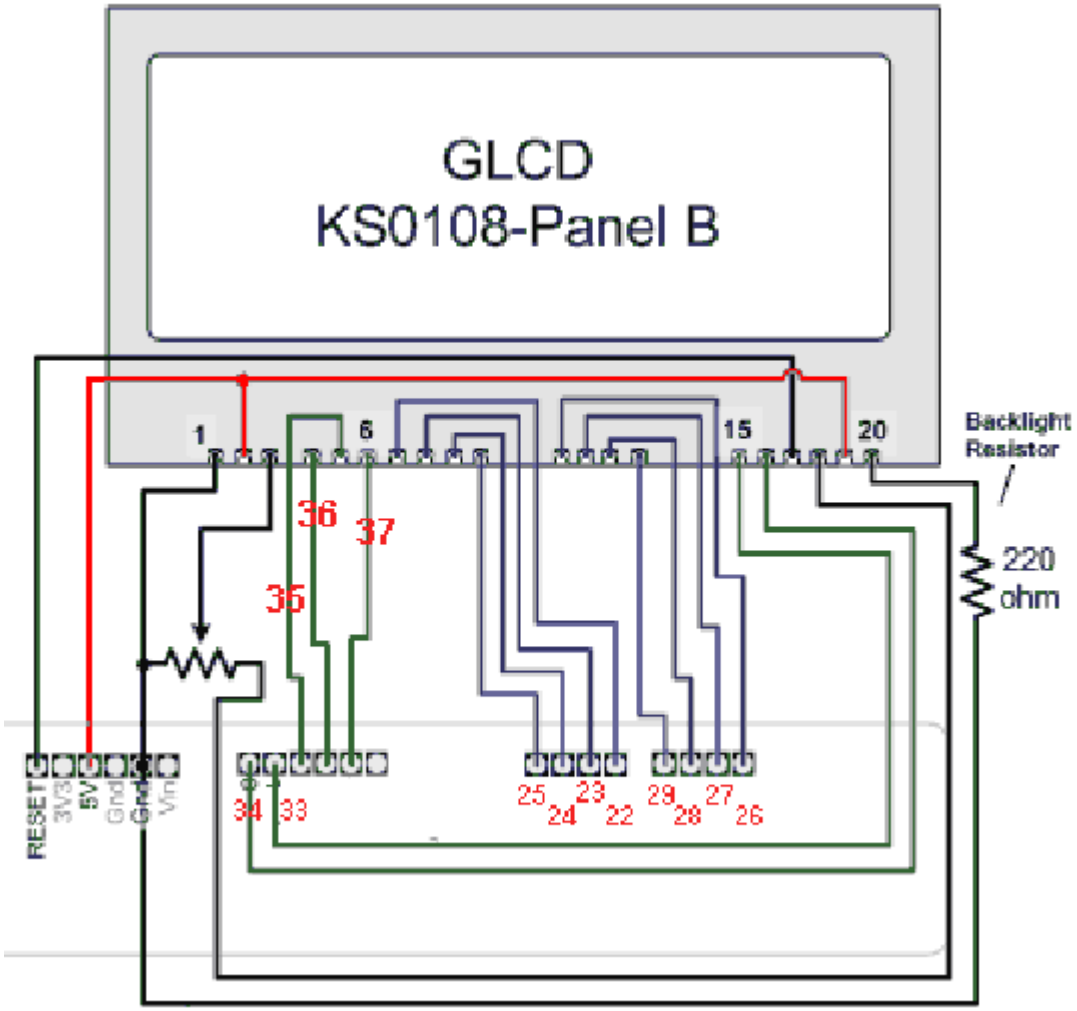
128*64 LCD Screen

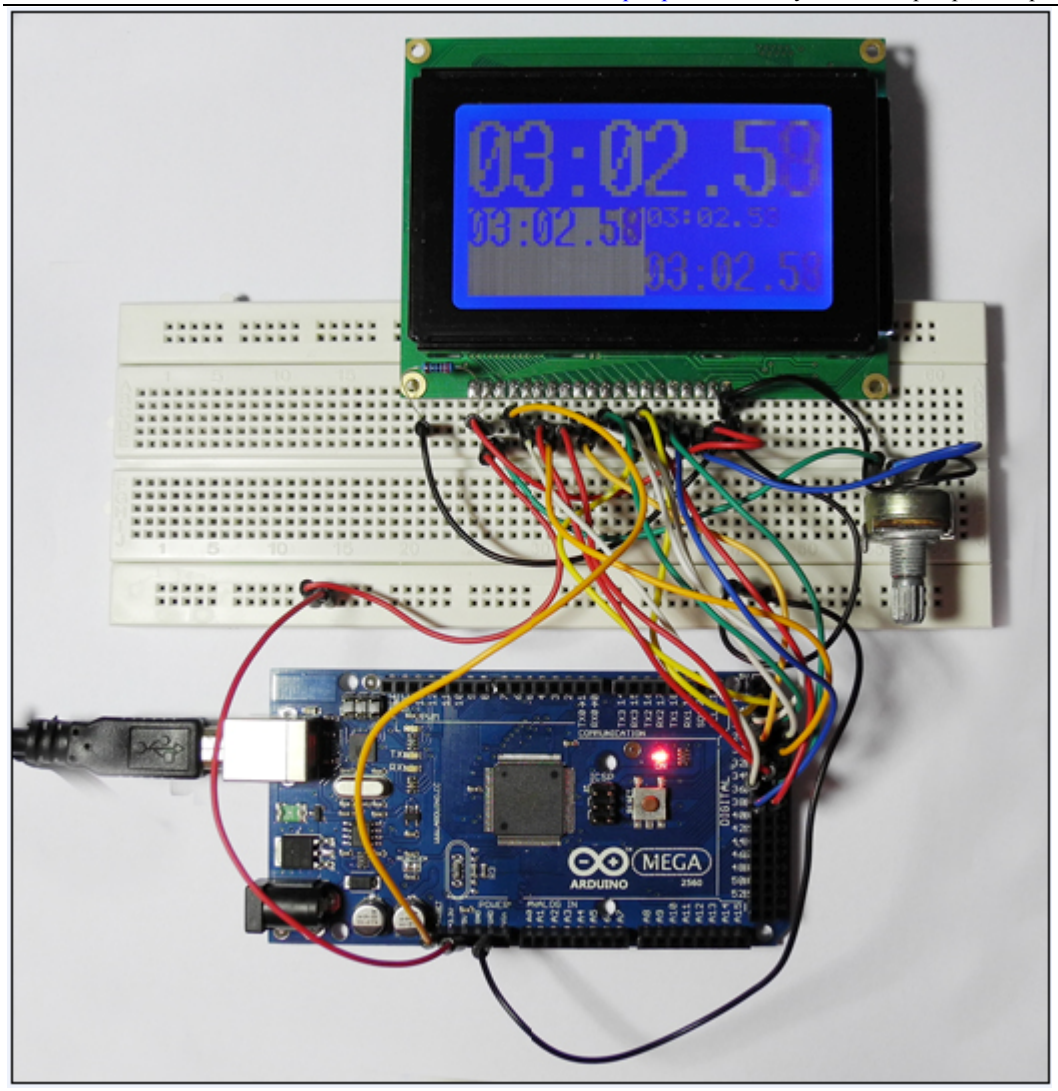
10k Potentiometer

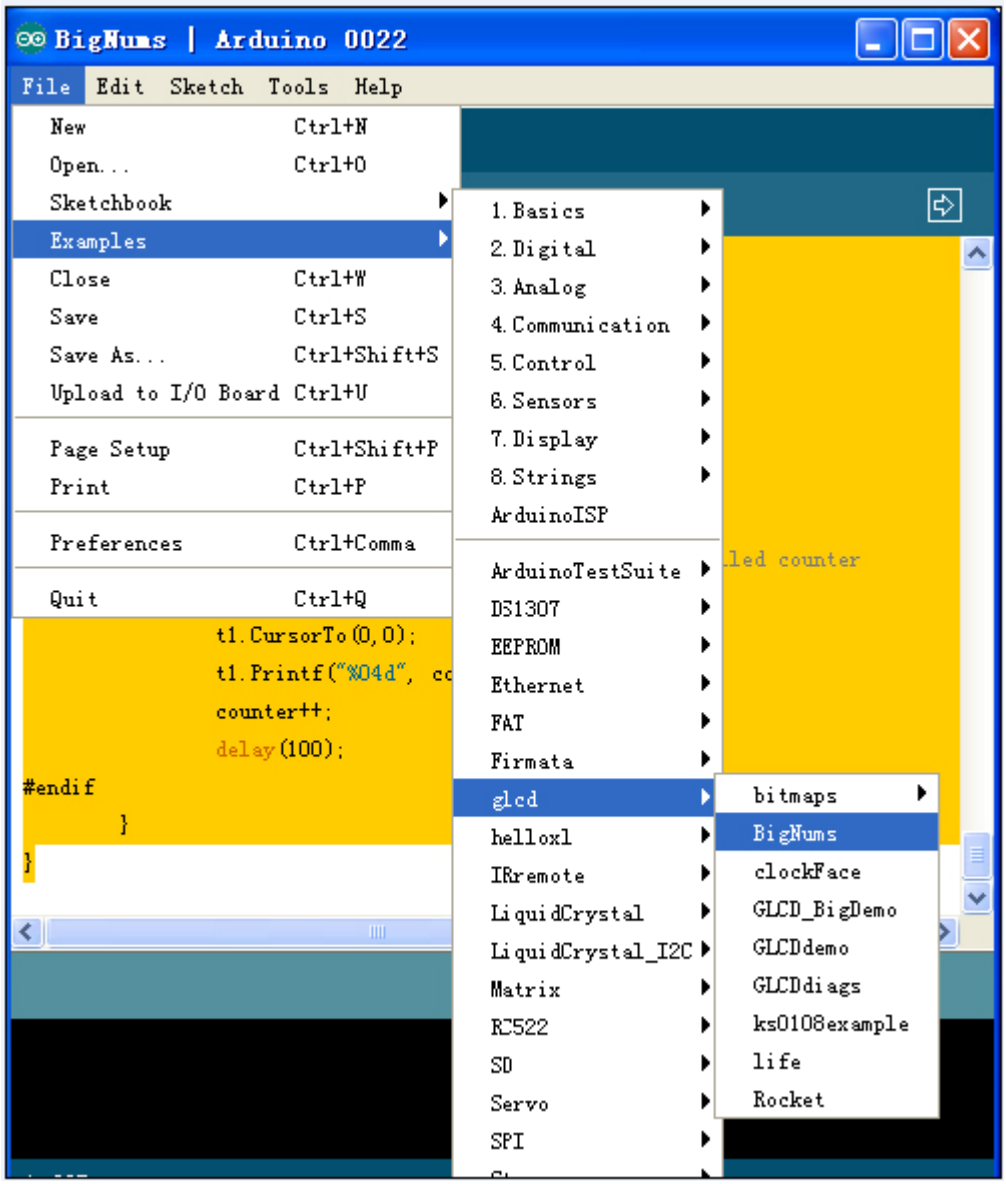
220 ohm Resistor

Connect it up

GLCD Function	Panel B	Mega Pins
+5 volts	2	+5V
GND	1	Gnd
VO (Contrast in)	3	To the middle of the Potentiometer
D0	7	22
D1	8	23
D2	9	24
D3	10	25
D4	11	26
D5	12	27
D6	13	28
D7	14	29
CSEL1	15	33
CSEL2	16	34
Reset	17	Reset
R_W	5	35
D_I	4	36
EN	6	37
VOUT（Contrast out）	18	10k or 20k preset
Backlight +5	19	+5V
Backlight Gnd	20	Gnd


GLCD
KS0108-Panel B
1
6
15
20
Backlight
Resistor
220
ohm
36
37
35
RESET
3V3
5V
Gnd
Gnd
Vm
34
33
25
24
23
22
29
28
27
26


03:02.59
03:02.59
03:02.59
03:02.59
MEGA
ARDUINO



Enter the code

```
/*
 * BigNums
 *
 * This sketch demonstrates how to use text areas, some of the large number
 * fonts, and the Printf() formatting function.
 *
 * Note: Printf() uses about 1.8k of code space. It can be expensive
 * if simple strings are being printed relative to using other simpler/smaller
 * functions such as Puts() or the Arduino Print class.
 * However, when doing doing lots of formatting or using hex number output the printf
```

```
 * function can be smaller.
 *
 * To save code space, try not to use both the arduino print class and
 * Printf().
 *
 * To save RAM space,
 * keep in mind that if using Printf() the formatting string will consum RAM space.
 * (Normal strings get copied to RAM so you can modify them)
 * Same is true when using Puts()
 * To avoid this, you can delcare your strings to only be in AVR progrmm memory by using
the PSTR macro.
 * but then you have to also use the Print_P() or Puts_P() functions instead.
 * Example:
 *
 * GLCD.Printf_P(PSTR("format string"), arg1, arg2, ...);
 * It can also be used for simple strings too:
 * GLCD.Printf_P(PSTR("Hello world"));
 * GLCD.Puts_P(PSTR("Hello world"));
 *
 * The fixnums fonts are not complete fonts. The font headers only contain
 * the characters: '+', ',', '-', '.', '/', '0', '1', '2', '3', '4', '5', '6', '7', '9'
 * There are no other characters other than those listed above, so if you try to draw
 * something else like say an 'V', 'F', 'C', or anything not listed above,
 * it will not draw anything for that character.
 *
 * This is intentional to create the possiblity of having large numbers but not eat
 * up lots of flash space. This should be enough to do things like a clock or numbers
 * for things like counters, displaying voltage or temperature, etc...
 *
 * 2011-04-23 Created by Bill Perry
 */

#include "glcd.h"

/*
 * individual fonts don't have to be included
 * if you include the allFonts.h header
 */
//#include "fonts/fixednums15x31.h"
//#include "fonts/fixednums8x16.h"
//#include "fonts/fixednums7x15.h"
//#include "fonts/SystemFont5x7.h"
```

```
#include "fonts/allFonts.h"

void setup() {
      GLCD.Init(NON_INVERTED);
}

/*
 * Define some text areas
 * Text areas can be defined when declared or can be defined
 * runtime.
 *
 * Note:
 * The GLCD object contains a built in text area that is pre-defined
 * to be then entire glcd display.
 * All the same printing functions exist for the GLCD object and user
 * define text areas.
 */

gText t1; // will define runtime later

#if DISPLAY_HEIGHT > 32
gText t2; // will define runtime later

/*
* define t3 to be half way down starting in the middle (plus a pixel) and
* 8 columns by 1 row tall using the system font.
*/
gText t3 = gText(GLCD.CenterX+1, GLCD.CenterY, 8, 1, SystemFont5x7);

gText t4; // will define runtime later.
#endif

void loop()
{
int hr, min, sec;
int counter = 0;;

      /*
       * Create 4 text areas in different ways
       * for demonstration.
       *
       * Text areas can be declared using DefineArea()
       *    in different ways:
```

```
     * - predefined area
     * - x1, y1, columns, rows
     * - x1,y1 to x2,y2
     *
     */

    /*
     * define t1 area based on a predefined area of the display
     */

    t1.DefineArea(textAreaTOP);
    t1.SelectFont(fixednums15x31);

#if DISPLAY_HEIGHT > 32
    /*
     * define t2 to be half way down on the left and
     * 8 columns by 2 rows tall
     */
    t2.DefineArea(0, GLCD.CenterY, 8, 2, fixednums7x15);

    /*
     * For demonstration, we will set the font color to white
     * and clear the entire text area so it is visible.
     */
    t2.SetFontColor(WHITE); // set font color
    t2.ClearArea(); // Clear entire text area so you can see the full boundaries.


    /*
     * t3    text area was defined when declared.
     */

    /*
     * t4 is a 1 line 8 colum text area that starts in the middle
     * and is 3/4 of the display height down.
     */

    t4.DefineArea(GLCD.CenterX, GLCD.Height * 3/4, 8, 1, fixednums7x15);
#endif

    hr = 0;
    min = 0;
    sec = 0;
```

```
    while(1)
    {

#if DISPLAY_WIDTH > 127
        t1.CursorToXY(0,0); // column & row is relative to text area
        /*
         * use a formatting string that is only in FLASH/(program memory) and *not* in
RAM.
         */
        t1.Printf_P(PSTR("%02d:%02d.%02d"), hr, min, sec);

#if DISPLAY_HEIGHT > 32
        t2.CursorTo(0,0); // column & row is relative to text area
        t2.Printf("%02d:%02d.%02d", hr, min, sec);

        /*
         * because text area 3 is exactly 8x1 characters in size
         * and that is what we print every time,
         * there is no need to position or clear the area
         * The new text will force out the old text.
         */
        t3.Printf("%02d:%02d.%02d", hr, min, sec);

        /*
         * You can also use the Arduino print class stuff as well.
         * If you do a println the library defers the newline until the next character.
         * So if you have a sigle line text area, it is automatically cleared when
         * the next character is printed.
         */

        /*
         * This creates the same formatted output as the printf above but
         * uses the Arduino Print class functions instead.
         */
        if(hr < 9)
            t4.print(0);
        t4.print(hr);
        t4.print(":");
        if(min < 9)
            t4.print(0);
        t4.print(min);
        t4.print(".");
        if(sec < 9)
```

```
            t4.print(0);
        t4.println(sec);
#endif
        sec++; // not really seconds but animates display
        if(sec > 59)
        {
            sec = 0;
            min++;
            if(min>59)
            {
                min = 0;
                hr++;
                if(hr>11)
                    hr = 0;
            }
        }
#else
        /*
         * For small displays, print a simple 0 filled counter
         */
        t1.CursorTo(0,0);
        t1.Printf("%04d", counter);
        counter++;
        delay(100);
#endif
    }
}
```

Project 11 - Hardware Overview

Check it in the file **12864lcd.pdf** which is in the “project_8 “

Please check the file “ **GLCD_Documentation.pdf** ” more about the GLCD library

GLCD Methods:

Here is a summary of the methods supported by this library.

Note that all coordinates start from 0 unless otherwise noted. 0,0 is the pixel on the upper left edge of the screen

Table of contents

Arduino print functions

GLCD Version 3 (Beta) Jun 9 2010

Text Areas

DefineArea()

Enhanced print functions

printFlash ()

printFlashln ()

Printf ()

Printf_P ()

CharWidth()

StringWidth()

StringWidth_P()

Legacy text output functions

PutChar()

Puts()

PrintNumber()

Puts_P()

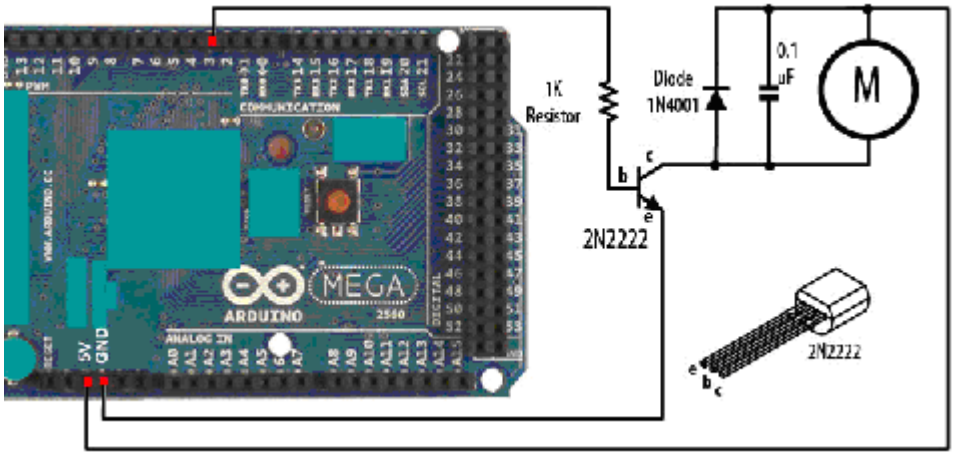
Project 12 - Vibration Motor

In this project we are going to use a vibration motor to vibrate under Arduino control. For example, the following project to shake for one second every minute.

What you will need

1 X 1K Resistor
1 X 1N4001 Diode
1 X 0.1uf Capacitance
1 X 3 - 5V motor
1 X 2N2222 Audion

Connect it up



Code :

```
/*
* Vibrate sketch
* Vibrate for one second every minute
*
*/
const int motorPin = 3; // vibration motor transistor is connected to pin 3
void setup()
{
      pinMode(motorPin, OUTPUT);
}
void loop()
{
   digitalWrite(motorPin, HIGH); //vibrate
   delay(1000); // delay one second
   digitalWrite(motorPin, LOW); // stop vibrating
   delay(59000); // wait 59 seconds.
}
```

Project 12 - Hardware Overview

Vibration motors require more power than an Arduino pin can provide, so a transistor is used to switch the motor current on and off. Almost any NPN transistor can be used; Our project shows the common 2N2222. A 1 kilohm resistor connects the output pin to the transistor base; the value is not critical, and you can use values up to 4.7 kilohm or so (the resistor prevents too much current). The diode absorbs (or snubs—it's sometimes called a snubber diode) voltages produced by the motor windings as it rotates. The capacitor absorbs voltage spikes produced when the brushes (contacts connecting electric current to the motor windings) open and close. And you can add the 33 ohm resistor that connect the +5v to the motor, so that to limit the amount of current flowing through the motor.

This sketch sets the output pin HIGH for one second (1,000 milliseconds) and then waits for 59 seconds. The transistor will turn on (conduct) when the pin is HIGH, allowing current to flow through the motor.

By the way, you can uses a sensor to make the motor vibrate .
Such as (with the addition of a photocell connected to analog pin 0) :

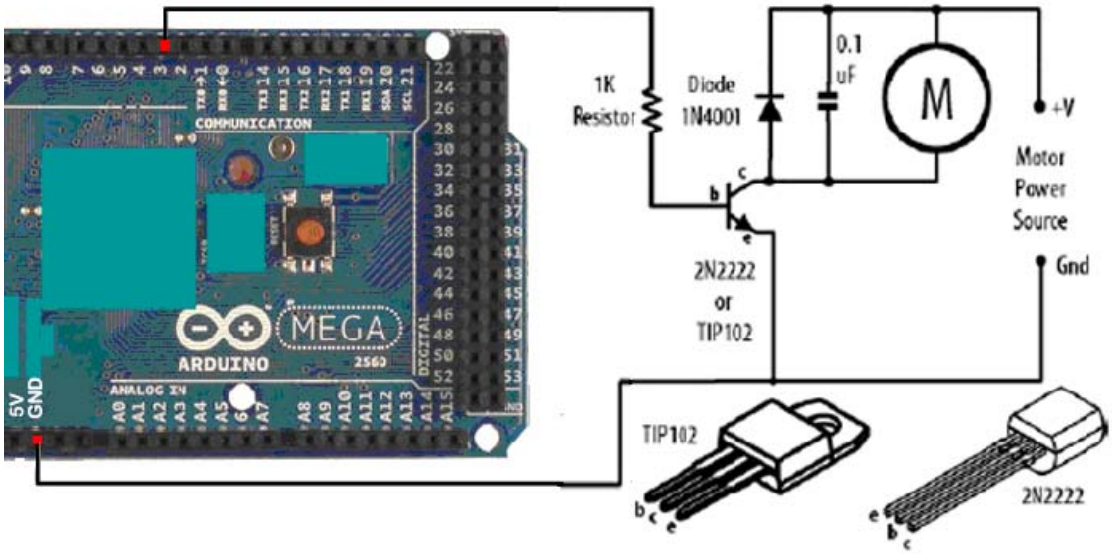
```
/*
* Vibrate_Photocell sketch
* Vibrate when photosensor detects light above ambient level
*
*/
const int motorPin = 3; // vibration motor transistor is connected to pin 3
const int sensorPin = 0; // Photodetector connected to analog input 0
int sensorAmbient = 0; // ambient light level (calibrated in setup)
const int thresholdMargin = 100; // how much above ambient needed to vibrate
void setup()
{
        pinMode(motorPin, OUTPUT);
        sensorAmbient = analogRead(sensorPin); // get startup light level;
}
void loop()
{
      int sensorValue = analogRead(sensorPin);
      if( sensorValue > sensorAmbient + thresholdMargin)
      {
        digitalWrite(motorPin, HIGH); //vibrate
      }
      else
      {
```

```
            digitalWrite(motorPin, LOW); // stop vibrating
        }
    }
```

Here the output pin is turned on when a light shines on the photocell. When the sketch starts, the background light level on the sensor is read and stored in the variable sensorAmbient. Light levels read in loop that are higher than this will turn on the vibration motor.

Addition project – (Driving a Brushed Motor Using a Transistor)

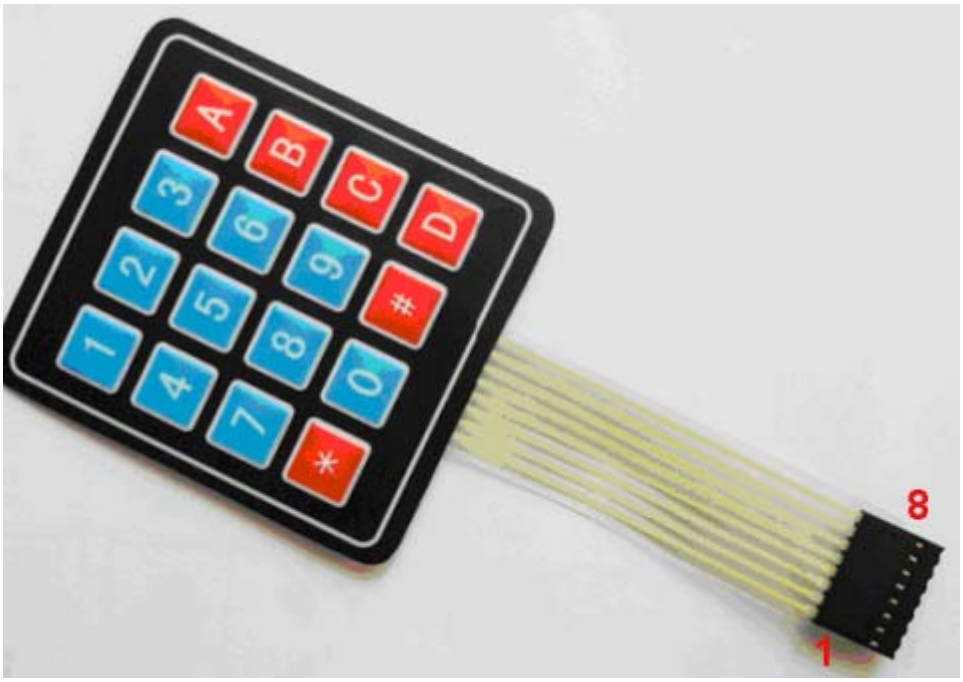
If You want to turn a motor on and off. You may want to control its speed. The motor only needs to turn in one direction.



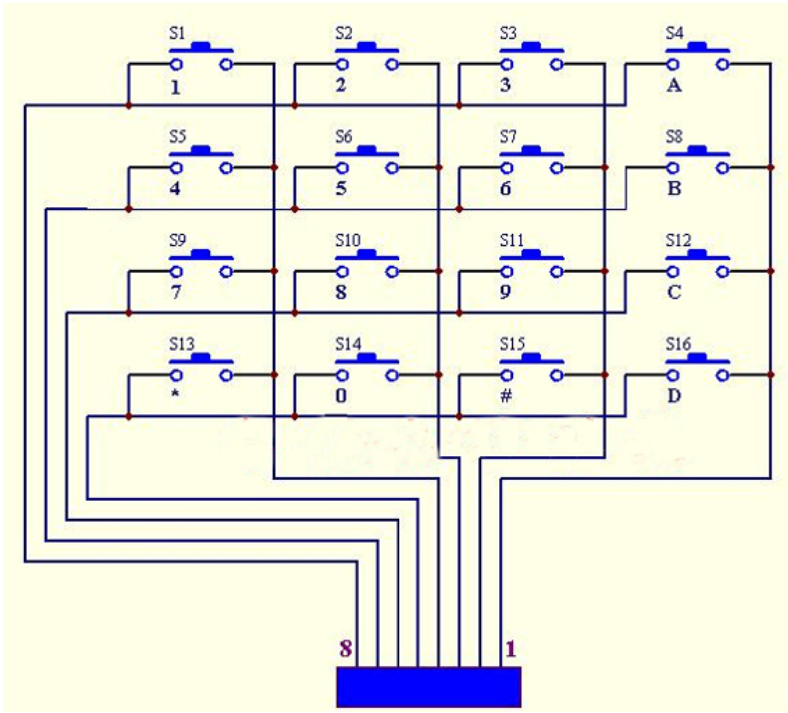
This sketch turns the motor on and off and controls its speed from commands received on the serial port

Project 13 – 4*4 Matrix Keypad

About the 4*4 Matrix Keypad



The Schematic of the keypad

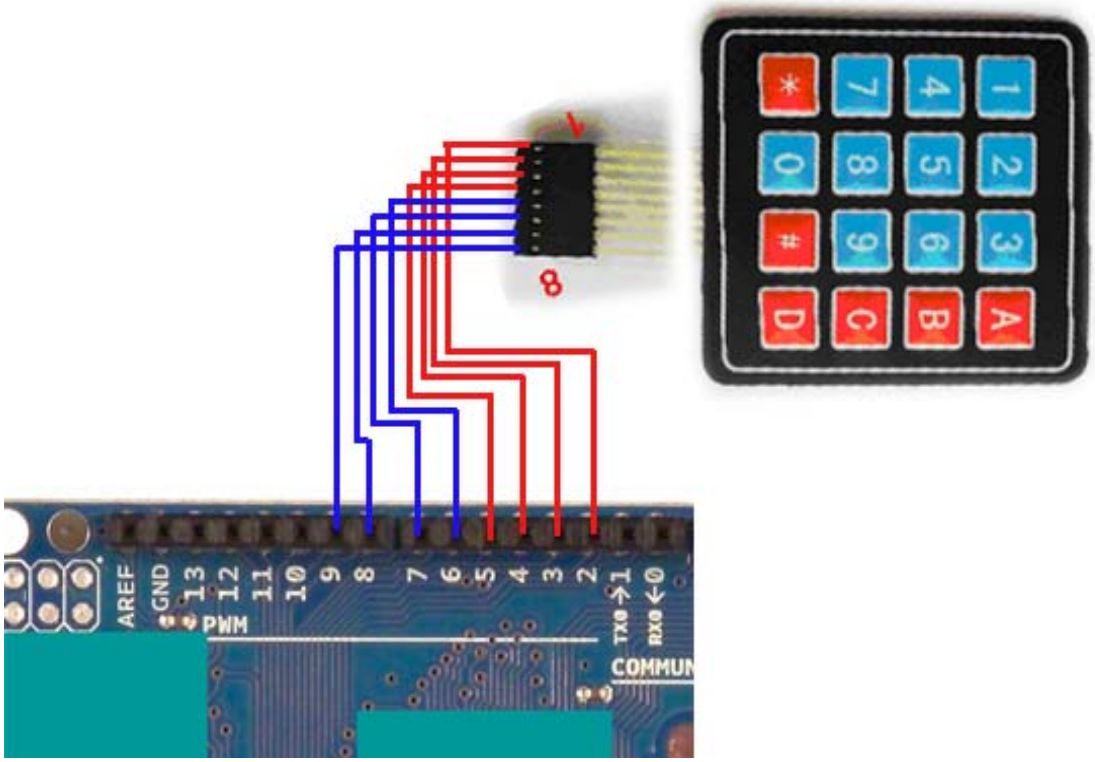


What you will need

1 X 4*4 matrix keypad

Lots of wire jumper.

Connect it up



Code :

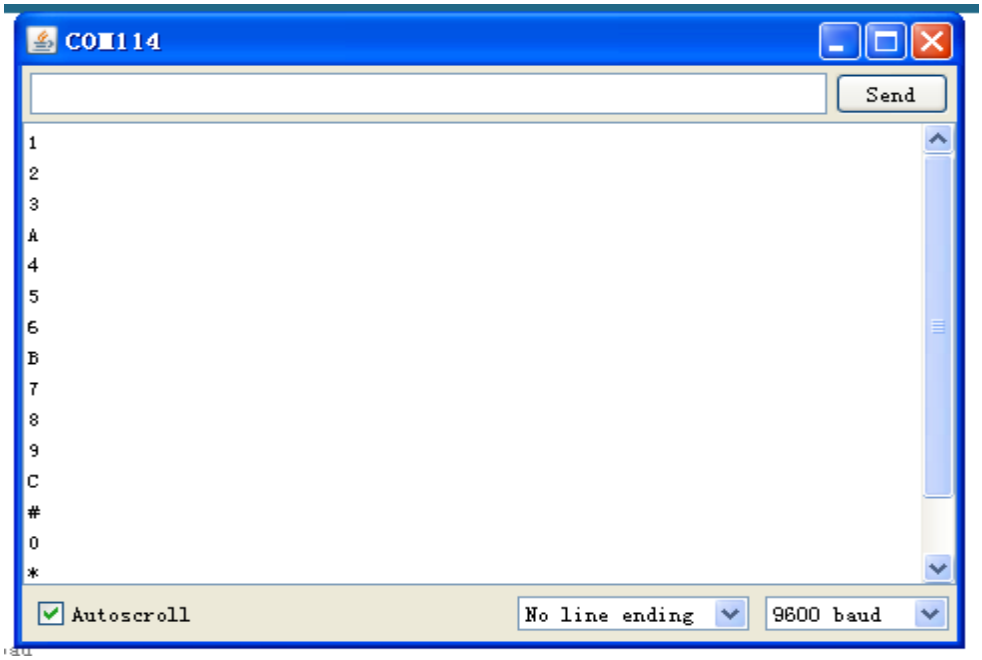
```
/*     @file EventSerialKeypad.pde
   ||   @version 1.0
   ||   @author Alexander Brevig
   ||   @contact alexanderbrevig@gmail.com
   ||
   ||   @description
   ||   | Demonstrates using the KeypadEvent.
   || #
*/
#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
char keys[ROWS][COLS] = {
   {'1','2','3','A'},
```

```
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {2,3,4,5}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {6,7,8,9}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
byte ledPin = 13;

boolean blink = false;

void setup(){
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);          // sets the digital pin as output
  digitalWrite(ledPin, HIGH);     // sets the LED on
  keypad.addEventListener(keypadEvent); //add an event listener for this keypad
}

void loop(){
  char key = keypad.getKey();

  if (key != NO_KEY) {
    Serial.println(key);
  }
  if (blink){
    digitalWrite(ledPin,!digitalRead(ledPin));
    delay(100);
  }
}

//take care of some special events
void keypadEvent(KeypadEvent key){
  switch (keypad.getState()){
    case PRESSED:
      switch (key){
        case '#': digitalWrite(ledPin,!digitalRead(ledPin)); break;
        case '*':
          digitalWrite(ledPin,!digitalRead(ledPin));
        break;
      }
    break;
    case RELEASED:
```

```
        switch (key){
          case '*':
            digitalWrite(ledPin,!digitalRead(ledPin));
            blink = false;
          break;
        }
      break;
      case HOLD:
        switch (key){
          case '*': blink = true; break;
        }
      break;
  }
}
```

The result of this test code

When you press the key “*”, the LED (pin 13) will turn on , and press it again, it will be turn off the LED. And in the serial tool, it will shows .



Project 13 - Code Overview

In this project, you will use the Keypad library . So put the keypad library to your arduino software library .

The Keypad library allows your Arduino to read a matrix type keypad.

About the Keypad Library :

Functions

void begin()

Initialize all variables

The constructor does this for you.

void begin(makeKeymap(userKeymap))

Initializes the internal keymap to be equal to userKeymap

[See Keypad/Examples/CustomKeypad/CustomKeypad.pde for an example]

The overloaded constructor #3 does this for you.

char getKey()

Returns the key that is pressed, if any.

KeypadState getState()

Returns the current state of the keypad.

The four states are IDLE, PRESSED, RELEASED and HOLD.

setHoldTime(unsigned int time)

Set the amount of milliseconds the user will have to hold a button until the HOLD state is triggered.

setDebounceTime(unsigned int time)

Set the amount of milliseconds the keypad will wait until it accepts a new keypress/keyEvent. This is the "time delay" debounce method.

addEventListener(keypadEvent)

Trigger an event if the keypad is used. You can load an example in the Arduino IDE (File -> Sketchbook -> Examples -> Library-Keypad -> EventSerialKeypad) or see the KeypadEvent Example code.

Notes on using the library

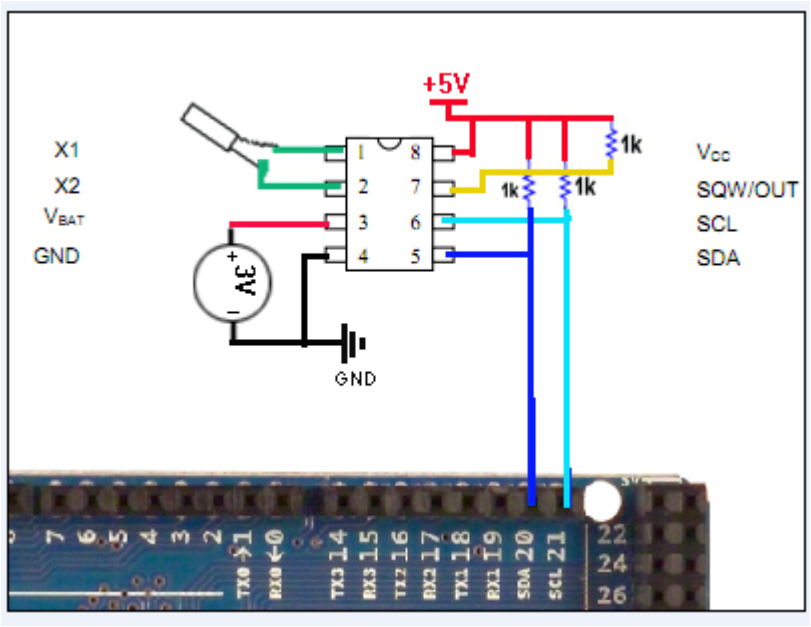
- The library is non-blocking which means you can press and hold the key all day long and your Arduino will continue processing the rest of your code.
- Consider, though, when you are writing your code that every delay() you use will take processing time away from the keypad. Something as short as delay(250) can make the keypad seem very unresponsive. And the same thing will happen if you sprinkle a bunch of delay(10)'s all through your code.
- The function getKey() returns a key value as soon as you press the key but it does not repeat automatically. Also, when you release the key you can track the KEY_RELEASED event if you are using the eventListener feature of the library.
- You will need pullup resistors, somewhere between 2k to 10k, on each of the row pins.

Project 14 –Real Time Clock 1307

What you will need

1 X DS1307 RTC chip
1 X 32.768kH quartz crystal
1 X battery socket
1 X CR2032 3V battery
3 X 1 K Ohm Resistors

Connect it up



Code :

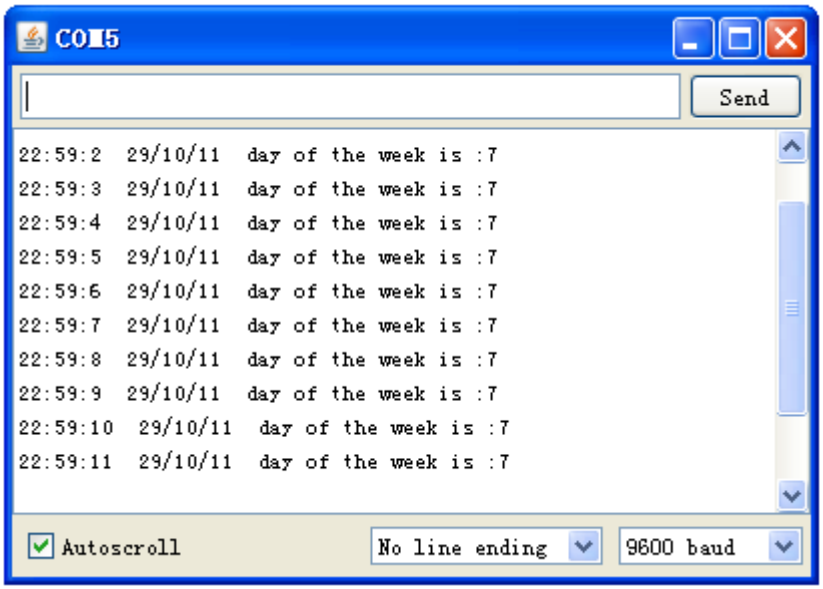
```
/* @file RTC1302.pde
|| @author Dr. Leong
|| @contact b2cqshop@gmail.com
|| WWW.B2CQSHOP.COM
|| @description
|| | Demonstrates using the KeypadEvent.
|| #
*/

#include <Wire.h>
#define DS1307_I2C_ADDRESS 0x68

int ledPin =  13;

// for RTC work
```

```
byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;


void setup()
{
  pinMode(ledPin, OUTPUT); // show the status
  Serial.begin(9600); // for debugging
  Serial.flush(); // need to flush serial buffer, otherwise first read from reset/power on may not be
correct
  Wire.begin(); // initialise i2c bus for DS1307 RTC

  // initial time values
  second = 0;
  minute = 59;
  hour = 22;
  dayOfWeek = 7;
  dayOfMonth = 29;
  month = 10;
  year = 11;
  setDateDs1307(second, minute, hour, dayOfWeek, dayOfMonth, month, year);
}

void loop()
{
    getDateDs1307(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year);
    Serial.print(hour,DEC);
    Serial.print(":");
    Serial.print(minute,DEC);
    Serial.print(":");
    Serial.print(second,DEC);
    Serial.print("    ");
    Serial.print(dayOfMonth,DEC);
    Serial.print("/");
    Serial.print(month,DEC);
    Serial.print("/");
    Serial.print(year,DEC);
    Serial.print("    day of the week is :");
    Serial.print(dayOfWeek,DEC);
    Serial.println("");
    digitalWrite(ledPin, HIGH);
    delay(500);
    digitalWrite(ledPin, LOW);
    delay(500);
```

```
}

void setDateDs1307(byte second,          // 0-59
byte minute,          // 0-59
byte hour,            // 1-23
byte dayOfWeek,       // 1-7
byte dayOfMonth,      // 1-28/29/30/31
byte month,           // 1-12
byte year)            // 0-99
{
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.send(0);
  Wire.send(decToBcd(second));      // 0 to bit 7 starts the clock
  Wire.send(decToBcd(minute));
  Wire.send(decToBcd(hour));        // If you want 12 hour am/pm you need to set
  // bit 6 (also need to change readDateDs1307)
  Wire.send(decToBcd(dayOfWeek));
  Wire.send(decToBcd(dayOfMonth));
  Wire.send(decToBcd(month));
  Wire.send(decToBcd(year));
  Wire.endTransmission();
}

// Gets the date and time from the ds1307
void getDateDs1307(byte *second,
byte *minute,
byte *hour,
byte *dayOfWeek,
byte *dayOfMonth,
byte *month,
byte *year)
{
  // Reset the register pointer
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.send(0);
  Wire.endTransmission();

  Wire.requestFrom(DS1307_I2C_ADDRESS, 7);

  // A few of these need masks because certain bits are control bits
  *second     = bcdToDec(Wire.receive() & 0x7f);
  *minute     = bcdToDec(Wire.receive());
  *hour       = bcdToDec(Wire.receive() & 0x3f);  // Need to change this if 12 hour am/pm
```

```
    *dayOfWeek    = bcdToDec(Wire.receive());
    *dayOfMonth = bcdToDec(Wire.receive());
    *month          = bcdToDec(Wire.receive());
    *year           = bcdToDec(Wire.receive());
}

// Convert normal decimal numbers to binary coded decimal
byte decToBcd(byte val)
{
   return ( (val/10*16) + (val%10) );
}
// Convert binary coded decimal to normal decimal numbers
byte bcdToDec(byte val)
{
   return ( (val/16*10) + (val%16) );
}
```

The result of this project is :



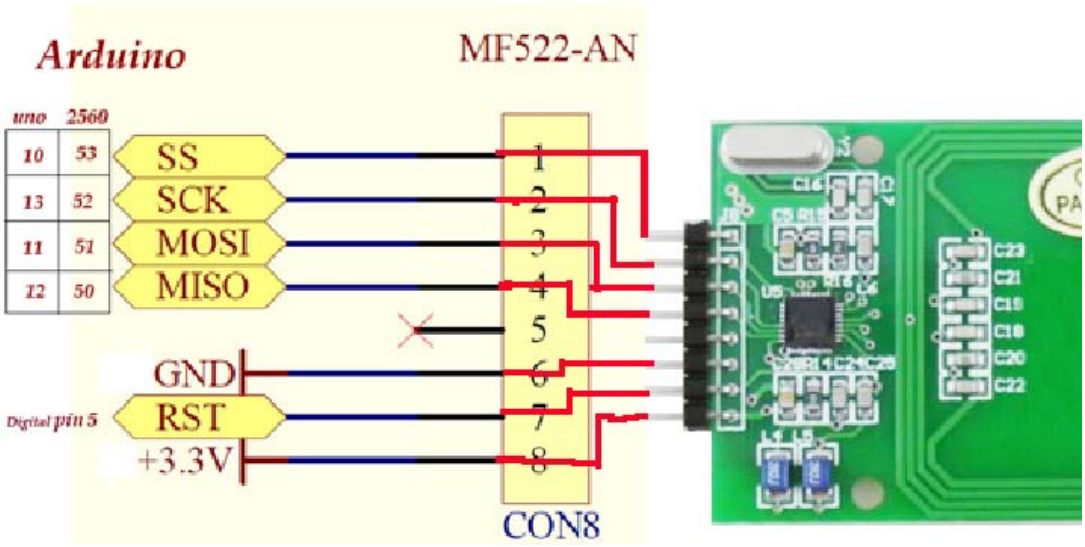
Project 15- RFID module

This MFRC522 chip and breakout is designed to be used by 3.3V systems. To use it with a 5V system such as an Arduino, a level shifter is required to convert the high voltages into 3.3V. If you have a 3.3V embedded system you won't have to use the shifter of course

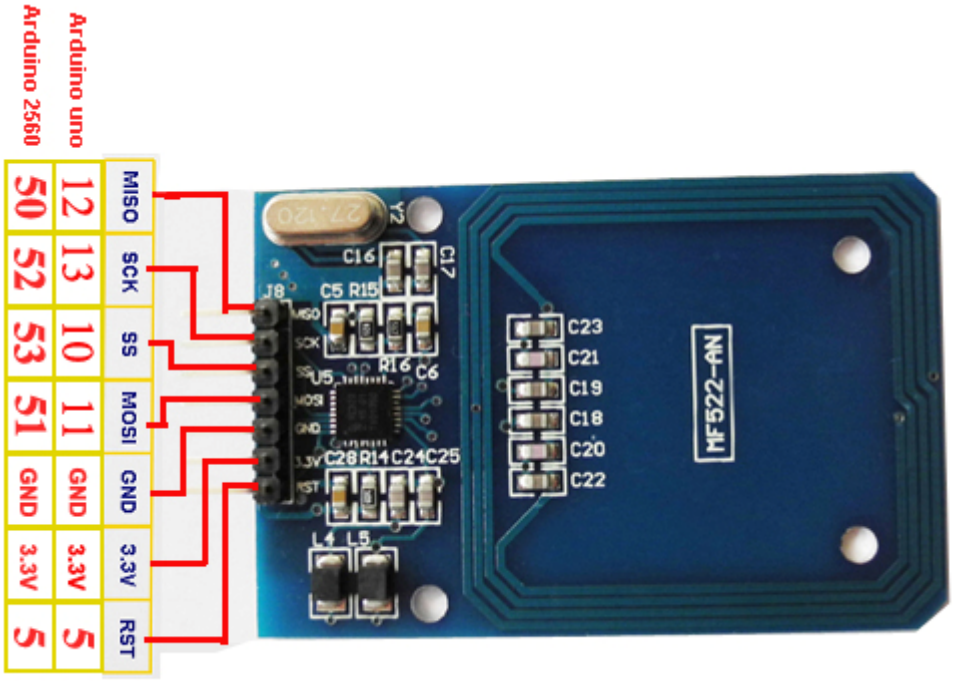
What you will need

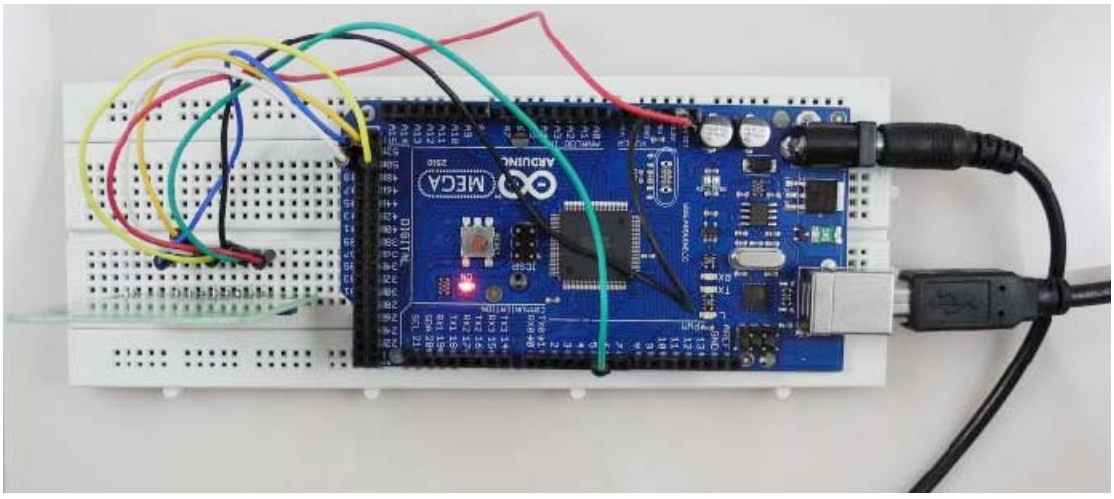
1 X RFID module
1 X Rectangle Tag
1 X Round Tag
Lots of jumper cables

Connect it up



The new version of the RFID, the connect pin is :


Arduino 2560
Arduino uno
50
52
53
51
GND
3.3V
5
12
13
10
11
GND
3.3V
5
MISO
SCK
SS
MOSI
GND
3.3V
RST
MF522-AN
C23
C21
C19
C18
C20
C22



Code :

```
/*
 * 文 件 名：RFID.pde
 * 创 建 者：Dr.Leong    ( WWW.B2CQSHOP.COM )
 * 创建日期：2011.09.19
 * 修 改 者：
 * 修改日期：
 * 功能描述：Mifare1 寻卡→防冲突→选卡→读写 接口
 */
 // the sensor communicates using SPI, so include the library:
#include <SPI.h>

#define    uchar      unsigned char
#define    uint  unsigned int

//数组最大长度
#define MAX_LEN 16

//////////////////////////////////////////////////////////////////
//set the pin
//////////////////////////////////////////////////////////////////
const int chipSelectPin = 53;
const int NRSTPD = 5;

//MF522 命令字
#define PCD_IDLE              0x00               //NO action;取消当前命令
#define PCD_AUTHENT           0x0E               //验证密钥
#define PCD_RECEIVE           0x08               //接收数据
#define PCD_TRANSMIT          0x04               //发送数据
#define PCD_TRANSCEIVE        0x0C               //发送并接收数据
#define PCD_RESETPHASE        0x0F               //复位
#define PCD_CALCCRC           0x03               //CRC 计算

//Mifare_One 卡片命令字
#define PICC_REQIDL           0x26               //寻天线区内未进入休眠状态
#define PICC_REQALL           0x52               //寻天线区内全部卡
#define PICC_ANTICOLL         0x93               //防冲撞
#define PICC_SElECTTAG        0x93               //选卡
#define PICC_AUTHENT1A        0x60               //验证 A 密钥
#define PICC_AUTHENT1B        0x61               //验证 B 密钥
#define PICC_READ             0x30               //读块
#define PICC_WRITE            0xA0               //写块
#define PICC_DECREMENT        0xC0               //扣款
```

```
#define PICC_INCREMENT          0xC1                //充值
#define PICC_RESTORE            0xC2                //调块数据到缓冲区
#define PICC_TRANSFER           0xB0                //保存缓冲区中数据
#define PICC_HALT               0x50                //休眠


//和 MF522 通讯时返回的错误代码
#define MI_OK                   0
#define MI_NOTAGERR             1
#define MI_ERR                  2


//------------------MFRC522 寄存器---------------
//Page 0:Command and Status
#define     Reserved00          0x00
#define     CommandReg          0x01
#define     CommIEnReg          0x02
#define     DivlEnReg           0x03
#define     CommIrqReg          0x04
#define     DivIrqReg           0x05
#define     ErrorReg            0x06
#define     Status1Reg          0x07
#define     Status2Reg          0x08
#define     FIFODataReg         0x09
#define     FIFOLevelReg        0x0A
#define     WaterLevelReg       0x0B
#define     ControlReg          0x0C
#define     BitFramingReg       0x0D
#define     CollReg             0x0E
#define     Reserved01          0x0F
//Page 1:Command
#define     Reserved10          0x10
#define     ModeReg             0x11
#define     TxModeReg           0x12
#define     RxModeReg           0x13
#define     TxControlReg        0x14
#define     TxAutoReg           0x15
#define     TxSelReg            0x16
#define     RxSelReg            0x17
#define     RxThresholdReg      0x18
#define     DemodReg            0x19
#define     Reserved11          0x1A
#define     Reserved12          0x1B
```

```
#define     MifareReg           0x1C
#define     Reserved13          0x1D
#define     Reserved14          0x1E
#define     SerialSpeedReg      0x1F
//Page 2:CFG
#define     Reserved20          0x20
#define     CRCResultRegM       0x21
#define     CRCResultRegL       0x22
#define     Reserved21          0x23
#define     ModWidthReg         0x24
#define     Reserved22          0x25
#define     RFCfgReg            0x26
#define     GsNReg              0x27
#define     CWGsPReg            0x28
#define     ModGsPReg           0x29
#define     TModeReg            0x2A
#define     TPrescalerReg       0x2B
#define     TReloadRegH         0x2C
#define     TReloadRegL         0x2D
#define     TCounterValueRegH   0x2E
#define     TCounterValueRegL   0x2F
//Page 3:TestRegister
#define     Reserved30          0x30
#define     TestSel1Reg         0x31
#define     TestSel2Reg         0x32
#define     TestPinEnReg        0x33
#define     TestPinValueReg     0x34
#define     TestBusReg          0x35
#define     AutoTestReg         0x36
#define     VersionReg          0x37
#define     AnalogTestReg       0x38
#define     TestDAC1Reg         0x39
#define     TestDAC2Reg         0x3A
#define     TestADCReg          0x3B
#define     Reserved31          0x3C
#define     Reserved32          0x3D
#define     Reserved33          0x3E
#define     Reserved34          0x3F
//-----------------------------------------------

//4 字节卡序列号，第 5 字节为校验字节
uchar serNum[5];
```

```
uchar   writeData[16]={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 100};  //初始化 100 元钱
uchar   moneyConsume = 18 ;  //消费 18 元
uchar   moneyAdd = 10 ;  //充值 10 元
//扇区 A 密码，16 个扇区，每个扇区密码 6Byte
 uchar sectorKeyA[16][16] = {{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                             {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                             //{0x19, 0x84, 0x07, 0x15, 0x76, 0x14},
                             {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                            };
 uchar sectorNewKeyA[16][16] = {{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                                {0xFF,  0xFF,  0xFF,  0xFF,  0xFF,  0xFF,
0xff,0x07,0x80,0x69, 0x19,0x84,0x07,0x15,0x76,0x14},
        //you can set another ket , such as   " 0x19, 0x84, 0x07, 0x15, 0x76, 0x14 "
        //{0x19, 0x84, 0x07, 0x15, 0x76, 0x14, 0xff,0x07,0x80,0x69, 0x19,0x84,0x07,0x15,0x76,0x14},
        // but when loop, please set the   sectorKeyA, the same key, so that RFID module can read the card
                                {0xFF,  0xFF,  0xFF,  0xFF,  0xFF,  0xFF,
0xff,0x07,0x80,0x69, 0x19,0x33,0x07,0x15,0x34,0x14},
                               };

void setup() {
    Serial.begin(9600);      // RFID reader SOUT pin connected to Serial RX pin at 2400bps
 // start the SPI library:
   SPI.begin();

// Set digital pin 10 as OUTPUT to connect it to the RFID /ENABLE pin
   pinMode(chipSelectPin,OUTPUT);
      digitalWrite(chipSelectPin, LOW);              // Activate the RFID reader
   pinMode(NRSTPD,OUTPUT);                   // Set digital pin 10 , Not Reset and Power-down
      digitalWrite(NRSTPD, HIGH);

   MFRC522_Init();
}

void loop()
{
      uchar i,tmp;
      uchar status;
            uchar str[MAX_LEN];
            uchar RC_size;
            uchar blockAddr;   //选择操作的块地址 0～63


           //寻卡，返回卡类型
```

```
        status = MFRC522_Request(PICC_REQIDL, str);
        if (status == MI_OK)
        {
                        Serial.println("Find out a card ");
            Serial.print(str[0],BIN);
                        Serial.print(" , ");
            Serial.print(str[1],BIN);
                        Serial.println(" ");
        }

        //防冲撞，返回卡的序列号 4 字节
        status = MFRC522_Anticoll(str);
        memcpy(serNum, str, 5);
        if (status == MI_OK)
        {

            Serial.println("The card's number is    : ");
            Serial.print(serNum[0],BIN);
                        Serial.print(" , ");
            Serial.print(serNum[1],BIN);
                        Serial.print(" , ");
            Serial.print(serNum[2],BIN);
                        Serial.print(" , ");
            Serial.print(serNum[3],BIN);
                        Serial.print(" , ");
            Serial.print(serNum[4],BIN);
                        Serial.println(" ");
        }

        //选卡，返回卡容量
        RC_size = MFRC522_SelectTag(serNum);
        if (RC_size != 0)

        {
                        Serial.print("The size of the card is    :     ");
            Serial.print(RC_size,DEC);
                        Serial.println(" K bits ");
        }

        //注册卡
        blockAddr = 7;              //数据块 7
        status = MFRC522_Auth(PICC_AUTHENT1A, blockAddr, sectorKeyA[blockAddr/4],
serNum); //认证
```

```
        if (status == MI_OK)
        {
            //写数据
            status = MFRC522_Write(blockAddr, sectorNewKeyA[blockAddr/4]);
                        Serial.print("set the new card password, and can modify the data of
the Sector ");
                        Serial.print(blockAddr/4,DEC);
                        Serial.println(" : ");
            for (i=0; i<6; i++)
                {
                        Serial.print(str[i],HEX);
                            Serial.print(" , ");
                }
                        Serial.println(" ");
                        blockAddr = blockAddr - 3 ;
                        status = MFRC522_Write(blockAddr, writeData);
                        if(status == MI_OK)
                        {
                            Serial.println("You are B2CQSHOP VIP Member, The card has
$100 !");
                        }
        }

        //读卡
        blockAddr = 7;          //数据块 7
        status          =          MFRC522_Auth(PICC_AUTHENT1A,          blockAddr,
sectorNewKeyA[blockAddr/4], serNum);    //认证
        if (status == MI_OK)
        {
            //读数据
                        blockAddr = blockAddr - 3 ;
                        status = MFRC522_Read(blockAddr, str);
            if (status == MI_OK)
            {
                                Serial.println("Read from the card ,the data is : ");
                for (i=0; i<16; i++)
                {
                            Serial.print(str[i],DEC);
                                Serial.print(" , ");
                }
                            Serial.println(" ");
            }
        }
```

```
                //消费
        blockAddr = 7;          //数据块 7
        status = MFRC522_Auth(PICC_AUTHENT1A, blockAddr, sectorNewKeyA[blockAddr/4], serNum);   //认证
        if (status == MI_OK)
        {
            //读数据
                        blockAddr = blockAddr - 3 ;
            status = MFRC522_Read(blockAddr, str);
            if (status == MI_OK)
            {
                        if( str[15] < moneyConsume )
                        {
                            Serial.println(" The money is not enough !");
                        }
                        else
                        {
                            str[15] = str[15] - moneyConsume;
                            status = MFRC522_Write(blockAddr, str);
                            if(status == MI_OK)
                            {
                                Serial.print("You pay $18 for items in B2CQSHOP.COM . Now, Your money balance is : $");
                        Serial.print(str[15],DEC);
                                Serial.println(" ");
                            }
                        }
            }
        }

                //充值
        blockAddr = 7;          //数据块 7
        status = MFRC522_Auth(PICC_AUTHENT1A, blockAddr, sectorNewKeyA[blockAddr/4], serNum);   //认证
        if (status == MI_OK)
        {
            //读数据
                        blockAddr = blockAddr - 3 ;
            status = MFRC522_Read(blockAddr, str);
            if (status == MI_OK)
            {
                        tmp = (int)(str[15] + moneyAdd) ;
```

```
                                //Serial.println(tmp,DEC);
                                if( tmp < (char)254 )
                                {
                                    Serial.println(" The money of the card can not be more than
255 !");
                                }
                                else
                                {
                                    str[15] = str[15] + moneyAdd ;
                                    status = MFRC522_Write(blockAddr, str);
                                    if(status == MI_OK)
                                    {
                                        Serial.print("You add $10 to your card in
B2CQSHOP.COM , Your money balance is :  $");
                                Serial.print(str[15],DEC);
                                        Serial.println(" ");
                                    }
                                }
                        }
                }
                        Serial.println(" ");
                MFRC522_Halt();             //命令卡片进入休眠状态

}

/*
 * 函 数 名：Write_MFRC5200
 * 功能描述：向 MFRC522 的某一寄存器写一个字节数据
 * 输入参数：addr--寄存器地址；val--要写入的值
 * 返 回 值：无
 */
void Write_MFRC522(uchar addr, uchar val)
{
    digitalWrite(chipSelectPin, LOW);

    //地址格式：0XXXXXX0
    SPI.transfer((addr<<1)&0x7E);
    SPI.transfer(val);

    digitalWrite(chipSelectPin, HIGH);
}
```

```
/*
 * 函 数 名：Read_MFRC522
 * 功能描述：从 MFRC522 的某一寄存器读一个字节数据
 * 输入参数：addr--寄存器地址
 * 返 回 值：返回读取到的一个字节数据
 */
uchar Read_MFRC522(uchar addr)
{
    uchar val;

    digitalWrite(chipSelectPin, LOW);

    //地址格式：1XXXXXX0
    SPI.transfer(((addr<<1)&0x7E) | 0x80);
    val =SPI.transfer(0x00);

    digitalWrite(chipSelectPin, HIGH);

    return val;
}

/*
 * 函 数 名：SetBitMask
 * 功能描述：置 RC522 寄存器位
 * 输入参数：reg--寄存器地址;mask--置位值
 * 返 回 值：无
 */
void SetBitMask(uchar reg, uchar mask)
{
    uchar tmp;
    tmp = Read_MFRC522(reg);
    Write_MFRC522(reg, tmp | mask);  // set bit mask
}


/*
 * 函 数 名：ClearBitMask
 * 功能描述：清 RC522 寄存器位
 * 输入参数：reg--寄存器地址;mask--清位值
 * 返 回 值：无
 */
void ClearBitMask(uchar reg, uchar mask)
{
```

```
    uchar tmp;
    tmp = Read_MFRC522(reg);
    Write_MFRC522(reg, tmp & (~mask));    // clear bit mask
}


/*
 * 函 数 名：AntennaOn
 * 功能描述：开启天线,每次启动或关闭天险发射之间应至少有 1ms 的间隔
 * 输入参数：无
 * 返 回 值：无
 */
void AntennaOn(void)
{
    uchar temp;

    temp = Read_MFRC522(TxControlReg);
    if (!(temp & 0x03))
    {
        SetBitMask(TxControlReg, 0x03);
    }
}


/*
 * 函 数 名：AntennaOff
 * 功能描述：关闭天线,每次启动或关闭天险发射之间应至少有 1ms 的间隔
 * 输入参数：无
 * 返 回 值：无
 */
void AntennaOff(void)
{
    ClearBitMask(TxControlReg, 0x03);
}


/*
 * 函 数 名：ResetMFRC522
 * 功能描述：复位 RC522
 * 输入参数：无
 * 返 回 值：无
 */
void MFRC522_Reset(void)
```

```
{
    Write_MFRC522(CommandReg, PCD_RESETPHASE);
}


/*
 * 函 数 名：InitMFRC522
 * 功能描述：初始化 RC522
 * 输入参数：无
 * 返 回 值：无
 */
void MFRC522_Init(void)
{
    digitalWrite(NRSTPD,HIGH);

    MFRC522_Reset();

    //Timer: TPrescaler*TreloadVal/6.78MHz = 24ms
    Write_MFRC522(TModeReg, 0x8D);      //Tauto=1; f(Timer) = 6.78MHz/TPreScaler
    Write_MFRC522(TPrescalerReg, 0x3E);  //TModeReg[3..0] + TPrescalerReg
    Write_MFRC522(TReloadRegL, 30);
    Write_MFRC522(TReloadRegH, 0);

    Write_MFRC522(TxAutoReg, 0x40);      //100%ASK
    Write_MFRC522(ModeReg, 0x3D);        //CRC 初始值 0x6363   ???

    //ClearBitMask(Status2Reg, 0x08);      //MFCrypto1On=0
    //Write_MFRC522(RxSelReg, 0x86);       //RxWait = RxSelReg[5..0]
    //Write_MFRC522(RFCfgReg, 0x7F);           //RxGain = 48dB

    AntennaOn();        //打开天线
}


/*
 * 函 数 名：MFRC522_Request
 * 功能描述：寻卡，读取卡类型号
 * 输入参数：reqMode--寻卡方式，
 *           TagType--返回卡片类型
 *                0x4400 = Mifare_UltraLight
 *                0x0400 = Mifare_One(S50)
 *                0x0200 = Mifare_One(S70)
 *                0x0800 = Mifare_Pro(X)
```

```
 *                      0x4403 = Mifare_DESFire
 * 返 回 值：成功返回 MI_OK
 */
uchar MFRC522_Request(uchar reqMode, uchar *TagType)
{
    uchar status;
    uint backBits;              //接收到的数据位数

    Write_MFRC522(BitFramingReg, 0x07);          //TxLastBists = BitFramingReg[2..0]    ???

    TagType[0] = reqMode;
    status = MFRC522_ToCard(PCD_TRANSCEIVE, TagType, 1, TagType, &backBits);

    if ((status != MI_OK) || (backBits != 0x10))
    {
        status = MI_ERR;
    }

    return status;
}


/*
 * 函 数 名：MFRC522_ToCard
 * 功能描述：RC522 和 ISO14443 卡通讯
 * 输入参数：command--MF522 命令字，
 *               sendData--通过 RC522 发送到卡片的数据,
 *               sendLen--发送的数据长度
 *               backData--接收到的卡片返回数据，
 *               backLen--返回数据的位长度
 * 返 回 值：成功返回 MI_OK
 */
uchar MFRC522_ToCard(uchar command, uchar *sendData, uchar sendLen, uchar *backData,
uint *backLen)
{
    uchar status = MI_ERR;
    uchar irqEn = 0x00;
    uchar waitIRq = 0x00;
    uchar lastBits;
    uchar n;
    uint i;

    switch (command)
```

```
{
    case PCD_AUTHENT:       //认证卡密
    {
        irqEn = 0x12;
        waitIRq = 0x10;
        break;
    }
    case PCD_TRANSCEIVE:    //发送 FIFO 中数据
    {
        irqEn = 0x77;
        waitIRq = 0x30;
        break;
    }
    default:
        break;
}

Write_MFRC522(CommIEnReg, irqEn|0x80);    //允许中断请求
ClearBitMask(CommIrqReg, 0x80);           //清除所有中断请求位
SetBitMask(FIFOLevelReg, 0x80);           //FlushBuffer=1, FIFO 初始化

Write_MFRC522(CommandReg, PCD_IDLE);    //NO action;取消当前命令   ???

//向 FIFO 中写入数据
for (i=0; i<sendLen; i++)
{
    Write_MFRC522(FIFODataReg, sendData[i]);
}

//执行命令
Write_MFRC522(CommandReg, command);
if (command == PCD_TRANSCEIVE)
{
    SetBitMask(BitFramingReg, 0x80);          //StartSend=1,transmission of data starts
}

//等待接收数据完成
i = 2000; //i 根据时钟频率调整，操作 M1 卡最大等待时间 25ms  ???
do
{
    //CommIrqReg[7..0]
    //Set1 TxIRq RxIRq IdleIRq HiAlerIRq LoAlertIRq ErrIRq TimerIRq
    n = Read_MFRC522(CommIrqReg);
```

```
        i--;
    }
    while ((i!=0) && !(n&0x01) && !(n&waitIRq));

    ClearBitMask(BitFramingReg, 0x80);                  //StartSend=0

    if (i != 0)
    {
        if(!(Read_MFRC522(ErrorReg) & 0x1B))    //BufferOvfl Collerr CRCErr ProtecolErr
        {
            status = MI_OK;
            if (n & irqEn & 0x01)
            {
                status = MI_NOTAGERR;                   //??
            }

            if (command == PCD_TRANSCEIVE)
            {
                n = Read_MFRC522(FIFOLevelReg);
                lastBits = Read_MFRC522(ControlReg) & 0x07;
                if (lastBits)
                {
                    *backLen = (n-1)*8 + lastBits;
                }
                else
                {
                    *backLen = n*8;
                }

                if (n == 0)
                {
                    n = 1;
                }
                if (n > MAX_LEN)
                {
                    n = MAX_LEN;
                }

                //读取 FIFO 中接收到的数据
                for (i=0; i<n; i++)
                {
                    backData[i] = Read_MFRC522(FIFODataReg);
                }
```

```
                }
            }
            else
            {
                status = MI_ERR;
            }

        }

        //SetBitMask(ControlReg,0x80);                  //timer stops
        //Write_MFRC522(CommandReg, PCD_IDLE);

        return status;
}


/*
 * 函 数 名：MFRC522_Anticoll
 * 功能描述：防冲突检测，读取选中卡片的卡序列号
 * 输入参数：serNum--返回 4 字节卡序列号,第 5 字节为校验字节
 * 返 回 值：成功返回 MI_OK
 */
uchar MFRC522_Anticoll(uchar *serNum)
{
        uchar status;
        uchar i;
        uchar serNumCheck=0;
        uint unLen;


        //ClearBitMask(Status2Reg, 0x08);           //TempSensclear
        //ClearBitMask(CollReg,0x80);               //ValuesAfterColl
        Write_MFRC522(BitFramingReg, 0x00);           //TxLastBists = BitFramingReg[2..0]

        serNum[0] = PICC_ANTICOLL;
        serNum[1] = 0x20;
        status = MFRC522_ToCard(PCD_TRANSCEIVE, serNum, 2, serNum, &unLen);

        if (status == MI_OK)
        {
            //校验卡序列号
            for (i=0; i<4; i++)
            {
```

```
            serNumCheck ^= serNum[i];
        }
        if (serNumCheck != serNum[i])
        {
            status = MI_ERR;
        }
    }

    //SetBitMask(CollReg, 0x80);            //ValuesAfterColl=1

    return status;
}


/*
 * 函 数 名：CalulateCRC
 * 功能描述：用 MF522 计算 CRC
 * 输入参数：pIndata--要读数 CRC 的数据，len--数据长度，pOutData--计算的 CRC 结果
 * 返 回 值：无
 */
void CalulateCRC(uchar *pIndata, uchar len, uchar *pOutData)
{
    uchar i, n;

    ClearBitMask(DivIrqReg, 0x04);              //CRCIrq = 0
    SetBitMask(FIFOLevelReg, 0x80);             //清 FIFO 指针
    //Write_MFRC522(CommandReg, PCD_IDLE);

    //向 FIFO 中写入数据
    for (i=0; i<len; i++)
    {
        Write_MFRC522(FIFODataReg, *(pIndata+i));
    }
    Write_MFRC522(CommandReg, PCD_CALCCRC);

    //等待 CRC 计算完成
    i = 0xFF;
    do
    {
        n = Read_MFRC522(DivIrqReg);
        i--;
    }
    while ((i!=0) && !(n&0x04));                //CRCIrq = 1
```

```
    //读取 CRC 计算结果
    pOutData[0] = Read_MFRC522(CRCResultRegL);
    pOutData[1] = Read_MFRC522(CRCResultRegM);
}


/*
 * 函 数 名：MFRC522_SelectTag
 * 功能描述：选卡，读取卡存储器容量
 * 输入参数：serNum--传入卡序列号
 * 返 回 值：成功返回卡容量
 */
uchar MFRC522_SelectTag(uchar *serNum)
{
    uchar i;
    uchar status;
    uchar size;
    uint recvBits;
    uchar buffer[9];

    //ClearBitMask(Status2Reg, 0x08);                    //MFCrypto1On=0

    buffer[0] = PICC_SElECTTAG;
    buffer[1] = 0x70;
    for (i=0; i<5; i++)
    {
        buffer[i+2] = *(serNum+i);
    }
    CalulateCRC(buffer, 7, &buffer[7]);            //??
    status = MFRC522_ToCard(PCD_TRANSCEIVE, buffer, 9, buffer, &recvBits);

    if ((status == MI_OK) && (recvBits == 0x18))
    {
        size = buffer[0];
    }
    else
    {
        size = 0;
    }

    return size;
}
```

```
/*
 * 函 数 名：MFRC522_Auth
 * 功能描述：验证卡片密码
 * 输入参数：authMode--密码验证模式
                    0x60 = 验证 A 密钥
                    0x61 = 验证 B 密钥
                BlockAddr--块地址
                Sectorkey--扇区密码
                serNum--卡片序列号，4 字节
 * 返 回 值：成功返回 MI_OK
 */
uchar MFRC522_Auth(uchar authMode, uchar BlockAddr, uchar *Sectorkey, uchar *serNum)
{
    uchar status;
    uint recvBits;
    uchar i;
    uchar buff[12];

    //验证指令+块地址＋扇区密码＋卡序列号
    buff[0] = authMode;
    buff[1] = BlockAddr;
    for (i=0; i<6; i++)
    {
        buff[i+2] = *(Sectorkey+i);
    }
    for (i=0; i<4; i++)
    {
        buff[i+8] = *(serNum+i);
    }
    status = MFRC522_ToCard(PCD_AUTHENT, buff, 12, buff, &recvBits);

    if ((status != MI_OK) || (!(Read_MFRC522(Status2Reg) & 0x08)))
    {
        status = MI_ERR;
    }

    return status;
}


/*
```

```
 * 函 数 名：MFRC522_Read
 * 功能描述：读块数据
 * 输入参数：blockAddr--块地址;recvData--读出的块数据
 * 返 回 值：成功返回 MI_OK
 */
uchar MFRC522_Read(uchar blockAddr, uchar *recvData)
{
    uchar status;
    uint unLen;

    recvData[0] = PICC_READ;
    recvData[1] = blockAddr;
    CalulateCRC(recvData,2, &recvData[2]);
    status = MFRC522_ToCard(PCD_TRANSCEIVE, recvData, 4, recvData, &unLen);

    if ((status != MI_OK) || (unLen != 0x90))
    {
        status = MI_ERR;
    }

    return status;
}


/*
 * 函 数 名：MFRC522_Write
 * 功能描述：写块数据
 * 输入参数：blockAddr--块地址;writeData--向块写 16 字节数据
 * 返 回 值：成功返回 MI_OK
 */
uchar MFRC522_Write(uchar blockAddr, uchar *writeData)
{
    uchar status;
    uint recvBits;
    uchar i;
    uchar buff[18];

    buff[0] = PICC_WRITE;
    buff[1] = blockAddr;
    CalulateCRC(buff, 2, &buff[2]);
    status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 4, buff, &recvBits);

    if ((status != MI_OK) || (recvBits != 4) || ((buff[0] & 0x0F) != 0x0A))
```

```
    {
        status = MI_ERR;
    }

    if (status == MI_OK)
    {
        for (i=0; i<16; i++)        //向 FIFO 写 16Byte 数据
        {
            buff[i] = *(writeData+i);
        }
        CalulateCRC(buff, 16, &buff[16]);
        status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 18, buff, &recvBits);

        if ((status != MI_OK) || (recvBits != 4) || ((buff[0] & 0x0F) != 0x0A))
        {
            status = MI_ERR;
        }
    }

    return status;
}


/*
 * 函 数 名：MFRC522_Halt
 * 功能描述：命令卡片进入休眠状态
 * 输入参数：无
 * 返 回 值：无
 */
void MFRC522_Halt(void)
{
    uchar status;
    uint unLen;
    uchar buff[4];

    buff[0] = PICC_HALT;
    buff[1] = 0;
    CalulateCRC(buff, 2, &buff[2]);

    status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 4, buff,&unLen);
}
```

Project 15- Hardware Overview

About NFC

NFC (Near Field Communication)

NFC (Near Field Communication) is a set of short-range (typically up to 10cm) wireless communication technologies designed to offer light-weight and secure communication between two devices. While NFC was invented by NXP (Phillips at the time), Nokia and Sony, the main body behind the NFC 'standard' today is the NFC Forum , who are responsible for publishing and maintaining a variety of standards relating to NFC technology.

NFC operates at 13.56MHz, and is based around an "initiator" and "target" model where the initiator generates a small magnetic field that powers the target, meaning that the target does not require a power source. This means of communication is referred to as **Passive Communication**, and is used to read and write to small, inexpensive 13.56MHz RFID tags based on standards like ISO14443A. **Active communication** (peer-to-peer) is also possible when both devices are powered, where each device alternately creates its own magentic field, with the secondary device as a target and vice versa in continuous rotation.

Passive Communication: ISO14443A Cards (Mifare, etc.)

Please check the **S50.pdf** about the Mifare Classic Cards .

MIFARE Classic cards come in 1K and 4K varieties. In here , we use the 1k card.

Active Communication (Peer-to-Peer)

Active or "Peer-to-Peer" communication is still based around the Initiator/Target model described earlier, but both devices are actively powered and switch roles from being an Initiator or a Target during the communication. When one device is initiating a conversation with the other, it enables it's magnetic field and the receiving device listens in (with it's own magnetic field disabled). Afterwards, the target/recipient device may need to respond and will in turn activate it's own magnetic field and the original device will be configured as the target. Despite two devices being present, only one magnetic field is active at a time, with each device constantly enabling or disabling its own magnetic field.

ToDo: Add better description of active mode, but I need to test it out a bit first myself!

About MiFare Card

Mifare Classic cards typically have a **4-byte NUID** that uniquely (within the numeric limits of the value) identifies the card. It's possible to have a 7 byte IDs as well, but the 4 byte models are far more common for Mifare Classic.

EEPROM Memory

Mifare Classic cards have either 1K or 4K of EEPROM memory. Each memory block can be configured with different access conditions, with two seperate authentication keys present in each block.

Mifare Classic cards are divided into section called **sectors** and **blocks**. Each "sector" has individual access rights, and contains a fixed number of "blocks" that are controlled by these access rights. Each block contains 16 bytes, and sectors contains either 4 blocks (1K/4K cards) for a total of 64 bytes per sector, or 16 blocks (4K cards only) for a total of 256 bytes per sector. The card types are organised as follows:

- **1K Cards** - 16 sectors of 4 blocks each (sectors 0..15)
- **4K Cards** - 32 sectors of 4 blocks each (sectors 0..31) and 8 sectors of 16 blocks each (sectors 32..39)

In Here we talk about the **4 block sectors**

1K and 4K cards both use 16 sectors of 4 blocks each, with the bottom 1K of memory on the 4K cards being organised identically to the 1K models for compatability reasons. These individual 4 block sectors (containing 64 byts each) have basic security features are can each be configured with seperate read/write access and two different 6-byte authentication keys (the keys can be different for each sector). Due to these security features (which are stored in the last block, called the **Sector Trailer**), only the bottom 3 blocks of each sector are actually available for data storage, meaning you have 48 bytes per 64 byte sector available for your own use.

Each 4 block sector is organised as follows, with four rows of 16 bytes each for a total of 64-bytes per sector. The first two sectors of any card are shown:

```
Sector  Block   Bytes                                                   Description
------  -----   -----                                                   -----------
                0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15

1       3       [-------KEY A-------]  [Access Bits]  [-------KEY B-------]  Sector Trailer
        2       [                          Data                          ]  Data
        1       [                          Data                          ]  Data
        0       [                          Data                          ]  Data

0       3       [-------KEY A-------]  [Access Bits]  [-------KEY B-------]  Sector Trailer
        2       [                          Data                          ]  Data
        1       [                          Data                          ]  Data
        0       [                   Manufacturer Data                    ]  Manufacturer Block
```

Sector Trailer (Block 3)

The sector trailer block contains the two secret keys (Key A and Key B), as well as the access conditions for the four blocks. It has the following structure:

```
Sector Trailer Bytes
----------------------------------------------------------------
0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
[       Key A       ]   [Access Bits]   [       Key B       ]
```

For more information in using Keys to access the clock contents, see Accessing Data Blocks further below.

Data Blocks (Blocks 0..2)

Data blocks are 16 bytes wide and, depending on the permissions set in the access bits, can be read from and written to. You are free to use the 16 data bytes in any way you wish. You can easily store text input, store four 32-bit integer values, a 16 character uri, etc.

Data Blocks as "Value Blocks"

An alternative to storing random data in the 16 byte-wide blocks is to configure them as "Value Blocks". Value blocks allow performing electronic purse functions (valid commands are: read, write, increment, decrement, restore, transfer).

Each Value block contains a single signed 32-bit value, and this value is stored 3 times for data integrity and security reasons. It is stored twice non-inverted, and once inverted. The last 4 bytes are used for a 1-byte address, which is stored 4 times (twice non-inverted, and twice inverted).

Data blocks configured as "Value Blocks" have the following structure:

```
Value Block Bytes
----------------------------------------------------------------
0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
[   Value   ]   [   ~Value  ]   [   Value   ]   [A  ~A  A   ~A]
```

Manufacturer Block (Sector 0, Block 0)

Sector 0 is special since it contains the Manufacturer Block. This block contains the manufacturer data, and is read-only. It should be avoided unless you know what you are doing.

Accessing EEPROM Memory

To access the EEPROM on the cards, you need to perform the following steps:

1. You must retrieve the 4-byte NUID of the card (this can sometimes be 7-bytes long as well, though rarely for Mifare Classic cards). This is required for the subsequent authentication process.
2. You must authenticate the sector you wish to access according to the access rules defined in the Sector Trailer block for that sector, by passing in the appropriate 6 byte Authentication Key (ex. 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF for new cards).
3. Once authenication has succeeded, and depending on the sector permissions, you can then read/write/increment/decrement the contents of the specific block. Note that you need to re-authenticate for each sector that you access, since each sector can have it's own distinct access keys and rights!

Note on Authentication

Before you can do access the sector's memory, you first need to "authenticate" according to the security settings stored in the Sector Trailer. By default, any new card will generally be configured to allow full access to every block in the sector using Key A and a value of 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF. Some other common keys that you may wish to try if this doesn't work are:

```
0XFF 0XFF 0XFF 0XFF 0XFF 0XFF

0XD3 0XF7 0XD3 0XF7 0XD3 0XF7

0XA0 0XA1 0XA2 0XA3 0XA4 0XA5

0XB0 0XB1 0XB2 0XB3 0XB4 0XB5

0X4D 0X3A 0X99 0XC3 0X51 0XDD
```

```
        0X1A 0X98 0X2C 0X7E 0X45 0X9A

        0XAA 0XBB 0XCC 0XDD 0XEE 0XFF

        0X00 0X00 0X00 0X00 0X00 0X00

        0XAB 0XCD 0XEF 0X12 0X34 0X56
```

Example of a New Mifare Classic 1K Card

The follow memory dump illustrates the structure of a 1K Mifare Classic Card, where the data and Sector Trailer blocks can be clearly seen:

```
[------------------------Start of Memory Dump-----------------------]

------------------------Sector 0--------------------

Block 0  8E 02 6F 66 85 08 04 00 62 63 64 65 66 67 68 69  ?.of?...bcdefghi

Block 1  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 2  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 3  00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ

------------------------Sector 1-------------------------

Block 4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 5  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 6  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 7  00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ

------------------------Sector 2-------------------------

Block 8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 9  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 11 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ

------------------------Sector 3-------------------------
```

```
Block 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 15 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 4-------------------------
Block 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 17 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 19 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 5-------------------------
Block 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 23 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 6-------------------------
Block 24 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 26 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 27 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 7-------------------------
Block 28 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 29 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 31 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
```

```
------------------------Sector 8-------------------------
Block 32 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 34 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 35 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 9-------------------------
Block 36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 37 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 38 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 39 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 10------------------------
Block 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 42 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 43 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 11------------------------
Block 44 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 45 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 46 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 47 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ
------------------------Sector 12------------------------
Block 48 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 49 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
Block 51 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ

------------------------Sector 13-------------------------

Block 52 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 53 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 54 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 55 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ

------------------------Sector 14-------------------------

Block 56 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 57 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 58 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 59 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ

------------------------Sector 15-------------------------

Block 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 61 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 62 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Block 63 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF  ......ÿ.?iÿÿÿÿÿÿ

[------------------------End of Memory Dump-------------------------]
```

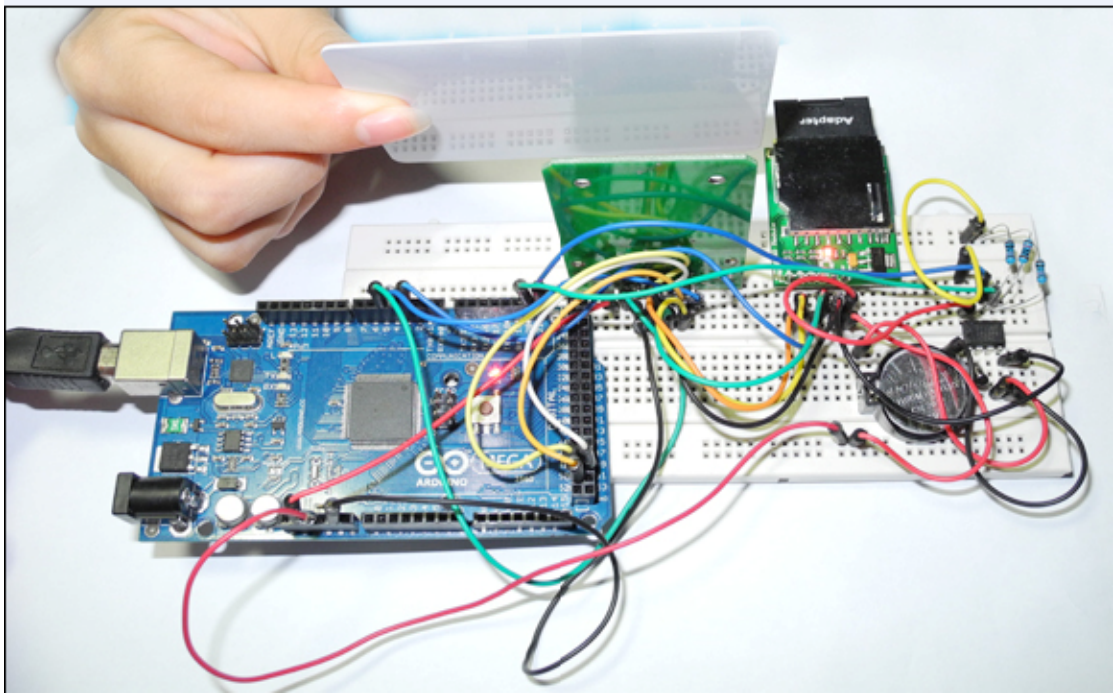
Project 16 – Big Project

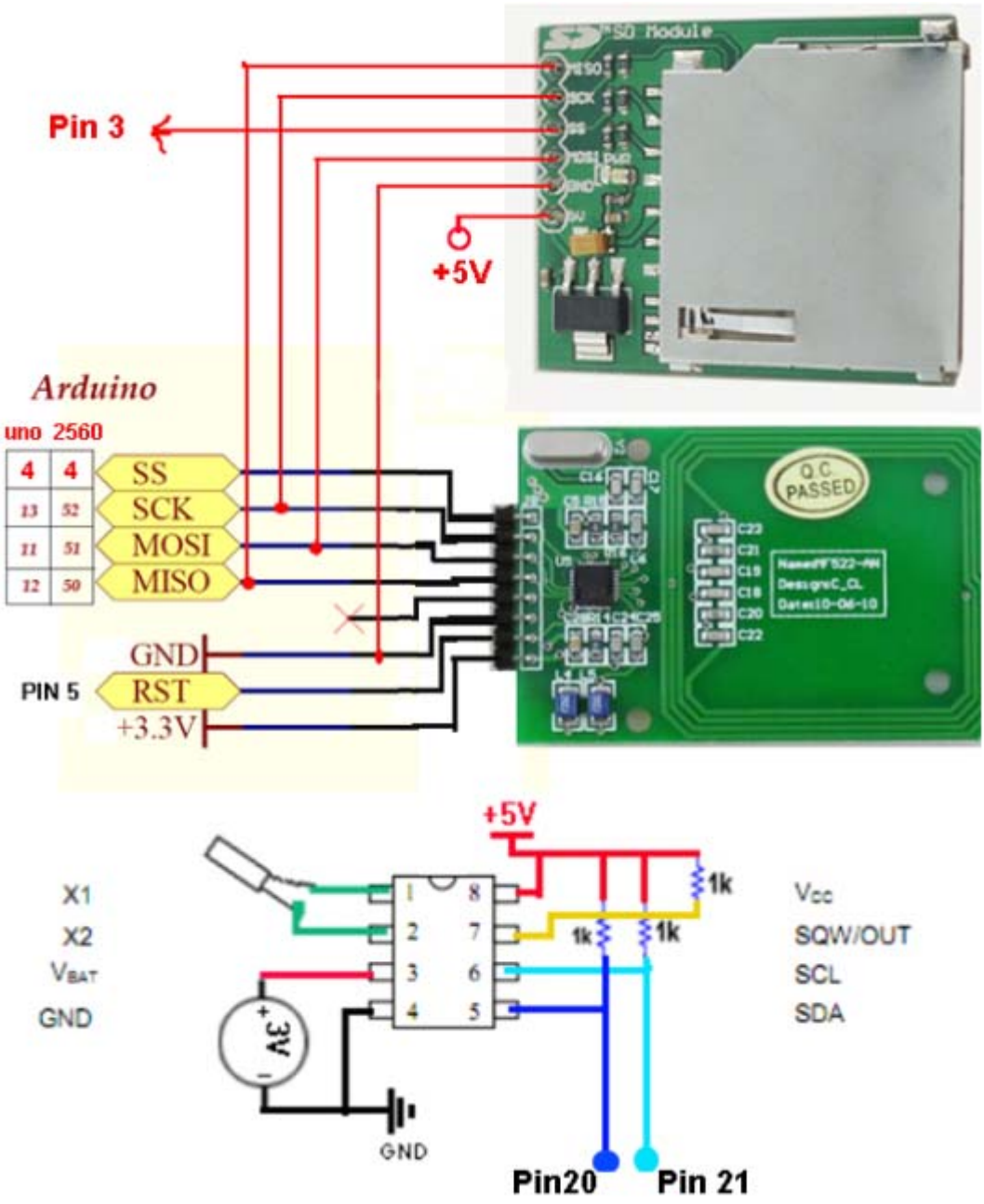
In this project, we will use two SPI module and one I2C module .

What you will need

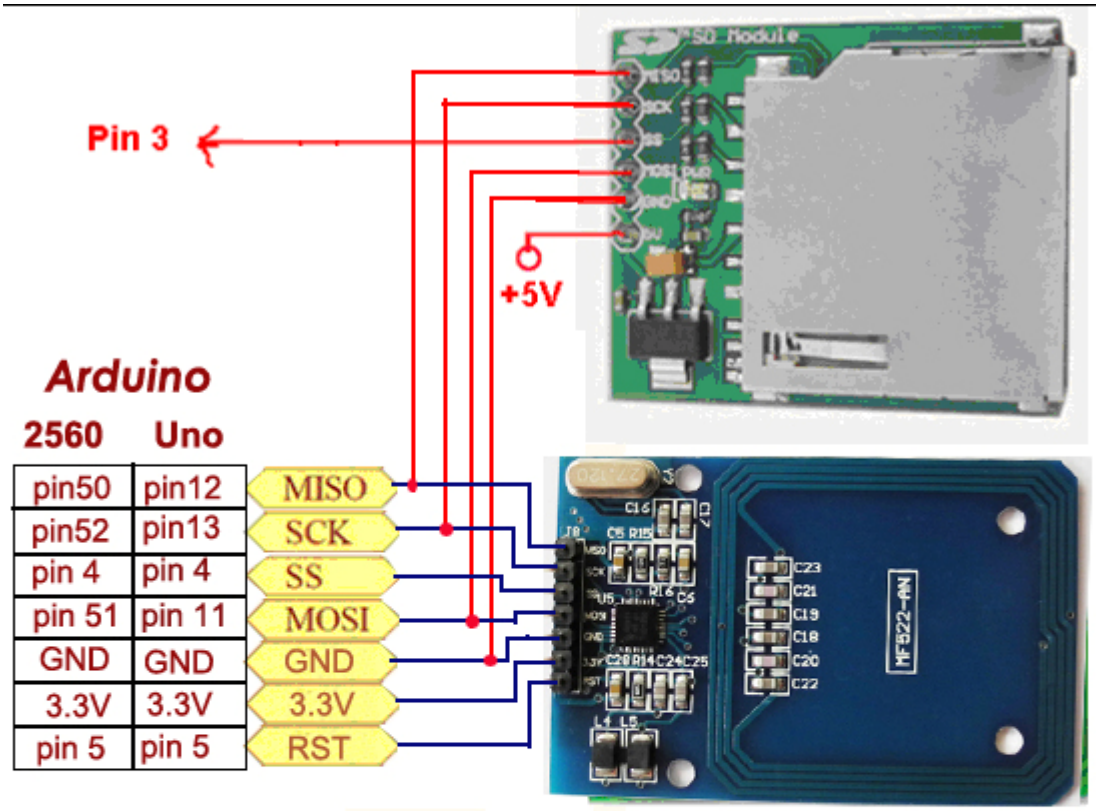
1 X DS1307 RTC chip
1 X 32.768kH quartz crystal
1 X battery socket
1 X CR2032 3V battery
3 X 1 K Ohm Resistors
1 X SD card module
1 X SD card
1 X RFID module .
1 X tag

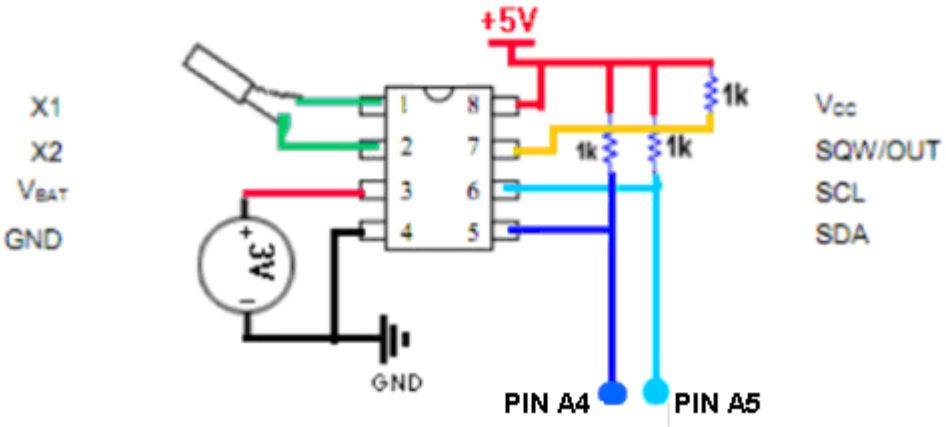
Connect it up





The New version of the RFID module, connect it like this .


Pin 3
+5V
Arduino
2560 Uno
pin50 pin12 MISO
pin52 pin13 SCK
pin 4 pin 4 SS
pin 51 pin 11 MOSI
GND GND GND
3.3V 3.3V 3.3V
pin 5 pin 5 RST
MF522-AN


+5V
X1
X2
VBAT
GND
1k
1k
1k
Vcc
SQW/OUT
SCL
SDA
3V
GND
PIN A4
PIN A5

Code :

```
/*
 * 文 件 名：BigProject.pde
 * 创 建 者：Evan    ( WWW.B2CQSHOP.COM )
 * 创建日期：2011.09.19
 * 修 改 者：
 * 修改日期：
 * 功能描述：Mifare1 寻卡→防冲突→选卡→读写 接口,存数据到卡， 和 SD
 */
 // the sensor communicates using SPI, so include the library:
#include <SPI.h>
#include <SdFat.h>
#include <SdFatUtil.h>
#include <Wire.h>


#define   uchar      unsigned char
#define   uint  unsigned int

//数组最大长度
#define MAX_LEN 16

//DS1307
#define DS1307_I2C_ADDRESS 0x68

////////////////////////////////////////////////////////////////////
//set the pin
////////////////////////////////////////////////////////////////////
//const int chipSelectPin = 4;
const int selectSd = 3;
const int selectRfid = 4;
const int NRSTPD = 5;

//MF522 命令字
#define PCD_IDLE              0x00               //NO action;取消当前命令
#define PCD_AUTHENT           0x0E               //验证密钥
#define PCD_RECEIVE           0x08               //接收数据
#define PCD_TRANSMIT          0x04               //发送数据
#define PCD_TRANSCEIVE        0x0C               //发送并接收数据
#define PCD_RESETPHASE        0x0F               //复位
#define PCD_CALCCRC           0x03               //CRC 计算
```

```
//Mifare_One 卡片命令字
#define PICC_REQIDL          0x26          //寻天线区内未进入休眠状态
#define PICC_REQALL          0x52          //寻天线区内全部卡
#define PICC_ANTICOLL        0x93          //防冲撞
#define PICC_SElECTTAG       0x93          //选卡
#define PICC_AUTHENT1A       0x60          //验证 A 密钥
#define PICC_AUTHENT1B       0x61          //验证 B 密钥
#define PICC_READ            0x30          //读块
#define PICC_WRITE           0xA0          //写块
#define PICC_DECREMENT       0xC0          //扣款
#define PICC_INCREMENT       0xC1          //充值
#define PICC_RESTORE         0xC2          //调块数据到缓冲区
#define PICC_TRANSFER        0xB0          //保存缓冲区中数据
#define PICC_HALT            0x50          //休眠


//和 MF522 通讯时返回的错误代码
#define MI_OK                0
#define MI_NOTAGERR          1
#define MI_ERR               2


//------------------MFRC522 寄存器---------------
//Page 0:Command and Status
#define     Reserved00          0x00
#define     CommandReg          0x01
#define     CommIEnReg          0x02
#define     DivlEnReg           0x03
#define     CommIrqReg          0x04
#define     DivIrqReg           0x05
#define     ErrorReg            0x06
#define     Status1Reg          0x07
#define     Status2Reg          0x08
#define     FIFODataReg         0x09
#define     FIFOLevelReg        0x0A
#define     WaterLevelReg       0x0B
#define     ControlReg          0x0C
#define     BitFramingReg       0x0D
#define     CollReg             0x0E
#define     Reserved01          0x0F
//Page 1:Command
#define     Reserved10          0x10
#define     ModeReg             0x11
```

```
#define     TxModeReg           0x12
#define     RxModeReg           0x13
#define     TxControlReg        0x14
#define     TxAutoReg           0x15
#define     TxSelReg            0x16
#define     RxSelReg            0x17
#define     RxThresholdReg      0x18
#define     DemodReg            0x19
#define     Reserved11          0x1A
#define     Reserved12          0x1B
#define     MifareReg           0x1C
#define     Reserved13          0x1D
#define     Reserved14          0x1E
#define     SerialSpeedReg      0x1F
//Page 2:CFG
#define     Reserved20          0x20
#define     CRCResultRegM       0x21
#define     CRCResultRegL       0x22
#define     Reserved21          0x23
#define     ModWidthReg         0x24
#define     Reserved22          0x25
#define     RFCfgReg            0x26
#define     GsNReg              0x27
#define     CWGsPReg            0x28
#define     ModGsPReg           0x29
#define     TModeReg            0x2A
#define     TPrescalerReg       0x2B
#define     TReloadRegH         0x2C
#define     TReloadRegL         0x2D
#define     TCounterValueRegH   0x2E
#define     TCounterValueRegL   0x2F
//Page 3:TestRegister
#define     Reserved30          0x30
#define     TestSel1Reg         0x31
#define     TestSel2Reg         0x32
#define     TestPinEnReg        0x33
#define     TestPinValueReg     0x34
#define     TestBusReg          0x35
#define     AutoTestReg         0x36
#define     VersionReg          0x37
#define     AnalogTestReg       0x38
#define     TestDAC1Reg         0x39
#define     TestDAC2Reg         0x3A
```

```
#define     TestADCReg          0x3B
#define     Reserved31          0x3C
#define     Reserved32          0x3D
#define     Reserved33          0x3E
#define     Reserved34          0x3F

// store error strings in flash to save RAM
#define error(s) Error_P(PSTR(s))

// for RTC work , DS1307
byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;

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

//4 字节卡序列号，第 5 字节为校验字节
uchar serNum[5];

uchar   writeData[16]={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 100};  //初始化  100 元钱
uchar   moneyConsume = 18 ;  //消费 18 元
uchar   moneyAdd = 10 ;  //充值 10 元


//  保存从 RFID 读出的数据
  uchar str_tem[MAX_LEN];
  uchar str_name[MAX_LEN];
  char RTC_time[] ="22:44:15 08/10/2011";
// initialise variables
// first four required for SD card work
    Sd2Card card;
    SdVolume volume;
    SdFile root;
    SdFile file;



void setup() {
    pinMode(selectSd,OUTPUT);                   // Set digital pin 3 as OUTPUT to connect it
to the RFID /ENABLE pin
    pinMode(selectRfid,OUTPUT);                 // Set digital pin 4 as OUTPUT to connect it to
the RFID /ENABLE pin
    pinMode(NRSTPD,OUTPUT);                     // Set digital pin 5 , Not Reset and
Power-down
```

```
  Serial.begin(9600); // for debugging
  Serial.flush(); // need to flush serial buffer, otherwise first read from reset/power on may not be
correct


    digitalWrite(selectSd, HIGH);
    digitalWrite(selectRfid, HIGH);
    digitalWrite(NRSTPD, HIGH);

// start the SPI library:
  SPI.begin();
  // initialise i2c bus for DS1307 RTC
  Wire.begin();
   //sd
  Serial.println();
  Serial.println("Type any character to start");
  while (!Serial.available());

  SelectOne();
  SD_Init();
       // write header
  file.writeError = 0;
  SelectOne();

       // The header of the file
    writeString(file, " Date");
    writeString(file, ", Card ID");
    writeString(file, ", Name ");
    writeString(file, ", Money Pay ");
    writeString(file, ", Money Add ");
    writeString(file, ", Balance ");
    writeCRLF(file);

  SelectOne();
  MFRC522_Init();

  SelectOne();

    // initial time values
  second = 0;
  minute = 59;
  hour = 22;
  dayOfWeek = 7;
```

```
  dayOfMonth = 9;
  month = 10;
  year = 11;
  //setDateDs1307(second, minute, hour, dayOfWeek, dayOfMonth, month, year);
}

void loop()
{
      uchar i,tmp;
      uchar status;
            uchar RC_size;
            uchar blockAddr;   //选择操作的块地址 0～63
            char data1;
            uchar    vipName[16] ={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
            //扇区 A 密码，16 个扇区，每个扇区密码 6Byte
 uchar sectorKeyA[16][16] = {{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                                          {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},//when you try it again,
please change it into " 0x19, 0x84, 0x07, 0x15, 0x76, 0x14 "
                                          {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                                         };
 uchar sectorNewKeyA[16][16] = {{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                                             {0xFF,      0xFF,      0xFF,      0xFF,      0xFF,      0xFF,
0xFF,0x07,0x80,0x69, 0x19,0x84,0x07,0x15,0x76,0x14},
                                             {0x19,      0x33,      0x07,      0x15,      0x34,      0x14,
0xFF,0x07,0x80,0x69, 0x19,0x33,0x07,0x15,0x34,0x14},
                                            };

        Serial.println("Please select the funtion :");
        Serial.println("      #1 : search the RFID tags");
        Serial.println("      #2 : Register A tags, and made it as your VIP member, and write to
buffer");
        Serial.println("      #3 : Customer Pay $18 for items, and write to buffer");
        Serial.println("      #4 : Customer Add $10 to your card, and write to buffer");
        Serial.println("      #5 : Write all of the information to SD CARD, and Exit");

         Serial.flush(); // need to flush serial buffer, otherwise first read from reset/power on may
not be correct

        while(!Serial.available()); // if a read has been attempted
              // read the incoming number on serial RX
            data1=Serial.read();

          switch(data1){
```

```
            case '1':
            {
              Serial.println("Funtion #1 :     ");
               SelectOne();
               //寻卡，返回卡类型
               status = MFRC522_Request(PICC_REQIDL, str_tem);
               if (status == MI_OK)
               {
                        Serial.println("Find out a card ");
                    Serial.print(str_tem[0],HEX);
                        Serial.print(" , ");
                    Serial.print(str_tem[1],HEX);
                        Serial.println(" ");
               }

               //防冲撞，返回卡的序列号  4 字节
               status = MFRC522_Anticoll(str_tem);
               memcpy(serNum, str_tem, 5);
               if (status == MI_OK)
               {
                        Serial.println("The card's number is    : ");
                    Serial.print(bcdToDec(serNum[0]),DEC);
                    Serial.print(bcdToDec(serNum[1]),DEC);
                    Serial.print(bcdToDec(serNum[2]),DEC);
                    Serial.print(bcdToDec(serNum[3]),DEC);
                    Serial.print(bcdToDec(serNum[4]),DEC);
                        Serial.println(" ");
               }
                    //选卡，返回卡容量
               RC_size = MFRC522_SelectTag(serNum);
               if (RC_size != 0)
                    {
                        Serial.print("The size of the card is    :     ");
                        Serial.print(RC_size,DEC);
                        Serial.println(" K ");
                    }

              SelectOne();
            //读卡
            blockAddr = 7;              //数据块 7
            status              =              MFRC522_Auth(PICC_AUTHENT1A,              blockAddr,
sectorNewKeyA[blockAddr/4], serNum);     //认证
            if (status == MI_OK)
```

```
        {
            //读数据
                                blockAddr = blockAddr - 3 ;
                                status = MFRC522_Read(blockAddr, str_tem);
            if (status == MI_OK)
            {
                                        Serial.println("Read from the card ,the data is : ");
                for (i=0; i<16; i++)
                {
                                Serial.print(str_tem[i],DEC);
                                        Serial.print(" , ");
                }
                                        Serial.println(" ");
            }
                                status = MFRC522_Read(5, str_name);
                                if(status == MI_OK)
                                {
                                        Serial.println("Read from the card ,the Name is : ");
                for (i=0; i<16; i++)
                {
                                   Serial.print((char)str_name[i]);
                }
                                        Serial.println(" ");
                                }
        }
                else
                {
                        Serial.println("You are not our VIP Member, please register it");
                }
         }
         break;

         case '2':
         {
                //注册卡
                Serial.println("Funtion #2 :       ");
                int index = 0 ;
                SelectOne();

        blockAddr = 7;              //数据块 7
        status = MFRC522_Auth(PICC_AUTHENT1A, blockAddr, sectorKeyA[blockAddr/4],
serNum); //认证
        if (status == MI_OK)
```

```
        {
            //写数据
            status = MFRC522_Write(blockAddr, sectorNewKeyA[blockAddr/4]);
                        Serial.print("set the new card password, and can modify the data of the Sector ");
                        Serial.print(blockAddr/4,DEC);
                        Serial.println(" : ");
            for (i=0; i<6; i++)
                {
                        Serial.print(sectorNewKeyA[blockAddr/4][i],HEX);
                            Serial.print(" , ");
                }
                        Serial.println(" ");
                        blockAddr = blockAddr - 3 ;
                        status = MFRC522_Write(blockAddr, writeData);
                        if(status == MI_OK)
                        {
                            Serial.println("You are B2CQSHOP VIP Member, The card has $100 !");
                        }

                        Serial.println("Please give us your name, (the name should be less than 16 chars) ");

                       while(!Serial.available()); // if a read has been attempted
                          // read the incoming number on serial RX
                          // Needed to allow time for the data to come in from the serial buffer.
                         delay(100);
                         i = Serial.available();
                         if (i>15){
                            i = 15;
                         }
                         while(i--)
                         {
                            vipName[index++] = Serial.read();
                         }
                        status = MFRC522_Write(5, vipName);
                        if(status == MI_OK)
                        {
                                    Serial.println("Your name is : ");
                                    for (i=0; i<16; i++)
                           {
```

```
                    Serial.print((char)vipName[i]);
                }
                    Serial.println(" ");

                SelectOne();

                readRTC(RTC_time);
                Serial.print("Now, The time is    :    ");
                  for(i=0;i<20;i++)
                   {
                     Serial.print(RTC_time[i]);
                   }
                Serial.println(" ");

               // write to the SD Card
               writeString(file,RTC_time); //write the time
               writeString(file, ",");
               for(i=0;i<5;i++)
               {
                  writeNumber(file, bcdToDec(serNum[i]));
               }
               writeString(file, ",");
               writeID(file, vipName);
               writeString(file, ", ");
               writeString(file, ", $100 ");
               writeString(file, ", $100 ");
               writeCRLF(file);
            }
        }
          else{
            Serial.println("You have been our VIP member!");
          }
         }
         break;

         case '3':
         {
              Serial.println("Funtion #3 :       ");
              SelectOne();
            //消费
        blockAddr = 7;              //数据块 7
        status              =              MFRC522_Auth(PICC_AUTHENT1A,              blockAddr,
sectorNewKeyA[blockAddr/4], serNum);    //认证
```

```
if (status == MI_OK)
{
    //读数据
                    blockAddr = blockAddr - 3 ;
    status = MFRC522_Read(blockAddr, str_tem);
    if (status == MI_OK)
    {
                    if( str_tem[15] < moneyConsume )
                    {
                        Serial.println(" The money is not enough !");
                    }
                    else
                    {
                        str_tem[15] = str_tem[15] - moneyConsume;
                        status = MFRC522_Write(blockAddr, str_tem);
                        if(status == MI_OK)
                        {
                            Serial.print("You pay $18 for items in
B2CQSHOP.COM . Now, Your money balance is :  $");
                    Serial.print(str_tem[15],DEC);
                        Serial.println(" ");
                        //the VIP member's name
                        MFRC522_Read(5, str_name);
                    SelectOne();
                     readRTC(RTC_time);
                     Serial.print("Now, The time is  :  ");
                        for(i=0;i<20;i++)
                          {
                            Serial.print(RTC_time[i]);
                          }
                     Serial.println(" ");

                    // write to the SD Card
                    writeString(file,RTC_time); //write the time
                      writeString(file, ",");
                      for(i=0;i<5;i++)
                      {
                        writeNumber(file, bcdToDec(serNum[i]));
                      }
                      writeString(file, ",");
                      writeID(file,  str_name);
                      writeString(file, ", $18 ");
                      writeString(file, ", ");
```

```
                                writeString(file, '', $");
                                writeNumber(file, str_tem[15]);
                                writeCRLF(file);
                            }
                        }
                    }
                }

                }
                break;
                case '4':
                {
                    Serial.println("Funtion  #4 :    ");
                    SelectOne();
                      //充值
                blockAddr = 7;          //数据块 7
                status = MFRC522_Auth(PICC_AUTHENT1A, blockAddr, sectorNewKeyA[blockAddr/4], serNum);   //认证
                if (status == MI_OK)
                {
                    //读数据
                            blockAddr = blockAddr - 3 ;
                    status = MFRC522_Read(blockAddr, str_tem);
                    if (status == MI_OK)
                    {
                            tmp = (int)(str_tem[15] + moneyAdd) ;
                            //Serial.println(tmp,DEC);
                            if( tmp < (char)254 )
                            {
                                Serial.println(" The money of the card can not be more than 255 !");
                            }
                            else
                            {
                                str_tem[15] = str_tem[15] + moneyAdd ;
                                status = MFRC522_Write(blockAddr, str_tem);
                                if(status == MI_OK)
                                {
                                    Serial.print("You  add  $10  to  your  card  in B2CQSHOP.COM , Your money balance is :  $");
                                Serial.print(str_tem[15],DEC);
                                    Serial.println(" ");
                                    //the VIP member's name
```

```
                                MFRC522_Read(5, str_name);

                                  SelectOne();

                                readRTC(RTC_time);
                                Serial.print("Now, The time is    :    ");
                                    for(i=0;i<20;i++)
                                     {
                                        Serial.print(RTC_time[i]);
                                     }
                                Serial.println(" ");

                               // write to the SD Card
                               writeString(file,RTC_time); //write the time
                                  writeString(file, ",");
                                  for(i=0;i<5;i++)
                                  {
                                     writeNumber(file, bcdToDec(serNum[i]));
                                  }
                                  writeString(file, ",");
                                  writeID(file, str_name);
                                  writeString(file, ",    ");
                                  writeString(file, ", $10");
                                  writeString(file, ", $");
                                  writeNumber(file, str_tem[15]);
                                  writeCRLF(file);
                               }
                           }
      }
}
 }
 break;

 case '5':
 {
      Serial.println("Funtion    #5 :       ");
      if (file.writeError) {
          error("The file is Close");
          break;
       }
       // close file and force write of all data to the SD card
       file.close();
       Serial.println("write to SD card , Done");
```

```
                SelectOne();
                MFRC522_Halt();                 //命令卡片进入休眠状态
            }
            break;

            default:
            break;
        }


}

/*
 * 函 数 名：Error_P
 * SD ERROR Message .
 * 返 回 值：无
 */
void Error_P(const char *str)
{
  PgmPrint("error: ");
  SerialPrintln_P(str);
  if (card.errorCode()) {
    PgmPrint("SD error: ");
    Serial.print(card.errorCode(), HEX);
    Serial.print(',');
    Serial.println(card.errorData(), HEX);
  }
  while(1);
}

//set the sd card or RFID module to active at one time
void SelectOne()
{
   digitalWrite(selectSd, HIGH);
   digitalWrite(selectRfid, HIGH);
}

// initialize the SD card reader to read and write
void SD_Init()
{
    // initialize the SD card at SPI_HALF_SPEED to avoid bus errors with breadboards.
    //   use SPI_FULL_SPEED for better performance.
```

```
  if (!card.init(SPI_HALF_SPEED,selectSd)) error("card.init failed"); //set the selectSd as SS , so that you select the right pin to SD

  // initialize a FAT volume
  if (!volume.init(&card)) error("volume.init failed");

  // open the root directory
  if (!root.openRoot(&volume)) error("openRoot failed");

  // create a new file
  char name[] = "LOGGER00.CSV";

    for (uint8_t i = 0; i < 100; i++) {
    name[6] = i/10 + '0';
    name[7] = i%10 + '0';
    if (file.open(&root, name, O_CREAT | O_EXCL | O_WRITE)) break;
  }
  Serial.print("Logging to: ");
  Serial.println(name);

}

/*
 * Write CR LF to a file
 */
void writeCRLF(SdFile& f) {
  f.write((uint8_t*)"\r\n", 2);
}
/*
 * Write an unsigned number to a file
 */
void writeNumber(SdFile& f, uint32_t n) {
  uint8_t buf[10];
  uint8_t i = 0;
  do {
    i++;
    buf[sizeof(buf) - i] = n%10 + '0';
    n /= 10;
  } while (n);
  f.write(&buf[sizeof(buf) - i], i);
}
/*
 * Write a string to a file
```

```
 */
void writeID(SdFile& f, unsigned char *str) {
   uint8_t n;
   for (n = 0; str[n]; n++);
   f.write((uint8_t *)str, n);
}
/*
 * Write a string to a file
 */
void writeString(SdFile& f, char *str) {
   uint8_t n;
   for (n = 0; str[n]; n++);
   f.write((uint8_t *)str, n);
}
// Convert normal decimal numbers to binary coded decimal
byte decToBcd(byte val)
{
   return ( (val/10*16) + (val%10) );
}
// Convert binary coded decimal to normal decimal numbers
byte bcdToDec(byte val)
{
   return ( (val/16*10) + (val%16) );
}

void setDateDs1307( byte second,          // 0-59
                    byte minute,          // 0-59
                    byte hour,            // 1-23
                    byte dayOfWeek,       // 1-7
                    byte dayOfMonth,      // 1-28/29/30/31
                    byte month,           // 1-12
                    byte year)            // 0-99
{
   Wire.beginTransmission(DS1307_I2C_ADDRESS);
   Wire.send(0);
   Wire.send(decToBcd(second));      // 0 to bit 7 starts the clock
   Wire.send(decToBcd(minute));
   Wire.send(decToBcd(hour));        // If you want 12 hour am/pm you need to set
   // bit 6 (also need to change readDateDs1307)
   Wire.send(decToBcd(dayOfWeek));
   Wire.send(decToBcd(dayOfMonth));
   Wire.send(decToBcd(month));
   Wire.send(decToBcd(year));
```

```
  Wire.endTransmission();
}

// Gets the date and time from the ds1307
void getDateDs1307( byte *second,
                    byte *minute,
                    byte *hour,
                    byte *dayOfWeek,
                    byte *dayOfMonth,
                    byte *month,
                    byte *year)
{
  // Reset the register pointer
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.send(0);
  Wire.endTransmission();

  Wire.requestFrom(DS1307_I2C_ADDRESS, 7);

  // A few of these need masks because certain bits are control bits
  *second     = bcdToDec(Wire.receive() & 0x7f);
  *minute     = bcdToDec(Wire.receive());
  *hour       = bcdToDec(Wire.receive() & 0x3f);  // Need to change this if 12 hour am/pm
  *dayOfWeek  = bcdToDec(Wire.receive());
  *dayOfMonth = bcdToDec(Wire.receive());
  *month      = bcdToDec(Wire.receive());
  *year       = bcdToDec(Wire.receive());
}

void readRTC(char *RTC_time)
{
  //unsigned char RTC_time[] ="22:44:15 08/10/2011";
  getDateDs1307(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year);
  RTC_time[0] = hour/10+'0';
  RTC_time[1] =  hour%10+'0';
  //RTC_time[2] = ':';
  RTC_time[3] =  minute/10+'0';
  RTC_time[4] =  minute%10+'0';
  //RTC_time[5] = ':';
  RTC_time[6] =  second/10+'0';
  RTC_time[7] =  second%10+'0';
  //RTC_time[8] = ' ';
  RTC_time[9] =  dayOfMonth/10+'0';
```

```
    RTC_time[10] =   dayOfMonth%10+'0';
    //RTC_time[11] = '/';
    RTC_time[12] =   month/10+'0';
    RTC_time[13] =   month%10+'0';
    //RTC_time[14] = '/';
    //RTC_time[15] = '/';
    //RTC_time[16] = '/';
    RTC_time[17] =   year/10+'0';
    RTC_time[18] =   year%10+'0';
}

/*
 * 函 数 名：Write_MFRC5200
 * 功能描述：向 MFRC522 的某一寄存器写一个字节数据
 * 输入参数：addr--寄存器地址；val--要写入的值
 * 返 回 值：无
 */
void Write_MFRC522(uchar addr, uchar val)
{
    digitalWrite(selectRfid, LOW);

    //地址格式：0XXXXXX0
    SPI.transfer((addr<<1)&0x7E);
    SPI.transfer(val);

    digitalWrite(selectRfid, HIGH);
}


/*
 * 函 数 名：Read_MFRC522
 * 功能描述：从 MFRC522 的某一寄存器读一个字节数据
 * 输入参数：addr--寄存器地址
 * 返 回 值：返回读取到的一个字节数据
 */
uchar Read_MFRC522(uchar addr)
{
    uchar val;

    digitalWrite(selectRfid, LOW);

    //地址格式：1XXXXXX0
    SPI.transfer(((addr<<1)&0x7E) | 0x80);
```

```
    val =SPI.transfer(0x00);

    digitalWrite(selectRfid, HIGH);

    return val;
}

/*
 * 函 数 名：SetBitMask
 * 功能描述：置 RC522 寄存器位
 * 输入参数：reg--寄存器地址;mask--置位值
 * 返 回 值：无
 */
void SetBitMask(uchar reg, uchar mask)
{
    uchar tmp;
    tmp = Read_MFRC522(reg);
    Write_MFRC522(reg, tmp | mask);    // set bit mask
}


/*
 * 函 数 名：ClearBitMask
 * 功能描述：清 RC522 寄存器位
 * 输入参数：reg--寄存器地址;mask--清位值
 * 返 回 值：无
 */
void ClearBitMask(uchar reg, uchar mask)
{
    uchar tmp;
    tmp = Read_MFRC522(reg);
    Write_MFRC522(reg, tmp & (~mask));    // clear bit mask
}


/*
 * 函 数 名：AntennaOn
 * 功能描述：开启天线,每次启动或关闭天险发射之间应至少有 1ms 的间隔
 * 输入参数：无
 * 返 回 值：无
 */
void AntennaOn(void)
{
```

```
    uchar temp;

    temp = Read_MFRC522(TxControlReg);
    if (!(temp & 0x03))
    {
        SetBitMask(TxControlReg, 0x03);
    }
}


/*
 * 函 数 名：AntennaOff
 * 功能描述：关闭天线,每次启动或关闭天险发射之间应至少有 1ms 的间隔
 * 输入参数：无
 * 返 回 值：无
 */
void AntennaOff(void)
{
    ClearBitMask(TxControlReg, 0x03);
}


/*
 * 函 数 名：ResetMFRC522
 * 功能描述：复位 RC522
 * 输入参数：无
 * 返 回 值：无
 */
void MFRC522_Reset(void)
{
    Write_MFRC522(CommandReg, PCD_RESETPHASE);
}


/*
 * 函 数 名：InitMFRC522
 * 功能描述：初始化 RC522
 * 输入参数：无
 * 返 回 值：无
 */
void MFRC522_Init(void)
{
    digitalWrite(NRSTPD,HIGH);
```

```
    MFRC522_Reset();

    //Timer: TPrescaler*TreloadVal/6.78MHz = 24ms
    Write_MFRC522(TModeReg, 0x8D);          //Tauto=1; f(Timer) = 6.78MHz/TPreScaler
    Write_MFRC522(TPrescalerReg, 0x3E);     //TModeReg[3..0] + TPrescalerReg
    Write_MFRC522(TReloadRegL, 30);
    Write_MFRC522(TReloadRegH, 0);

    Write_MFRC522(TxAutoReg, 0x40);         //100%ASK
    Write_MFRC522(ModeReg, 0x3D);           //CRC 初始值 0x6363   ???

    //ClearBitMask(Status2Reg, 0x08);           //MFCrypto1On=0
    //Write_MFRC522(RxSelReg, 0x86);            //RxWait = RxSelReg[5..0]
    //Write_MFRC522(RFCfgReg, 0x7F);                //RxGain = 48dB

    AntennaOn();        //打开天线
}


/*
 * 函 数 名：MFRC522_Request
 * 功能描述：寻卡，读取卡类型号
 * 输入参数：reqMode--寻卡方式，
 *               TagType--返回卡片类型
 *                    0x4400 = Mifare_UltraLight
 *                    0x0400 = Mifare_One(S50)
 *                    0x0200 = Mifare_One(S70)
 *                    0x0800 = Mifare_Pro(X)
 *                    0x4403 = Mifare_DESFire
 * 返 回 值：成功返回 MI_OK
 */
uchar MFRC522_Request(uchar reqMode, uchar *TagType)
{
    uchar status;
    uint backBits;              //接收到的数据位数

    Write_MFRC522(BitFramingReg, 0x07);         //TxLastBists = BitFramingReg[2..0]    ???

    TagType[0] = reqMode;
    status = MFRC522_ToCard(PCD_TRANSCEIVE, TagType, 1, TagType, &backBits);

    if ((status != MI_OK) || (backBits != 0x10))
```

```
    {
        status = MI_ERR;
    }

    return status;
}


/*
 * 函 数 名：MFRC522_ToCard
 * 功能描述：RC522 和 ISO14443 卡通讯
 * 输入参数：command--MF522 命令字，
 *           sendData--通过 RC522 发送到卡片的数据,
 *           sendLen--发送的数据长度
 *           backData--接收到的卡片返回数据，
 *           backLen--返回数据的位长度
 * 返 回 值：成功返回 MI_OK
 */
uchar MFRC522_ToCard(uchar command, uchar *sendData, uchar sendLen, uchar *backData,
uint *backLen)
{
    uchar status = MI_ERR;
    uchar irqEn = 0x00;
    uchar waitIRq = 0x00;
    uchar lastBits;
    uchar n;
    uint i;

    switch (command)
    {
        case PCD_AUTHENT:       //认证卡密
        {
            irqEn = 0x12;
            waitIRq = 0x10;
            break;
        }
        case PCD_TRANSCEIVE:    //发送 FIFO 中数据
        {
            irqEn = 0x77;
            waitIRq = 0x30;
            break;
        }
        default:
```

```
            break;
    }

    Write_MFRC522(CommIEnReg, irqEn|0x80);     //允许中断请求
    ClearBitMask(CommIrqReg, 0x80);            //清除所有中断请求位
    SetBitMask(FIFOLevelReg, 0x80);            //FlushBuffer=1, FIFO 初始化

    Write_MFRC522(CommandReg, PCD_IDLE);       //NO action;取消当前命令   ???

    //向 FIFO 中写入数据
    for (i=0; i<sendLen; i++)
    {
        Write_MFRC522(FIFODataReg, sendData[i]);
    }

    //执行命令
    Write_MFRC522(CommandReg, command);
    if (command == PCD_TRANSCEIVE)
    {
        SetBitMask(BitFramingReg, 0x80);       //StartSend=1,transmission of data starts
    }

    //等待接收数据完成
    i = 2000; //i 根据时钟频率调整，操作 M1 卡最大等待时间 25ms  ???
    do
    {
        //CommIrqReg[7..0]
        //Set1 TxIRq RxIRq IdleIRq HiAlerIRq LoAlertIRq ErrIRq TimerIRq
        n = Read_MFRC522(CommIrqReg);
        i--;
    }
    while ((i!=0) && !(n&0x01) && !(n&waitIRq));

    ClearBitMask(BitFramingReg, 0x80);         //StartSend=0

    if (i != 0)
    {
        if(!(Read_MFRC522(ErrorReg) & 0x1B))   //BufferOvfl Collerr CRCErr ProtecolErr
        {
            status = MI_OK;
            if (n & irqEn & 0x01)
            {
                status = MI_NOTAGERR;          //??
```

```
            }

            if (command == PCD_TRANSCEIVE)
            {
                n = Read_MFRC522(FIFOLevelReg);
                lastBits = Read_MFRC522(ControlReg) & 0x07;
                if (lastBits)
                {
                    *backLen = (n-1)*8 + lastBits;
                }
                else
                {
                    *backLen = n*8;
                }

                if (n == 0)
                {
                    n = 1;
                }
                if (n > MAX_LEN)
                {
                    n = MAX_LEN;
                }

                //读取 FIFO 中接收到的数据
                for (i=0; i<n; i++)
                {
                    backData[i] = Read_MFRC522(FIFODataReg);
                }
            }
        }
        else
        {
            status = MI_ERR;
        }

    }

    //SetBitMask(ControlReg,0x80);              //timer stops
    //Write_MFRC522(CommandReg, PCD_IDLE);

    return status;
}
```

```
/*
 * 函 数 名：MFRC522_Anticoll
 * 功能描述：防冲突检测，读取选中卡片的卡序列号
 * 输入参数：serNum--返回 4 字节卡序列号,第 5 字节为校验字节
 * 返 回 值：成功返回 MI_OK
 */
uchar MFRC522_Anticoll(uchar *serNum)
{
    uchar status;
    uchar i;
    uchar serNumCheck=0;
    uint unLen;


    //ClearBitMask(Status2Reg, 0x08);          //TempSensclear
    //ClearBitMask(CollReg,0x80);              //ValuesAfterColl
    Write_MFRC522(BitFramingReg, 0x00);        //TxLastBists = BitFramingReg[2..0]

    serNum[0] = PICC_ANTICOLL;
    serNum[1] = 0x20;
    status = MFRC522_ToCard(PCD_TRANSCEIVE, serNum, 2, serNum, &unLen);

    if (status == MI_OK)
    {
        //校验卡序列号
        for (i=0; i<4; i++)
        {
            serNumCheck ^= serNum[i];
        }
        if (serNumCheck != serNum[i])
        {
            status = MI_ERR;
        }
    }

    //SetBitMask(CollReg, 0x80);           //ValuesAfterColl=1

    return status;
}
```

```
/*
 * 函 数 名：CalulateCRC
 * 功能描述：用 MF522 计算 CRC
 * 输入参数：pIndata--要读数 CRC 的数据，len--数据长度，pOutData--计算的 CRC 结果
 * 返 回 值：无
 */
void CalulateCRC(uchar *pIndata, uchar len, uchar *pOutData)
{
    uchar i, n;

    ClearBitMask(DivIrqReg, 0x04);              //CRCIrq = 0
    SetBitMask(FIFOLevelReg, 0x80);                 //清 FIFO 指针
    //Write_MFRC522(CommandReg, PCD_IDLE);

    //向 FIFO 中写入数据
    for (i=0; i<len; i++)
    {
        Write_MFRC522(FIFODataReg, *(pIndata+i));
    }
    Write_MFRC522(CommandReg, PCD_CALCCRC);

    //等待 CRC 计算完成
    i = 0xFF;
    do
    {
        n = Read_MFRC522(DivIrqReg);
        i--;
    }
    while ((i!=0) && !(n&0x04));                //CRCIrq = 1

    //读取 CRC 计算结果
    pOutData[0] = Read_MFRC522(CRCResultRegL);
    pOutData[1] = Read_MFRC522(CRCResultRegM);
}


/*
 * 函 数 名：MFRC522_SelectTag
 * 功能描述：选卡，读取卡存储器容量
 * 输入参数：serNum--传入卡序列号
 * 返 回 值：成功返回卡容量
 */
uchar MFRC522_SelectTag(uchar *serNum)
```

```
{
    uchar i;
    uchar status;
    uchar size;
    uint recvBits;
    uchar buffer[9];

    //ClearBitMask(Status2Reg, 0x08);                    //MFCrypto1On=0

    buffer[0] = PICC_SElECTTAG;
    buffer[1] = 0x70;
    for (i=0; i<5; i++)
    {
        buffer[i+2] = *(serNum+i);
    }
    CalulateCRC(buffer, 7, &buffer[7]);          //??
    status = MFRC522_ToCard(PCD_TRANSCEIVE, buffer, 9, buffer, &recvBits);

    if ((status == MI_OK) && (recvBits == 0x18))
    {
        size = buffer[0];
    }
    else
    {
        size = 0;
    }

    return size;
}


/*
 * 函 数 名：MFRC522_Auth
 * 功能描述：验证卡片密码
 * 输入参数：authMode--密码验证模式
                0x60 = 验证 A 密钥
                0x61 = 验证 B 密钥
              BlockAddr--块地址
              Sectorkey--扇区密码
              serNum--卡片序列号，4 字节
 * 返 回 值：成功返回 MI_OK
 */
uchar MFRC522_Auth(uchar authMode, uchar BlockAddr, uchar *Sectorkey, uchar *serNum)
```

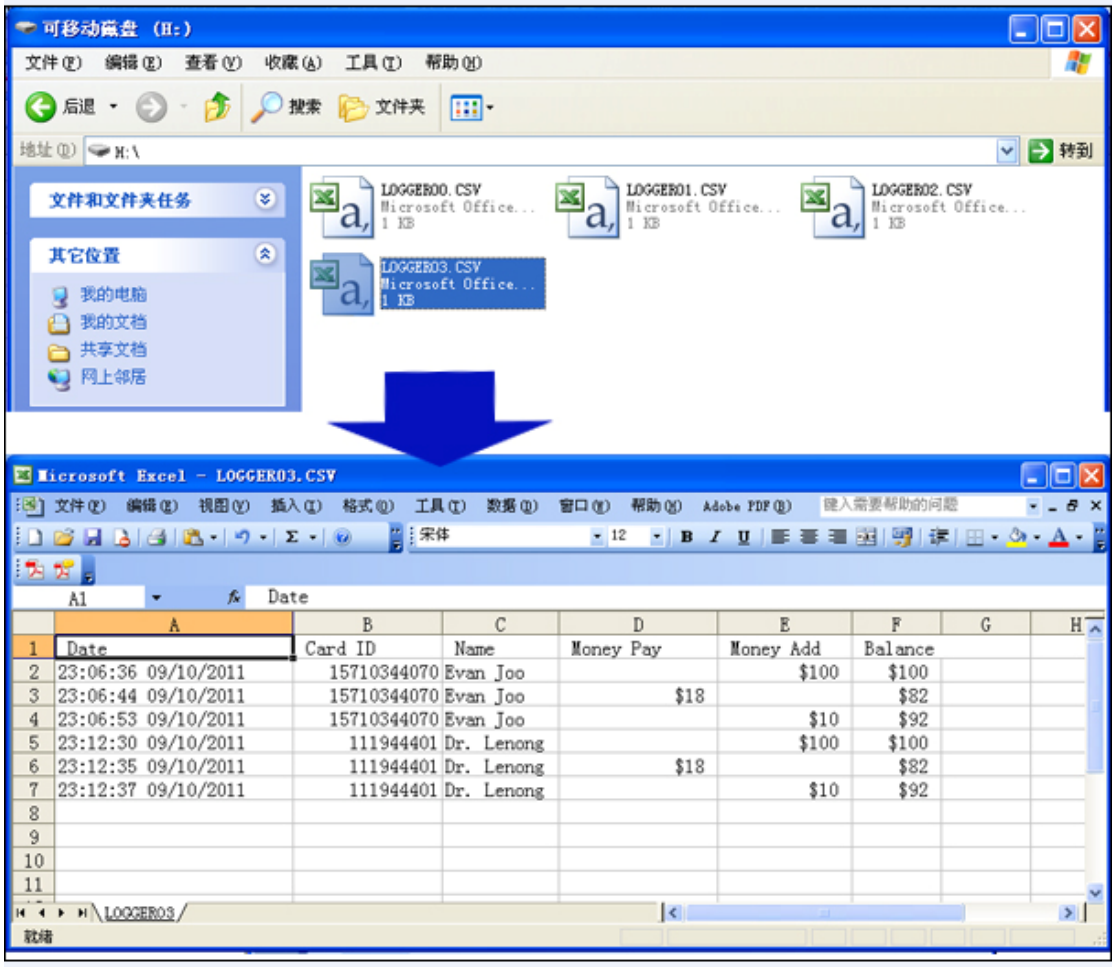
```
{
    uchar status;
    uint recvBits;
    uchar i;
    uchar buff[12];

    //验证指令+块地址＋扇区密码＋卡序列号
    buff[0] = authMode;
    buff[1] = BlockAddr;
    for (i=0; i<6; i++)
    {
        buff[i+2] = *(Sectorkey+i);
    }
    for (i=0; i<4; i++)
    {
        buff[i+8] = *(serNum+i);
    }
    status = MFRC522_ToCard(PCD_AUTHENT, buff, 12, buff, &recvBits);

    if ((status != MI_OK) || (!(Read_MFRC522(Status2Reg) & 0x08)))
    {
        status = MI_ERR;
    }

    return status;
}


/*
 * 函 数 名：MFRC522_Read
 * 功能描述：读块数据
 * 输入参数：blockAddr--块地址;recvData--读出的块数据
 * 返 回 值：成功返回 MI_OK
 */
uchar MFRC522_Read(uchar blockAddr, uchar *recvData)
{
    uchar status;
    uint unLen;

    recvData[0] = PICC_READ;
    recvData[1] = blockAddr;
    CalulateCRC(recvData,2, &recvData[2]);
    status = MFRC522_ToCard(PCD_TRANSCEIVE, recvData, 4, recvData, &unLen);
```

```
    if ((status != MI_OK) || (unLen != 0x90))
    {
        status = MI_ERR;
    }

    return status;
}


/*
 * 函 数 名：MFRC522_Write
 * 功能描述：写块数据
 * 输入参数：blockAddr--块地址;writeData--向块写 16 字节数据
 * 返 回 值：成功返回 MI_OK
 */
uchar MFRC522_Write(uchar blockAddr, uchar *writeData)
{
    uchar status;
    uint recvBits;
    uchar i;
    uchar buff[18];

    buff[0] = PICC_WRITE;
    buff[1] = blockAddr;
    CalulateCRC(buff, 2, &buff[2]);
    status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 4, buff, &recvBits);

    if ((status != MI_OK) || (recvBits != 4) || ((buff[0] & 0x0F) != 0x0A))
    {
        status = MI_ERR;
    }

    if (status == MI_OK)
    {
        for (i=0; i<16; i++)        //向 FIFO 写 16Byte 数据
        {
            buff[i] = *(writeData+i);
        }
        CalulateCRC(buff, 16, &buff[16]);
        status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 18, buff, &recvBits);

        if ((status != MI_OK) || (recvBits != 4) || ((buff[0] & 0x0F) != 0x0A))
```

```
            {
                status = MI_ERR;
            }
        }

        return status;
}


/*
 * 函 数 名：MFRC522_Halt
 * 功能描述：命令卡片进入休眠状态
 * 输入参数：无
 * 返 回 值：无
 */
void MFRC522_Halt(void)
{
        uchar status;
        uint unLen;
        uchar buff[4];

        buff[0] = PICC_HALT;
        buff[1] = 0;
        CalulateCRC(buff, 2, &buff[2]);

        status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 4, buff,&unLen);
}
```

The result of this project :


COM32
Send
Type any character to start
Logging to: LOGGER18.CSV
Please select the funtion :
#1 : search the RFID tags
#2 : Register A tags, and made it as your VIP member, and write to buffer
#3 : Customer Pay $18 for items, and write to buffer
#4 : Customer Add $10 to your card, and write to buffer
#5 : Write all of the information to SD CARD, and Exit
Funtion #1 :
Find out a card
4 , 0
The card's number is :
15710344070
The size of the card is : 8 K
Funtion #2 :
set the new card password, and can modify the data of the Sector 1 :
FF , FF , FF , FF , FF , FF ,
You are B2CQSHOP VIP Member, The card has $100 !
Please give us your name, (the name should be less than 16 chars)
Your name is :
Evan Joo
Now, The time is : 23:06:36 09/10/2011
Funtion #3 :
You pay $18 for items in B2CQSHOP.COM . Now, Your money balance is : $82
Now, The time is : 23:06:44 09/10/2011
Funtion #4 :
You add $10 to your card in B2CQSHOP.COM , Your money balance is : $92
Now, The time is : 23:06:53 09/10/2011
Funtion #1 :
You are not our VIP Member, please register it
Funtion #5 :
write to SD card , Done
Autoscroll
No line ending
9600 baud



	A	B	C	D	E	F	G	H
1	Date	Card ID	Name	Money Pay	Money Add	Balance		
2	23:06:36 09/10/2011	15710344070	Evan Joo		$100	$100		
3	23:06:44 09/10/2011	15710344070	Evan Joo	$18		$82		
4	23:06:53 09/10/2011	15710344070	Evan Joo		$10	$92		
5	23:12:30 09/10/2011	111944401	Dr. Lenong		$100	$100		
6	23:12:35 09/10/2011	111944401	Dr. Lenong	$18		$82		
7	23:12:37 09/10/2011	111944401	Dr. Lenong		$10	$92		
8								
9								
10								
11								