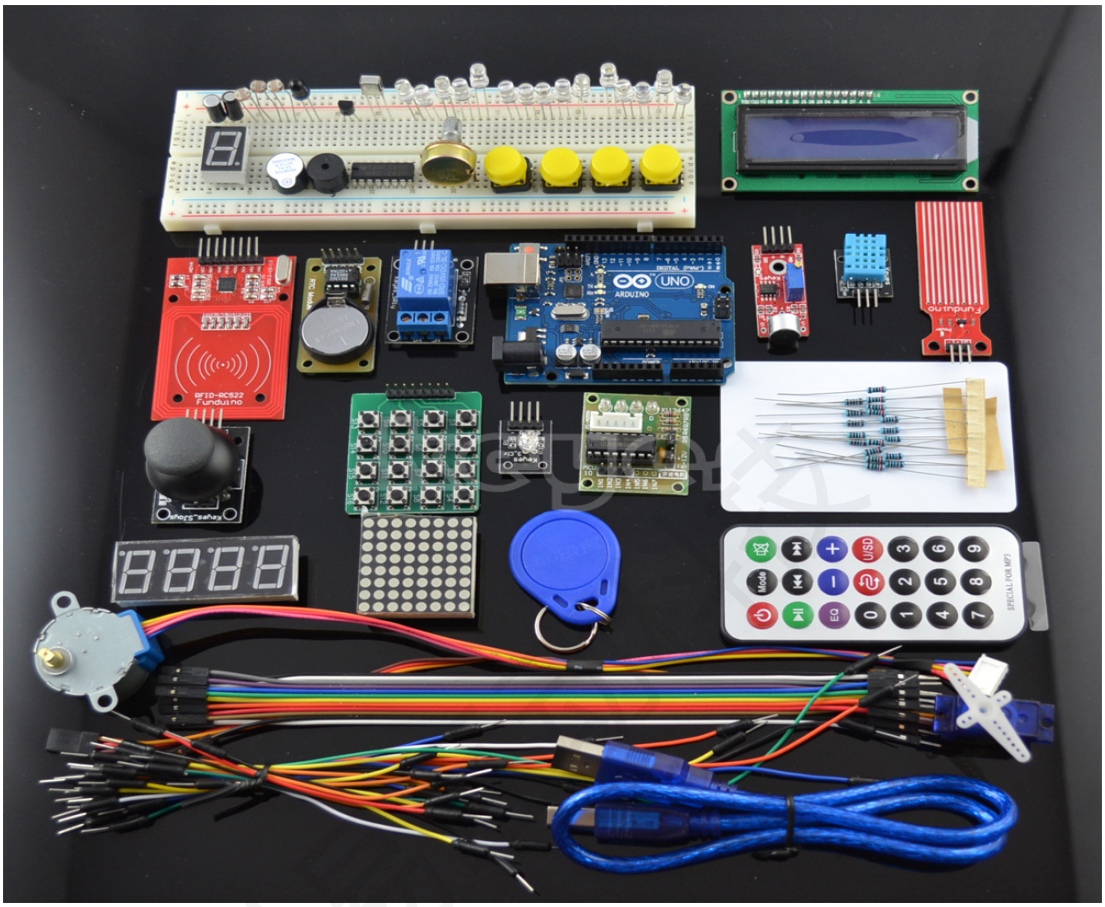
RFID learning kit for Arduino

ARDUINO RFID 学习套件

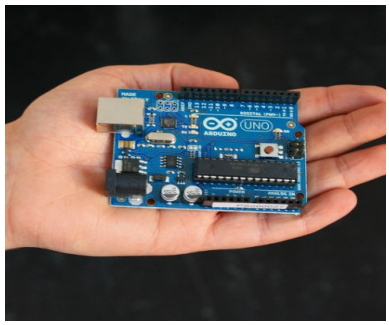


Routine 1, Hello World!

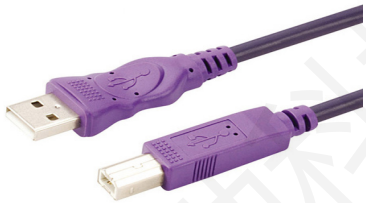
A first-come, first practice no other auxiliary components, only one Arduino and a download cable simple experiment, let's Arduino say "Hello World!", Which is a let Arduino and PC communications experiment, which is an entry test, hoping to lead us into the Arduino world.

This experiment we need to use the experimental hardware are:

Arduino controller



USB download cable



We will be in accordance with the above stated Arduino driver installed, we open the Arduino software, write a program to Arduino's orders received by us on the show "Hello World!" String, of course, you can have Arduino without accepting any instructions directly constantly echo "Hello World!", is actually very simple, a

if () statement can make your Arduino to follow your instructions, and we'll borrow Arduino comes with digital 13 LED, so Arduino LED flashes when it receives the command, and then displays "Hello World!"

The following give us a reference program.

```
int val ;/ / define a variable val
int ledpin = 13 ;/ / define the digital interface 13
void setup ()
{
```

Serial.begin (9600) ;/ / set the baud rate to 9600, where the software settings keep consistent. When access to a specific device (such as: Bluetooth), we also keep other equipment baud rate to reach consensus.

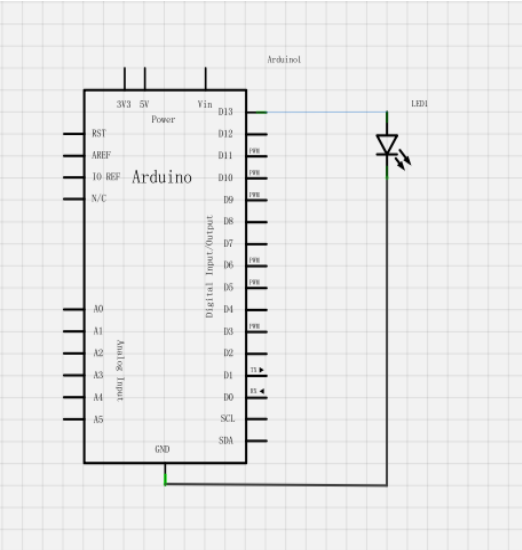
pinMode (ledpin, OUTPUT) ;/ / set the digital output interface 13 is, Arduino, we use the I / O port should be carried out like this definition.

```
}
void loop ()
{
val = Serial.read () ;/ / read the PC sends a command to the Arduino or characters, and the instruction or character assigned val
if (val == 'R') / / determine the received command or character is "R".
{/ / If you receive a "R" character
digitalWrite (ledpin, HIGH) ;/ / lit Digital 13 LED.
delay (500);
digitalWrite (ledpin, LOW) ;/ / Off Digital 13 LED
delay (500);
Serial.println ("Hello World!") ;/ / Displays "Hello World!" String
}
}
```


Routine 2, LED flashing experiment

LED lights are more basic experiment experimental one, this time we use 13 feet comes with LED lights to complete this experiment,

Small light experiment schematics



Good link circuit according to the diagram, you can start writing programs, and we still make LED lights flashing, lit one second off one second. This procedure is very simple and routine that comes with Arduino, like in the Blink

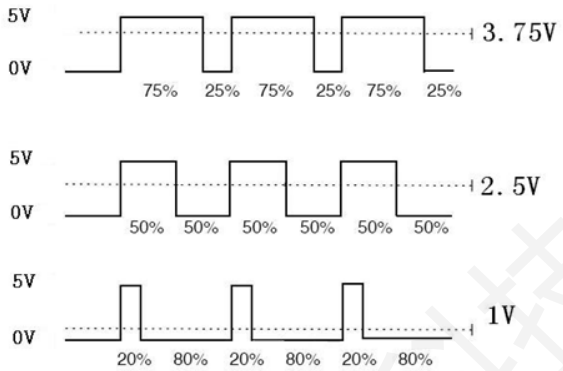
Reference procedure is as follows:

```
int ledPin = 13; / / define the interface number 10
void setup ()
{
pinMode (ledPin, OUTPUT) ;/ / define a small lamp interface output interface
}
void loop ()
{
digitalWrite (ledPin, HIGH); / / lit a small lamp
delay (1000); / / Delay 1 second
digitalWrite (ledPin, LOW); / / extinguish small lights
delay (1000); / / Delay 1 second
}
```

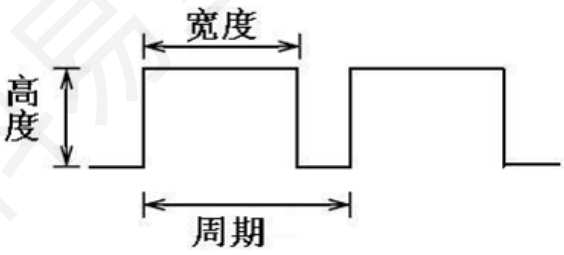
After downloading the program, you can see our 13 external small lights flashing, so our little lights flashing experiment
Is complete.

Routine 3, PWM control light levels experiments

Pulse Width Modulation is commonly referred to PWM, translated as pulse width modulation, referred to as pulse width modulation. Pulse width modulation (PWM) is an analog signal level for digital coding method, because the computer can not output the analog voltage, only output 0 or 5V digital voltage values, the counters through the use of high-resolution, the usage-side wave is modulated by the modulation method to a specific analog signal level to be encoded. Digital PWM signal is still, because in any given moment, the full DC supply is either a 5V (ON), is either 0V (OFF). Voltage or current source is one kind of pass (ON) or off (OFF) repeating pulse sequence is applied to analog load. Pass the time that the DC supply is applied to the load when the off-time is when the supply is switched off. Sufficient bandwidth, any analog value can be encoded with PWM. The voltage value is set by the output on and off time for calculations. Output Voltage = (ON time / pulse time) * Maximum voltage value

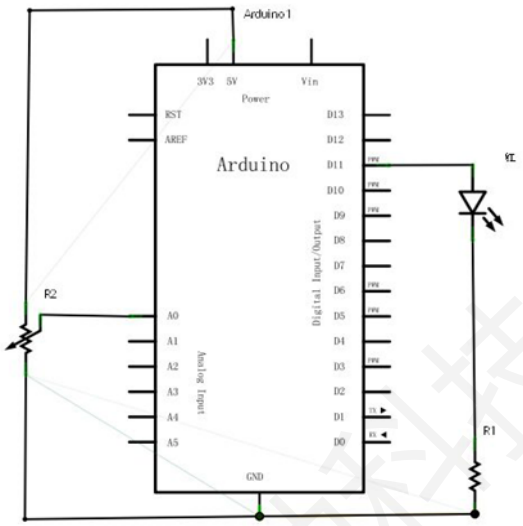


PWM is used in many places, dimming lights, motor speed, sound production and so on.
Here are some of the PWM three basic parameters:



1, the pulse width variation amplitude (min / max)
2, the pulse period (1 second inverse number of the pulse frequency)
3, the voltage height (eg: 0V-5V)
Arduino controller has six PWM interface are digital interface 3,5,6,9,10,11, before we have done a small lamp button control experiments, it is a digital interface digital signal control experiments, we have done potentiometer experiment, this time we have to complete a small lamp potentiometer control experiments.

Required components are:
Potentiometer module * 1
Red M5 DIP LED * 1
220Ω line resistance
Bread board * 1
Breadboard Jumper * a tie
Potentiometer input analog value is the analog port we received, we received a small lamp PWM interface, so that by generating different PWM signal can make a small lamp with brightness different variations.
Let's follow the diagram to connect the physical map.



In the preparation of the program, we will use the simulation write analogWrite (PWM interface, analog value) function, for the simulation write analogWrite () function, which usage is also very simple, we read in this experiment potentiometer analog value signal and assigns PWM interface makes small lights produce a corresponding change in brightness, and then displayed on the screen read analog value, we can understand the value of this program is to read in a simulated experimental procedures that will more PWM interface analog value is assigned to this section, the following for everyone to provide a reference source.
Reference source:

```
int potpin = 0 ;/ / define analog interface 0
int ledpin = 11 ;/ / define the digital interface 11 (PWM output)
int val = 0 ;/ / temporary values of the variables from the sensor
void setup ()
{
pinMode (ledpin, OUTPUT) ;/ / define the digital interface 11 as output
Serial.begin (9600) ;/ / set the baud rate to 9600
/ / NOTE: analog interface is automatically set to the input
}
```

```
void loop ()
{
val = analogRead (potpin) ;/ / read sensor analog values and assigned to val
Serial.println (val) ;/ / display val variable
analogWrite (ledpin, val / 4) ;/ / turn on the LED and set the brightness (PWM output max 255)
delay (10) ;/ / delay of 0.01 seconds
}
```

After downloading the program, we rotate the potentiometer knob only see the change in the value on the screen can also clearly see our breadboard brightness LED lights also changes.

Routine 4, LED flashing experiment

LED lights are more basic experiment experimental one, "Hello World!" Experiment has been built using the Arduino to LED, this time we use other I / O ports and an external DIP LED lights to complete this experiment, we need to the experimental equipment in addition to each experiment must Arduino controller and USB download cable outside

Other devices are as follows:

Red M5 DIP LED * 1

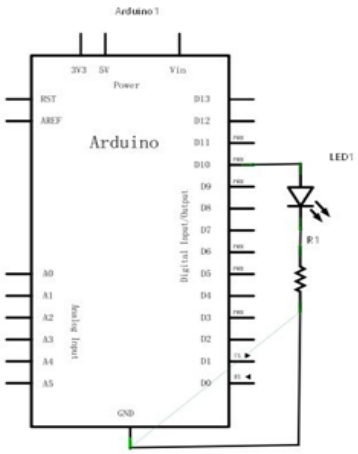
220Ω line resistance * 1

Bread board * 1

Breadboard Jumper * a tie

Next we follow the little lamp experiments link schematic physical map, where we use the number 10 interface. When using a light emitting diode LED, the current limiting resistor to be connected, here 220Ω resistor currents can burn or light emitting diodes.

Small light experiment schematics



Good link circuit according to the diagram, you can start writing programs, and we still make LED lights flashing, lit one second off one second. This procedure is very simple and comes with Arduino Blink similar routines where just 13 digital interface for doing 10 digital interface.

Reference procedure is as follows:

```
int ledPin = 10; / / define the interface number 10
void setup ()
{
pinMode (ledPin, OUTPUT) ;/ / define a small lamp interface output interface
}
void loop ()
{
digitalWrite (ledPin, HIGH); / / lit a small lamp
delay (1000); / / Delay 1 second
digitalWrite (ledPin, LOW); / / extinguish small lights
delay (1000); / / Delay 1 second
}
```

After downloading the program, you can see our 10 external small lights flashing, so our little lights flashing experiment

Is complete.

Routine 5 Advertising lights effect experiment

1) experimental device

□ Led Lights: 6

□ 220Ω resistor: 6

□ colorful experimental breadboard jumper: Several

2) test the connection

Wiring methods in accordance with two tubes, LED lights turn received the six numbers 1 to 6

pin. Figure:
Advertising lights wiring experiment
3) experimental principle

In life we often see some led lights from a variety of colors consisting of billboards, billboards various locations to Tx led lights constantly changed, then the formation of a variety of effects. This section is the use of experimental programming analog advertising lights led lights effect.
Programming Reference:

```
int BASE = 2; / / the first one LED connected to the I / O pins
int NUM = 6; / / LED's total

void setup ()
{
  for (int i = BASE; i <BASE NUM; i)
  {
   pinMode (i, OUTPUT); / / set the digital I / O pin as an output
  }
}

void loop ()
{
  for (int i = BASE; i <BASE NUM; i)
  {
   digitalWrite (i, LOW); / / set the digital I / O pin output is "low", that gradually turn off the lights
    delay (200); / / delay
  }
  for (int i = BASE; i <BASE NUM; i)
  {
   digitalWrite (i, HIGH); / / set the digital I / O pin output is "low", that gradually lights
   delay (200); / / delay
  }
}
```

Routine 6. Traffic light design experiments

Above we have completed a single small light control experiment, then we do a little more complicated traffic light experiments, in fact, clever friends can see out of this experiment is to test a single small light above experiments extended to three colors the small lights, you can

achieve our experiment simulated traffic lights. We accomplish this experiment required components in addition to Arduino controller and the download cable is also required hardware is as follows:

Red M5 DIP LED * 1

Yellow M5 DIP LED * 1

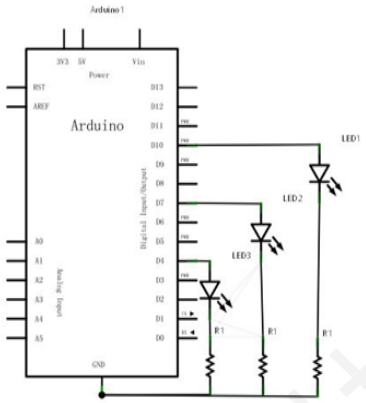
Green M5 DIP LED * 1

220Ω resistor * 3

Bread board * 1

Breadboard Jumper * a tie

Ready we can start these components, we can follow the above experimental replicability of small lights flashing, here is that we provide reference schematics, respectively, we use digital 10,7,4, interface.



Since it is a traffic light simulation experiments, a small red-yellow-green lights flashing time will simulate real traffic lights, we use the Arduino delay () function to control the delay time relative to the C language would simply many of the.

Here is a reference program:

```
int redled = 10; / / define the interface number 10
int yellowled = 7; / / define the number 7 Interface
int greenled = 4; / / define the number 4 Interface
void setup ()
{
pinMode (redled, OUTPUT) ;/ / define a small red light interface output interface
pinMode (yellowled, OUTPUT); / / define the yellow light interface output interface
pinMode (greenled, OUTPUT); / / define the small green light interface output interface
}
void loop ()
{
digitalWrite (redled, HIGH) ;/ / lit red lights
delay (1000) ;/ / delay of 1 second
digitalWrite (redled, LOW); / / off red light
digitalWrite (yellowled, HIGH) ;/ / light yellow light
delay (200) ;/ / delay of 0.2 seconds
digitalWrite (yellowled, LOW) ;/ / off yellow light
```

```
digitalWrite (greenled, HIGH) ;/ / flashes the green LED
delay (1000) ;/ / delay of 1 second
digitalWrite (greenled, LOW) ;/ / green LED off
}
```

Complete the download process can be seen after the control of our own design of traffic lights.

Routine 7 buzzer experiments

Arduino can be done with a lot of interactive work, the most common and most commonly used is the sound and light show in front has been in use LED lights in the experiment, the experiment let everyone circuit sound, with a voice the most common is the buzzer and speaker components, the comparison of the two buzzer easier and ease the present study, we buzzer.
The following components are to be prepared:
Buzzer * 1
Buttons * 1
Bread board * 1
Breadboard Jumper * a tie
According to the following diagram to connect the circuit

Connect the circuit to the attention of the positive and negative point is that the buzzer of the points, can be seen below on the right physical map buzzer red and black two wiring. Connect the circuit program in this regard is very simple, with small lights front control buttons experimental procedure is similar because of the buzzer output control interface Digital interface also can control the high and low buzzer sounds.
Reference source:

```
int buzzer = 8 ;/ / setting controls the digital IO foot buzzer
void setup ()
{
pinMode (buzzer, OUTPUT) ;/ / set the digital IO pin mode, OUTPUT out of Wen
}
void loop ()
{
unsigned char i, j ;/ / define variables
while (1)
{
for (i = 0; i <80; i + +) / / Wen a frequency sound
{
digitalWrite (buzzer, HIGH) ;/ / send voice
delay (1) ;/ / Delay 1ms
```

```
digitalWrite (buzzer, LOW) ;/ / do not send voice
delay (1) ;/ / delay ms
}
for (i = 0; i <100; i + +) / / Wen Qie out another frequency sound
{
digitalWrite (buzzer, HIGH) ;/ / send voice
delay (2) ;/ / delay 2ms
digitalWrite (buzzer, LOW) ;/ / do not send voice
delay (2) ;/ / delay 2ms
}
}
}
```

After downloading the program, the buzzer experiments are done.

Routine 8. Tilt switch experiments

Tilt switch control led lights out

Experimental devices

Ball switch: a Led Light:

A 220Ω resistance: 1

Colorful experimental breadboard jumper: Several

2, the experimental connection

In accordance with the Arduino tutorial panels, expansion board, bread board connected, download cable connected. Then led lights connected to a digital 8-pin, ball switch connected to the analog 5 pin.

3, one end of the experimental principle when the switch is tilted below the horizontal position, the switch look pass, analog port voltage is about 5V (digital binary representation of 1023), led lights lit. When the other end of the lower horizontal position tilt switch stops, so analog port voltage is 0V (binary digit is 0), led lights extinguished. Analog port in the program voltage value is greater than about 2.5V (expressed as number two In Assistant System 512), you can know whether the tilt switch look through.

4, the program code

Code is as follows: void setup ()

```
{
pinMode (8, OUTPUT) ;/ / set the digital 8 pin Wen-out mode
}
void loop ()
{
int i ;/ / define the amount of hair i
while (1)
{
```

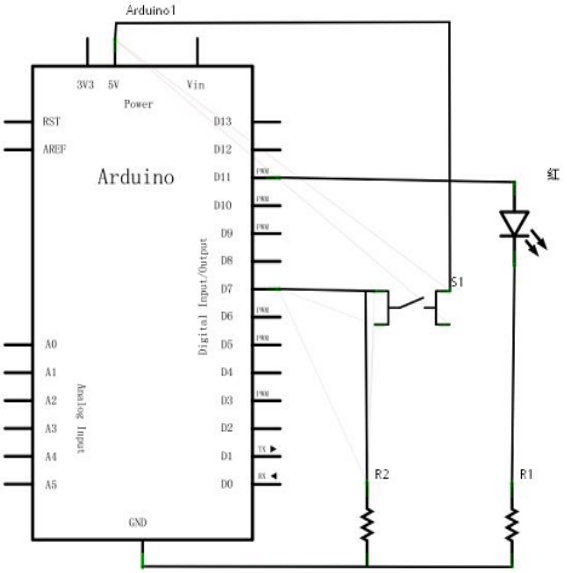
```
i = analogRead (5) ;/ / read the [analog voltage value 5
if (i> 200) / / if greater than 512 (2.5V)
{
digitalWrite (8, HIGH) ;/ / led lights lit
}
else / / Otherwise
{
digitalWrite (8, LOW) ;/ / off led lights
}
}
}
```

Download the program to test the board after the board we can tilt observation led lights Qie state. When the golden end below the horizontal position of the tilt switch look pass, lights led lights; below the horizontal position when the silver side tilt switch constricted ended analog
Around the mouth voltage is 0V (expressed as number two In Assistant system 0), led lights extinguished.
Master the program, we can follow their own Qie thought experiment, receive as a guest you can control other devices such as a buzzer, etc.

Routines 9. Keys to control LED experiment

I / O port interface and the meaning is INPUT OUTPUT jacks, so far we have designed experiments are just a small light applied to the Arduino I / O port output function, try this experiment we used the Arduino I / O port input function reads the external device is the output value, we use a button and an LED lights to complete an input-output combination experiments, so that we can easily understand the I / O function. Key switch everyone should be more understanding, belonging switch (digital) components, when pressed closed (ON) state. To complete this experiment
The components used are as follows:
Key switch * 1
Red M5 DIP LED * 1
220Ω resistor * 1
10KΩ resistor * 1
Bread board * 1
Breadboard Jumper * a tie
We will receive the number 7 key interfaces, a small red light to a digital 11 Interface (Arduino controller 0-13 Digital I / O interfaces can be used to access keys and a small lamp, but try not to choose interfaces 0 and 1, 0, and an interface is multiplexed interface functions, in addition to I / O port function is serial communication interface, download the program belongs to communicate

with the PC, it should be kept vacant interfaces 0 and 1, so as to avoid the hassle of plug wire and try not to use 0 1 Interface), according to the following schematic circuit connected.



The following start writing programs, we let a small button lights up when pressed, according to the previous study believe that this program is very easy to write out several experiments with respect to the previous program, this experiment more of a conditional statement, here we using an if statement, Arduino program he wrote statements are based on the C language, the C's conditional statement naturally also applies to Arduino, like while, swich and so on. Here we are used according to personal preferences

To the use of simple and easy to understand for everyone to do if statement presentation routines.

We analyze the circuit shows when the button is pressed, the interface can read out the number 7 is high, then we make the digital output high 11 allows small lights, the program we determine whether the low seven figures To make digital 11 is low for low output small lamp is not lit, the principle above.

Reference source:

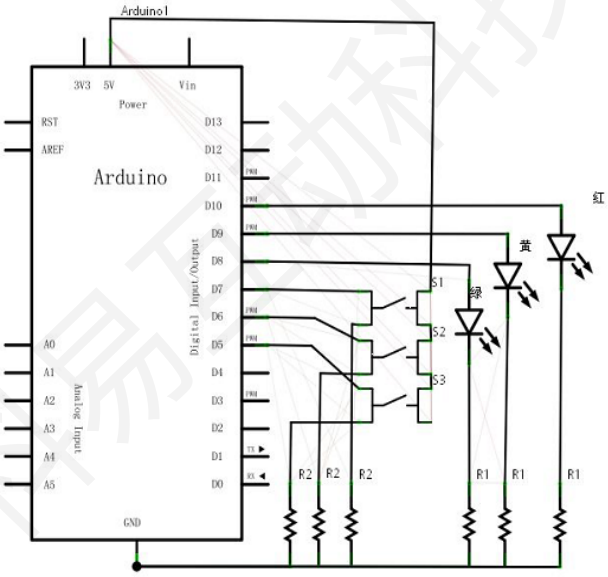
```
int ledpin = 11 ;/ / define the interface number 11
int inpin = 7 ;/ / define the number 7 Interface
int val ;/ / define a variable val
void setup ()
{
pinMode (ledpin, OUTPUT) ;/ / define a small lamp interface output interface
pinMode (inpin, INPUT) ;/ / define the key interface for the input interface
}
void loop ()
{
val = digitalRead (inpin) ;/ / read digital 7-level value assigned to val
if (val == LOW) / / test button is pressed, the button lights up when pressed small
```

```
{DigitalWrite (ledpin, LOW);}
else
{DigitalWrite (ledpin, HIGH);}
}
```

After downloading the program we have this little lamp with the button to complete the experiment, the experimental principle is very simple, widely used in various circuits and appliances, all in real life is not difficult to find on various devices, such as everyone's phone when you press any key when the backlight is lit, this is a typical application, the following is an experiment in one of the most simple life application examples ------ Responder.

Routine 10. Responder design experiments

After completion of the above experiments believed to have a lot of friends can independently complete this experiment, the present study is to small lights above the button control experiments extended to three buttons correspond to three small lights, taking six digital I / O interface.
Principle here is not to say with the above experiments, the following reference to the schematic attached.



Refer to the source as follows:

```
int redled = 10;
int yellowled = 9;
int greenled = 8;
int redpin = 7;
int yellowpin = 6;
```

```
int greenpin = 5;
int red;
int yellow;
int green;
void setup ()
{
pinMode (redled, OUTPUT);
pinMode (yellowled, OUTPUT);
pinMode (greenled, OUTPUT);
pinMode (redpin, INPUT);
pinMode (yellowpin, INPUT);
pinMode (greenpin, INPUT);
}
void loop ()
{
red = digitalRead (redpin);
if (red == LOW)
{DigitalWrite (redled, LOW);}
else
{DigitalWrite (redled, HIGH);}
yellow = digitalRead (yellowpin);
if (yellow == LOW)
{DigitalWrite (yellowled, LOW);}
else
{DigitalWrite (yellowled, HIGH);}
green = digitalRead (greenpin);
if (green == LOW)
{DigitalWrite (greenled, LOW);}
else
{DigitalWrite (greenled, HIGH);}
}
```

This program interfaces with the increase in the previous procedure except no inconsistencies outside, so do not do analysis program notes.
After downloading the program, we have to produce their own simple Responder is complete.

Routine 11. Analog value read experiment

The experiment we have to start learning about analog I / O interface to use, Arduino with Analog 0 - Analog 5 Total 6 analog interfaces, which six interface can also be counted as multiplexed

interface functions, in addition to analog interface functions, which 6 interface can be used as a digital interface, numbered numeral 14 - the number 19, a simple understanding of the future, following on to start our experiment. Potentiometer is more familiar typical analog value output element, the experiment will be done with it.

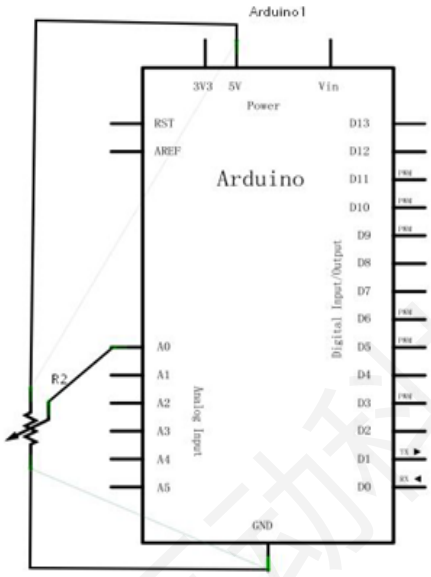
Required components are:

Potentiometer * 1

Bread board * 1

Breadboard Jumper * a tie

In this study, we will be the resistance of the potentiometer into analog value is read out, and then displayed on the screen, which is why we need to complete their subsequent experimental features examples of applications that must be mastered. We need the following diagram to connect the physical map



We are using an analog 0 interface.

Procedures for the preparation is very simple, a analogRead (); statement can be read out analog port values, Arduino 328 is a 10-bit A / D acquisition, so the read analog value range is 0-1023, the experimental procedure There is also a difficulty value is displayed on the screen this problem, it is very simple to learn. First, we want to voidsetup () which set the baud rate, the indicated value is Arduino communicate with the PC, so the Arduino software and PC baud rate should be set in the same order to show the correct value, otherwise it will be garbled or does not appear in the lower right corner of the monitor window of Arduino software has a button that you can set the baud rate, the baud rate set here need to talk program void setup () which set the baud rate the same procedure to set the baud rate of the statement is Serial.begin (); including

No value for the baud rate. The second is a statement of the displayed value, Serial.print (); or Serial.println (); all OK, unlike the latter value End show automatically after carriage return, the former is not, to explain more about the statement in front of

Introduction there is not more to say.
Here is a reference source:

```
int potpin = 0 ;/ / define analog interface 0
int ledpin = 13 ;/ / define the digital interface 13
int val = 0 ;/ / will define the variable val, and the initial value 0
void setup ()
{
pinMode (ledpin, OUTPUT) ;/ / output interface defines the digital interface
Serial.begin (9600) ;/ / set the baud rate to 9600
}
void loop ()
{
digitalWrite (ledpin, HIGH) ;/ / digital interface 13 of the LED lights
delay (50) ;/ / delay of 0.05 seconds
digitalWrite (ledpin, LOW) ;/ / off LED digital interface 13
delay (50) ;/ / delay of 0.05 seconds
val = analogRead (potpin) ;/ / read the value of analog interface 0, and assign val
Serial.println (val) ;/ / shows the value of val
}
```

Reference procedure borrowed Arduino digital 13 comes with LED lights, small lights every read value will flash once.
Here is read analog values.

This experiment is done here, when you turn the potentiometer knob when you can see the changes in the value on the screen, read the analog value of this method will always accompany us, the analog value is read very frequently used functions, as Many sensors are analog value output, we read the analog value, and then the corresponding algorithm processing, can be applied to the functions we need to implement in the.

Routine 12. Experimental sound and light control

1, the experimental device
Photoresistor: 1
Buzzer: 1
Colorful experimental breadboard jumper: Several
2, the experimental connection

In accordance with the Arduino tutorial panel, prototype board, bread board connected, download cable connected. Photoresistor a termination in figure 6, the other end is connected to the anode from bee name, bee funerary negative electrode and GND.
3, the experimental principle

The application process preceding sections read analog port voltage value method photoresistor connected directly to the digital port. Programs similar to the second buzzer procedures, there is no light, the normal sound, but the sound special little; when there is light, the photosensitive resistor decreases, so will increase the voltage across the buzzer The buzzer sounds fat. The stronger the light, the smaller the resistance, the buzzer louder.

Program description:

```
void setup ()
{
pinMode (6, OUTPUT);
}
void loop ()
{
while (1)
{
char i, j;
while (1)
{
for (i = 0; i <80; i) / / Wen Qie one frequency sound
{
digitalWrite (6, HIGH);
delay (1);
digitalWrite (6, LOW);
delay (1);
}
for (i = 0; i <100; i) / / Wen Qie out another frequency sound
{
digitalWrite (6, HIGH);
delay (2);
digitalWrite (6, LOW);
delay (2);
}
}
}
}
```

Download the program to experiment board, you can use a flashlight or other light objects received irradiation photosensitive resistance, can be heard
There is light buzzer louder.
Master the program, we can design experiments yourself, you can also use photoresistor control led lamp brightness.

Routine 13. Flames alarm experiment

A flame sensor introduction

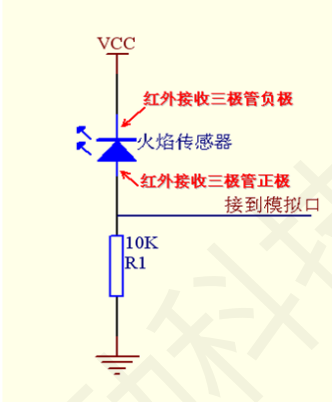
1, recognizing the flame sensor

Flame sensor (ie, infrared receiver Transistor) is a robot designed to search for the source of fire sensors, this sensor particularly sensitive to fire. Kind shown in Figure:

2, the working principle

Infrared flame sensor is very sensitive to the characteristics of the flame, the use of a special infrared receiver to detect the flame, and the flame luminosity changes into the low level signal is input to the central processor, the central processor changes the signal to make the appropriate procedures.

3, the flame sensor connection



Infrared receiver transistor short lead cathode end, long lead terminal is positive. The negative to the following diagram 5V interface, then the positive and 10K resistor connected to the other end of the resistor to GND interface, the last from the positive terminal of the flame sensor column where access a jumper, the other end of the jumper connected to the analog mouth. Figure

Second, the experimental flame alarm

1, the experimental device

□ □ flame sensor: 1

□ □ Buzzer: 1

□ □ 10K Resistance: 1

□ □ colorful experimental breadboard jumper: Several

2, the experimental connection

1) Connect the buzzer

First, in accordance with the Arduino tutorial panel, prototype board, bread board connected, download cable connected. Remove the cartridge from the experiment buzzer, buzzer according to the connection method second experiment, the buzzer is connected to the digital 8TH. Buzzer complete connection.

2) The flame sensor connection

From the experimental flame sensor out of the box, as described in this section of the flame sensor wiring method, the flame sensor connected to analog 5. Complete the entire experiment connection.

3, the experimental principle

And without a flame near the flame near both cases, the voltage value read analog port is changing. Actual measured with a multimeter shows that in the absence of flames approached, the voltage value of the analog port read about 0.3V; when a flame near the analog voltage is 1.0V mouth read about the flame near the closer voltage value, the more large.

Therefore, the beginning of the program, can be stored without a flame, the voltage analog port i. Then continuous loop reads analog port voltage value j, to do with the difference between the stored value k = ji, the difference k 0.6v do not compare. K If the difference is greater than 0.6V (digital binary value 123), it is judged that a flame near the buzzer sounds for alarms; if the difference is less than 0.6v buzzer does not sound.

4, the program code

```
int flame = A5 ;/ / define the flame interface analog 0 interface
 int Beep = 8 ;/ / buzzer interface defines the interface number 7
 int val = 0 ;/ / define numeric variables
val void setup ()
{PinMode (Beep, OUTPUT) ;/ / define LED as output interface
 pinMode (flame, INPUT) ;/ / define the buzzer as the input interface
 Serial.begin (9600) ;/ / set the baud rate to 9600}
void loop () {val = analogRead (flame) ;/ / read the analog value flame sensor
Serial.println (val) ;/ / output analog values, and print them out
 if (val> = 600) / / When the analog value is greater than 600 when the buzzer sounds
 {DigitalWrite (Beep, HIGH);} else {digitalWrite (Beep, LOW);}}
```

5, the program features

The program can simulate a flame alarm at the situation in the absence of the flame when everything is normal, when there is an alarm as soon as the flame made tips

Routine 14 Arduino to do with 0-5V range voltmeter

Note: This design is not experimental circuit protection circuit design is relatively complex, so do not use more than two or more AA batteries, and not be used to measure battery or other power! !

Need to prepare a list of electronic components

Arduino control board a

Breadboard a
Bread line six
1KΩ resistance of a
A USB cable

Here we look at the design of the circuit connection diagram

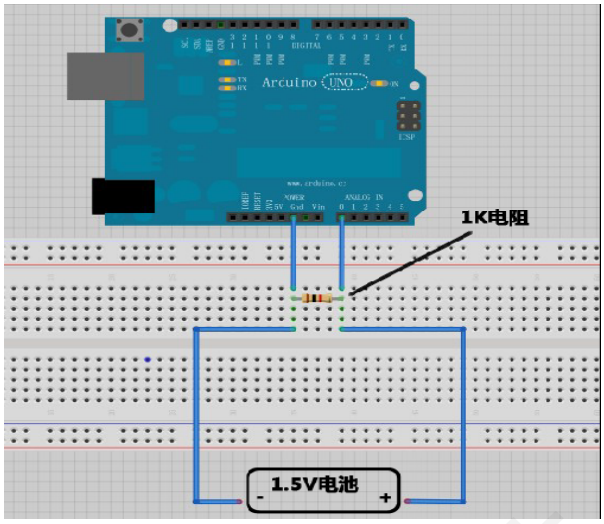


Figure 1K resistors used on the measurement object side floating case, the GND reference level to the measuring port, to avoid interference floating interfaces,
In accordance with the circuit connection diagram corresponds to the physical circuit structures

Program source code is as follows

```
float temp; / / create a float variable temp as a storage space to store data preparation
void setup ()
{
  Serial.begin (9600); / / use 9600 baud rate serial communication
}
void loop ()
{

  int V1 = analogRead (A0);
/ / Read voltage from A0 mouth integer type data into the newly created variable V1, analog port voltage measurement range of 0-5V returns a value of 0-1024
  float vol = V1 * (5.0 / 1023.0);
/ / We will be converted into the actual value of V1 voltage value into a float variable vol
  if (vol == temp)
/ / This part of the judgment is used to filter duplicate data, only the second voltage value when the output and the last mixed
  {
    temp = vol; / / After the completion of the comparison, the ratio of this value into a variable temp for comparison
```

```
  }
  else
  {
    Serial.print (vol); / / serial output voltage value, and do not wrap
    Serial.println ("V"); / / serial output character V, and line breaks
    temp = vol;
    delay (1000); / / Wait a second output is completed, the data used to control the refresh rate.
  }
}
```

Click to open the serial monitor

Then the red wire measuring battery anode, black line in the measuring cell will be in the negative serial monitor once a second refresh rate of the voltage value, the voltage value twice the normal fluctuations, because, after all, the low precision of the test .

Routine 15. Voice detection module

For sound detection
1, AO, analog output, real-time output voltage signal of the microphone
2, there is a mounting screw hole 3mm
3, the use 5v DC power supply
4, with analog output
5, high sensitive microphone and high sensitivity.
6, a power indicator light

ARDUIINO code:

2 analog outputs:

```
int sensorPin = A5; / / select the input pin for the potentiometer
int ledPin = 13; / / select the pin for the LED
int sensorValue = 0; / / variable to store the value coming from the sensor

void setup () {
  pinMode (ledPin, OUTPUT);
  Serial.begin (9600);
}

void loop () {
```

```
  sensorValue = analogRead (sensorPin);
  digitalWrite (ledPin, HIGH);
 delay (sensorValue);
  digitalWrite (ledPin, LOW);
 delay (sensorValue);
 Serial.println (sensorValue, DEC);
}
```

Routine 16. LM35 temperature sensor experiment

LM35 is a very common and easy to use temperature sensor element, the components need only one application LM35 components, using only an analog interface can be, on the difficulty lies in the algorithm will read the analog value is converted to the actual temperature.

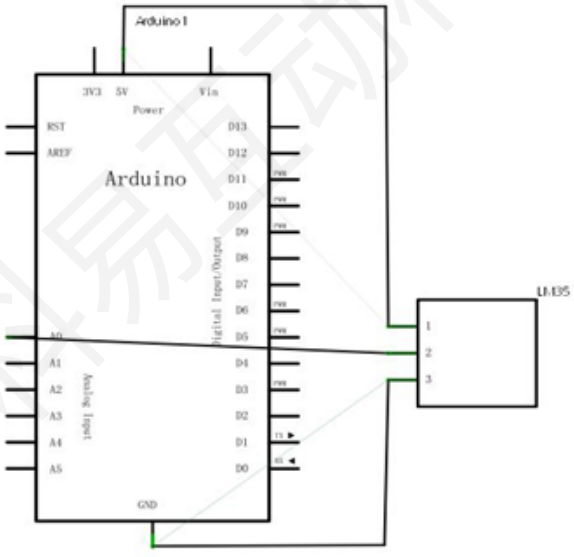
Required components are as follows.

Inline LM35 * 1

Bread board * 1

Breadboard Jumper * a tie

Connect the circuit according to the following diagram.



Reference source:

```
int potPin = 0; / / define the analog interface 0 LM35 temperature sensor connection
void setup ()
{
```

```
Serial.begin (9600) ;/ / set the baud rate
}
void loop ()
{
int val ;/ / define variables
int dat ;/ / define variables
val = analogRead (0) ;/ / read sensor analog values and assigned to val
dat = (125 * val) >> 8 ;/ / temperature formula
Serial.print ("Tep:") ;/ / output as a string represents the temperature display Tep
Serial.print (dat) ;/ / output shows the value of dat
Serial.println ("C") ;/ / output display as a C string
delay (500) ;/ / delay of 0.5 seconds
}
```

After downloading the program to open the monitor window you can see the current temperature.

Routine 17. Thermostat cups

Today we use arduino make an auto-sensing cup circuit, let's design a good circuit, through lm35 temperature sensor sensing temperature, so different color led display indicates the temperature

Original List

arduino controller 1

Breadboard 1

Bread plate line a box

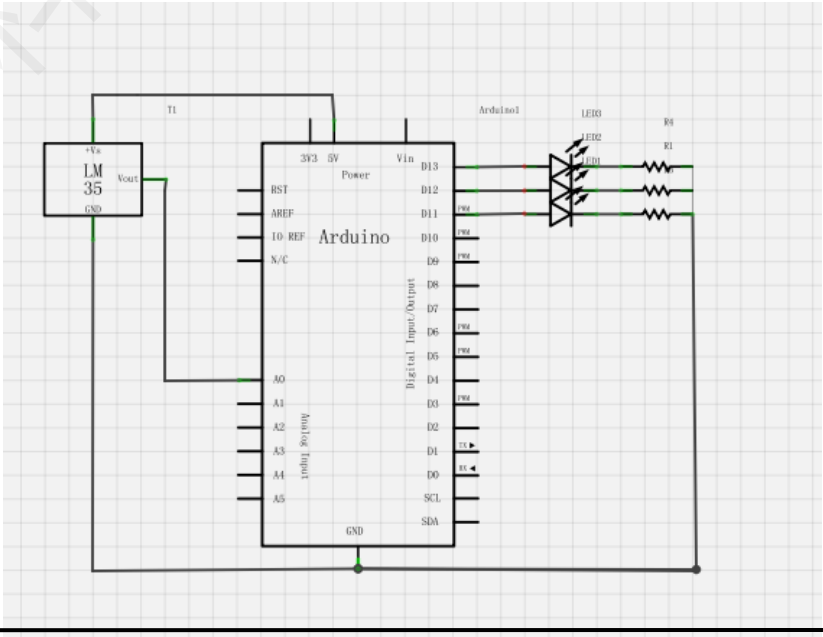
Each one red, yellow, blue LED

220Ω resistor 3

lm35 temperature sensor 1

USB data cable 1

Schematic



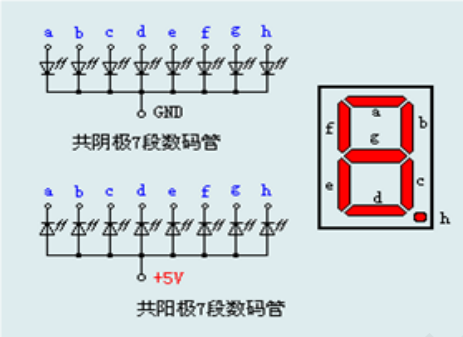
Procedure is as follows

```
void setup () {
  pinMode (13, OUTPUT);
  pinMode (12, OUTPUT);
  pinMode (11, OUTPUT);
}
void loop () {
  int vol = analogRead (A0) * (5.0 / 1023.0 * 100); / / read the LM35 temperature
if (vol <= 31) / / set the value of the temperature of the low temperature region, and led display
{
  digitalWrite (13, HIGH);
  digitalWrite (12, LOW);
  digitalWrite (11, LOW);
}
else if (vol> = 32 && vol <= 40)
 {
   digitalWrite (13, LOW);

  digitalWrite (12, HIGH);
  digitalWrite (11, LOW);
}
else if (vol> = 41) / / hot zone temperature setting
{
   digitalWrite (13, LOW);
  digitalWrite (12, LOW);
  digitalWrite (11, HIGH);
}
 }
```


Routine 18. Nixie tube display experiment

LED is a semiconductor light emitting device, the basic unit is a light emitting diode. Digital tube divided by the number of segments and eight seven-segment LED digital tube, eight digital tube

one more than the seven-segment digital tube light emitting diode unit (one more decimal display), used in this experiment is eight digital tube. Light emitting diode unit is connected by way into common anode and common cathode LED digital tube. Common anode LED refers to light emitting diode anode connected all together to form a common anode (COM) digital control. Common anode tube should be applied in the public received very COM +5 V, when a field emission cathode of the diode is low, the corresponding field on the light. When a field cathode is high, the corresponding field is not bright. Common cathode LED refers to light emitting diode cathode connected all together to form a common cathode (COM) digital control. Common cathode LED should be applied in the public pole connected to ground GND COM, when a field of light-emitting diode anode is high, the corresponding field on the light. When the anode of a field is low, the corresponding field is not bright.



Each section of the digital tube is composed of light-emitting diodes, so when used with the same light-emitting diode, also connected current limiting resistor, or the current through the light emitting diode meeting burned. In this study, using a common cathode LED, common cathode LED should be applied in the common electrode coupled to GND, when a field of light-emitting diode anode is low, the corresponding field on the point off. When the anode of a field is high, the corresponding field on the light. Introduction to theory,

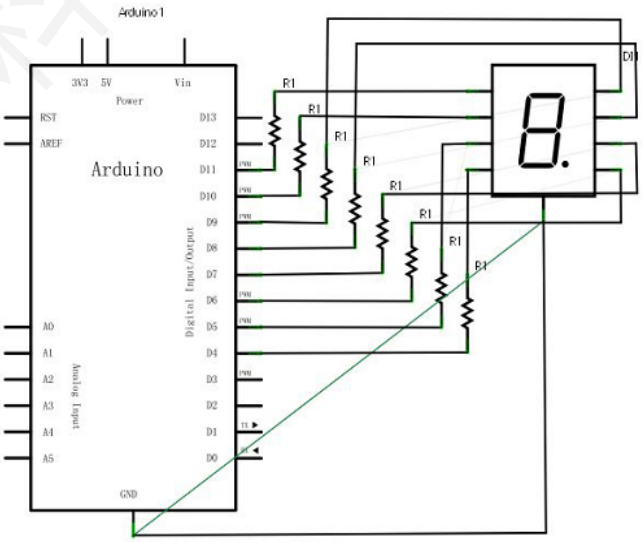
We began to prepare experimental components.

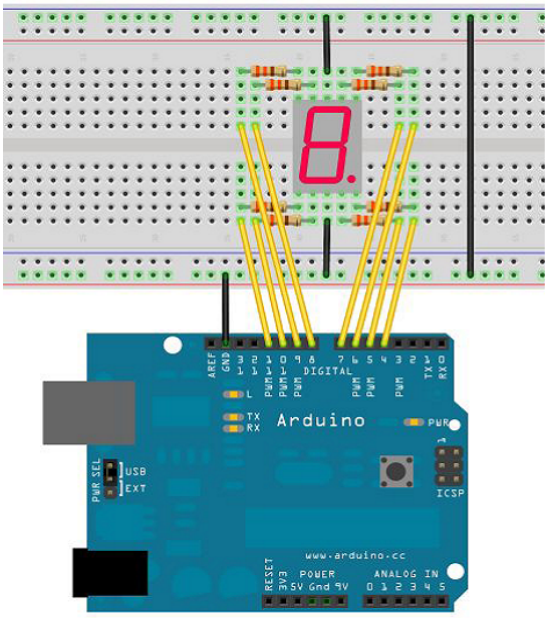
Eight digital tube * 1

220Ω line resistance * 8

Breadboard Breadboard Jumper * 1 * 1 tie

We refer to physical connection diagram according to the schematic circuit connected.





There are seven-segment LED display digital segment, there is a decimal point segments. When asked digital display numbers, as long as the corresponding segment lights can be. For example: Let the digital display number 1, then b, c section can be lit. Each number will be written as a subroutine. Every 2s in the main display a number, so digital control loop display 1 to 8 digits. Each time a digital display is determined by the delay time, time is set larger, the displayed time is longer, time is set smaller, the displayed time is short.

Reference source code:

```
/ / Set control each segment digital IO pin
int a = 7 ;/ / define the digital interface to connect a seven segment LED
int b = 6 ;/ / define the connection b Digital Interface 6-segment LED
int c = 5 ;/ / define paragraph (c) Digital Interface 5 digital connection
int d = 11 ;/ / define the digital interface 11 is connected to d-segment digital tube
int e = 10 ;/ / define the digital interface 10 is connected to e-segment digital tube
int f = 8 ;/ / define the digital interface 8 digital tube connection f
int g = 9 ;/ / define the digital interface 9 g of the digital control connection
int dp = 4 ;/ / define the digital interface 4 digital tube connecting dp
void digital_1 (void) / / display the number 1
{
unsigned char j;
digitalWrite (c, HIGH) ;/ / to the digital interface 5-pin high, lit paragraph (c)
digitalWrite (b, HIGH) ;/ / lit paragraph b
for (j = 7; j <= 11; j + +) / / off remaining segments
digitalWrite (j, LOW);
digitalWrite (dp, LOW) ;/ / off decimal point DP segment
}
void digital_2 (void) / / display number 2
{
unsigned char j;
```

```
digitalWrite (b, HIGH);
digitalWrite (a, HIGH);
for (j = 9; j <= 11; j + +)
digitalWrite (j, HIGH);
digitalWrite (dp, LOW);
digitalWrite (c, LOW);
digitalWrite (f, LOW);
}
void digital_3 (void) / / display the number 3
{
unsigned char j;
digitalWrite (g, HIGH);
digitalWrite (d, HIGH);
for (j = 5; j <= 7; j + +)
digitalWrite (j, HIGH);
digitalWrite (dp, LOW);
digitalWrite (f, LOW);
digitalWrite (e, LOW);
}
void digital_4 (void) / / show 4
{
digitalWrite (c, HIGH);
digitalWrite (b, HIGH);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
digitalWrite (dp, LOW);
digitalWrite (a, LOW);
digitalWrite (e, LOW);
digitalWrite (d, LOW);
}
void digital_5 (void) / / display the number 5
{
unsigned char j;
for (j = 7; j <= 9; j + +)
digitalWrite (j, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (dp, LOW);
digitalWrite (b, LOW);
digitalWrite (e, LOW);
}
void digital_6 (void) / / display the number 6
{
unsigned char j;
```

```
for (j = 7; j <= 11; j + +)
digitalWrite (j, HIGH);
digitalWrite (c, HIGH);
digitalWrite (dp, LOW);
digitalWrite (b, LOW);
}
void digital_7 (void) / / display the number 7
{
unsigned char j;
for (j = 5; j <= 7; j + +)
digitalWrite (j, HIGH);
digitalWrite (dp, LOW);
for (j = 8; j <= 11; j + +)
digitalWrite (j, LOW);
}
void digital_8 (void) / / display the number 8
{
unsigned char j;
for (j = 5; j <= 11; j + +)
digitalWrite (j, HIGH);
digitalWrite (dp, LOW);
}
void setup ()
{
int i ;/ / define variables
for (i = 4; i <= 11; i + +)
pinMode (i, OUTPUT) ;/ / set 4 to 11 pin to output mode
}
void loop ()
{
while (1)
{
digital_1 () ;/ / display numbers 1
delay (2000) ;/ / delay 2s
digital_2 () ;/ / display number 2
delay (1000); / / delay 1s
digital_3 () ;/ / display the number 3
delay (1000); / / delay 1s
digital_4 () ;/ / show 4
delay (1000); / / delay 1s
digital_5 () ;/ / display the number 5
delay (1000); / / delay 1s
digital_6 () ;/ / display the number 6
delay (1000); / / delay 1s
```

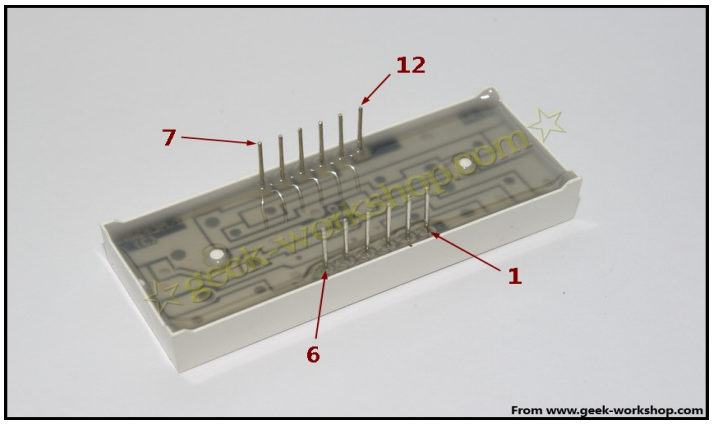
```
digital_7 () ;/ / display the number 7
delay (1000); / / delay 1s
digital_8 () ;/ / display the number 8
delay (1000); / / delay 1s
}
}
```

In the setup () previously defined a series of digital display routines, these routines can be easily defined in
loop () to use, simply use the subroutine name written on top.

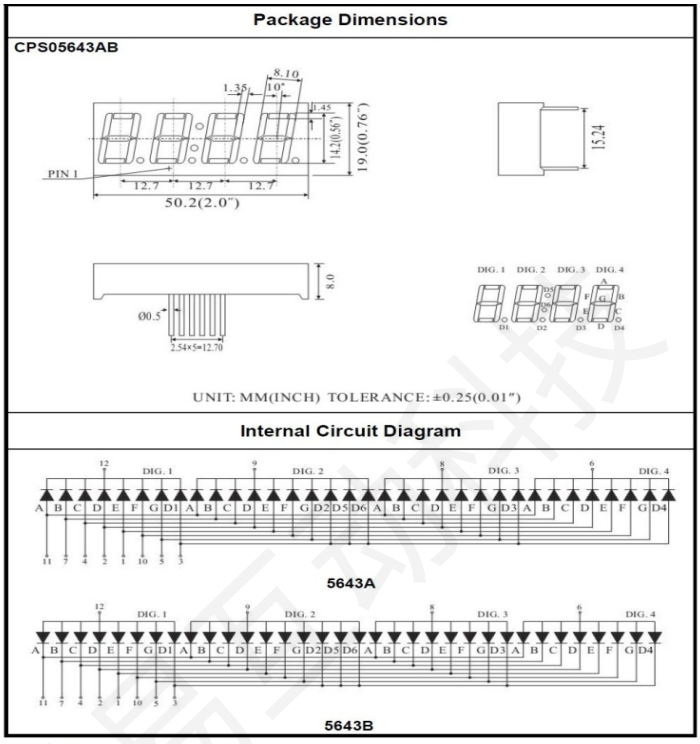
Routine 19.arduino drive four digital tube

We conducted this experiment is to use arduino drive a common anode four digital tube. Driven digital control current limiting resistor is certainly essential, current limiting resistor connection, there are two, one is in the d1-d4 anode, then a total of four. This benefit is demand connection resistance is relatively small, but it will produce different numbers displayed on each one will be different brightness, one of the brightest, 8 darkest. Another connection that is connected to the other eight pins, this connection is highlighted uniform, but with more resistance. The experiment uses eight 220Ω resistor (because there is no 100Ω resistor, so use 220Ω substitute, 100 ohm brightness will be relatively high).

4 There are 12 digital pins are placed in front of the decimal point down, bottom left is one, the other pin order to rotate counterclockwise. Upper left corner of the largest on the 12th pin.

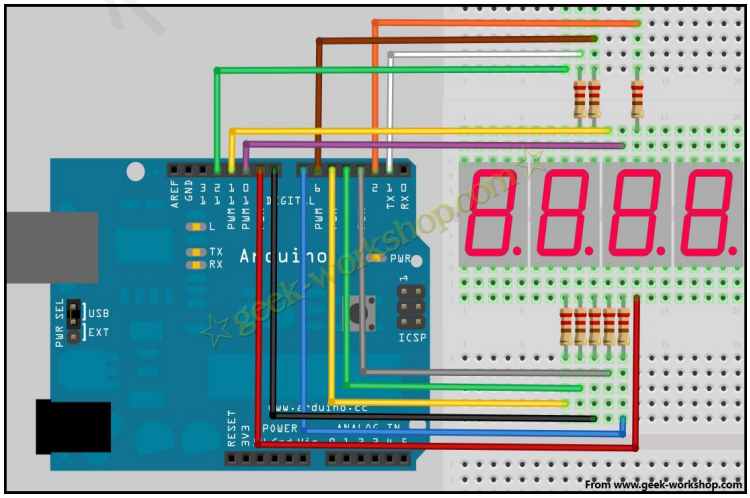


Below is the instruction manual for the digital tube



Four Digits Displays Series

Below is a hardware connection diagram



ARDUINO CODECOPY

```
/ / Set the cathode interfaces
int a = 1;
int b = 2;
int c = 3;
int d = 4;
int e = 5;
int f = 6;
int g = 7;
int p = 8;
/ / Set the anode Interface
int d4 = 9;
int d3 = 10;
int d2 = 11;
int d1 = 12;
/ / Set variables
long n = 0;
int x = 100;
int del = 55; / / here to fine-tune the value of the clock

void setup ()
{
  pinMode (d1, OUTPUT);
  pinMode (d2, OUTPUT);
  pinMode (d3, OUTPUT);
  pinMode (d4, OUTPUT);
  pinMode (a, OUTPUT);
  pinMode (b, OUTPUT);
  pinMode (c, OUTPUT);
  pinMode (d, OUTPUT);
  pinMode (e, OUTPUT);
  pinMode (f, OUTPUT);
  pinMode (g, OUTPUT);
  pinMode (p, OUTPUT);
}

void loop ()
```

```
{
  clearLEDs ();
  pickDigit (1);
  pickNumber ((n/x/1000)% 10);
  delayMicroseconds (del);

  clearLEDs ();
  pickDigit (2);
  pickNumber ((n/x/100)% 10);
  delayMicroseconds (del);

  clearLEDs ();
  pickDigit (3);
  dispDec (3);
  pickNumber ((n/x/10)% 10);
  delayMicroseconds (del);

  clearLEDs ();
  pickDigit (4);
  pickNumber (n / x% 10);
  delayMicroseconds (del);

  n + +;

  if (digitalRead (13) == HIGH)
  {
    n = 0;
  }
}

void pickDigit (int x) / / define pickDigit (x), whose role is to open the port dx
{
  digitalWrite (d1, LOW);
  digitalWrite (d2, LOW);
  digitalWrite (d3, LOW);
  digitalWrite (d4, LOW);

  switch (x)
  {
  case 1:
    digitalWrite (d1, HIGH);
    break;
  case 2:
    digitalWrite (d2, HIGH);
```

```
    break;
  case 3:
    digitalWrite (d3, HIGH);
    break;
  default:
    digitalWrite (d4, HIGH);
    break;
  }
}

void pickNumber (int x) / / define pickNumber (x), whose role is to show digital x-
{
  switch (x)
  {
  default:
    zero ();
    break;
  case 1:
    one ();
    break;
  case 2:
    two ();
    break;
  case 3:
    three ();
    break;
  case 4:
    four ();
    break;
  case 5:
    five ();
    break;
  case 6:
    six ();
    break;
  case 7:
    seven ();
    break;
  case 8:
    eight ();
    break;
  case 9:
    nine ();
    break;
```

```
  }
}

void dispDec (int x) / / set to open the decimal point
{
  digitalWrite (p, LOW);
}

void clearLEDs () / / clear the screen
{
  digitalWrite (a, HIGH);
  digitalWrite (b, HIGH);
  digitalWrite (c, HIGH);
  digitalWrite (d, HIGH);
  digitalWrite (e, HIGH);
  digitalWrite (f, HIGH);
  digitalWrite (g, HIGH);
  digitalWrite (p, HIGH);
}

void zero () / / define those figures 0:00 cathode pin switch
{
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, HIGH);
}

void one () / / define those figures 1:00 cathode pin switch
{
  digitalWrite (a, HIGH);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, HIGH);
  digitalWrite (e, HIGH);
  digitalWrite (f, HIGH);
  digitalWrite (g, HIGH);
}

void two () / / define those figures 2:00 cathode pin switch
{
```

```
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, HIGH);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, HIGH);
  digitalWrite (g, LOW);
}

void three () / / define those figures 3:00 cathode pin switch
{
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, HIGH);
  digitalWrite (f, HIGH);
  digitalWrite (g, LOW);
}

void four () / / define those figures 4:00 cathode pin switch
{
  digitalWrite (a, HIGH);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, HIGH);
  digitalWrite (e, HIGH);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
}

void five () / / define those figures 5:00 cathode pin switch
{
  digitalWrite (a, LOW);
  digitalWrite (b, HIGH);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, HIGH);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
}

void six () / / define those figures 6:00 cathode pin switch
{
```

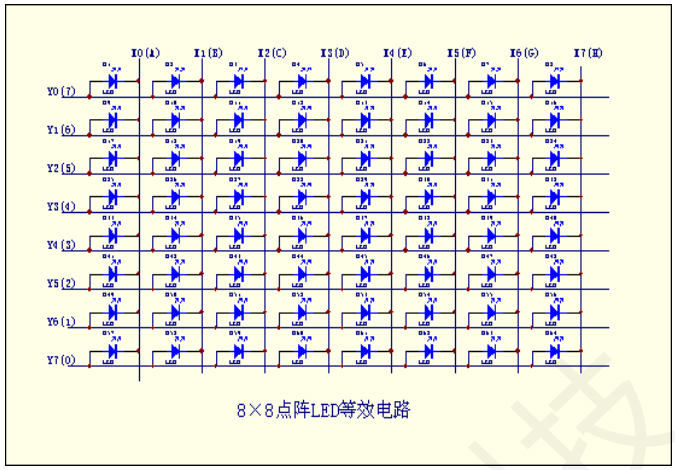
```
  digitalWrite (a, LOW);
  digitalWrite (b, HIGH);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
}

void seven () / / define those figures 7:00 cathode pin switch
{
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, HIGH);
  digitalWrite (e, HIGH);
  digitalWrite (f, HIGH);
  digitalWrite (g, HIGH);
}

void eight () / / define those figures 8:00 cathode pin switch
{
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
}

void nine () / / define those figures 9:00 cathode pin switch
{
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, HIGH);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
}
```

Copy the following code downloaded to the control board and see the effect.

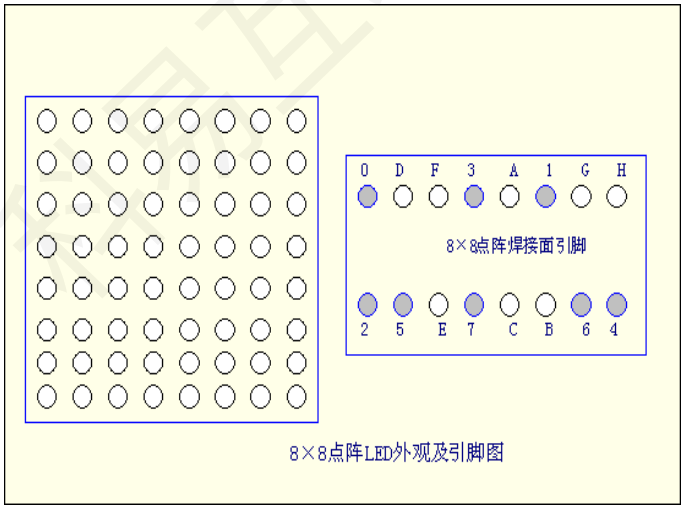
Routine 20.8x8 experimental lattice

Outline

1.8 * 8 dot matrix diagram



2.8 * 8 dot matrix physical map



The picture shows the 8 × 8 dot matrix LED appearance and pin diagram, the equivalent circuit shown in Figure (2) shows, as long as the corresponding X, Y axis forward bias, you can make the

LED lights up. For example if you want to make the top left LED is lit, Y0 = 1, X0 = 0 can be. Application time limit resistor can be placed X-axis or Y-axis

3.8 * 8 dot matrix scanning

Commonly used scanning LED display, the actual use of the following three methods

(1) point scanning

(2) Scan the ranks

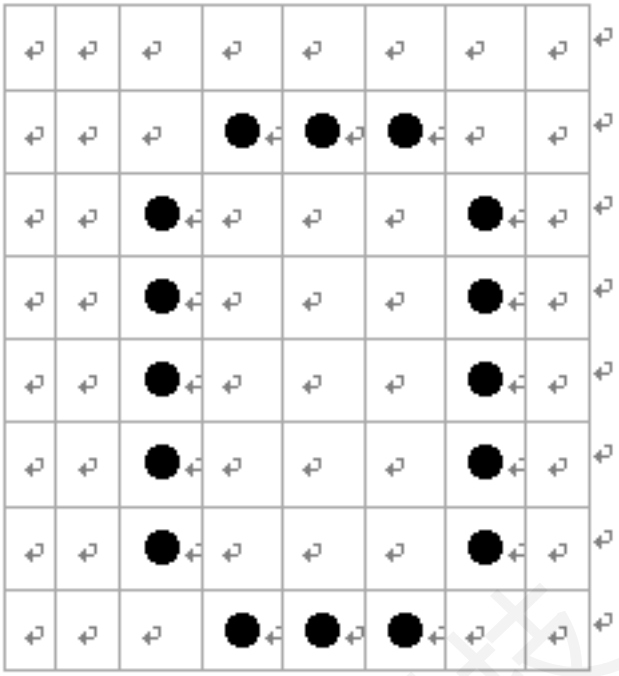
16 × 64 = 1024Hz, cycle to less than 1ms. If you use the second and the third way, the frequency must be greater than 16 × 8 = 128Hz, 7.8ms cycle is less than required to meet the persistence of vision. Moreover once driven a column or row (8 LED) when required plus increase the current drive circuit, or LED brightness is insufficient.

3.8 Application examples * 8 dot matrix

Lattice internal structure and shape as, 8X8 matrix comprises a total of 64 light-emitting diodes, and each light-emitting diodes are placed on the row lines and column lines crossing point, when the corresponding row is set to a level set in a column 0 level, the corresponding LED will light; like a little bit of light to the first, then 9 feet to 13 feet high then low, then the first point on the light; if you want the first line lit, 9 feet to pick high level, and (13,3,4,10,6,11,15,16) These pins are tied low, then the first line will be lit; If you want to The first column is lit, then the first 13 feet to low, and (9,14,8,12,1,7,2,5) then high, then the first column will be lit.

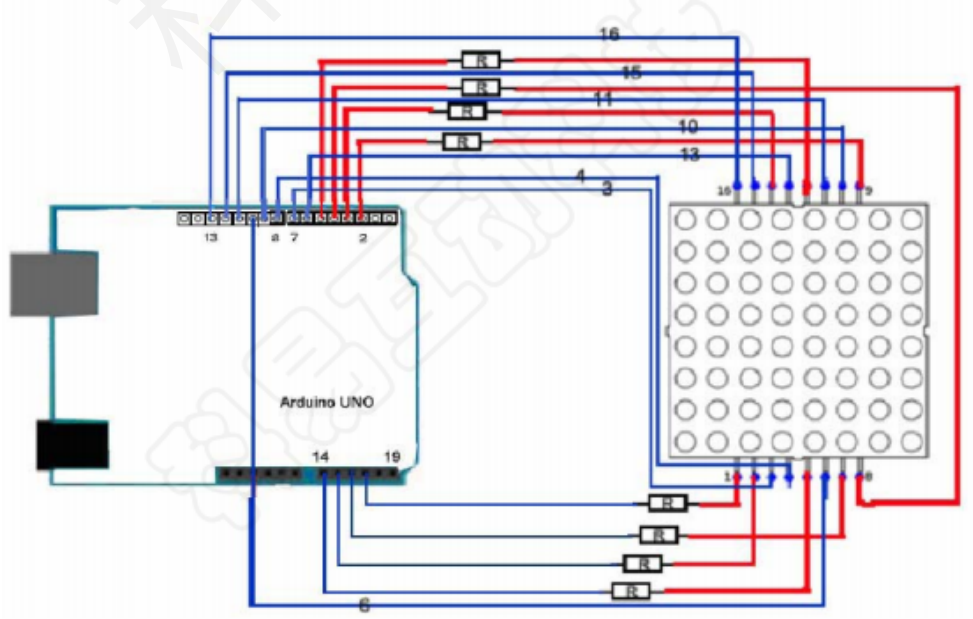
Generally, we use the dot matrix display Chinese characters is 16 * 16 dot matrix font Times New Roman, the so-called 16 * 16, is every character in the longitudinal and transverse area of the 16-point display. That is starting a four 8 * 8 dot matrix combined into a 16 * 16 dot matrix. As shown below, to display "you" then the corresponding points will light up, because our lattice in the column line is active low, while the row line is active high, so to show "You "character, then it's bit code information to be negated, ie all columns (13 to 16 feet) Delivery (1111011101111111,0 xF7, 0x7F), and the first line (9 feet) to send a signal, and then the first line to send 0. Sending the data to display the second line (13 to 16 feet) Delivery (1111011101111111,0 xF7, 0x7F), while the second line (14 feet) to send a signal. And so on, as long as each row of data display interval is short enough, the use of human visual suspensive effect, so that the data sent to 16 times after scanning the line 16 will see a "you"; second way is to

send data font signal to the row line and then scan the column line is the same reason. Similarly to "you" to illustrate, line 16 (9,14,8,12,1,7,2,5) sent (0000000000000000,0 x00, 0x00) and the first column (13 feet) to send, "0" . Similarly scan the second column. Line when the line 16 times to send data out line scan 16 times a "you" will show up.

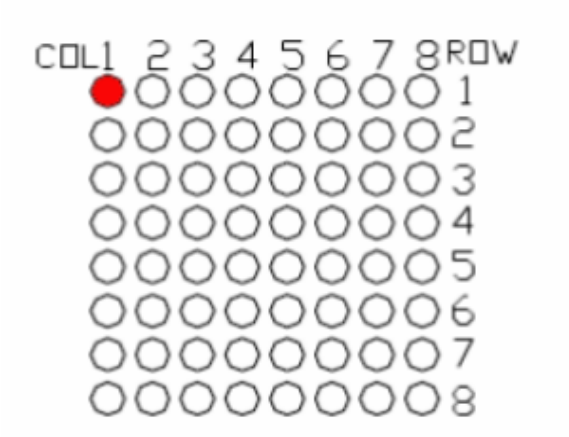


Thus, the formation of the column code 00H, 00H, 3EH, 41H, 41H, 3EH, 00H, 00H; as long as these codes are successively sent to the corresponding column lines above can achieve the "0" digital display.

In this study, the wiring diagram.



A 8X8 matrix LED lit LED as follows:



****
****

Example code:

```
/ / The pin to control ROW
const int row1 = 2; / / the number of the row pin 9
const int row2 = 3; / / the number of the row pin 14
const int row3 = 4; / / the number of the row pin 8
const int row4 = 5; / / the number of the row pin 12
const int row5 = 17; / / the number of the row pin 1
const int row6 = 16; / / the number of the row pin 7
const int row7 = 15; / / the number of the row pin 2
const int row8 = 14; / / the number of the row pin 5
/ / The pin to control COl
const int col1 = 6; / / the number of the col pin 13
const int col2 = 7; / / the number of the col pin 3
const int col3 = 8; / / the number of the col pin 4
const int col4 = 9; / / the number of the col pin 10
const int col5 = 10; / / the number of the col pin 6
const int col6 = 11; / / the number of the col pin 11
const int col7 = 12; / / the number of the col pin 15
const int col8 = 13; / / the number of the col pin 16
void setup () {
int i = 0;
for (i = 2; i <18; i + +)
{
pinMode (i, OUTPUT);
```

```
}
pinMode (row5, OUTPUT);
pinMode (row6, OUTPUT);
pinMode (row7, OUTPUT);
pinMode (row8, OUTPUT);
for (i = 2; i <18; i + +) {
digitalWrite (i, LOW);
}
digitalWrite (row5, LOW);
digitalWrite (row6, LOW);
digitalWrite (row7, LOW);
digitalWrite (row8, LOW);
}
void loop () {
int i;
/ / The row # 1 and col # 1 of the LEDs turn on
digitalWrite (row1, HIGH);
digitalWrite (row2, LOW);
digitalWrite (row3, LOW);
digitalWrite (row4, LOW);
digitalWrite (row5, LOW);
digitalWrite (row6, LOW);
digitalWrite (row7, LOW);
digitalWrite (row8, LOW);
digitalWrite (col1, LOW);
digitalWrite (col2, HIGH);
digitalWrite (col3, HIGH);
digitalWrite (col4, HIGH);
digitalWrite (col5, HIGH);
digitalWrite (col6, HIGH);
digitalWrite (col7, HIGH);
digitalWrite (col8, HIGH);
delay (1000);
/ / Turn off all
for (i = 2; i <18; i + +) {
digitalWrite (i, LOW);
}
delay (1000);
}
```

********** **********

Additional experiments code is as follows:

The letter A is displayed, then the lattice position set by dynamic scanning display.

Code:

```
**********************************************
*********************************
# 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 * /
/ **
 ** "A"
 # Define A {/ /
    {0, 0, 0, 0, 0, 0, 1, 0}, / / 0x02
    {0, 0, 0, 0, 1, 1, 0, 0}, / / 0x0C
    {0, 0, 0, 1, 1, 0, 0, 0}, / / 0x18
    {0, 1, 1, 0, 1, 0, 0, 0}, / / 0x68
    {0, 1, 1, 0, 1, 0, 0, 0}, / / 0x68
    {0, 0, 0, 1, 1, 0, 0, 0}, / / 0x18
    {0, 0, 0, 0, 1, 1, 0, 0}, / / 0x0C
    {0, 0, 0, 0, 0, 0, 1, 0} / / 0x02
}
** /
# 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 = 17; / / the number of the row pin 4
const int row6 = 16; / / the number of the row pin 3
const int row7 = 15; / / the number of the row pin 2
const int row8 = 14; / / 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 <18; i + +)
  {
   pinMode (i, OUTPUT);
  }

  for (i = 2; i <18; i + +) {
   digitalWrite (i, 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));
    }
   }
  }
}
```

Routine 21 tri-color LED RGB Module

Using three-color full color LED manufacturing
Module has three outputs:
1. R, red output,
2. G, green output,
3. B, blue output.
Module features:

Three groups of signal outputs can be programmed microcontroller R, G, B three colors mixed to achieve full-color effect,

Experimental code:

```
int ledPin = 13; / / LED is connected to digital pin 13
int redPin = 11; / / R petal on RGB LED module connected to digital pin 11
int greenPin = 9; / / G petal on RGB LED module connected to digital pin 9
int bluePin = 10; / / B petal on RGB LED module connected to digital pin 10

void setup ()
{
      pinMode (ledPin, OUTPUT); / / sets the ledPin to be an output
      pinMode (redPin, OUTPUT); / / sets the redPin to be an output
      pinMode (greenPin, OUTPUT); / / sets the greenPin to be an output
      pinMode (bluePin, OUTPUT); / / sets the bluePin to be an output
}

void loop () / / run over and over again
{
       / / Basic colors:
```

```
        color (255, 0, 0); / / turn the RGB LED red
        delay (1000); / / delay for 1 second
        color (0,255, 0); / / turn the RGB LED green
        delay (1000); / / delay for 1 second
        color (0, 0, 255); / / turn the RGB LED blue
        delay (1000); / / delay for 1 second

        / / Example blended colors:
        color (255,255,0); / / turn the RGB LED yellow
        delay (1000); / / delay for 1 second
        color (255,255,255); / / turn the RGB LED white
        delay (1000); / / delay for 1 second
        color (128,0,255); / / turn the RGB LED purple
        delay (1000); / / delay for 1 second
        color (0,0,0); / / turn the RGB LED off
        delay (1000); / / delay for 1 second
}

void color (unsigned char red, unsigned char green, unsigned char blue) / / the color generating function
{
        analogWrite (redPin, 255-red);
        analogWrite (bluePin, 255-blue);
        analogWrite (greenPin, 255-green);
}
```

Routine 22.74HC595 experiment

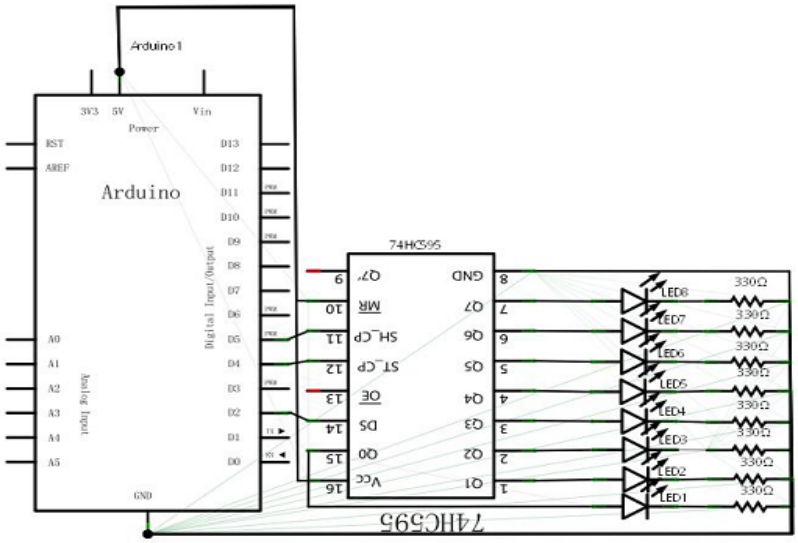
74HC595 simply is an 8-bit shift register and a memory, and tri-state output. Here we use it to control 8 LED lights. Why do we use it 74HC595 to control small lights? There will be a lot of my friends would ask this question, I want to ask is if the mere use our Arduino control 8 small lights, then to take the number of I / O it? The answer is 8, but our Arduino 168 has several I / O ports do? Plus analog interface also 20 bars, eight small lights which takes up too much resources, we aim to use 74HC595 is to reduce the I / O port number to use. We can use later with 74HC595 3 digital I / O port control 8 LED lights not Miya. Here we have to prepare the components.

74HC595 DIP chips * 1
Red M5 DIP LED * 4
Green M5 DIP LED * 4
220Ω line resistance * 8

Bread board * 1

Breadboard Jumper * a tie

We are ready components by connecting the circuit diagram below.



This circuit may seem complicated, we carefully analyze later incorporated by reference in kind will find very simple.

Here is a reference source:

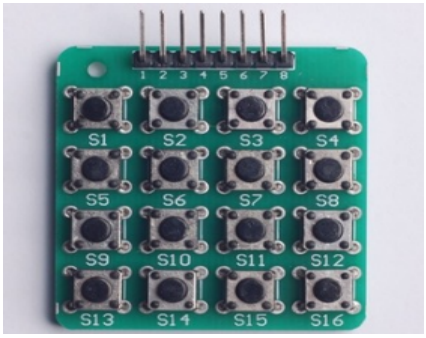
```
int data = 2;
int clock = 4;
int latch = 5;
int ledState = 0;
const int ON = HIGH;
const int OFF = LOW;
void setup ()
{
pinMode (data, OUTPUT);
pinMode (clock, OUTPUT);
pinMode (latch, OUTPUT);
}
void loop ()
{
int delayTime = 100;
for (int i = 0; i <256; i + +)
{
updateLEDs (i);
delay (delayTime);
}
}
```

```
void updateLEDs (int value)
{
digitalWrite (latch, LOW);
shiftOut (data, clock, MSBFIRST, value);
digitalWrite (latch, HIGH);
}
void updateLEDsLong (int value)
{
digitalWrite (latch, LOW);
for (int i = 0; i <8; i + +)
{
int bit = value & B10000000;
value = value << 1;
if (bit == 128) {digitalWrite (data, HIGH);}
else {digitalWrite (data, LOW);}
digitalWrite (clock, HIGH);
delay (1);
digitalWrite (clock, LOW);
}
digitalWrite (latch, HIGH);
}
int bits [] = {B00000001, B00000010, B00000100, B00001000, B00010000, B00100000,
B01000000, B10000000};
int masks [] = {B11111110, B11111101, B11111011, B11110111, B11101111, B11011111,
B10111111, B01111111};
void changeLED (int led, int state)
{
ledState = ledState & masks [led];
if (state == ON) {ledState = ledState | bits [led];}
updateLEDs (ledState);
}
```

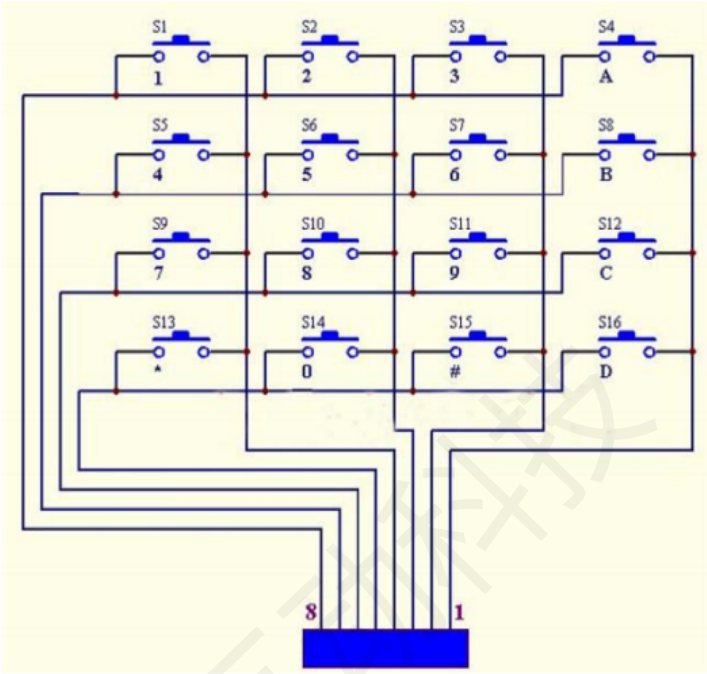
After downloading the program we can see eight small light flashes a wonderful scene.

23.4x4 Button to display the experimental routine

A) Overview



4 * 4 membrane keypad pin, look on the map. The principle is as follows:



Experimental devices

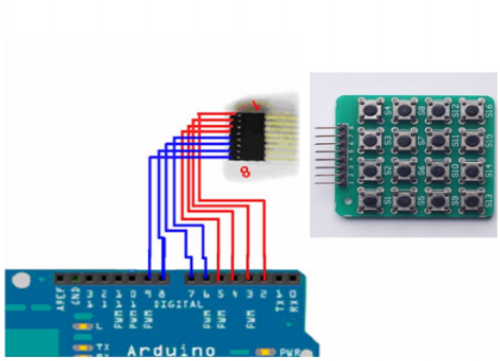
4 * 4 buttons: 1

Colorful experimental breadboard jumper: Several

Bread board: 1

2) test the connection

In the following diagram schematics, 4 * 4 membrane keys will 1-8 2-9 feet turn to the digital. Figure:



Advertising lights wiring experiment

4) code

Program code in the "4x4 button to display the experiment" folder. The first key to library files "Keypad.zip", Xie

Pressure to the arduino IDE installation folder under the "libraries" folder.

Code is as follows:

```
# Include <Keypad.h>
const byte ROWS = 4; / / define four rows
const byte COLS = 4; / / define four
char keys [ROWS] [COLS] = {
{'1 ', '2', '3 ',' A '},
{'4 ', '5', '6 ',' B '},
{'7 ', '8', '9 ',' C '},
{'*', '0 ',' # ',' D '}
};
/ / Connect 4 * 4 keypad row-bit port, the corresponding digital IO ports panel
byte rowPins [ROWS] = {2,3,4,5};
/ / Connect 4 * 4 buttons faithfully port, the corresponding digital IO ports panel
byte colPins [COLS] = {6,7,8,9};
/ / Call the function library function Keypad
Keypad keypad = Keypad (makeKeymap (keys), rowPins, colPins, ROWS, COLS);
void setup () {
Serial.begin (9600);
}
void loop () {
char key = keypad.getKey ();
if (key! = NO_KEY) {
Serial.println (key);
}
}
```

5) Download the program

In accordance with the arduino tutorial program download method will download the program to test the board.

6) Program Function

Download the program to experiment board, open the serial port tool, then press a key on the keyboard, the display on the serial port tool

Shows the key value. Figure, we press the "#", the display is as follows:

4x4 keypad control light experiment

1) Experiment connection:

Using the previous chapter the wiring diagram. And here, we borrow the 13 pin connector on the control board of a small lamp.

2) experimental code:

```
# Include <Keypad.h>
const byte ROWS = 4; / / define four rows
const byte COLS = 4; / / define four
char keys [ROWS] [COLS] = {
{'1 ', '2', '3 ',' A '},
{'4 ', '5', '6 ',' B '},
{'7 ', '8', '9 ',' C '},
{'*', '0 ',' # ',' D '}
};
/ / Connect 4 * 4 keypad row-bit port, the corresponding digital IO ports panel
byte rowPins [ROWS] = {2,3,4,5};
/ / Connect 4 * 4 buttons faithfully port, the corresponding digital IO ports panel
byte colPins [COLS] = {6,7,8,9};
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;
}
}
```

3) The experimental results

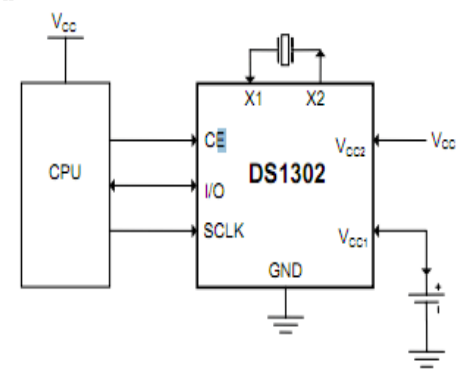
The program, when you press the button "*" hold time, the control panel comes with 13 feet of LED lights will remain lit until the release

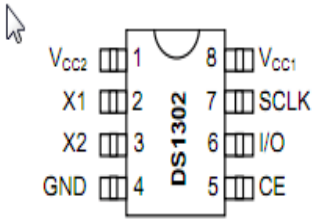
Key "*"; when pressing the key "#", and then released, the 13 feet of small lights will remain lit, then click the "#", the

Small lights go out!

Routine 24.Arduino connection DS1302 clock module

Is the maxim Maxim DS1302 clock module production to support the year, month, day, hour, minute, second, week display. Support for the back-up battery trickle charge. Can. Arduino with only three data cable can be used.





DS1302 data sheet can be seen here: http://www.maxim-ic.com/datasheet/index.mvp/id/2685/t/al

DS1302 circuit is very simple, if Pegboard homemade, you can refer to the following chart:

Connection method:
CE (DS1302 pin5) -> Arduino D5
IO (DS1302 pin6) -> Arduino D6
SCLK (DS1302 pin7) -> Arduino D7
Vcc2 (DS1302 pin1) -> Arduino +5 v
GND (DS1302 pin4) -> Arduino GND

Before use, first extract the files to the following libraries arduino-0023 \ libraries folder below

Arduino connected DS1302
Code Source: http://quadpoint.org/projects/arduino-ds1302
Increases the serial port to adjust the time code
* /
# Include <stdio.h>

```
# Include <string.h>
# Include <DS1302.h>

/ * Interface Definition
CE (DS1302 pin5) -> Arduino D5
IO (DS1302 pin6) -> Arduino D6
SCLK (DS1302 pin7) -> Arduino D7
* /
uint8_t CE_PIN = 5;
uint8_t IO_PIN = 6;
uint8_t SCLK_PIN = 7;

/ * Date variable buffer * /
char buf [50];
char day [10];
/ * Serial data cache * /
String comdata = "";
int numdata [7] = {0}, j = 0, mark = 0;
/ * Create DS1302 object * /
DS1302 rtc (CE_PIN, IO_PIN, SCLK_PIN);


void print_time ()
{
  / * Get the current time from the DS1302 * /
  Time t = rtc.time ();
  / * The week from digital to name * /
  memset (day, 0, sizeof (day));
  switch (t.day)
  {
  case 1: strcpy (day, "Sunday"); break;
  case 2: strcpy (day, "Monday"); break;
  case 3: strcpy (day, "Tuesday"); break;
  case 4: strcpy (day, "Wednesday"); break;
  case 5: strcpy (day, "Thursday"); break;
  case 6: strcpy (day, "Friday"); break;
  case 7: strcpy (day, "Saturday"); break;
  }
  / * The date code format to make up for output buf * /
  snprintf (buf, sizeof (buf), "% s% 04d-% 02d-% 02d% 02d:% 02d:% 02d", day, t.yr, t.mon,
t.date, t.hr, t. min, t.sec);
  / * Output the date to the serial port * /
  Serial.println (buf);
}
```

```
void setup ()
{
  Serial.begin (9600);
  rtc.write_protect (false);
  rtc.halt (false);
}

void loop ()
{

  / * When the serial port has data, the data is spliced into a variable comdata * /
  while (Serial.available ()> 0)
  {
    comdata + = char (Serial.read ());
    delay (2);
    mark = 1;
  }
  / * A comma-separated string comdata decomposition, the decomposition results become converted into digital to numdata [] array * /
  if (mark == 1)
  {
    Serial.print ("You inputed:");
    Serial.println (comdata);
    for (int i = 0; i <comdata.length (); i + +)
    {
      if (comdata [i] == ',' | | comdata [i] == 0x10 | | comdata [i] == 0x13)
      {
        j + +;
      }
      else
      {
        numdata [j] = numdata [j] * 10 + (comdata [i] - '0 ');
      }
    }
    / * Make up the converted numdata time format, write DS1302 * /
    Time t (numdata [0], numdata [1], numdata [2], numdata [3], numdata [4], numdata [5], numdata [6]);
    rtc.time (t);
    mark = 0; j = 0;
    / * Empty comdata variable to wait for the next input * /
    comdata = String ("");
    / * Empty numdata * /
```

```
    for (int i = 0; i <7; i + +) numdata [i] = 0;
  }

  / * Print the current time * /
  print_time ();
  delay (1000);
```

Open the Arduino serial debugger, you can see the current time. If you need to adjust the time, only need to enter the current date and time in port, separated by commas

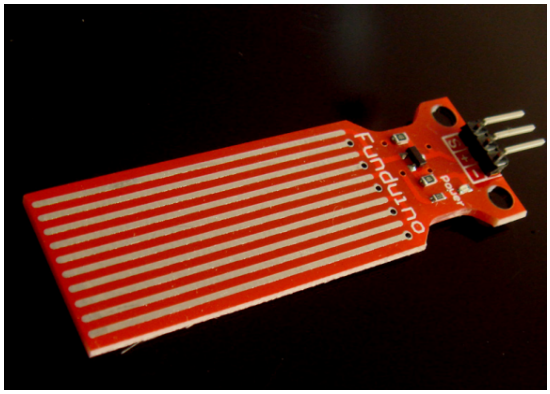
Format is:

Year, month, day, hour, minute, second, day of week

Number of weeks: Sunday = 1, MOnday = 2, ... Saturday = 7
For example, today is at 11:23:40 on November 17, 2011 Thursday
You can fill 2011,11,17,11,22,40,5

Routine 25.Water Sensor Module

1, Introduction
2013 newest Water Sensor is an easy-to-use, compact and lightweight, cost-effective high water level, water droplets identification and detection sensors. This sensor operates
Principle is exposed through a series of parallel wires stitches measure the size of water droplets. And domestic and foreign similar
Products, not only small, powerful, and cleverly designed with the following features:
First, the amount of water to analog conversion;
Second, plasticity, based on the sensor output analog values;
Third, low power consumption, high sensitivity;
Fourth, you can directly to the microprocessor or other logic circuitry connected for a variety of development boards and controllers, such as: Arduino controller, STC MCU, AVR microcontroller and so on.



2, Specifications

1 Product Name: water level sensor

(2) Item No.: K-0135

(3) Operating voltage: DC5V

4 Working current: less than 20mA

5 Sensor Type: Analog

6 Detection Area: 40mm x16mm

7 Production process: FR4 double-sided HASL

8 Fixed Hole Size: 3.0mm

9 user-friendly design: half-moon-slip handle depression

10 Operating temperature: 10 ℃ -30 ℃

11 Working Humidity: 10% ~ 90% non-condensing

12 Product Weight: 3g

13 Product Dimensions: 65mm x 20mm x 8mm

3., Water Sensor module test

We use the Arduino controller to be a test, need to use hardware device is as follows:

1, Arduino controller × 1

2, Arduino sensor expansion board × 1

3, Water Sensor Module × 1

4, 3P sensor cable × 2

5, IR & LED Modue (Red) × 1

6, USB data communication cable × 1

Water Sensor using DuPont line will connect to the Arduino sensor expansion board interface on A1. Using sensors

Line will be connected to the red piranha lights D8 on Arduino sensor expansion board. After completing the hardware connection, the code is compiled

Downloaded to the Arduino inside. Arduino experimental code below.

```
int analogPin = 1; / / level sensor connected to an analog port
int led = 12; / / Piranha LED connected to digital port 12
int val = 0; / / define the variable val initial value is 0
int data = 0; / / define a variable data initial value is 0
void setup ()
{
pinMode (led, OUTPUT); / / define led pin as an output
Serial.begin (9600); / / set the baud rate to 9600
```

```
}

void loop ()
{
val = analogRead (analogPin); / / read the analog value to give the variable val
if (val> 700) {/ / determine the variable val is greater than 700
digitalWrite (led, HIGH); / / variable val is greater than 700, the light Piranha LED
}
else {
digitalWrite (led, LOW); / / variable val is less than 700, the light goes off piranha
}
data = val; / / variable val assigned to the variable data
Serial.println (data); / / serial print variable data
delay (100);
}
```

After the above steps are completed, we test the low water level, see experimental phenomena:
Water level does not reach the warning value, Piranha LED is not lit

Water level reaches the warning level and beyond, piranha lights, initiate alarm

Routine 26. DHT11 temperature and humidity sensor

First, the product introduction

In our daily life, temperature and humidity on our lives has a great impact, especially for factory Production, if we are not well mastered and take relevant measures, then it brings will be a great loss, not

Over Well now, there is a temperature sensor that can measure not only but also measured humidity, it really can solve our problems

Yet. Well, following up on learning how to use it, it brings convenience to your life. . . . .

Second, the module related presentations

DHT11 digital temperature and humidity sensor is a calibrated digital signal output temperature and humidity combined sensor, which

Application-specific modules capture technology and digital temperature and humidity sensor technology to ensure that products with high reliability and excellent

Long-term stability. The product has excellent quality, fast response, anti-interference ability, high cost and other advantages. Single

Wire serial interface that allows quick and easy system integration. Ultra-small size, low power consumption, signal transmission distance

Up to 20 meters, making it to the class of applications and even the most demanding applications is the best choice. Products for the 4-pin single row

Pin package, easy connection.

Third, the technical parameters

Supply voltage: 3.3 ~ 5.5V DC

Output: single-bus digital signal

Measuring range: Humidity 20-90% RH, Temperature 0 ~ 50 ℃

Accuracy: Humidity + -5% RH, temperature + -2 ℃

Resolution: Humidity 1% RH, temperature 1 ℃

Long-term stability: <± 1% RH / Year

IV Notes

1, to avoid the use of the condensation conditions

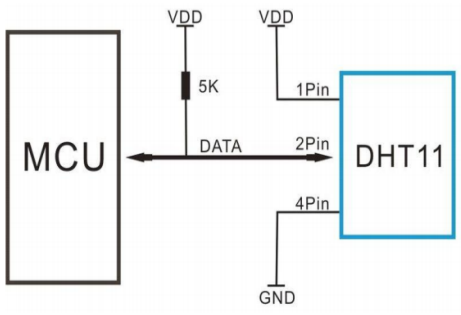
2, long-term storage temperature 10-40 ℃, humidity below 60%

3, the use of power and ground connection to be correct, so as not to damage the sensor

Five, instructions for use

About DHT11 specific timing problems we can refer to its datasheet, see the following modules, such as our company

He connected with the Arduino board



The above is typical of its connection with the processor, the following are specific connection reference
Module "+" Termination +5 V output, "-" Terminal GND, "S" Termination digital port on the 7th pin (Of course, this
You can define your own digital pins); connection is very simple, we only have the following test phase. . . . . .

Six, module function test
Hardware Requirements
Arduino controller × 1
USB data cable × 1
DHT 11 module × 1
In order to facilitate testing, we have written a short test code, for reference only

```
int DHpin = 8;
byte dat [5];
byte read_data ()
{
byte data;
for (int i = 0; i <8; i + +)
{
if (digitalRead (DHpin) == LOW)
{
while (digitalRead (DHpin) == LOW); / / wait for 50us;
delayMicroseconds (30); / / determine the duration of the high level to determine the data is '0 'or '1';
if (digitalRead (DHpin) == HIGH)
data | = (1 << (7-i)); / / high front and low in the post;

while (digitalRead (DHpin) == HIGH); / / data '1 ', wait for the next one receiver;
}
}
return data;
}
void start_test ()
{
```

```
digitalWrite (DHpin, LOW); / / bus down, send start signal;
delay (30); / / delay greater than 18ms, so DHT11 start signal can be detected;
digitalWrite (DHpin, HIGH);
delayMicroseconds (40); / / Wait DHT11 response;
pinMode (DHpin, INPUT);
while (digitalRead (DHpin) == HIGH);
delayMicroseconds (80); / / DHT11 a response, pulled the bus 80us;
if (digitalRead (DHpin) == LOW);
delayMicroseconds (80); / / DHT11 80us after the bus pulled to start sending data;
for (int i = 0; i <4; i + +) / / receives temperature and humidity data, the parity bit is not considered;
dat [i] = read_data ();
pinMode (DHpin, OUTPUT);

digitalWrite (DHpin, HIGH); / / sending data once after releasing the bus, wait for the host to open the next
Start signal;
}
void setup ()
{
Serial.begin (9600);
pinMode (DHpin, OUTPUT);
}
void loop ()
{
start_test ();
Serial.print ("Current humdity =");
Serial.print (dat [0], DEC); / / display the humidity-bit integer;
Serial.print ('.');
Serial.print (dat [1], DEC); / / display the humidity decimal places;
Serial.println ('%');
Serial.print ("Current temperature =");
Serial.print (dat [2], DEC); / / display the temperature of integer bits;
Serial.print ('.');

Serial.print (dat [3], DEC); / / display the temperature of decimal places;
Serial.println ('C');
delay (700);
}
```

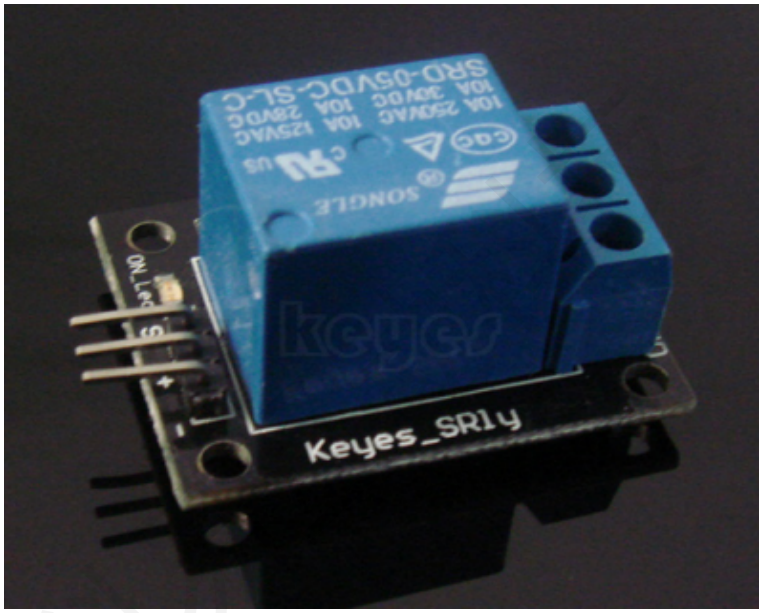
Well, we look at the test code compiler, compile the results we can see, really want to see now
At ambient temperature and humidity in the end is how much of that they are invisible, can be really curious **
Anthracene, we burn the program into Arduino board, and then wait to open the Serial
Monitor window, watching the results came out, wow, is not it a little excited!

We then hand hold it, wait a minute. . . . . What happens to look at the screen?

Look Na, elevated temperature display, next time we can use it to test the temperature of the palm, ha ha. . . . . So then we
Kazakhstan few breaths try it

And imagine the same humidity significantly larger, ha, this thing really is pretty good. . . . Are interested, you can
Be on your own hands frequently be in place, so your heart there at the end.

Routine 27. Relay Module



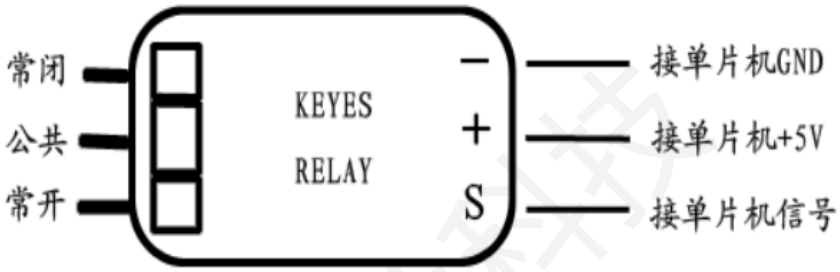
I. Introduction
When the relay is an input (excitation
Reed amount) meet the requirements change, the output circuit in the electrical manipulation occurred predetermined amount charged a step change in an electrical
Makers. The company produces relay module can be connected to 240V AC or 28V DC power into a variety of other electrical parts
Line control. Can be achieved using single-chip timing control switching purposes. Can be used in anti-theft alarm, toys, building
Let other fields. Relay is an electrically controlled device. It has a control system (also known as the input circuit) and the control system

(Also known as the output circuit) the interaction between. Commonly used in automation control circuit, it is actually a small
Current to control a large current operation "automatic switch." Therefore, the circuit automatically adjusts the play, safety protection, transfer
Conversion circuit and so on. Particularly suitable for single-chip control strong electric devices.
In the control and use is also very convenient, just give input corresponding output relay different levels, you can
Achieved by controlling the relay control purposes other devices, in addition, in the multi-channel relay PCB layout on the use of two lines
Layout, user-lead connections. While in the circuit of a DC diode added greatly improved relay
Module to engage current ability to prevent the transistor being burned. In addition, we added a relay this power indicator
Lights (except relay all the way), the indicator is red. In brightest relay also adds a status indicator. Can
To let everyone real-time observation of the relay switch status.
Second, the module classification introduced
1, one relay

A, the main purpose
Relay is a function of the automatic isolation switching elements, are widely used in remote control, telemetry, communications, automatic control,
Mechatronics and power electronic devices, is one of the most important control elements.
Boils down to the following effect:
1) expand the control range: for example, multi-contact relay control signal reaches a certain value, you can not press the contact group
Different forms, and for access, breaking, connected multi-channel circuits.
2) Zoom: for example, sensitive relays, relays, etc., with a very small amount of control, you can control a large
The power of the circuit.
3) Integrated signal: for example, when a plurality of control signals in the form prescribed input multi-winding relay, by comparison mechanized
Together, to achieve the desired control effect.
4) automatic, remote control and monitoring: for example, the automatic device relays together with other appliances, can be composed of program control
Wire line, in order to achieve automatic operation

B, Note
1) Rated voltage: refers to normal working hours relay coil voltage required,
The control circuit is a control voltage. According to the relay model, can be ac
Pressure, it can be a DC voltage.
2) DC resistance: refers to the relay coil DC resistance, measured by the multimeter.
3) Pick-up current: refers to the relay pull action can produce a minimum current. In normal use, the current will be given
Be slightly larger than the pull current, so that the relay can be operated stably. The work of the

coil voltage is applied, generally do not
To more than 1.5 times the rated working voltage, otherwise it will have a greater current to the coil burnt.
4) release current: refers to the relay generates the maximum current release action. When the relay state current is reduced to a
Certain extent, the relay will revert to the release of unpowered state. Then the current is much smaller than the pull current.
5) contact switch voltage and current: is the relay to allow the applied voltage and current. It determines the relay to control
Voltage and current size, use can not exceed this value, it will be very easy to damage the relay contacts.
C, module test
Pin Description below



Description: COM to VCC, NO then we have to control the LED anode, which
Like when the relay turns on, LED lights will be lit;
To complete the look of this test must be prepared to what what they specifically
Arduino controller × 1
USB data cable × 1
1 relay module × 1
Led indicator × 1
The resistance of the resistance 330 × 1
Of course, we have the following physical connections for a specific reference
Well, Here is a simple relay control test procedure:
int relay = 10; / / relay turns trigger signal - active high;

```
void setup ()
{
pinMode (relay, OUTPUT); / / Define port attribute is output;
}
void loop ()
{
digitalWrite (relay, HIGH); / / relay conduction;
delay (1000);
digitalWrite (relay, LOW); / / relay switch is turned off;
delay (1000);
}
```

Program Description: The program notes in the conduction and disconnection refers to the way that we want that we are using the NO side,

When S relay switches into high and hit the NO side, the switch is turned on, connected to the LED will be lit, otherwise the switch

Hit the NC side, NO direction disconnect, LED light goes out;

You will see the test results led lights flashing interval 1s;

Routine 28.I2C LCD1602 Module

- I. Introduction

As we all know, LCD and LED displays like although they greatly enriched the human-computer interaction, but

We have a common characteristic is that is connected to the controller must take more IO line, which for some peripheral interface

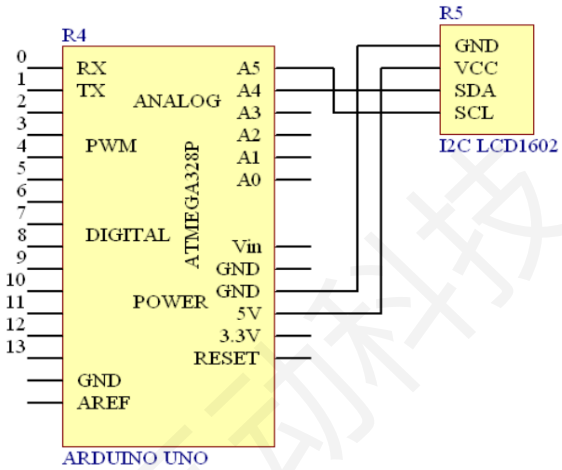
Rich enough controllers are a big problem, but also limits the controller's other functions, for this, we band

I2C interface LCD1602 can be a good solution to this problem, but it is relatively simple to use.

Second, product characteristics

Interface: I2C Interface I2C Address: 0x27
Pin definition: VCC, GND, SDA, SCL
Working voltage: 5V Dimensions: 27.7mm × 42.6mm
Contrast adjustment: via potentiometer
IO interface using only two
Third, the use
Because the module is I2C interface, so we re-use I2C protocol to follow, to include the appropriate header files
Job, and must be added to the library, will be described below.

Or take a look at how it is connected to Arduino control board



So when in use as long as we follow this schematic wiring on it, details follow introduction.
Fourth, the module test
Hardware Requirements
1, Arduino controller × 1
2, USB data cable × 1
3, I2C LCD1602 Module × 1

Testing requirements on so many things, the same as last time, this time we do a simple test, we displayed on the LCD
His characters "www.KEYES.com";
Well, look at the test code
include <Wire.h>
include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd (0x27, 16,2); / / set the LCD address to 0x27 for a 16 chars and 2

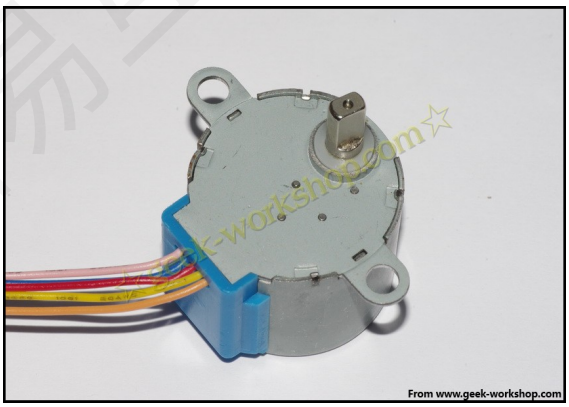
```
line display
void setup ()
{
lcd.init (); / / initialize the lcd
/ / Print a message to the LCD.
lcd.backlight ();
lcd.print ("www.KEYES.com");
}
void loop ()
{
}
```

When the test code sure to Wire, LiquidCrystal_I2C two header files to add to our library
This is a guarantee of the module!

Let's look under the experimental results, there are screenshots
Well, the test results are normal, normal use, then we would not be so next time you use LCD strenuous.

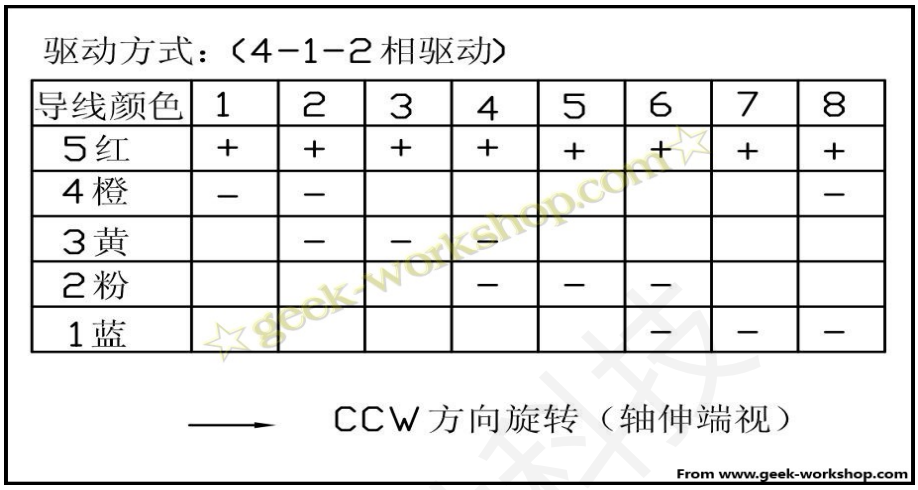
Routine 29. Stepper motor test

Stepper motor is an electrical pulse into the angular displacement of the actuator. Plainly speaking: When receiving a stepper drive pulse signal, it will drive a stepper motor to set the direction of rotation of a fixed angle (and step angle). You can control the number of pulses to control the angular displacement, so as to achieve the purpose of accurate positioning; and you can also control the pulse frequency to control the motor rotation speed and acceleration so as to achieve the purpose of speed.



The following experiment was used in the stepping motor

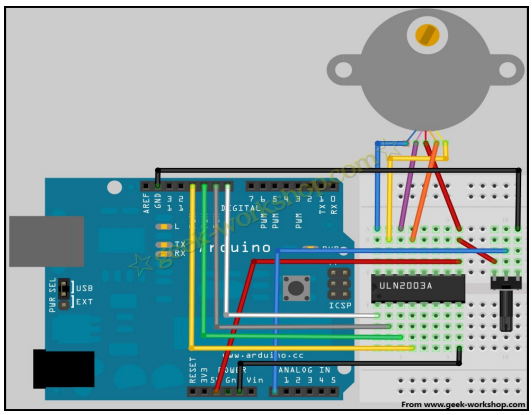
Stepping motor is used must be carefully view the documentation, confirm that the four-phase or two-phase, how to connect each line, this experiment is to use four-phase stepper motor, different colored lines defined as follows:



导线颜色	1	2	3	4	5	6	7	8
5红	+	+	+	+	+	+	+	+
4橙	–	–						–
3黄		–	–	–				
2粉				–	–	–		
1蓝						–	–	–

Stepping Motor
Diameter: 28mm
Voltage: 5V
Step Angle: 5.625 x 1/64
Reduction ratio: 1/64
5 Line 4 phase can be driven by ordinary uln2003 chip can also be connected into a two-phase use

The stepper motor load power consumption of 50mA or less, with 64 times the speed reducer, the output torque is relatively large, can drive heavy loads, very suitable for development board. NOTE: This step motor with a reducer 64 times, and without gear stepper motor compared to the speed becomes slower, to facilitate the observation, in the output shaft of a small cardboard glue.



Stepper motor (four-phase five-wire) driver board (UL2003) test plate

Stepper motor driver board (UL2003) test plate

Dimensions: 31 × 35mm
Hardware connection diagram is as follows

The code is downloaded to the control board to see the effect arduino

```
/ *
* Stepper motor rotating follower potentiometer
* (Or other sensors) using the input analog port number 0
* Use the arduino IDE comes Stepper.h library files
* /

# Include <Stepper.h>

/ / Set here is the number of revolution of the stepper motor step
# Define STEPS 100

/ / Attached to set the number of steps of the stepper motor and pin
Stepper stepper (STEPS, 8, 9, 10, 11);

/ / Define a variable to store the history of reading
int previous = 0;

void setup ()
{
  / / Set the motor speed of 90 per minute step
  stepper.setSpeed (90);
}

void loop ()
{
  / / Get the sensor readings
  int val = analogRead (0);

  / / Move to the current reading minus the number of steps historical readings
  stepper.step (val - previous);

  / / Save history readings
```

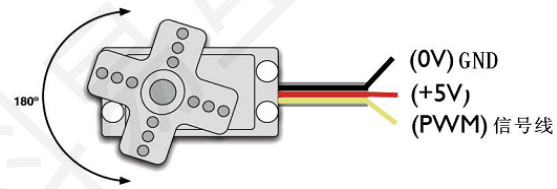
```
 previous = val;
}
/ *
 * Stepper motor rotating follower potentiometer
 * (Or other sensors) using the input analog port number 0
 * Use the arduino IDE comes Stepper.h library files
 * /

# Include <Stepper.h>

/ / Set here is the number of revolution of the stepper motor step
# Define STEPS 100

/ / Attached to set the number of steps of the stepper motor and pin
Stepper stepper (STEPS, 8, 9, 10, 11);

/ / Define a variable to store the history of reading
int previous = 0;

void setup ()
{
  / / Set the motor speed of 90 per minute step
  stepper.setSpeed (90);
}

void loop ()
{
  / / Get the sensor readings
  int val = analogRead (0);

  / / Move to the current reading minus the number of steps historical readings
  stepper.step (val - previous);

  / / Save history readings
  previous = val;
}
```

Routine 30. Servo control experiments

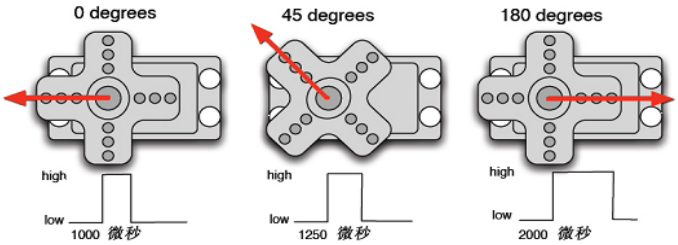
A position servo servo drive, mainly by the housing, a circuit board, coreless motors, gears and the

position detector of the composition. Its working principle is the microcontroller sends a signal to the receiver or servos, which has an internal reference circuit, resulting in a period of 20ms, the width of 1.5ms reference signal will get the DC bias voltage and the potentiometer voltage comparator, access differential voltage output. Judgment of the IC via the circuit board the direction of rotation and then starts rotating non-core motor drive, the power transmitted through the reduction gear arm, and back to the signal from the position detector to determine whether the positioning has been reached. For changing the angle and that require a control system can be maintained. When the motor speed is constant, the reduction gear driven by cascading rotary potentiometer so that the voltage difference is 0, the motor stops rotating. General steering angle of rotation range is 0 to 180 degrees.

There are many standard servos, but all the servos have an external three lines, respectively, with brown, red, orange and three colors to distinguish, because servos different brands, colors will vary, brown is the ground wire, red for positive power supply line, orange for the signal lines.



Angle of rotation of the servo by adjusting the PWM (pulse width modulation) signal duty cycle to be achieved, standard PWM (pulse width modulation) signal period is fixed at 20ms (50Hz), in theory, should be distributed at 1ms pulse width to between 2ms, but, in fact, be 0.5ms to 2.5ms pulse width between pulse width and servo corner 0 ° ~ 180 ° correspond. It is worth noting the place, because servos different brands, for the same signal, different brands of servo rotation angle will be different.



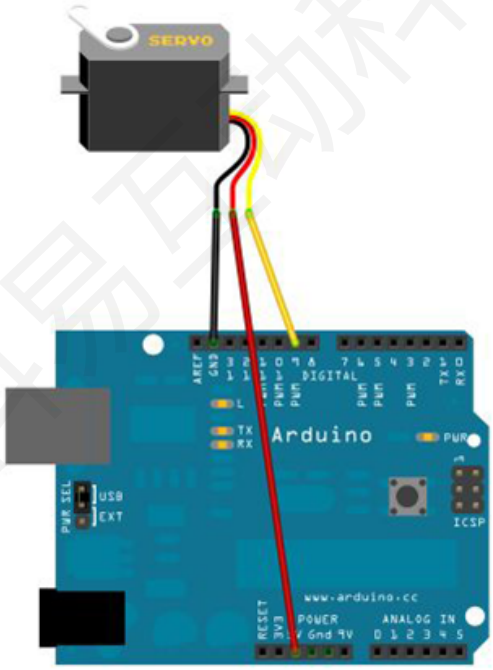
After understanding the basics since we can learn to control a servo, and this experiment need only servos a few components, jumpers a bundle on it.

servos * 1

Breadboard Jumper * a tie

Arduino control with steering in two ways, one is through the Arduino ordinary digital sensor interface produce different duty cycle square wave, analog servo positioning generate a PWM signal, the second is the direct use of Arduino comes Servo function steering control, this control method has the advantage of programming, the disadvantage is only control two-way steering, because Arduino own function only use digital 9,10 interface. Arduino drive capability is limited, so when you need to control one or more servos require an external power supply.

Method One



The number 9 on the interface connected to the steering gear.

The servo write a program that lets the user enter the number of the rotation angle corresponding to the number of position and angle to the screen printing.

Reference source A:

```
int servopin = 9 ;/ / define the digital interface to connect the servo servo signal line 9
int myangle ;/ / define the angle variables
int pulsewidth ;/ / define variable pulse width
int val;
void servopulse (int servopin, int myangle) / / define a pulse function
{
pulsewidth = (myangle * 11) +500 ;/ / the angle value into a pulse width 500-2480
digitalWrite (servopin, HIGH) ;/ / the servo interface level to high
delayMicroseconds (pulsewidth) ;/ / number of microseconds delay pulse width value
digitalWrite (servopin, LOW) ;/ / the servo interface level to Low
delay (20-pulsewidth/1000);
}
void setup ()
{
pinMode (servopin, OUTPUT) ;/ / set servo interface output interface
Serial.begin (9600) ;/ / connect to the serial port, the baud rate is 9600
Serial.println ("servo = o_seral_simple ready");
}
void loop () / / will be 0-9's 0-180 number into perspective, and let the number of times the corresponding LED flashes
{
val = Serial.read () ;/ / read the value of the serial port
if (val> '0 '&& val <= '9')
{
val = val-'0 ';/ / will be converted to numeric variables characteristic quantities
val = val * (180/9) ;/ / will figure into perspective
Serial.print ("moving servo to");
Serial.print (val, DEC);
Serial.println ();
for (int i = 0; i <= 50; i + +) / / give enough time to let it go to the steering angle specified
{
servopulse (servopin, val) ;/ / reference pulse function
}
}
}
```

Method Two

First detailed analysis of the Arduino Servo function and its own statement, to introduce a few common statement function steering bar.

1, attach (Interface) - set the servo interface, only the number 9 or 10 interfaces available.

2, write (angle) - for setting the steering angle of rotation of statements, the settable angle range is 0 ° to 180 °.

3, read () - used to read the steering angle of the statement, it is understood to read the last one write () command
Value.
4, attached () - determine the servo parameters have been sent to the servo where interface.
5, detach () - the steering gear and its interface is separated from the interface (figure 9 or 10 interfaces) may continue to be used as a PWM interface.
Note: The above statement writing format is "steering variable name. Specific statement ()" For example: myservo.attach (9).
Will still be connected to the digital servo 9 connector on top.
Reference source B:

```
# Include <Servo.h> / / define header files, there is one thing to note, you can directly click on the menu bar at the Arduino software Sketch> Importlibrary> Servo, Servo function call, you can also directly enter the # include <Servo.h>, But at the input to the attention of the # include and <Servo.h> should be a space between, otherwise a compile-time error.
Servo myservo ;/ / define a variable name servos
void setup ()
{
myservo.attach (9) ;/ / define Servo Interface (9,10 all OK, shortcomings can only control 2)
}
void loop ()
{
myservo.write (90) ;/ / set the steering angle of rotation
}
```

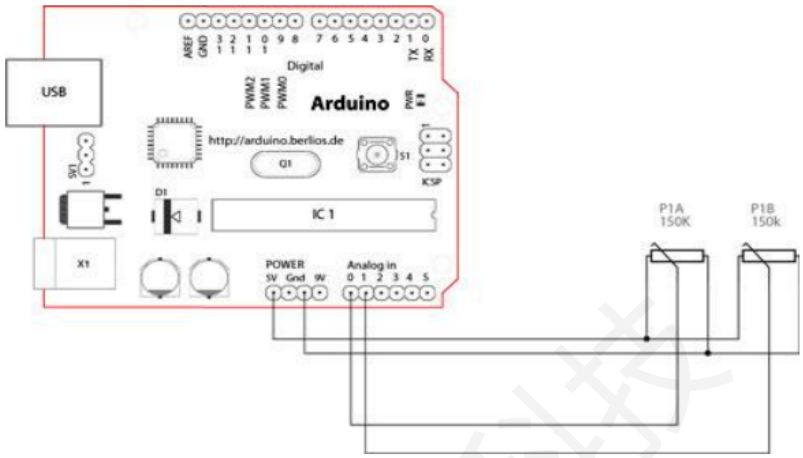
These are the two methods of steering control, their advantages and disadvantages according to their own preferences and the need for choice.

Routine 31. Game joystick axis sensor module

First, the product description
The company produces PS2 game joystick axis sensor module consists of using original quality metal PS2 joystick potentiometer system
For, with (X, Y) 2-axis analog output, (Z) 1 digital output channel button. With Arduino sensor expansion board can be made
For remote control and other interactive work. In addition the product in order to allow customers to more easily fit arduino expansion boards and other standard interfaces
Mouth, in the design of the X, Y, Z-axis circuit leads individually, you can use three dedicated lines really pin ARDUINO
Plug into the expansion board for use. It is convenient.
Second, product characteristics

It is like a game console joystick, you can control the joystick module input x, y, z of

PS2 joystick game controller module Joystick
Values, and to achieve a particular value in a function, it can be considered a combination of buttons and a potentiometer. Data
Type of x, y dimension for the analog input signal is a digital input signal z dimension, therefore, x and y connected to the analog port
Pin sensor end, and z port is connected to the digital port.
Third, the use On how to use, we first look at how it works now, so we know it is there in the end
How, which we find it helpful to use, there is a functional diagram below, we take a look



Now we should clear it, in fact, it is a potentiometer Well, x, y dimension of the data output is analog
Port readout voltage value, is not a little surprised. Of course, this is not shown above, z-dimensional data output, in fact, it is more
Simple, we know that z-dimensional output only 0 and 1, then it can be achieved through a button bar. Now on we should
Surface of saying, it is a potentiometer and button combination (To be honest, if you do not understand it just to see that
Sentence is a bit foggy it?).
After reading the chart I believe we all know how to use it right Arduino, x, y dimension we received two analog ports

Read their values, and z dimensions we are to the digital port, so that the line, plus the power and ground, so fine. . . .
Fourth, the module test
Let's look at this test what things we have, in fact, not much. . . .
Arduino controller × 1

USB data cable × 1
Game sensor module × 1
Here x I connected an analog port 0, y even an analog port 1, z I connect to the digital port 7, the relevant port
No. You can look at the individual situation, but properties can not be wrong. Code is as follows

```
int sensorPin = 5;

int value = 0;
void setup () {
pinMode (7, OUTPUT);
Serial.begin (9600);
}
void loop () {
value = analogRead (0);
Serial.print ("X:");
Serial.print (value, DEC);
value = analogRead (1);
Serial.print ("| Y:");
Serial.print (value, DEC);
value = digitalRead (7);
Serial.print ("| Z:");
Serial.println (value, DEC);
delay (100);
}
```

Program function: it can play sensor status (x, y, z three-dimensional data) in real time response to the computer screen (we use the
Serial Monitor window), here I cut a figure for everyone to look at it

Is not it, the test is successful, or more I play, we have the opportunity to see for yourself. . . . .
I believe we carefully read the above schematic would say, in fact, I can also DIY
A yo, but might not look so good, it is not so smoothly control it, ah, yes, imitation principle,
Prepare their own two potentiometers and a key entirely possible Yeah, this would leave you complete it, the following test code,
We can take a try. . .

```
int JoyStick_X = 0; / / x
int JoyStick_Y = 1; / / y
int JoyStick_Z = 3; / / key
void setup ()
{
pinMode (JoyStick_X, INPUT);
pinMode (JoyStick_Y, INPUT);
pinMode (JoyStick_Z, INPUT);
Serial.begin (9600); / / 9600 bps
}
```

```
void loop ()
{
int x, y, z;
x = analogRead (JoyStick_X);
y = analogRead (JoyStick_Y);
z = digitalRead (JoyStick_Z);
Serial.print (x, DEC);
Serial.print (",");
Serial.print (y, DEC);
Serial.print (",");
Serial.println (z, DEC);
delay (100);
}
```

Routine 32.Infrared remote control

1, the infrared receiver introduction

First, what is the infrared receiver?

Infrared remote control signals is a series of binary pulse code. In order to make the wireless transmission from another infrared signal interference, which often are first modulated on a particular carrier frequency, and then emitted by the infrared emitting diodes, infrared receiver and other clutter filter will have , Gua receiving the particular frequency signal and restore the binary pulse code, which is demodulated.

Second, the principle

Built-in infrared receiver tube will launch tube out Qie weak optical signal into an electrical signal, this signal is amplified by the amplifier inside the IC, and then through automatic gain control, band pass filter, demodulator, waveform shaping restore the remote control transmitter out of the original encoded signal via the receiver input to the output pin coded identification on the electrical circuit.

Third, the infrared receiver pin and connection

Infrared receiver has three pins as shown below:

When used to receive analog port VOUT, GND received experimental board GND, VCC received experimental board +5 v.

Infrared remote experiment

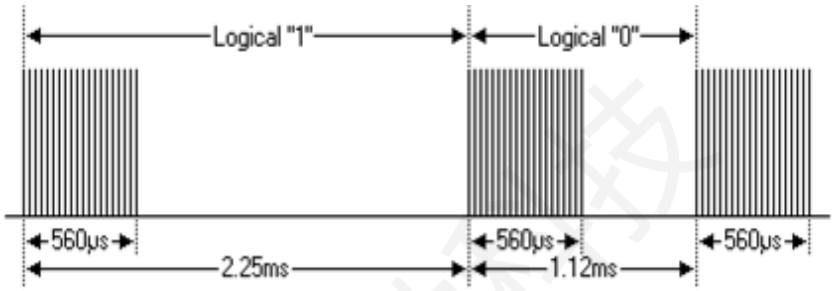
1, the experimental device

□ □ IR Remote Control: 1

□ □ infrared receiver: 1

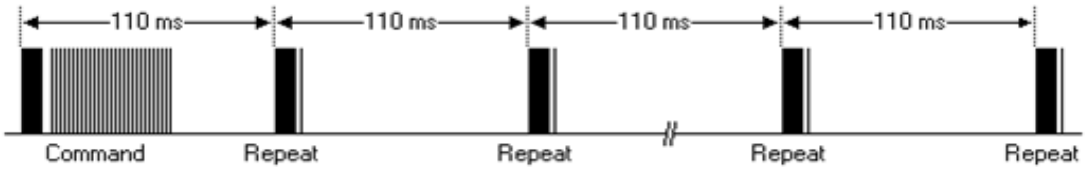
□ □ LED lights: 6
□ 220Ω resistance: 6
□ □ colorful bread line: Several

2, the experimental connection
First, the board is connected; followed by the infrared receiver connected as described above, will receive digital 11 VOUT pin through a resistor connected to the LED lights digital pins 2,3,4,5,6,7. Sample Returning to complete the circuit part of the connection.
3, the experimental principle
To decode of a remote control the remote control must know the encoding. This product uses the controller's code manner: NEC protocol. NEC to introduce the following protocol:
• NEC protocol: Features: (1) 8 address bits, 8-bit command
(2) In order to address bits and reliability command bits are transmitted twice
(3) Pulse Position Modulation
(4) the carrier frequency 38khz
(5) Every one Qie time of 1.125ms 2.25ms Ring
• logic 0 and 1 is defined as shown below



Have agreed as follows:
• immediately release button is pressed the transmit pulse:

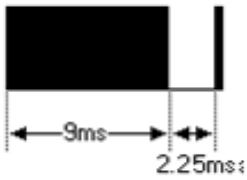
The above picture shows a typical pulse NEC protocol sequence. Note: This is first sent LSB (least significant bit) of the agreement. Qie pulse propagation in the above address 0x59 command 0x16. A message is generated by a high level 9ms start, followed by a low level of 4.5ms, (composed of lead back to two-level search yards) and then by the address code and command code. Address and command transfer twice. Second invert all bits, can be used for the acknowledgment received message used. Total transmission time is constant, since every point of its length to take anti-duplication. If you're not interested, you can ignore this reliability negated, you can also expand the address and command to every 16!



Some time before the release button is pressed the transmit pulse:

A command is sent once, even if the buttons on the remote control is still pressed. When the button has been pressed, the first 110ms Qie pulse as the previous figure, and then transmitted once every 110ms repetitive code. Repeat the code to return a high level by a 9ms 2.25ms pulse and a low level and high level composition 560μs Qie.

• Repeat Pulse



Note: Integration of the pulse waveform into the receiver after the reception in advance as to the integration In Assistant decoding, signal amplification and shaping, therefore to note: in the absence of infrared signal, its output is high, a signal is low when flat, so the output signal level is just the opposite transmitter. Receiver pulse we can see through the oscilloscope, waveform understand the program combines see.

Code

```
# Include <IRremote.h>
int RECV_PIN = 11;
int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
int LED5 = 6;
int LED6 = 7;
long on1 = 0x00FFA25D;
long off1 = 0x00FFE01F;
long on2 = 0x00FF629D;
long off2 = 0x00FFA857;
long on3 = 0x00FFE21D;
long off3 = 0x00FF906F;
long on4 = 0x00FF22DD;
long off4 = 0x00FF6897;
```

```
long on5 = 0x00FF02FD;
long off5 = 0x00FF9867;
long on6 = 0x00FFC23D;
long off6 = 0x00FFB047;
IRrecv irrecv (RECV_PIN);
decode_results results;
/ / Dumps out the decode_results structure.
/ / Call this after IRrecv :: decode ()
/ / Void * to work around compiler issue
/ / Void dump (void * v) {
/ / Decode_results * results = (decode_results *) v
void dump (decode_results * results) {
  int count = results-> rawlen;
  if (results-> decode_type == UNKNOWN)
    {
     Serial.println ("Could not decode message");
    }
  else
   {
    if (results-> decode_type == NEC)
      {
       Serial.print ("Decoded NEC:");
      }
    else if (results-> decode_type == SONY)
      {
       Serial.print ("Decoded SONY:");
      }
    else if (results-> decode_type == RC5)
      {
       Serial.print ("Decoded RC5:");
      }
    else if (results-> decode_type == RC6)
      {
       Serial.print ("Decoded RC6:");
      }
     Serial.print (results-> value, HEX);
     Serial.print ("(");
     Serial.print (results-> bits, DEC);
     Serial.println ("bits)");
   }
     Serial.print ("Raw (");
     Serial.print (count, DEC);
     Serial.print ("):");
```

```
  for (int i = 0; i <count; i + +)
    {
     if ((i% 2) == 1) {
     Serial.print (results-> rawbuf [i] * USECPERTICK, DEC);
     }
    else
     {
     Serial.print (- (int) results-> rawbuf [i] * USECPERTICK, DEC);
     }
    Serial.print ("");
     }
     Serial.println ("");
     }

void setup ()
 {
  pinMode (RECV_PIN, INPUT);
  pinMode (LED1, OUTPUT);
  pinMode (LED2, OUTPUT);
  pinMode (LED3, OUTPUT);
  pinMode (LED4, OUTPUT);
  pinMode (LED5, OUTPUT);
  pinMode (LED6, OUTPUT);
  pinMode (13, OUTPUT);
  Serial.begin (9600);

  irrecv.enableIRIn (); / / Start the receiver
 }

int on = 0;
unsigned long last = millis ();

void loop ()
{
  if (irrecv.decode (& results))
   {
    / / If it's been at least 1/4 second since the last
    / / IR received, toggle the relay
    if (millis () - last> 250)
      {
       on =! on;
/ / DigitalWrite (8, on? HIGH: LOW);
       digitalWrite (13, on? HIGH: LOW);
       dump (& results);
```

```
    }
  if (results.value == on1)
    digitalWrite (LED1, HIGH);
  if (results.value == off1)
    digitalWrite (LED1, LOW);
  if (results.value == on2)
    digitalWrite (LED2, HIGH);
  if (results.value == off2)
    digitalWrite (LED2, LOW);
  if (results.value == on3)
    digitalWrite (LED3, HIGH);
  if (results.value == off3)
    digitalWrite (LED3, LOW);
  if (results.value == on4)
    digitalWrite (LED4, HIGH);
  if (results.value == off4)
    digitalWrite (LED4, LOW);
  if (results.value == on5)
    digitalWrite (LED5, HIGH);
  if (results.value == off5)
    digitalWrite (LED5, LOW);
  if (results.value == on6)
    digitalWrite (LED6, HIGH);
  if (results.value == off6)
    digitalWrite (LED6, LOW);
  last = millis ();
  irrecv.resume (); / / Receive the next value
 }
}
```

Fifth, the program features

Emitted from the remote control decoding coded pulses, based on the decoding result of the appropriate action. Back like you can use the remote control remote control of your device, let it listen to your command.

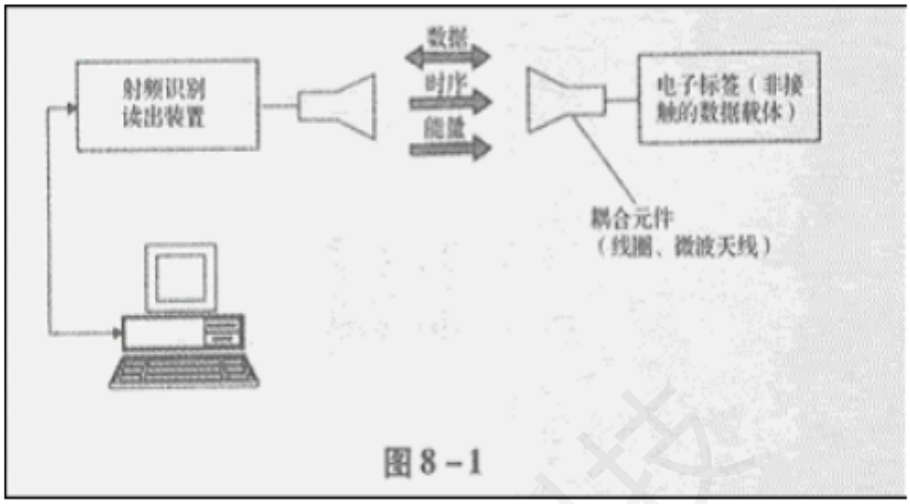
Routine 33.RFID reader experiment

A) Overview

RF technology is also referred to as RFID, RFID is English radio frequency identification "acronym, called radio frequency identification

Do not technology, referred to as radio frequency technology.

RFID works



The basic model of radio frequency identification system shown in Fig.

Wherein, the electronic tag is also known as radio frequency tags, transponder, data carrier; reader, also known as reading means, a scanner,

Communicator, the reader (depending on whether the wireless electronic tag data to be rewritten).

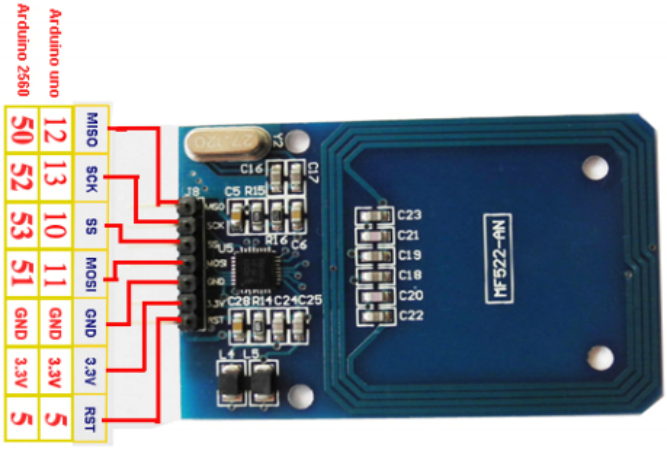
Electronic tags and readers through the coupling element between

The RF signal space (non-contact) is coupled, the coupling channel, according to the timing relationship, to achieve energy transfer, the data exchange

Replaced.

This module, we must use the +3.3 V power supply, otherwise it will burn modules.

2) specific wiring diagram below.



4) Program code:

RFID reader experiment

A) Overview

RF technology is also referred to as RFID, RFID is English radio frequency identification "acronym, called radio frequency identification

Do not technology, referred to as radio frequency technology.

RFID works

The basic model of radio frequency identification system shown in Fig.

Wherein, the electronic tag is also known as radio frequency tags, transponder, data carrier; reader, also known as reading means, a scanner,

Communicator, the reader (depending on whether the wireless electronic tag data to be rewritten). Electronic tags and readers through the coupling element between

The RF signal space (non-contact) is coupled, the coupling channel, according to the timing relationship, to achieve energy transfer, the data exchange

Replaced.

This module, we must use the +3.3 V power supply, otherwise it will burn modules.

2) specific wiring diagram below.

4) Program code:

```
# Include <SPI.h>
# Define uchar unsigned char
# Define uint unsigned int
/ / Maximum length of the array
# Define MAX_LEN 16
/ / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /
/ / Set the pin
/ / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /
const int chipSelectPin = 10; / / if the control panel for the UNO, 328,168
const int chipSelectPin = 53; / / If the control panel is mega 2560,1280
const int NRSTPD = 5;
/ / MF522 command word
# Define PCD_IDLE 0x00 / / NO action; cancel the current command
# Define PCD_AUTHENT 0x0E / / authentication key
# Define PCD_RECEIVE 0x08 / / receive data
# Define PCD_TRANSMIT 0x04 / / Transmit Data
# Define PCD_TRANSCEIVE 0x0C / / Send and receive data
# Define PCD_RESETPHASE 0x0F / / Reset
# Define PCD_CALCCRC 0x03 / / CRC calculation
/ / Mifare_One card command word
# Define PICC_REQIDL 0x26 / / find the antenna area does not enter hibernation
# Define PICC_REQALL 0x52 / / find all the cards antenna area
# Define PICC_ANTICOLL 0x93 / / anti-collision
```

```
# Define PICC_SElECTTAG 0x93 / / election card
# Define PICC_AUTHENT1A 0x60 / / authentication key A
# Define PICC_AUTHENT1B 0x61 / / authentication key B
# Define PICC_READ 0x30 / / Read Block
# Define PICC_WRITE 0xA0 / / write block
# Define PICC_DECREMENT 0xC0 / /
# Define PICC_INCREMENT 0xC1 / /
# Define PICC_RESTORE 0xC2 / / transfer block data to the buffer
# Define PICC_TRANSFER 0xB0 / / save the data in the buffer
# Define PICC_HALT 0x50 / / Sleep
/ / And MF522 communication error code is returned when
# Define MI_OK 0
# Define MI_NOTAGERR 1
# Define MI_ERR 2
/ / ------------------ MFRC522 register ---------------
/ / 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 bytes card serial number, the first 5 bytes for the checksum byte
uchar serNum [5];
uchar writeDate [16] = {'T', 'e', 'n', 'g', '', 'B', 'o', 0, 0, 0, 0, 0, 0, 0, 0, 0};
/ / Sector A password, 16 sectors, each sector password 6Byte
```

```
uchar sectorKeyA [16] [16] = {{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
{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, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xff, 0x07, 0x80, 0x69, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
};
void setup () {
Serial.begin (9600); / / RFID reader SOUT pin connected to Serial
RX pin at 2400bps
/ / Start the SPI library:
SPI.begin ();
pinMode (chipSelectPin, OUTPUT); / / Set digital pin 10 as OUTPUT to connect
it to the RFID / ENABLE pin
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; / / select the operating block addresses 0 to 63
/ / Find the card, back card type
status = MFRC522_Request (PICC_REQIDL, str);
if (status == MI_OK)
{
}
/ / Anti-collision, return card serial number 4 bytes
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 (serNum [1], BIN);
Serial.print (serNum [2], BIN);
```

```
Serial.print (serNum [3], BIN);
Serial.print (serNum [4], BIN);
Serial.println ("");
}
/ / Election card, return to card capacity
RC_size = MFRC522_SelectTag (serNum);
if (RC_size! = 0)
{}
/ / Write data card
blockAddr = 7; / / data block 7
status = MFRC522_Auth (PICC_AUTHENT1A, blockAddr, sectorKeyA [blockAddr / 4],
serNum); / / Certification
if (status == MI_OK)
{
/ / Write data
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);
/ / Write data
blockAddr = blockAddr - 3;
status = MFRC522_Write (blockAddr, writeDate);
if (status == MI_OK)
{
Serial.println ("OK!");
}
}
/ / Reader
blockAddr = 7; / / data block 7
status = MFRC522_Auth (PICC_AUTHENT1A, blockAddr,
sectorNewKeyA [blockAddr / 4], serNum); / / Certification
if (status == MI_OK)
{
/ / Read data
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]);
}
Serial.println ("");
```

```
}
}
Serial.println ("");
MFRC522_Halt (); / / command card into hibernation
}
/ *
* Function Name: Write_MFRC5200
* Description: MFRC522 of a register to write a byte of data
* Input Parameters: addr - register address; val - the value to be written
* Return value: None
* /
.....
```

Margins due to the data is very long, so the code, where not all the shows.

This experiment, when the IC card is approached, RFID module will write data to the IC card and RFID module and then read from the IC card

The data, and display on the monitor window.

Routine 34. Access Control System Experiment

A) Overview

Experiments in this chapter, we will use the RFID module, and relays, as well as red, green LED lights developed a door

Ban system.

When we took the pre-set password IC card, RFID module to the card, if the password is incorrect, then the relay

Normally open, the red light, which means that the door does not open; if the password is correct, the relay is closed, the green light lit up, which means to open the door.

Practical application, simply take two small lights circuit slightly modified, even to the door of the controller on it.

Wiring diagram, please combine the previous chapter "RFID reader experiment" and Chapter 17 of the "relay control experiment" Connection

Figure.

2) the program code:

Please refer to the "Access Control System Experiment" program folder.

How to use: 1) first download "SetPassword.pde" code to the control panel, connect the RFID module,

Open the serial port of the monitoring window, the IC card close to the module, when the show "The password has been changed!" (Secret

Code has been modified completed).

2) then download "DoorCon.pde" code to the control panel, while the relay control experiments cable

Lu is also connected to. When the IC card is close to module, RFID module detects the card, if the password is inconsistent, then the relay normally open,

Red light, the door closed; if the password is the same, then the relay is closed, the green light, the door is opened.

Through this experiment, we can be creative, would also integrated into the keyboard, to achieve more extensive project.

Next, we rely on free play it!