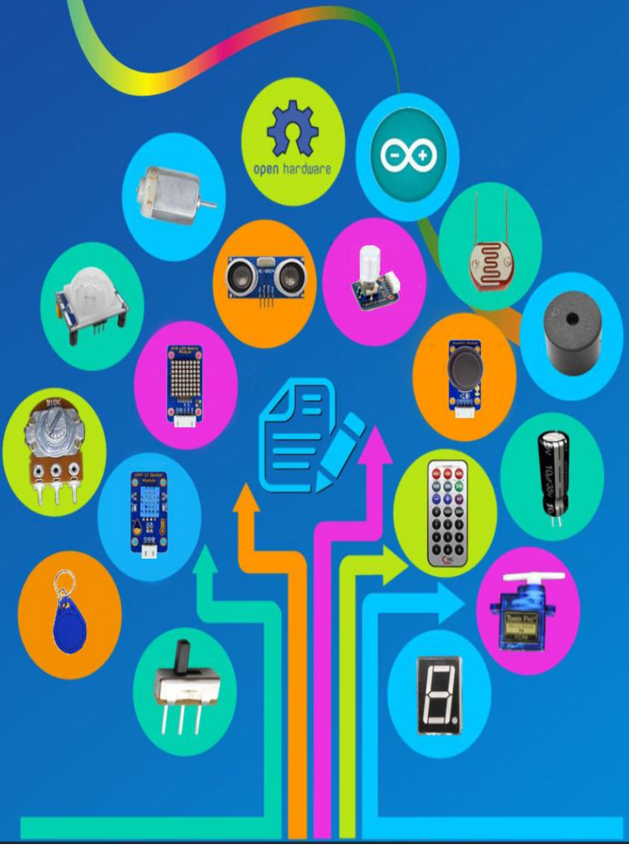
Adeept

Adeept RFID Kit For Arduino MEGA 2560

Sharing Perfects Innovation



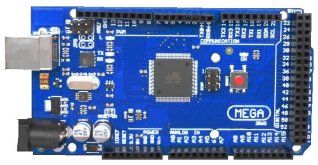
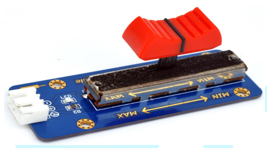
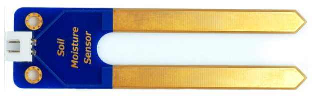
Preface

Adeept is a technical service team of open source software and hardware. Dedicated to applying the Internet and the latest industrial technology in open source area, we strive to provide best hardware support and software service for general makers and electronic enthusiasts around the world. We aim to create infinite possibilities with sharing. No matter what field you are in, we can lead you into the electronic world and bring your ideas into reality.

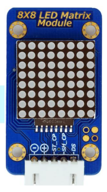
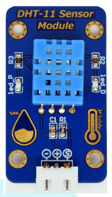
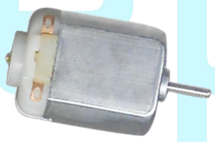
This is an entry-level learning kit for Arduino. Some common electronic components and sensors are included. Through the learning, you will get a better understanding of Arduino, and be able to make fascinating works based on Arduino.

If you have any problems for learning, please contact us at support@adeept.com, or please ask questions in our forum www.adeept.com. We will do our best to help you solve the problem.

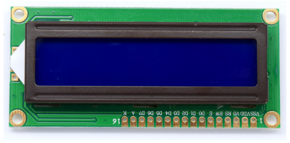
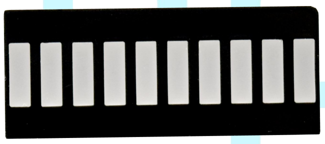
Component List

NO.	Name	Picture	Qty
1	Arduino 2560 Board		1
2	RC522 RFID Module		1
3	RFID ID Round Tag		1
4	RFID ID Card		1
5	Ultrasonic Distance Sensor		1
6	Slide Potentiometer Module		1
7	Rotary Encoder Module		1
8	Soil Moisture Sensor Module		1
9	CM Module		1

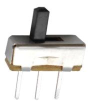
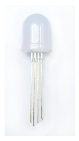
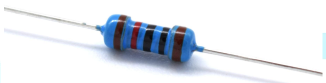
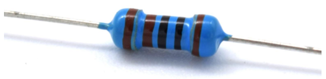
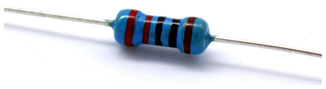
Adeept

10	Servo		1
11	Joystick Module		1
12	IR Receiver HX1838		1
13	8X8 LED Matrix Module		1
14	Remote Controller Module		1
15	DHT-11 Sensor Module		1
16	DC Motor		1
17	L9110 Motor Driver		1
18	Relay Module		1

Adeept

19	LCD1602 Module		1
20	7-segment Display		1
21	4-digit 7-segment Display		1
22	Active Buzzer		1
23	Passive Buzzer		1
24	Analog Temperature Sensor(Thermistor)		2
25	Light Sensor(Photoresistor)		2
26	LED Bar Graph		1
27	4*4 Matrix Keyboard		1

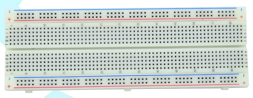
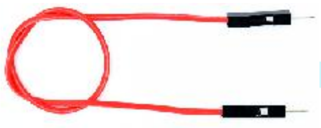
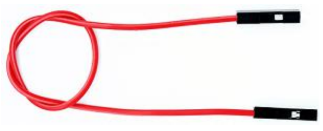
Adeept

28	Switch		2
29	RGB LED		1
30	Red LED		8
31	Green LED		4
32	Yellow LED		4
33	Blue LED		4
34	Resistor(220Ω)		16
35	Resistor(1KΩ)		10
36	Resistor(10KΩ)		5

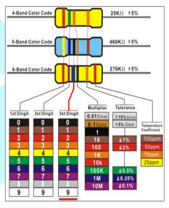
Adeept

37	Capacitor(104)		5
38	Capacitor(10uF)		2
39	Button(Large)		4
40	Button(Small)		4
41	Button Cap(Red)		1
42	Button Cap(White)		1
43	Button Cap(Blue)		2
44	NPN Transistor(8050)		2
45	PNP Transistor(8550)		2

Adeept

46	Potentiometer(10KΩ)		2
47	1N4148 Diode		2
48	1N4001 Diode		2
49	9V Battery Clip		1
50	Breadboard		1
51	USB Cable		1
52	Male to Male Jumper Wires		40
53	Male to Female Jumper Wires		20
54	Female to Female Jumper Wires		20

Adeept

55	3-Pin Wires		2
56	4-Pin Wires		1
57	5-Pin Wires		2
58	2-Pin Female to Female Wires		1
59	Band Resistor Card		1

Adeept

Contents

Adeept

Adeept

Adeept

About Arduino

What is Arduino?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.

ARDUINO BOARD

Arduino senses the environment by receiving inputs from many sensors, and affects its surroundings by controlling lights, motors, and other actuators.

ARDUINO SOFTWARE

You can tell your Arduino what to do by writing code in the Arduino programming language and using the Arduino development environment.

Before the development of Arduino program, the first thing you have to do is to install Arduino IDE software. The software provides you with the basic development environment that is required for developing Arduino program. You need the following URL to download Arduino IDE:
http://www.arduino.cc/en/Main/Software

For different operating system platforms, the way of using Arduino IDE is different. Please refer to the following links:
Windows User : http://www.arduino.cc/en/Guide/Windows
Mac OS X User : http://www.arduino.cc/en/Guide/MacOSX
Linux User : http://playground.arduino.cc/Learning/Linux

For more detailed information about Arduino IDE, please refer to the following link:
http://www.arduino.cc/en/Guide/HomePage

Adeept

About Processing

What is Processing?

Processing is a programming language, development environment, and online community. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. Initially created to serve as a software sketchbook and to teach computer programming fundamentals within a visual context, Processing evolved into a development tool for professionals. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production.

» Free to download and open source
» Interactive programs with 2D, 3D or PDF output
» OpenGL integration for accelerated 3D
» For GNU/Linux, Mac OS X, and Windows
» Over 100 libraries extend the core software

PROCESSING SOFTWARE

Download Processing:
https://www.processing.org/download/
For more detailed information about Processing IDE, please refer to the following link:
https://www.processing.org/reference/environment/

Adeept

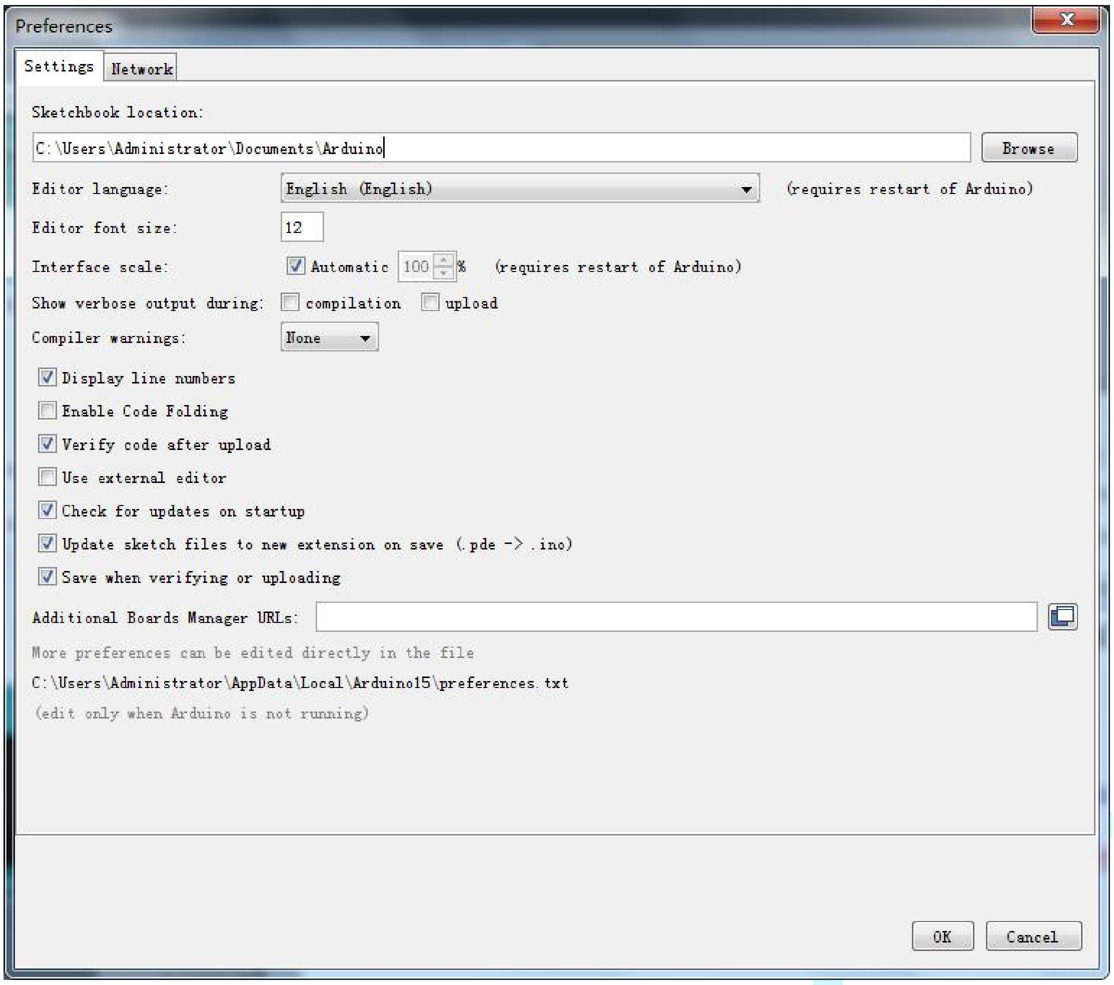
Getting Started with Ardublock

1. Download ardublock-all.jar:

https://www.adrive.com/public/SKNa2A/ardublock-beta-20140702.jar

2. In Arduino IDE, open menu”Arduino”->”Preference”

3. Find”Sketchbook location”


Preferences

Settings | Network

Sketchbook location:
C:\Users\Administrator\Documents\Arduino [Browse]

Editor language: English (English) (requires restart of Arduino)
Editor font size: 12
Interface scale: [✓] Automatic 100 % (requires restart of Arduino)
Show verbose output during: [ ] compilation [ ] upload
Compiler warnings: None

[✓] Display line numbers
[ ] Enable Code Folding
[✓] Verify code after upload
[ ] Use external editor
[✓] Check for updates on startup
[✓] Update sketch files to new extension on save (.pde -> .ino)
[✓] Save when verifying or uploading

Additional Boards Manager URLs:

More preferences can be edited directly in the file
C:\Users\Administrator\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

[OK] [Cancel]

- In Mac,it’s by default “Documents/Arduino”under user’s home directory.
- In Linux,it’s by default “sketchbook”under user’s home directory.
- In Windows, it’s by default “Documents/Arduino”under user’s home directory.

4 Copy ardublock-all.jar to tools/ArdublockTool/tool/ardublock-all.jar under “Sketchbook loaction”, Assume the user is “Administrator”.

- In Mac,

/Users/Administrator/Documents/Arduino/tools/ArduBlockTool/tool/ardublock-all.jar

●In Linux,

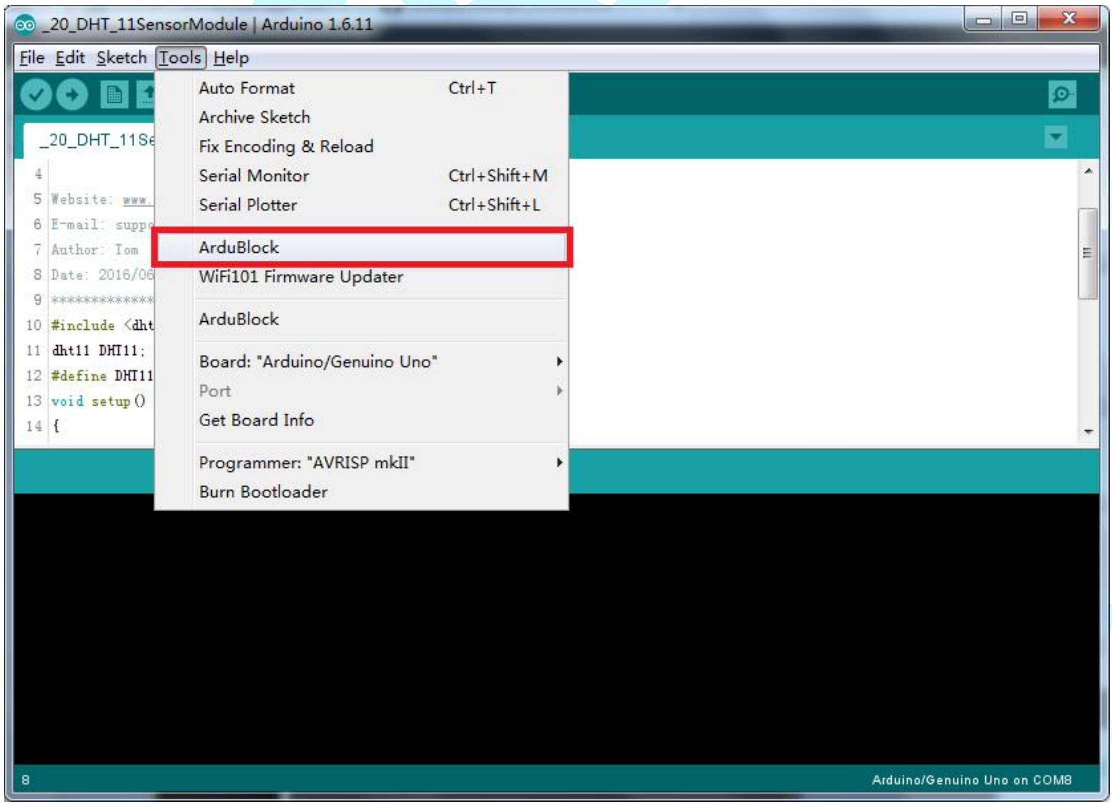
/home/Administrator/sketchbook/tools/ArduBlockTool/tool/ardublock-all.jar

●In Windows,

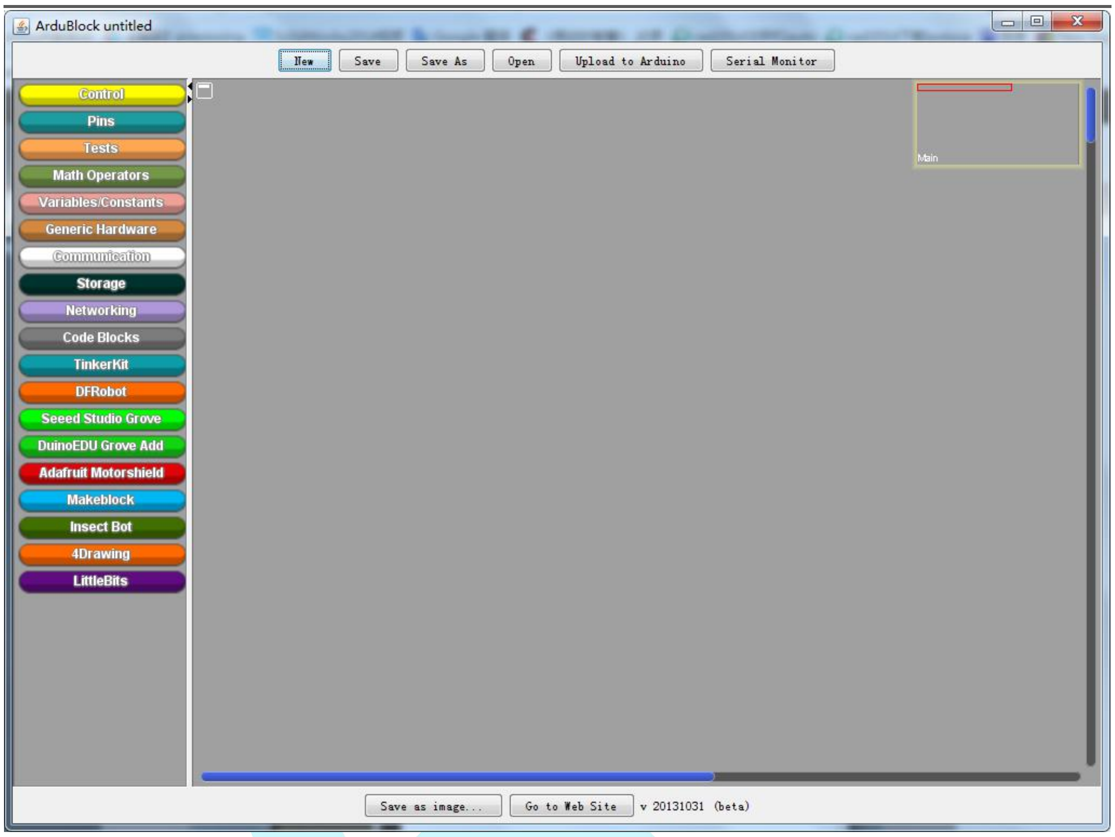
C:/Users/Administrator/Documents/Arduino/tools/ArduBlockTool/tool/ardublock-all.jar

* Be careful, the name of folder"ArduBlockTool"under tools folder is case sensitive.

5. Start the Arduino IDE and find ArduBlock under the Tool menu.


_20_DHT_11SensorModule | Arduino 1.6.11
File Edit Sketch Tools Help
_20_DHT_11Se
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
ArduBlock
WiFi101 Firmware Updater
ArduBlock
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
8
Arduino/Genuino Uno on COM8

```

Website: www.
E-mail: suppo
Author: Tom
Date: 2016/08
*************
#include <dht
dht11 DHT11;
#define DHT11
void setup()
{
```


ArduBlock untitled
New
Save
Save As
Open
Upload to Arduino
Serial Monitor
Control
Pins
Tests
Math Operators
Variables/Constants
Generic Hardware
Communication
Storage
Networking
Code Blocks
TinkerKit
DFRobot
Seeed Studio Grove
DuinoEDU Grove Add
Adafruit Motorshield
Makeblock
Insect Bot
4Drawing
LittleBits
Main
Save as image...
Go to Web Site
v 20131031 (beta)

Lesson 1 Blinking LED

Overview

In this tutorial, we will start the journey of learning Arduino MEGA 2560. In the first lesson, we will learn how to make a LED blinking.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* 220Ω Resistor
- 1* LED
- 1* Breadboard
- 2* Jumper Wires

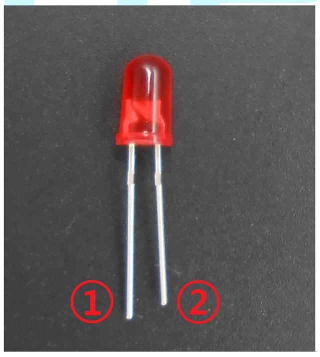
Principle

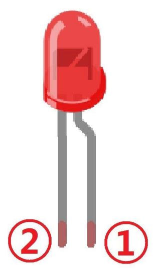
In this lesson, we will program the Arduino's GPIO output high(+5V) and low level(0V), and then make the LED which is connected to the Arduino’s GPIO flicker with a certain frequency.

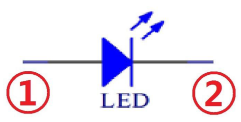
1. What is the LED?

The LED is the abbreviation of light emitting diode. It is usually made of gallium arsenide, gallium phosphide semiconductor materials. The LED has two electrodes, a positive electrode and a negative electrode, it will light only when a forward current passes, and it can be red, blue, green or yellow light, etc. The color of light depends on the materials it was made.

In general, the drive current for LED is 5-20mA. Therefore, in reality it usually needs an extra resistor for current limitation so as to protect the LED.







① **Anode**
② **Cathode**

Adeept

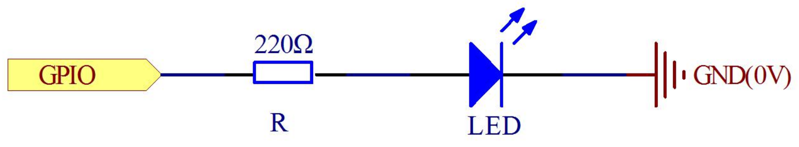
2. What is the resistor?

The main function of the resistor is to limit current. In the circuit, the character 'R' represents resistor, and the unit of resistor is ohm(Ω).

The band resistor is used in this experiment. A band resistor is one whose surface is coated with some particular color through which the resistance can be identified directly.

There are two methods for connecting LED to Arduino's GPIO:

①



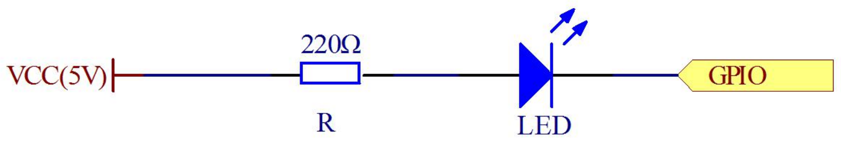
As shown in the schematic diagram above, the anode of LED is connected to Arduino's GPIO via a resistor, and the cathode of LED is connected to the ground(GND). When the GPIO output high level, the LED is on; when the GPIO output low level, the LED is off.

The size of the current-limiting resistor is calculated as follows: 5~20mA current is required to make an LED on, and the output voltage of the Arduino MEGA 2560's GPIO is 5V, so we can get the resistance :

R = U / I = 5V / (5~20mA) = 250Ω~1KΩ

Since the LED has a certain resistance, thus we choose a 220ohm resistor.

②



As shown in the schematic diagram above, the anode of LED is connected to VCC(+5V), and the cathode of LED is connected to the Arduino's GPIO. When the GPIO output low level, the LED is on; when the GPIO output high level, the LED is off.

The experiment is based on method ①, we select Arduino's D8 pin to control the LED. When the Arduino's D8 pin is programmed to output high level, then the LED will be on, next delay for the amount of time, and then programmed the D8 pin to low level to make the LED off. Continue to perform the above process, you can get a blinking LED.

3. Key functions:

● setup()

The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each powerup or reset of the Arduino board.

●loop()

After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

●pinMode()

Configures the specified pin to behave either as an input or an output.

As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pullups.

●digitalWrite()

Write a HIGH or a LOW value to a digital pin.

If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin. It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor.

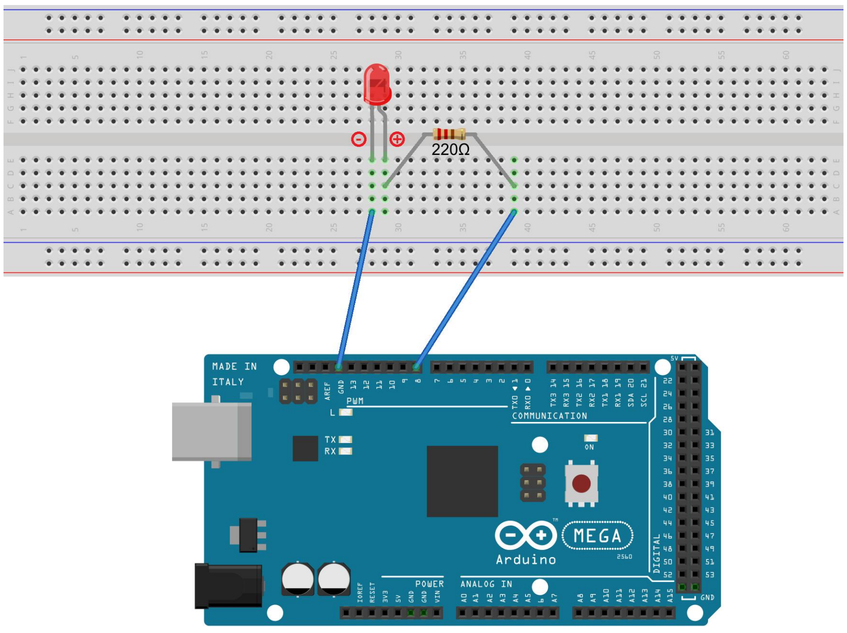
●delay()

Pauses the program for the amount of time (in miliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

Procedures

1. Build the circuit

Adeept

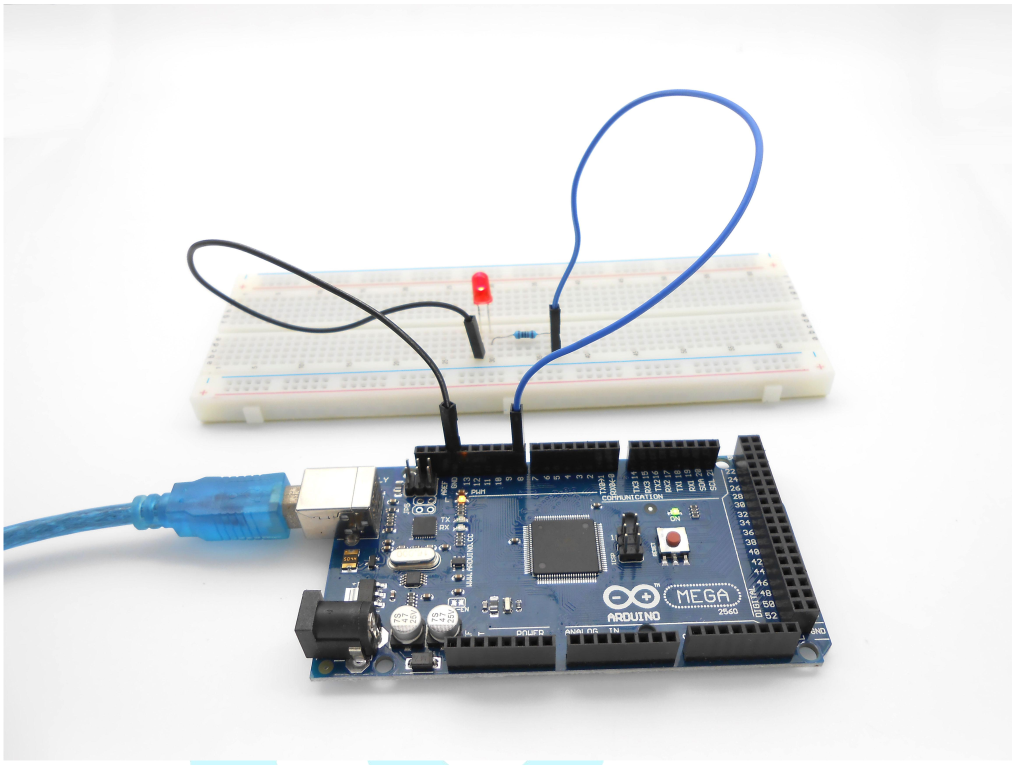


2. Program

```
/***********************************************************
File name:   01_blinkingLed.ino
Description:  Lit LED, let LED blinks.
Website: www.adeept.com
E-mail: support@adeept.com
Author: Tom
Date: 2016/10/15
***********************************************************/
int ledPin=8; //definition digital 8 pins as pin to control the LED
void setup()
{
   pinMode(ledPin,OUTPUT);    //Set the digital 8 port mode, OUTPUT:
Output mode
}
void loop()
{
   digitalWrite(ledPin,HIGH); //HIGH is set to about 5V PIN8
   delay(1000);             //Set the delay time, 1000 = 1S
   digitalWrite(ledPin,LOW);  //LOW is set to about 5V PIN8
   delay(1000);             //Set the delay time, 1000 = 1S
}
```

3. Compile the program and upload to Arduino MEGA 2560 board

Now, you can see the LED is blinking.



Adeept

Lesson 2 Active Buzzer

Overview

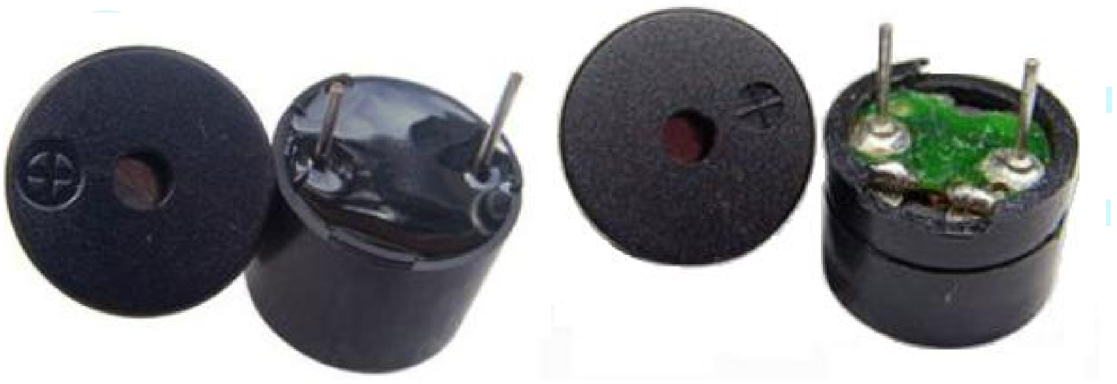
In this lesson, we will learn how to program the Arduino to make an active buzzer sound.

Requirement

- 1* Arduino MEGA 2560
- 1* USB cable
- 1* Active buzzer
- 1* 1 kΩ Resistor
- 1* NPN Transistor (S8050)
- 1* Breadboard
- Several Jumper Wires

Principle

A buzzer or beeper is an audio signaling device. As a type of electronic buzzer with integrated structure, which use DC power supply, are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic equipments, telephones, timers and other electronic products for voice devices. Buzzers can be categorized as active and passive buzzers (See the following pictures).

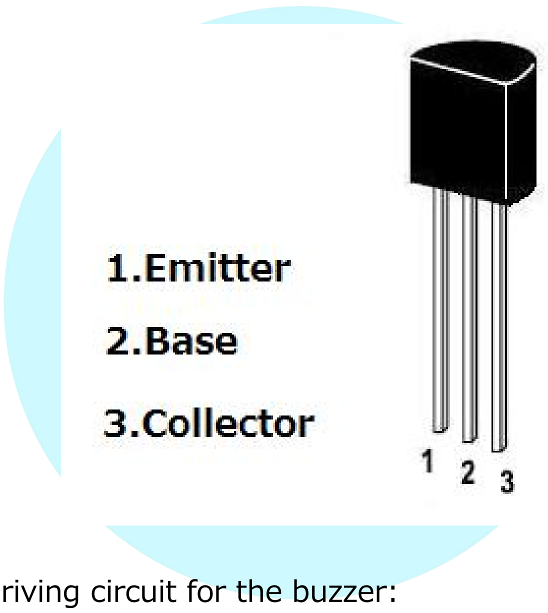


When you place the pins of buzzers upward, you can see that two buzzers are different, the buzzer that green circuit board exposed is the passive buzzer.

In this study, the buzzer we used is active buzzer. Active buzzer will sound as long as the power supply. We can program to make the Arduino output alternating high and low level, so that the buzzer sounds.

A slightly larger current is needed to make a buzzer sound. However, the output current of Arduino's GPIO is weak, so we need a transistor to drive the buzzer.

The main function of transistor is blowing up the voltage or current. The transistor can also be used to control the circuit conduction or deadline. And the transistor is divided into two kinds, one kind is NPN, for instance, the S8050 we provided; another kind is PNP transistor such as the S8550 we provided. The transistor we used is as shown in below:



There are two driving circuit for the buzzer:

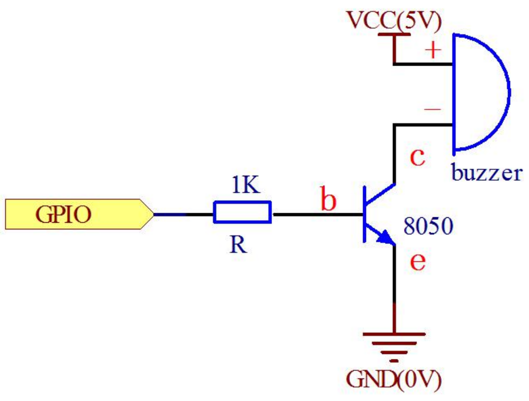


Figure1

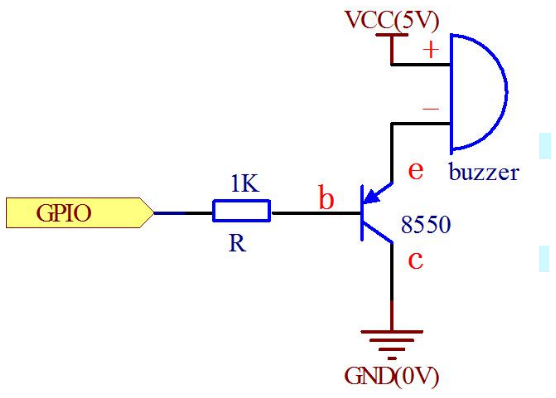


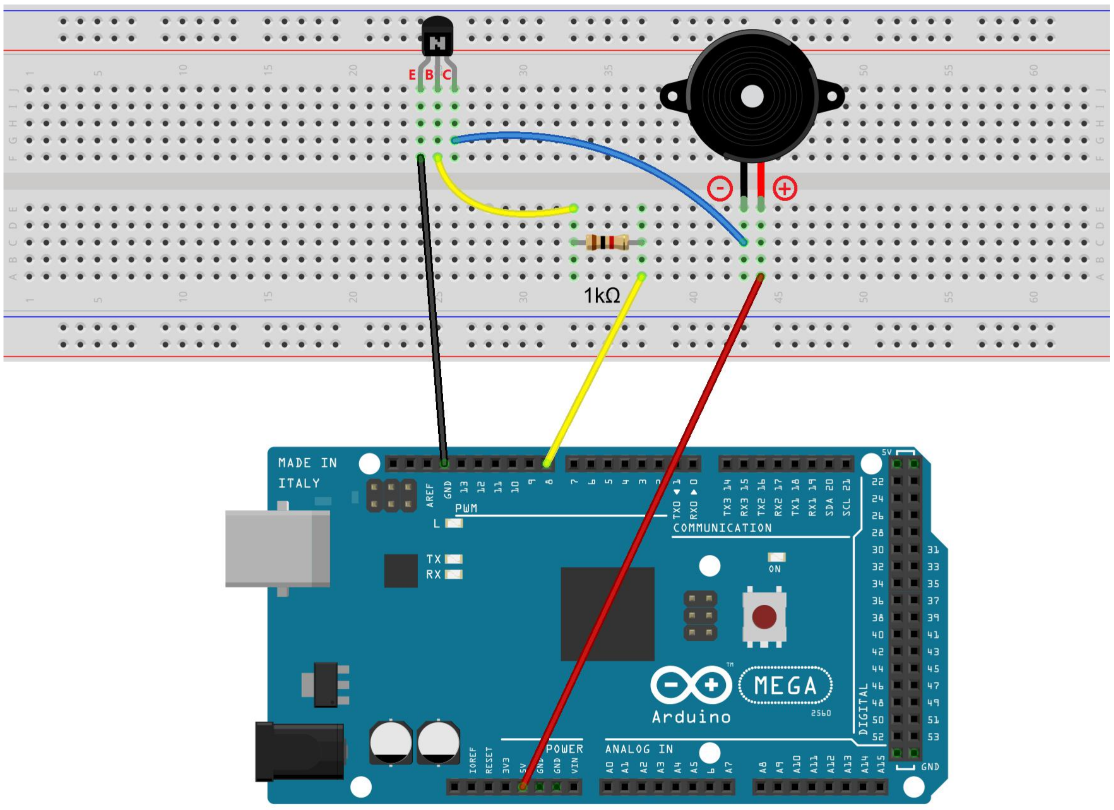
Figure2

Figure 1: Set the Arduino GPIO as a high level, the transistor S8050 will conduct, and then the buzzer will sound; set the Arduino GPIO as low level, the transistor S8050 will cut off, then the buzzer will stop.

Figure 2: Set the Arduino GPIO as low level, the transistor S8550 will conduct, and the buzzer will sound; set the Arduino GPIO as a high level, the transistor S8550 will cut off, then the buzzer will stop.

Procedures

1. Build the circuit

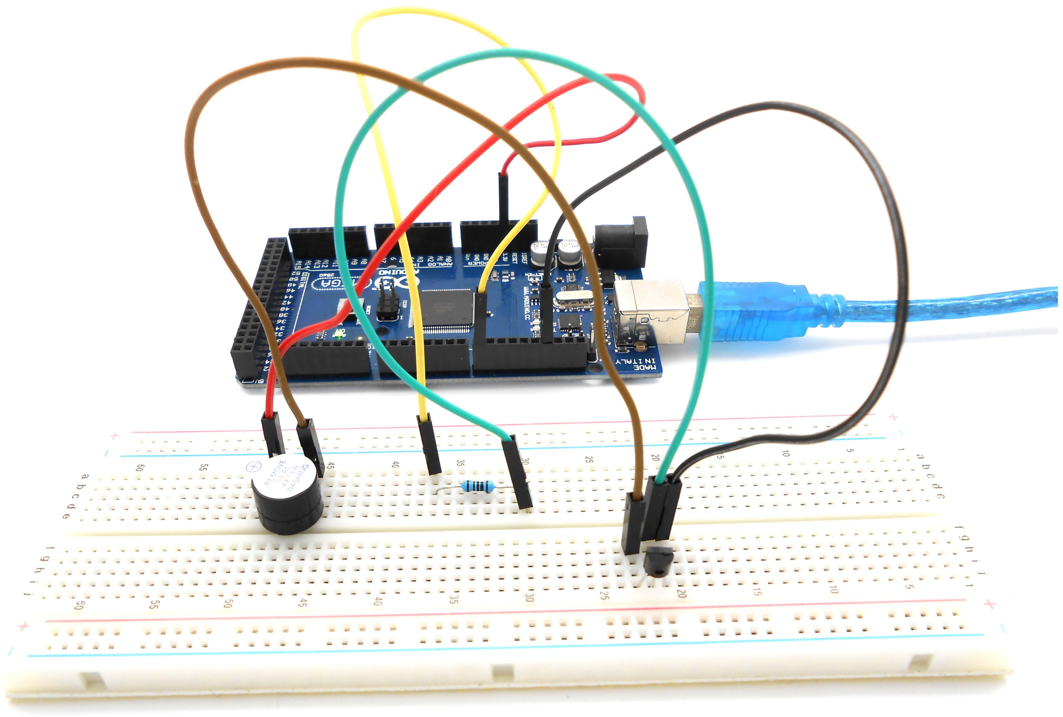


2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, you should be able to hear the sound of the buzzer.

Adeept



Summary

By learning this lesson, we have mastered the basic principle of the buzzer and the transistor. We also learned how to program the Arduino and then control the buzzer. I hope you can use what you have learned in this lesson to do some interesting things.

Adeept

Lesson 3 Controlling an LED with a button

Overview

In this lesson, we will learn how to detect the state of a button, and then toggle the state of LED based on the state of the button.

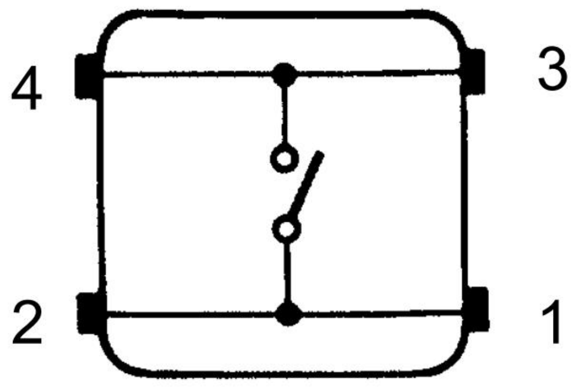
Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* Button
- 1* LED
- 1* 10KΩ Resistor
- 1* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

Principle

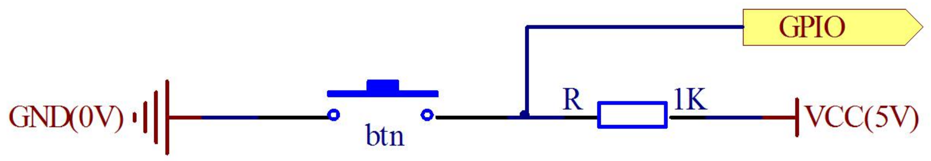
1. Button

Buttons are a common component used to control electronic devices. They are usually used as switches to connect or disconnect circuits. Although buttons come in a variety of sizes and shapes, the one used in this experiment will be a 12mm button as shown in the following pictures. Pins pointed out by the arrows of same color are meant to be connected.

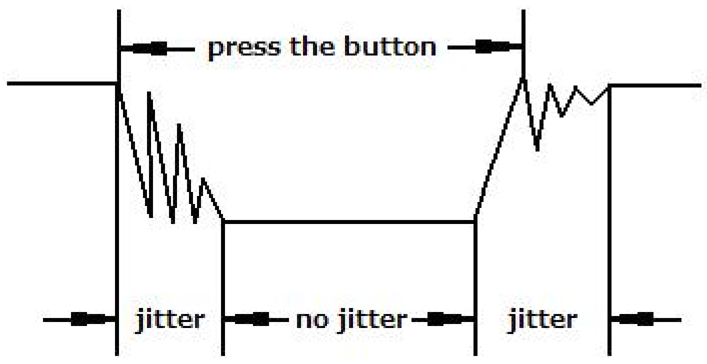


The button we used is a normally open type button. The two contacts of a button is in the off state under the normal conditions, only when the button is pressed they are closed.

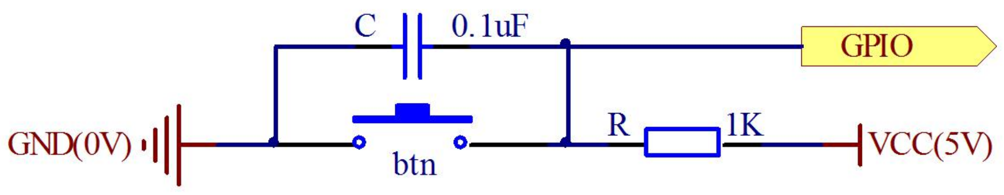
The schematic diagram we used is as follows:



The button jitter must be happen in the process of using. The jitter waveform is as the flowing picture:



Each time you press the button, the Arduino will think you have pressed the button many times due to the jitter of the button. We must to deal with the jitter of buttons before we use the button. We can through the software programming method to remove the jitter of buttons, and you can use a capacitance to remove the jitter of buttons. We introduce the software method. First, we detect whether the level of button interface is low level or high level. When the level we detected is low level, 5~10 MS delay is needed, and then detect whether the level of button interface is low or high. If the signal is low, we can confirm that the button is pressed once. You can also use a 0.1 uF capacitance to clean up the jitter of buttons. The schematic diagram is shown in below:



2. interrupt

Hardware interrupts were introduced as a way to reduce wasting the processor's valuable time in polling loops, waiting for external events. They may be implemented in hardware as a distinct system with control lines, or they may be integrated into the memory subsystem.

3. Key functions:

●attachInterrupt(interrupt, ISR, mode)

Specifies a named Interrupt Service Routine (ISR) to call when an interrupt occurs. Replaces any previous function that was attached to the interrupt. Most Arduino boards have two external interrupts: numbers 0 (on digital pin 2) and 1 (on digital pin 3).

Generally, an ISR should be as short and fast as possible. If your sketch uses multiple ISRs, only one can run at a time, other interrupts will be ignored (turned off) until the current one is finished. as delay() and millis() both rely on interrupts, they will not work while an ISR is running. delayMicroseconds(), which does not rely on interrupts, will work as expected.

Syntax

attachInterrupt(pin, ISR, mode)

Parameters

pin: the pin number

ISR: the ISR will be called when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.

mode: defines when the interrupt should be triggered. Four constants are predefined as valid values:

-LOW to trigger the interrupt whenever the pin is low,

-CHANGE to trigger the interrupt whenever the pin changes value

-RISING to trigger when the pin goes from low to high,

-FALLING for when the pin goes from high to low.

●digitalRead()

Reads the value from a specified digital pin, either HIGH or LOW.

Syntax

digitalRead(pin)

Parameters

pin: the number of the digital pin you want to read (int)

Returns

HIGH or LOW

●delayMicroseconds(us)

Pauses the program for the amount of time (in microseconds) specified as parameter. There are a thousand microseconds in a millisecond, and a million microseconds in a second.

Currently, the largest value that will produce an accurate delay is 16383. This could change in future Arduino releases. For delays longer than a few thousand microseconds, you should use delay() instead.

Syntax

delayMicroseconds(us)

Parameters

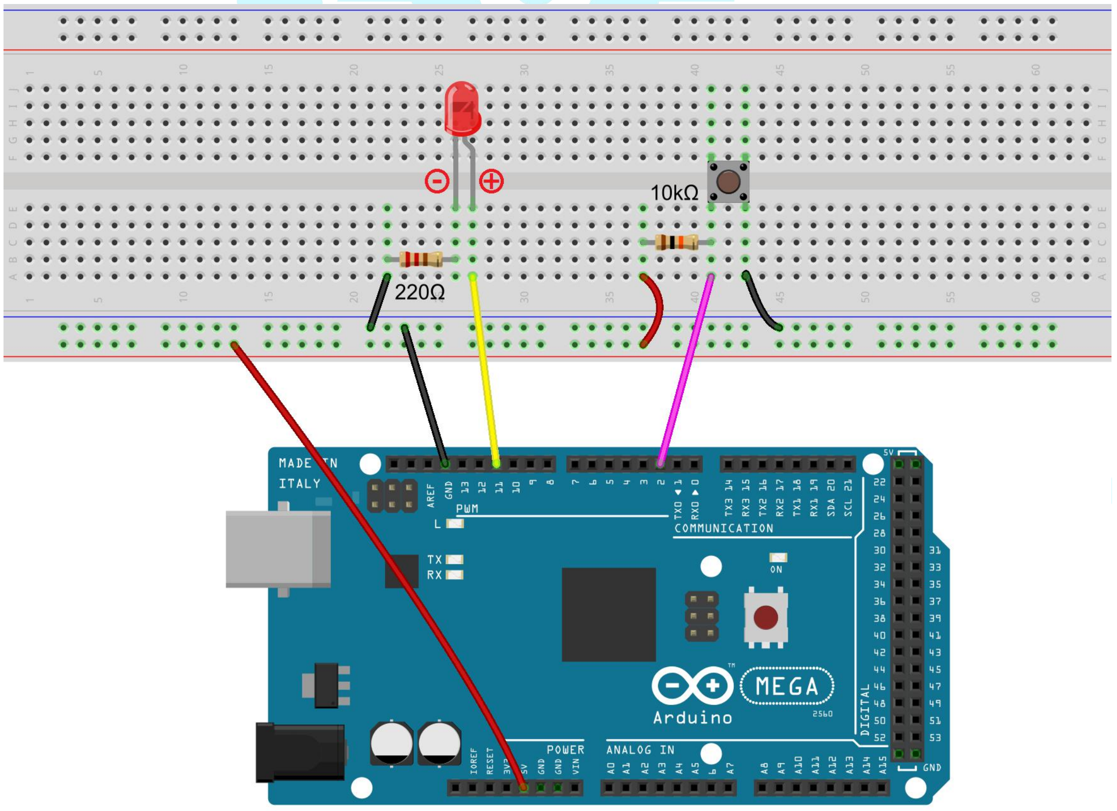
us: the number of microseconds to pause (unsigned int)

Returns

None

Procedures

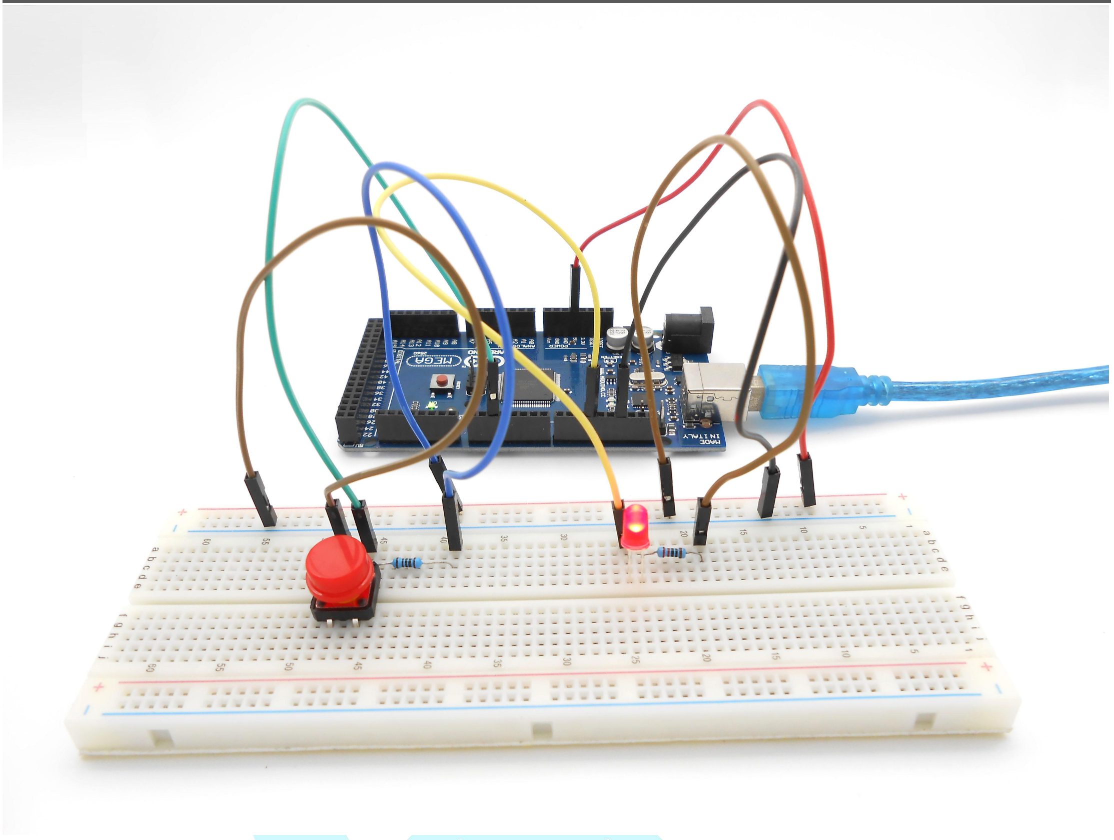
1. Build the circuit



2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

When you press the button, you can see the state of the LED will be toggled. (ON->OFF, OFF->ON).



Summary

Through this lesson, you should have learned how to use the Arduino MEGA 2560 detects an external button state, and then toggle the state of LED relying on the state of the button detected before.

Adeept

Lesson 4 Relay Module

Introduction

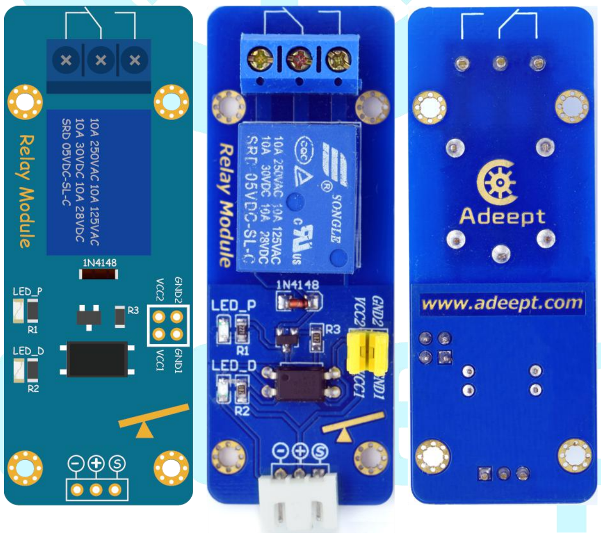
The relay is an electronic and electrical component that controls large currents by small currents. In the course of building an Arduino project, generally many large current or high volume devices like solenoid valve, lamp and motor cannot be connected directly to digital I/Os of the Arduino board. At this moment, a relay can save your project.

Components

- 1 * Arduino MEGA 2560
- 1 * Relay Module
- 1 * LED Module
- 1 * USB Cable
- 1 * 3-Pin Wires
- 3 * Hookup Wire Set
- 1 * Breadboard

Experimental Principle

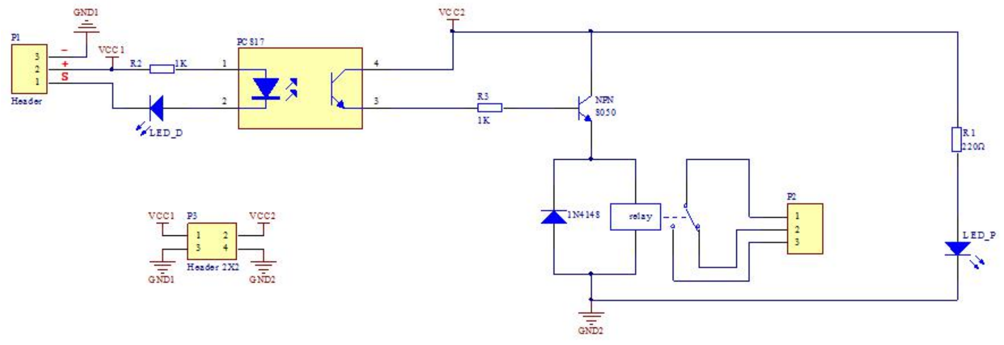
The Fritzing image:



Pin definition:

S	Digital Data Input
+	VCC1
-	GND1
VCC1	VCC1
GND1	GND1
VCC2	VCC1
GND2	GND2

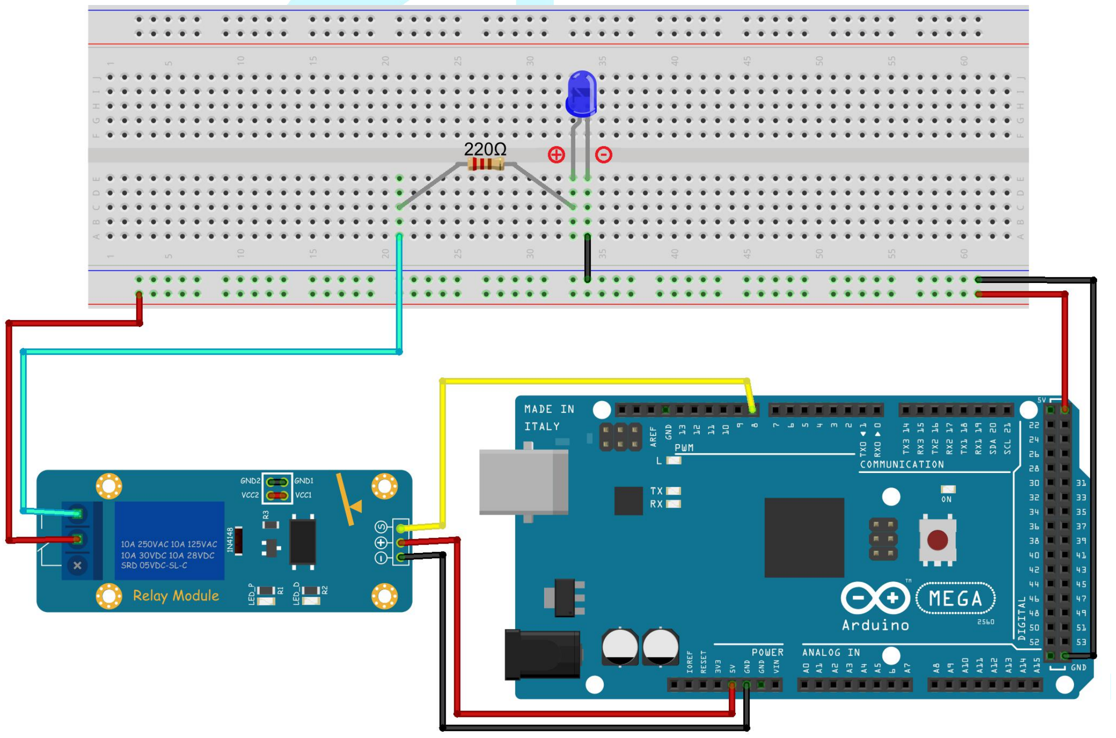
The schematic diagram:



This experiment is to control an LED to brighten and dim by a relay.

Experimental Procedures

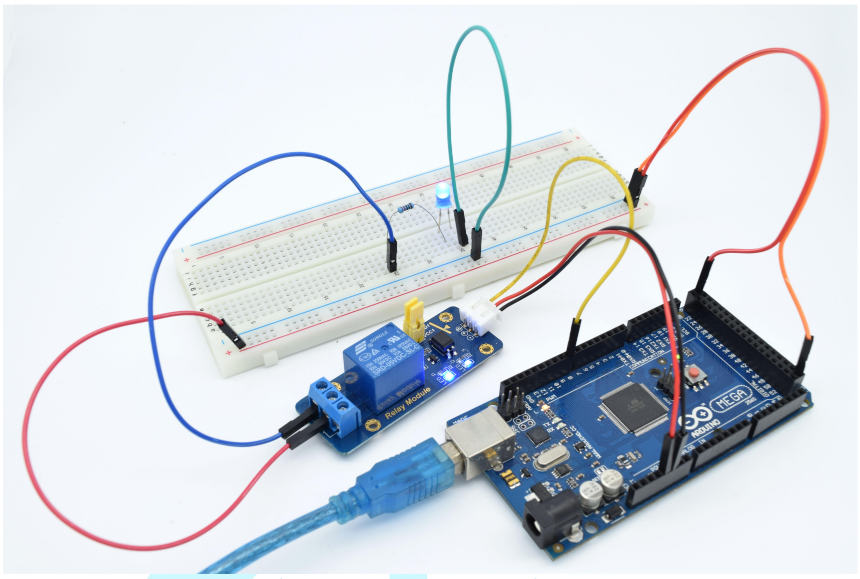
Step 1: Build the circuit



Step 2: Program _04_Relay.ino

Step 3: Compile and download the sketch to the 2560 board.

Now you can see the LED on the LED Module flickers every 2s and can hear the sound of relay closing and opening.



Adeept

Lesson 5 Serial Port

Overview

In this lesson, we will program the Arduino MEGA 2560 to achieve function of send and receive data through the serial port. The Arduino receiving data which send from PC, and then controlling an LED according to the received data, then return the state of LED to the PC's serial port monitor.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* LED
- 1* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

Principle

1. Serial ports

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART). It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.

You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to begin().

To use these pins to communicate with your personal computer, you will need an additional USB-to-serial adaptor, as they are not connected to the MEGA 2560's USB-to-serial adaptor. To use them to communicate with an external TTL serial device, connect the TX pin to your device's RX pin, the RX to your device's TX pin, and the ground of your MEGA 2560 to your device's ground. (Don't connect these pins directly to an RS232 serial port; they operate at +/- 12V and can damage your Arduino board.)

2. Key function

●begin()

Sets the data rate in bits per second (baud) for serial data transmission. For communicating with the computer, use one of these rates: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200. You can, however, specify other rates - for example, to communicate over pins 0 and 1 with a component that requires a particular baud rate.

Syntax

Serial.begin(speed)

Parameters

speed: in bits per second (baud) - long

Returns

nothing

●print()

Prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. Characters and strings are sent as is. For example:

Serial.print(78) gives "78"

Serial.print(1.23456) gives "1.23"

Serial.print('N') gives "N"

Serial.print("Hello world.") gives "Hello world."

An optional second parameter specifies the base (format) to use; permitted values are BIN (binary, or base 2), OCT (octal, or base 8), DEC (decimal, or base 10), HEX (hexadecimal, or base 16). For floating point numbers, this parameter specifies the number of decimal places to use. For example:

Serial.print(78, BIN) gives "1001110"

Serial.print(78, OCT) gives "116"

Serial.print(78, DEC) gives "78"

Serial.print(78, HEX) gives "4E"

Serial.println(1.23456, 0) gives "1"

Serial.println(1.23456, 2) gives "1.23"

Serial.println(1.23456, 4) gives "1.2346"

You can pass flash-memory based strings to Serial.print() by wrapping them with F(). For example :

Serial.print(F("Hello World"))

To send a single byte, use Serial.write().

Syntax

Adeept

Serial.print(val)

Serial.print(val, format)

Parameters

val: the value to print - any data type format: specifies the number base (for integral data types) or number of decimal places (for floating point types)

Returns

byte print() will return the number of bytes written, though reading that number is optional

●println()

Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n'). This command takes the same forms as Serial.print().

Syntax

Serial.println(val)

Serial.println(val, format)

Parameters

val: the value to print - any data type

format: specifies the number base (for integral data types) or number of decimal places (for floating point types)

Returns

byte

println() will return the number of bytes written, though reading that number is optional

●read()

Reads incoming serial data. read() inherits from the Stream utility class.

Syntax

Serial.read()

Parameters

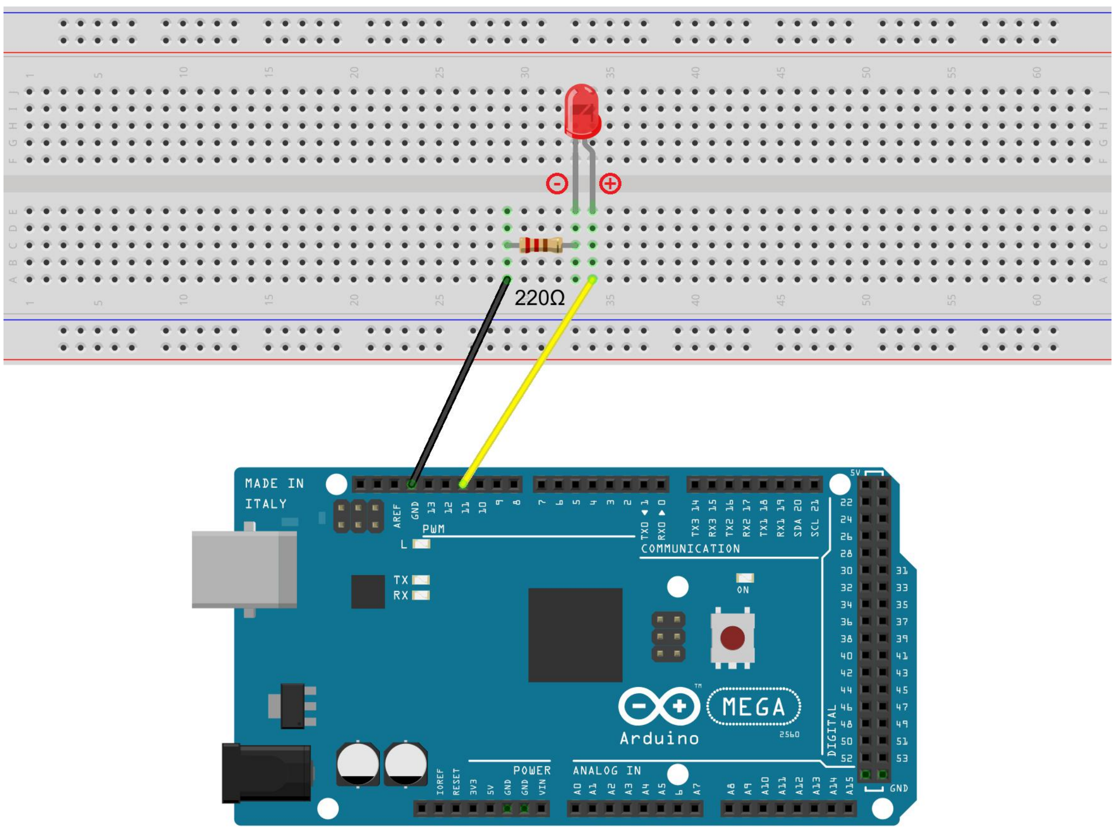
None

Returns

the first byte of incoming serial data available (or -1 if no data is available) - int

Procedures

1. Build the circuit

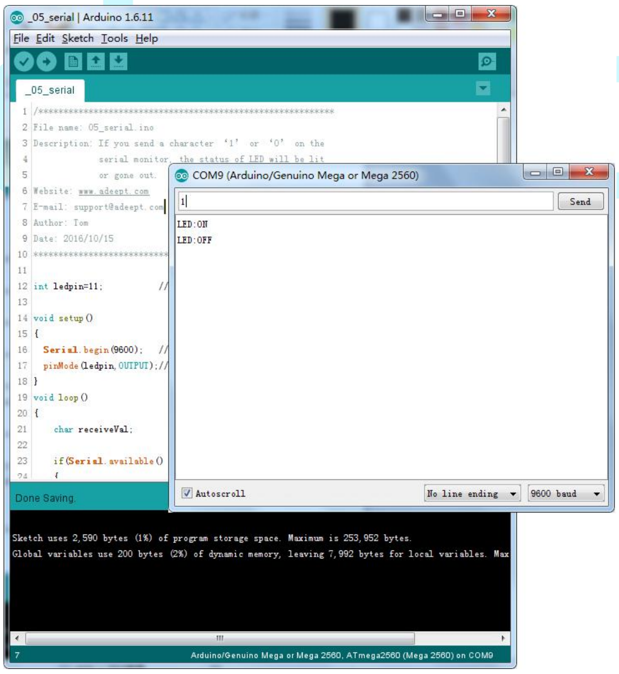


2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Open the port monitor, and then select the appropriate baud rate according to the program.

Now, if you send a character‘1’or‘0’on the serial monitor, the state of LED will be lit or gone out.


_05_serial | Arduino 1.6.11
File Edit Sketch Tools Help
_05_serial
COM9 (Arduino/Genuino Mega or Mega 2560)
Send
Autoscroll
No line ending
9600 baud
Done Saving.
7
Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

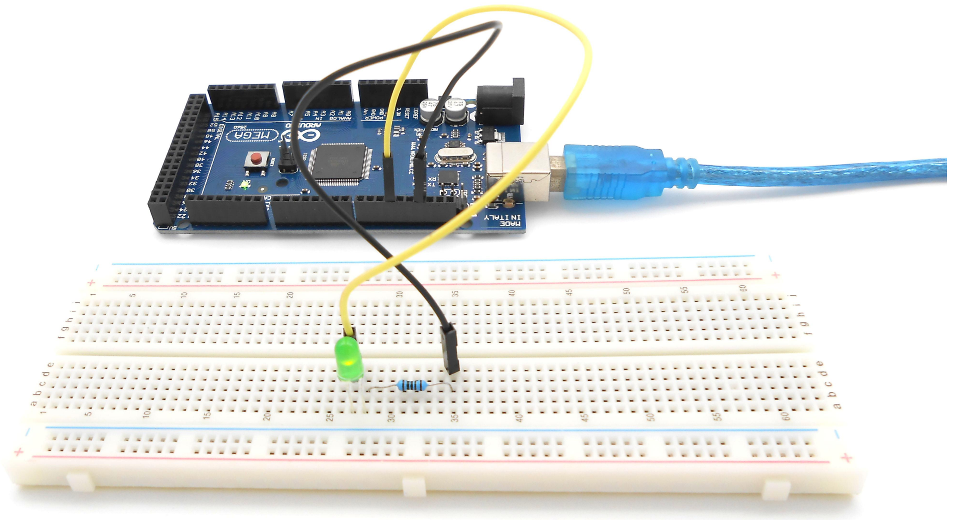
```
/*************************************************************
File name: 05_serial.ino
Description: If you send a character ‘1’ or ‘0’ on the
             serial monitor, the status of LED will be lit
             or gone out.
Website: www.adeept.com
E-mail: support@adeept.com
Author: Tom
Date: 2016/10/15
***************************

int ledpin=11;          //

void setup()
{
  Serial.begin(9600);   //
  pinMode(ledpin,OUTPUT);//
}
void loop()
{
    char receiveVal;

    if(Serial.available()
    {
```

```
1
```

```
LED:ON
LED:OFF
```

```
Sketch uses 2,590 bytes (1%) of program storage space. Maximum is 253,952 bytes.
Global variables use 200 bytes (2%) of dynamic memory, leaving 7,992 bytes for local variables. Max
```

Adeept



Summary

Through this lesson, you should have understood that the computer can send data to Arduino MEGA 2560 via the serial port, and then control the state of LED. I hope you can use your head to make more interesting things based on this lesson.

Adeept

Lesson 6 LED Flowing Lights

Overview

In the first class, we have learned how to make an LED blink by programming the Arduino. Today, we will use the Arduino to control 8 LEDs, so that 8 LEDs showing the result of flowing.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 8* LED
- 8* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

Principle

The principle of this experiment is very simple. It is very similar with the first class.

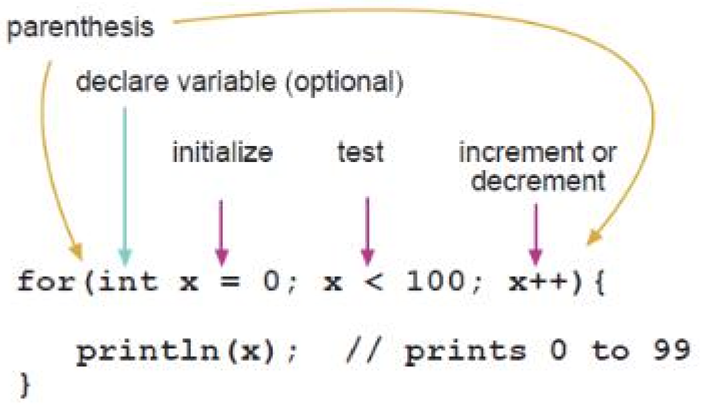
Key function:

● for statements

The for statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The for statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

There are three parts to the for loop header:

```
for (initialization; condition; increment) {
//statement(s);
}
```

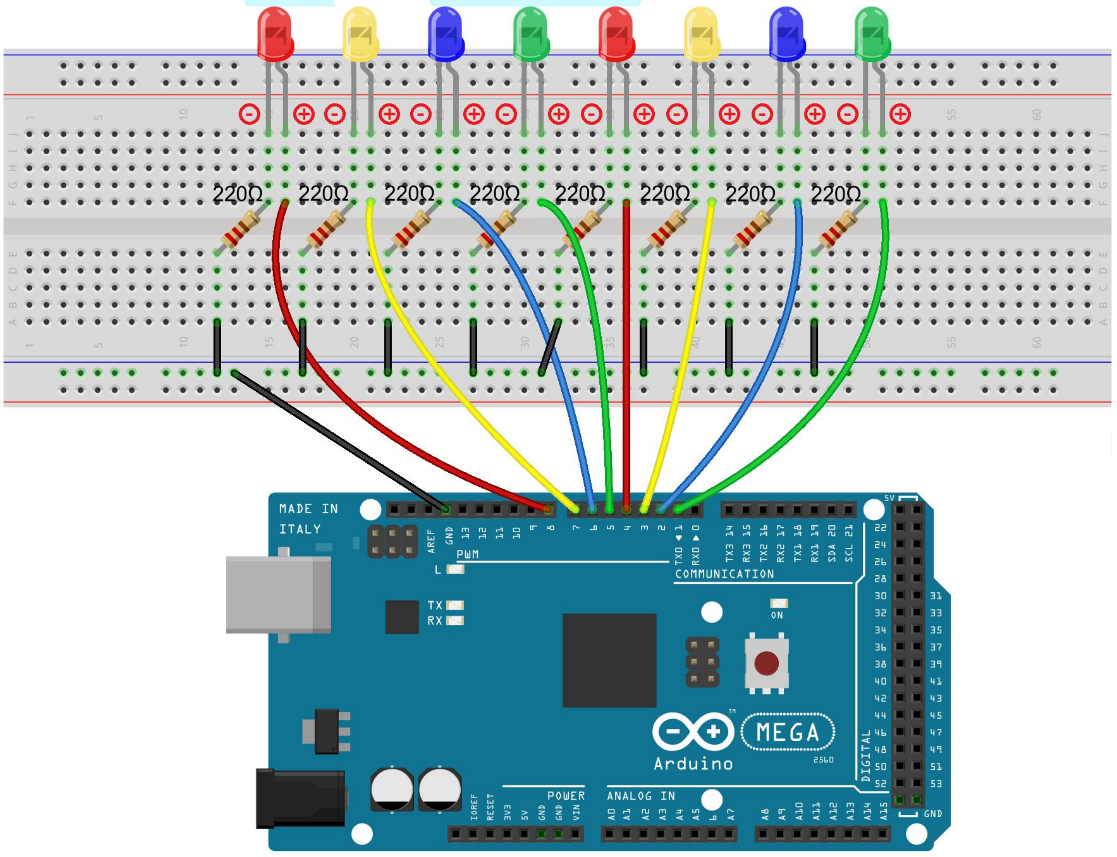


```
for(int x = 0; x < 100; x++){
    println(x);  // prints 0 to 99
}
```

The initialization happens first and exactly once. Each time through the loop, the condition is tested; if it's true, the statement block, and the increment is executed, then the condition is tested again. When the condition becomes false, the loop ends.

Procedures

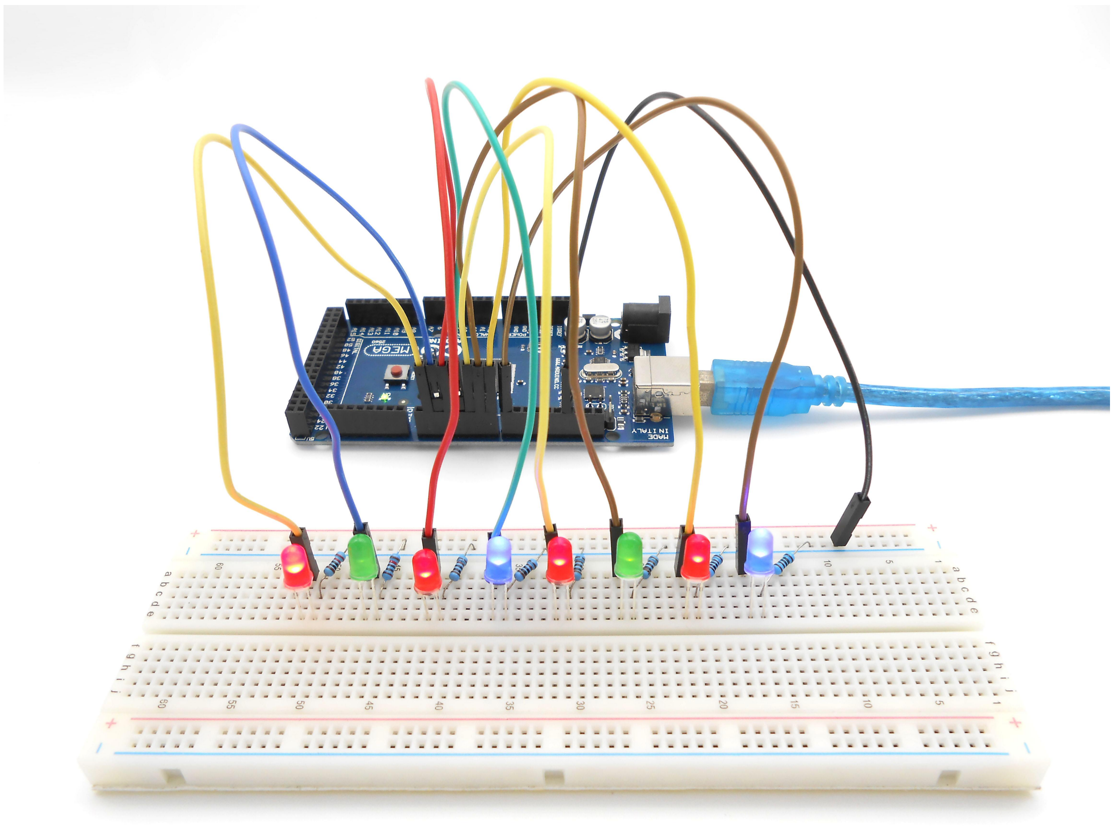
1. Build the circuit



2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, you should see 8 LEDs are lit in sequence from the right green one to the left, next from the left to the right one. And then repeat the above phenomenon.



Summary

Through this simple and fun experiment, we have learned more skilled programming about the Arduino. In addition, you can also modify the circuit and code we provided to achieve even more dazzling effect.

Adeept

Lesson 7 LED bar graph display

Overview

In this lesson, we will learn how to control a LED bar graph by programming the Arduino.

Requirement

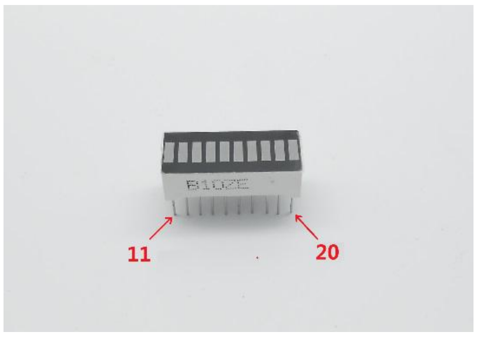
- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* 10KΩ Potentiometer
- 10* 220Ω Resistor
- 1* LED Bar Graph
- 1* Breadboard
- Several Jumper Wires

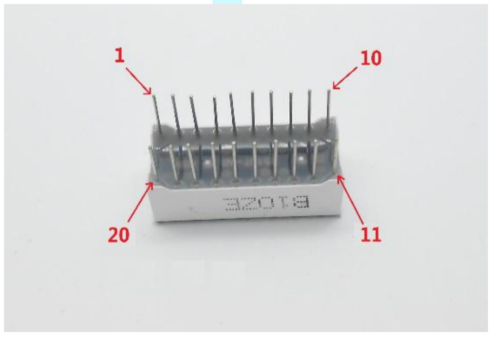
Principle

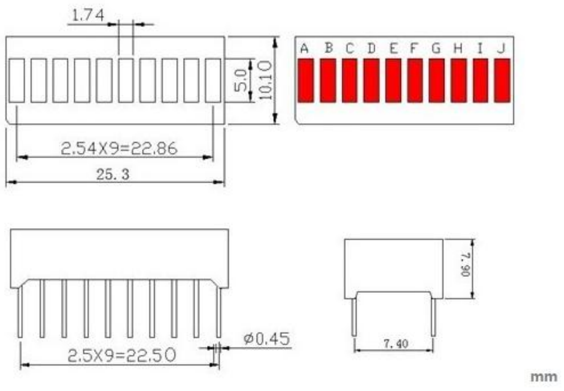
The bar graph - a series of LEDs in a line, such as you see on an audio display is a common hardware display for analog sensors. It's made up of a series of LEDs in a row, an analog input like a potentiometer, and a little code in between. You can buy multi-LED bar graph displays fairly cheaply. This tutorial demonstrates how to control a series of LEDs in a row, but can be applied to any series of digital outputs.

This tutorial borrows from the For Loop and Arrays tutorial as well as the Analog Input tutorial.

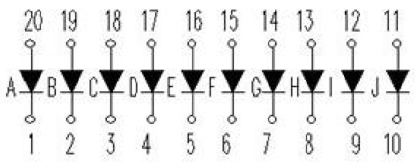
The sketch works like this: first you read the input. You map the input value to the output range, in this case ten LEDs. Then you set up a ***for*** loop to iterate over the outputs. If the output's number in the series is lower than the mapped input range, you turn it on. If not, you turn it off.







The internal schematic diagram for the LED bar graph is shown below:

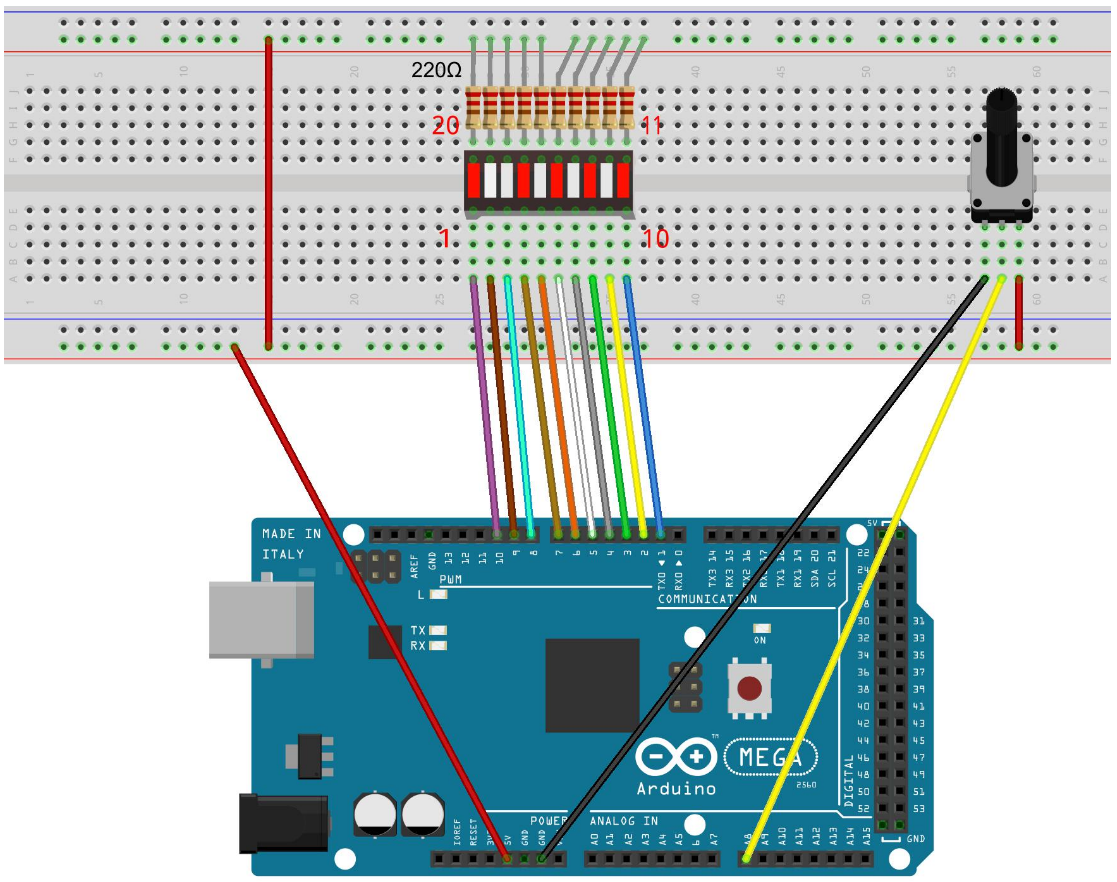


A potentiometer, informally a pot, is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat.

Procedures

1. Build the circuit

Adeept

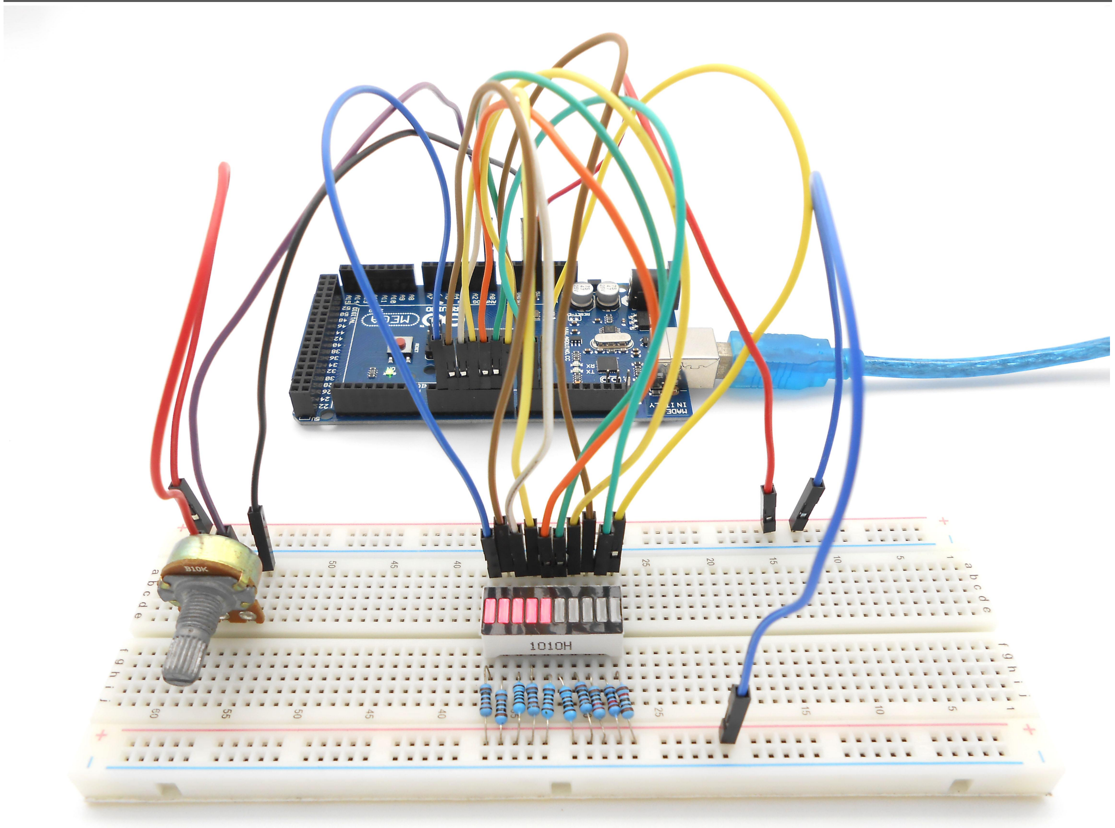


2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, when you turn the knob of the potentiometer, you will see that the number of LED in the LED bar graph will be changed.

Adeept



Adeept

Lesson 8 Breathing LED

Overview

In this lesson, we will learn how to program the Arduino to generate PWM signal. And use the PWM square-wave signal control an LED gradually becomes brighter and then gradually becomes dark like the animal's breathing.

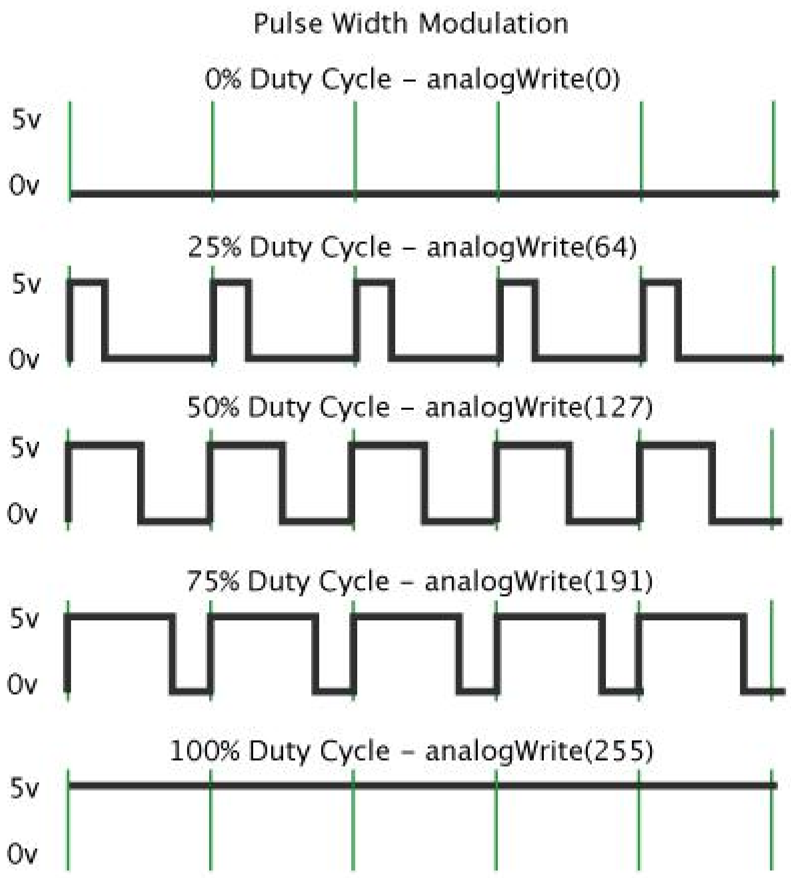
Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* LED
- 1* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

Principle

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

In the graphic below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to analogWrite() is on a scale of 0 - 255, such that analogWrite(255) requests a 100% duty cycle (always on), and analogWrite(127) is a 50% duty cycle (on half the time) for example.


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

Key function:

●analogWrite()

Writes an analog value (PWM wave) to a pin. Can be used to light an LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite() on the same pin). You do not need to call pinMode() to set the pin as an output before calling analogWrite().

Syntax

analogWrite(pin, value)

Parameters

pin: the pin to write to.

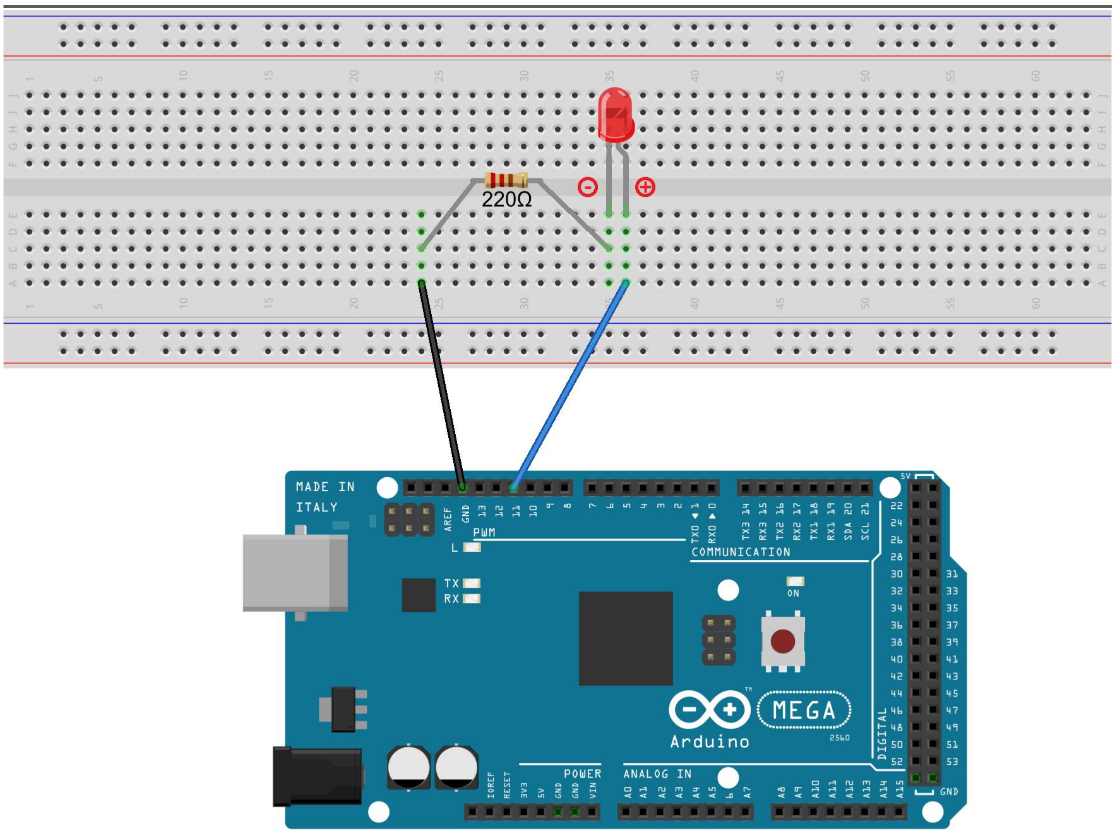
value: the duty cycle: between 0 (always off) and 255 (always on).

Returns

nothing

Procedures

1. Build the circuit

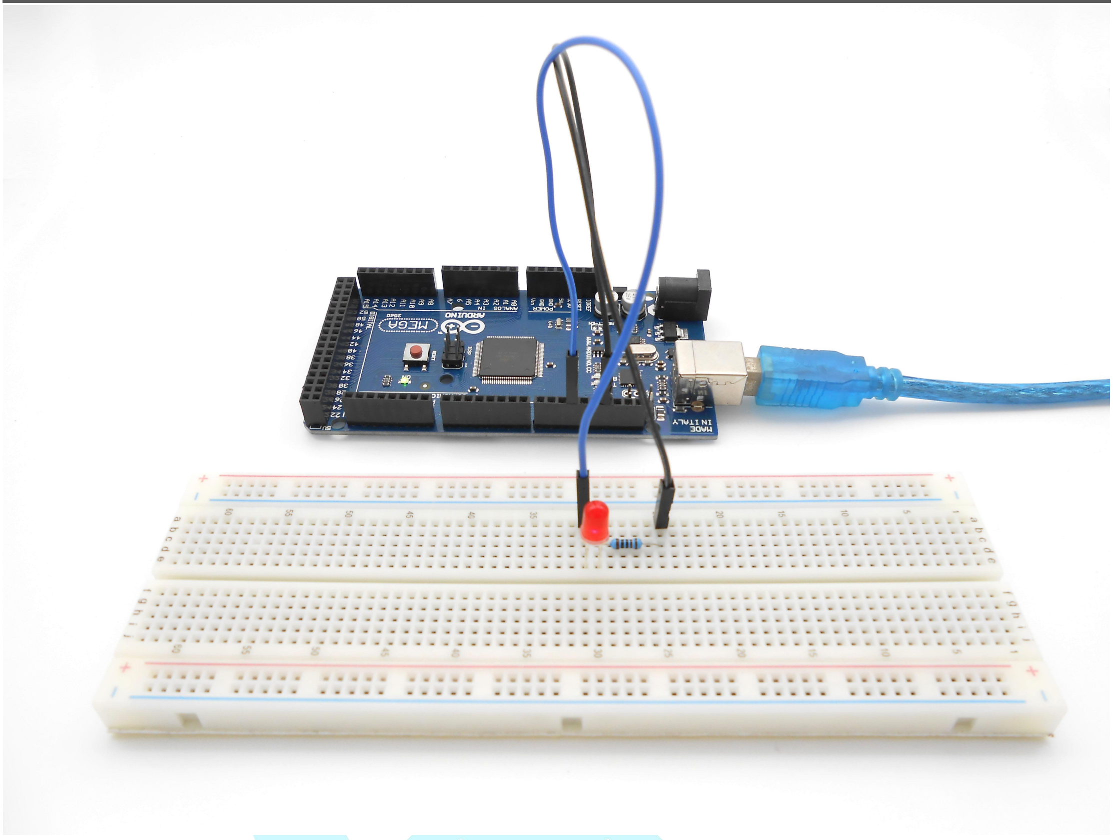


2. Program

3. Compile the program and upload to Arduino MEGA 2560 board.

Now, you should see the LED gradually from dark to brighter, and then from brighter to dark, continuing to repeat the process, its rhythm like the animal's breathing.

Adeept



Summary

By learning this lesson, I believe that you have understood the basic principles of the PWM, and mastered the PWM programming on the Arduino platform.

Adeept

Lesson 9 Controlling a RGB LED by PWM

Overview

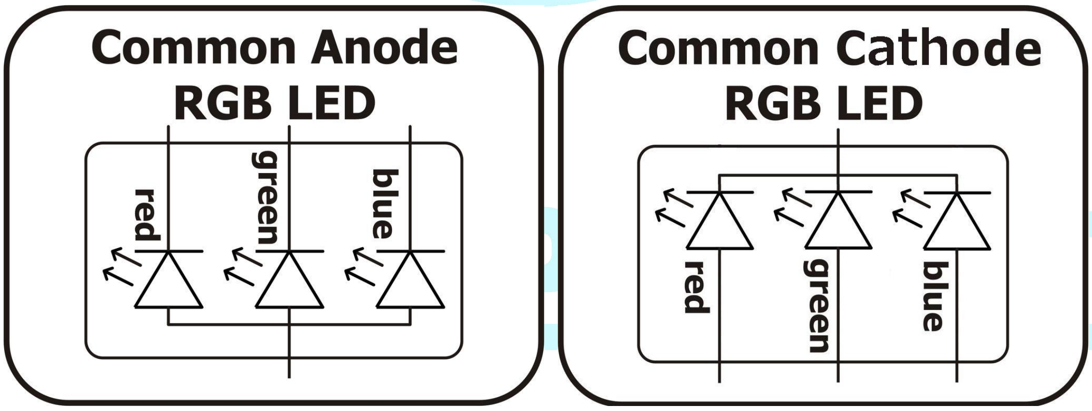
In this lesson, we will program the Arduino for RGB LED control, and make RGB LED emits a various of colors of light.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* RGB LED
- 3* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

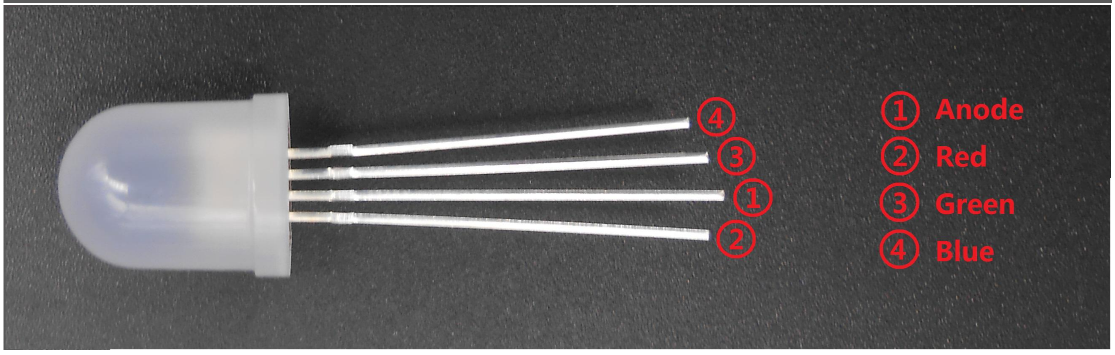
Principle

RGB LEDs consist of three LEDs. Each LED actually has one red, one green and one blue light. These three colored LEDs are capable of producing any color. Tri-color LEDs with red, green, and blue emitters, in general using a four-wire connection with one common lead (anode or cathode). These LEDs can have either common anode or common cathode leads.



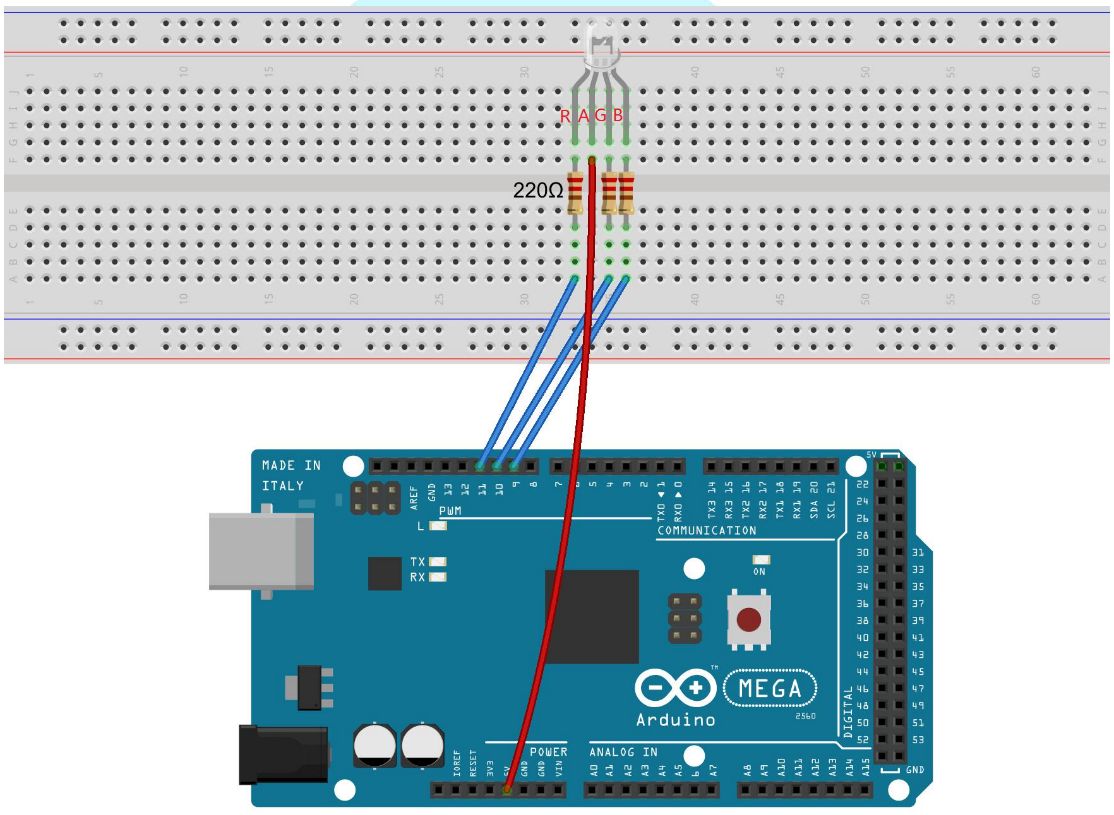
What we used in this experiment is the common anode RGB LED. The longest pin is the common anode of three LEDs. The pin is connected to the +5V pin of the Arduino, and the three remaining pins are connected to the Arduino's D9, D10, D11 pins through a current limiting resistor.

In this way, we can control the color of RGB LED by 3-channel PWM signal.



Procedures

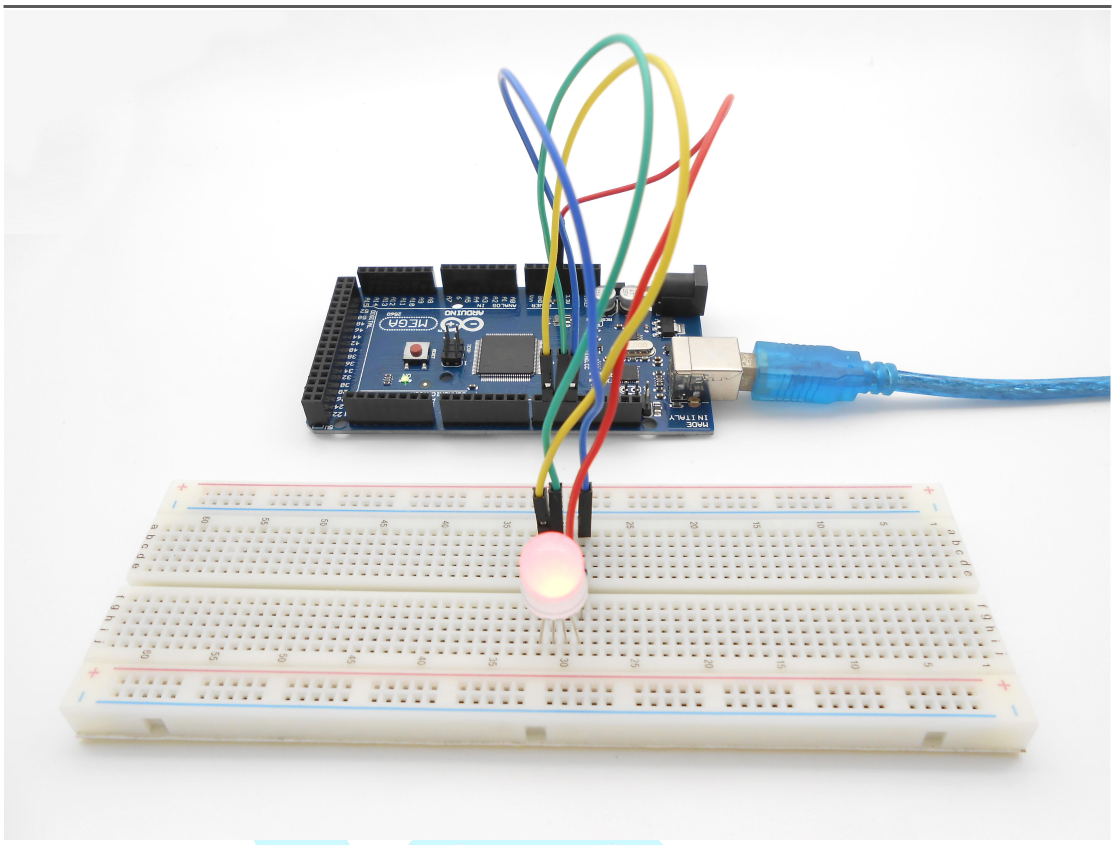
1. Build the circuit



2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, you can see the RGB LED emitting red, green, blue, yellow, white and purple light, then the RGB LED will be off, each state continues 1s, after repeating the above procedure.



Summary

By learning this lesson, I believe you have already known the principle and the programming of RGB LED. I hope you can use your imagination to achieve even more cool ideas based on this lesson.

Adeept

Lesson 10 Play the Music

Overview

In this lesson, we will program the Arduino to control a passive buzzer, and then make the passive buzzer play music.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* NPN Transistor (8050)
- 1* 1KΩ Resistor
- 1* Passive Buzzer
- 1* LED
- 1* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

Principle

As long as you send the square wave signals to a passive buzzer with different frequency, then the passive buzzer will make different sound.

Key function:

●tone()

Generates a square wave of the specified frequency (and 50% duty cycle) on a pin. A duration can be specified, otherwise the wave continues until a call to noTone(). The pin can be connected to a piezo buzzer or other speaker to play tones.

Only one tone can be generated at a time. If a tone is already playing on a different pin, the call to tone() will have no effect. If the tone is playing on the same pin, the call will set its frequency.

Use of the tone() function will interfere with PWM output on pins 3 and 11 (on boards other than the Mega).

NOTE: if you want to play different pitches on multiple pins, you need to call noTone() on one pin before calling tone() on the next pin.

Syntax

tone(pin, frequency)

tone(pin, frequency, duration)

Parameters

pin: the pin on which to generate the tone

frequency: the frequency of the tone in hertz - unsigned int

duration: the duration of the tone in milliseconds (optional) - unsigned long

Returns

nothing

● noTone()

Stops the generation of a square wave triggered by tone(). Has no effect if no tone is being generated.

NOTE: if you want to play different pitches on multiple pins, you need to call noTone() on one pin before calling tone() on the next pin.

Syntax

noTone(pin)

Parameters

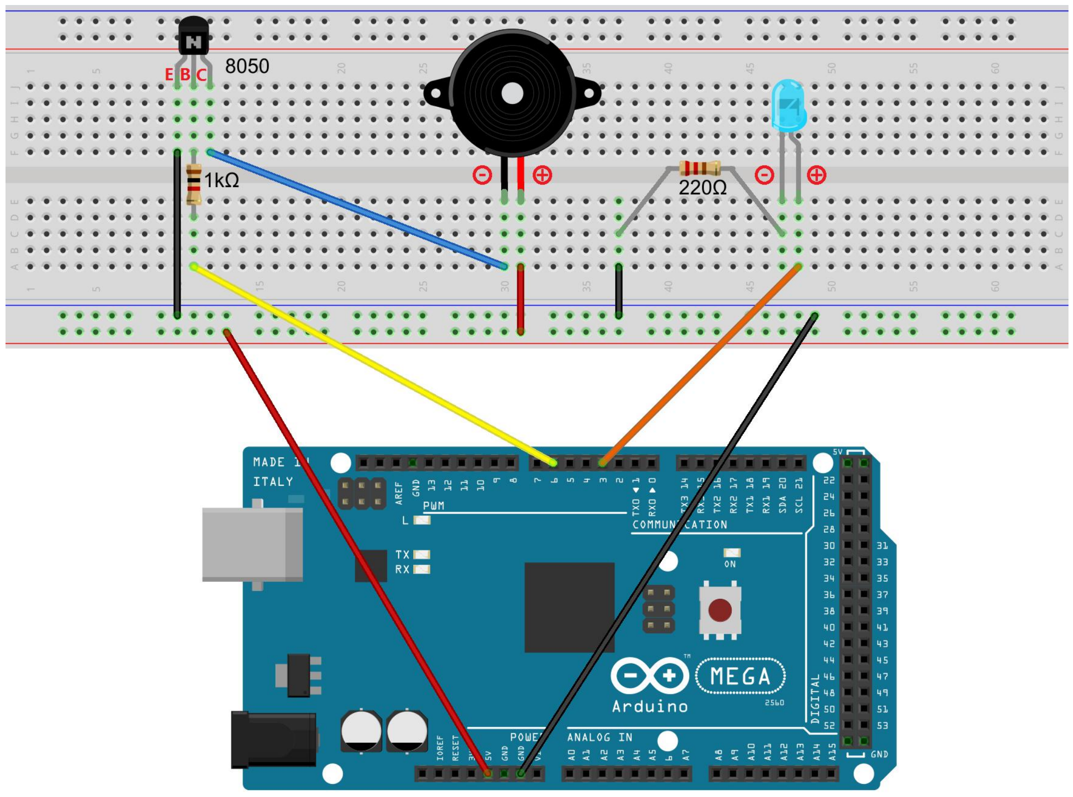
pin: the pin on which to stop generating the tone

Returns

nothing

Procedures

1. Build the circuit

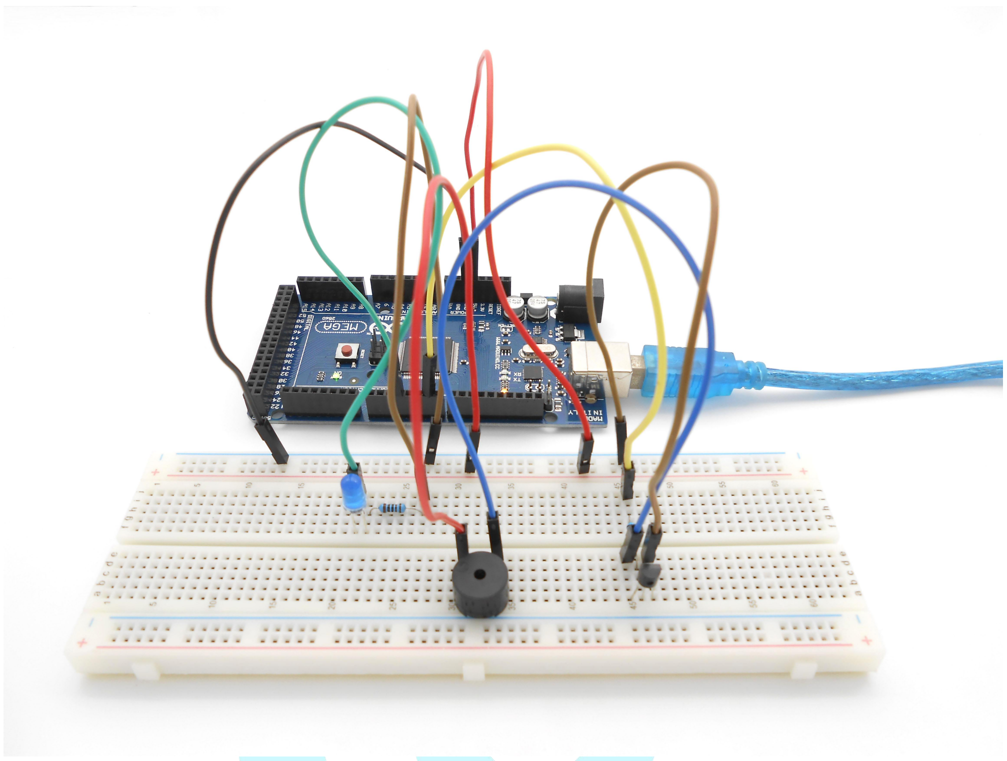


2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, you can hear the music of passive buzzer with blinking an LED.

Adeept



Lesson 11 LCD1602 display

Overview

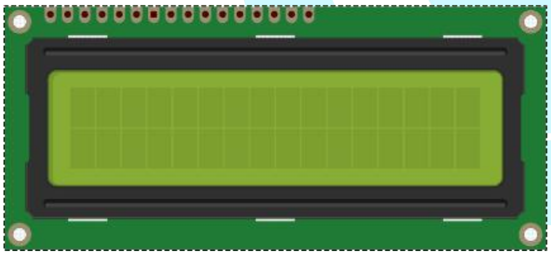
In this lesson, we will learn how to use a character display device—LCD1602 on the Arduino platform. First, we make the LCD1602 display a string "Hello Geeks!" scrolling, then display "Adeept" and "www.adeept.com" static.

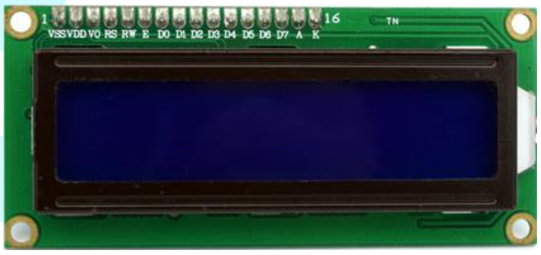
Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* LCD1602
- 1* 10KΩ Potentiometer
- 1* Breadboard
- Several Jumper Wires

Principle

The LCD1602 image:





Pin definition:

VSS	GND
VDD	VCC
VO	Analog input
RS	Digital input
RW	Digital input
E	Digital input
D0	Digital input
D1	Digital input
D2	Digital input
D3	Digital input
D4	Digital input
D5	Digital input
D6	Digital input
D7	Digital input
A	VCC
K	GND

LCD1602 is a kind of character LCD display. The LCD has a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

● A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

● A Read/Write (R/W) pin that selects reading mode or writing mode

● An Enable pin that enables writing to the registers

● 8 data pins (D0-D7). The state of these pins (high or low) are the bits that you're writing to a register when you write, or the values when you read.

● There's also a display contrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on or off the LED backlight respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The LiquidCrystal Library simplifies this for you so you don't need to know the low-level instructions.

The Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode, so example shows how to control a 2x16 LCD in 4-bit mode.

A potentiometer , informally a pot, is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat.

Key function:

●begin()

Specifies the dimensions (width and height) of the display.

Syntax

lcd.begin(cols, rows)

Parameters

lcd: a variable of type LiquidCrystal

cols: the number of columns that the display has

rows: the number of rows that the display has

●setCursor()

Position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed.

Syntax

lcd.setCursor(col, row)

Parameters

lcd: a variable of type LiquidCrystal

col: the column at which to position the cursor (with 0 being the first column)

row: the row at which to position the cursor (with 0 being the first row)

●scrollDisplayLeft()

Scrolls the contents of the display (text and cursor) one space to the left.

Syntax

lcd.scrollDisplayLeft()

Parameters

lcd: a variable of type LiquidCrystal

Example

scrollDisplayLeft() and scrollDisplayRight()

See also

scrollDisplayRight()

●print()

Prints text to the LCD.

Syntax

lcd.print(data)

lcd.print(data, BASE)

Parameters

lcd: a variable of type LiquidCrystal

data: the data to print (char, byte, int, long, or string)

BASE (optional): the base in which to print numbers: BIN for binary (base 2), DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

Returns

byte

print() will return the number of bytes written, though reading that number is optional

●clear()

Clears the LCD screen and positions the cursor in the upper-left corner.

Adeept

Syntax

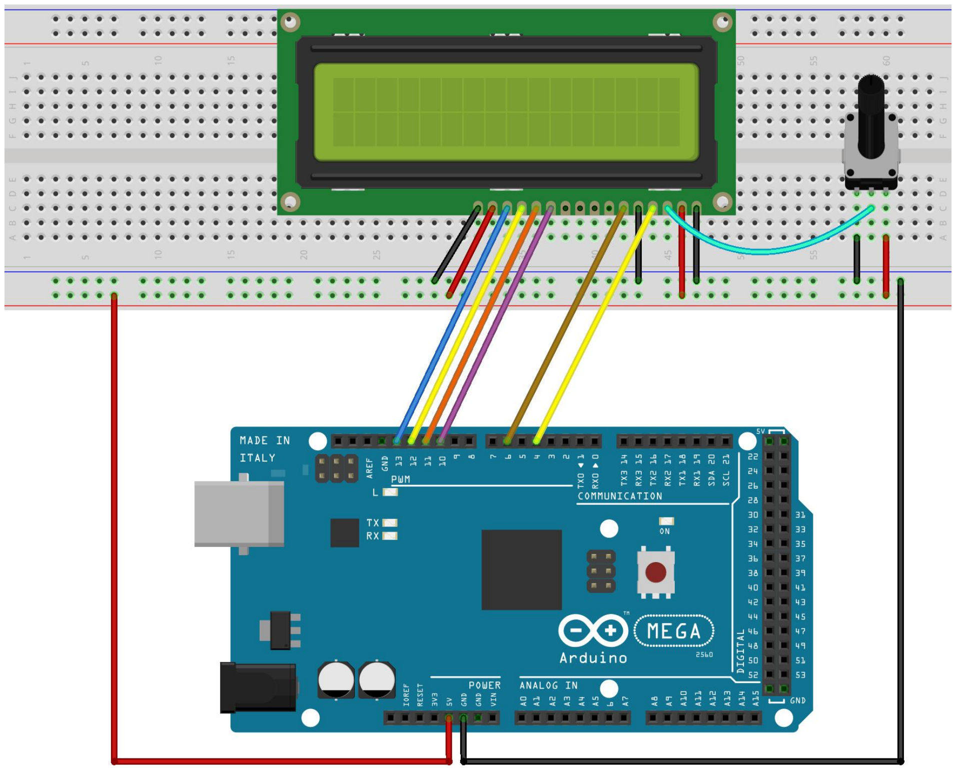
lcd.clear()

Parameters

lcd: a variable of type LiquidCrystal

Procedures

1. Build the circuit

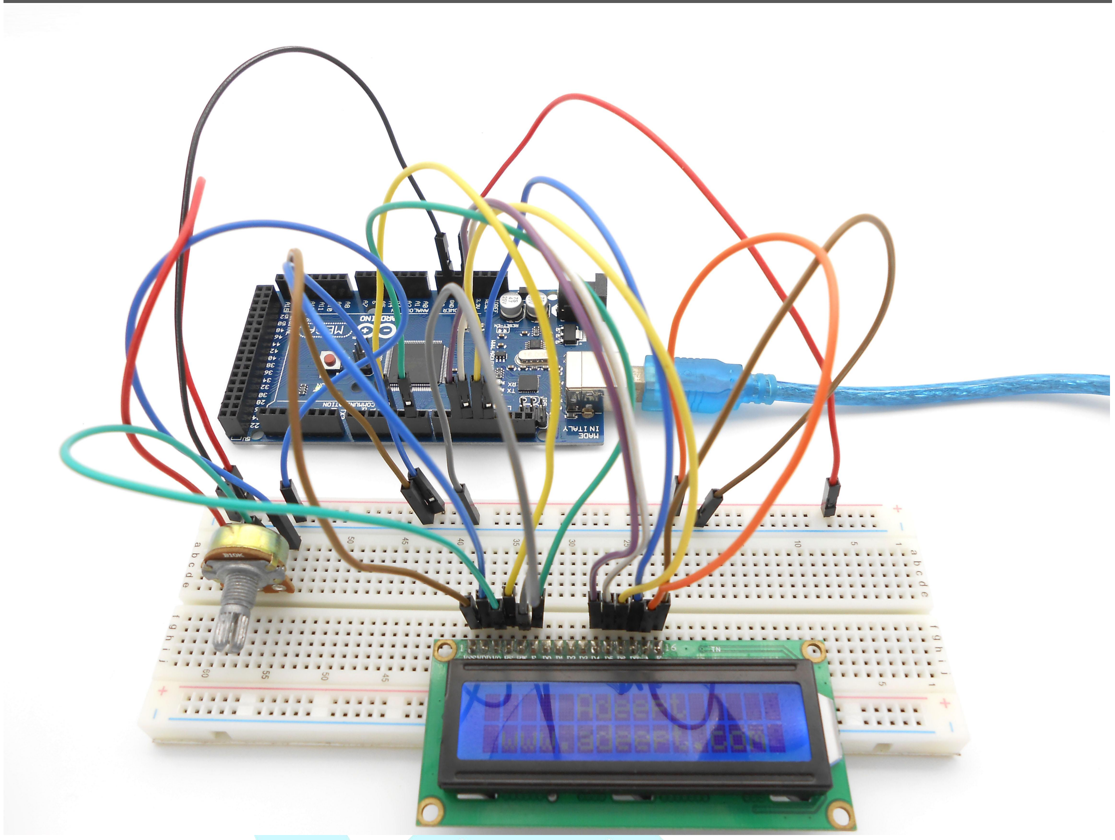


2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, you can see the string "Hello Geeks!" is shown on the LCD1602 scrolling, and then the string "Adeept" and "www.adeept.com" is displayed on the LCD1602 static.

Adeept



Summary

I believe that you have already mastered the driver of LCD1602 through this lesson. I hope you can make something more interesting base on this lesson and the previous lesson learned.

Adeept

Lesson 12 A Simple Voltmeter

Overview

In this lesson, we will make a simple voltmeter with Arduino MEGA 2560 and LCD1602, the range of this voltmeter is 0~5V. Then, we will measure the voltage of the potentiometer's adjustment end with the simple voltmeter and display it on the LCD1602.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* LCD1602
- 2* Potentiometer
- 1* Breadboard
- Several Jumper Wires

Principle

The basic principle of this experiment: Converting the analog voltage that the Arduino collected to digital quantity by the ADC(analog-to-digital converter) through programming, then display the voltage on the LCD1602.

Connect the three wires from the potentiometer to your Arduino board. The first goes to ground from one of the outer pins of the potentiometer. The second goes from analog input 0 to the middle pin of the potentiometer. The third goes from 5 volts to the other outer pin of the potentiometer.

By turning the shaft of the potentiometer, you change the amount of resistance on either side of the wiper which is connected to the center pin of the potentiometer. This changes the voltage at the center pin. When the resistance between the center and the side connected to 5 volts is close to zero (and the resistance on the other side is close to 10 kilohms), the voltage at the center pin nears 5 volts. When the resistances are reversed, the voltage at the center pin nears 0 volts, or ground. This voltage is the analog voltage that you're reading as an input.

The Arduino has a circuit inside called an analog-to-digital converter that reads this changing voltage and converts it to a number between 0 and 1023. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and the input value is 0. When the shaft is turned all the way in the opposite direction, there are 5 volts going to the pin and the input value is

1023. In between, analogRead( ) returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

Key functions:

● analogRead()

Reads the value from the specified analog pin. The Arduino MEGA 2560 board contains a 16 channel (8 channels on the Mini and Nano, 6 on the UNO), 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit. The input range and resolution can be changed using analogReference( ).

It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

Syntax

analogRead(pin)

Parameters

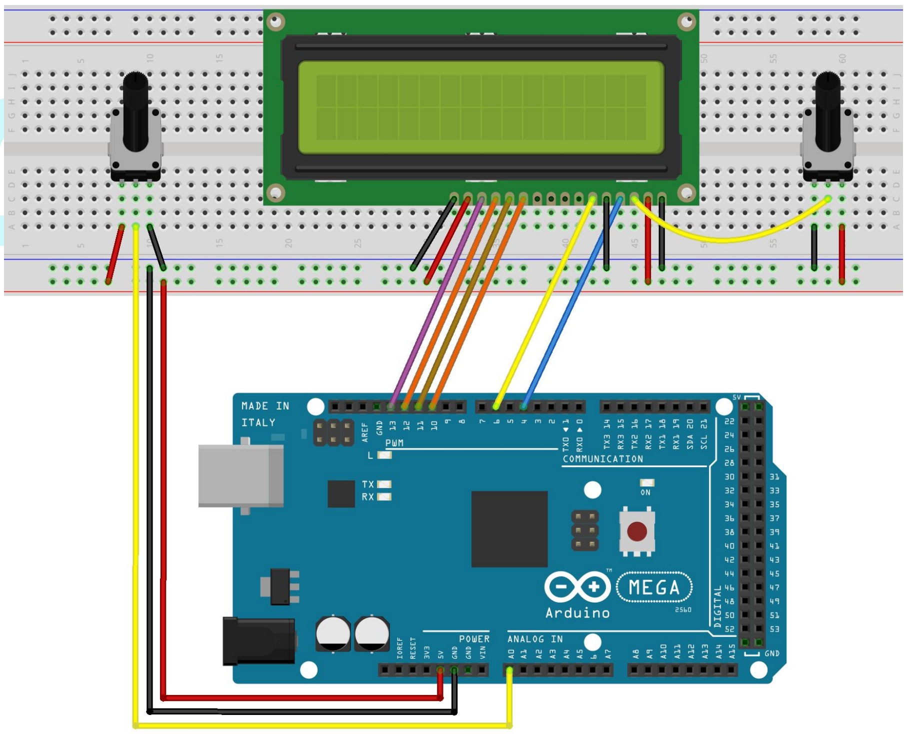
pin: the number of the analog input pin to read from (0 to 5 on most boards, 0 to 7 on the Mini and Nano, 0 to 15 on the Mega)

Returns

int (0 to 1023)

Procedures

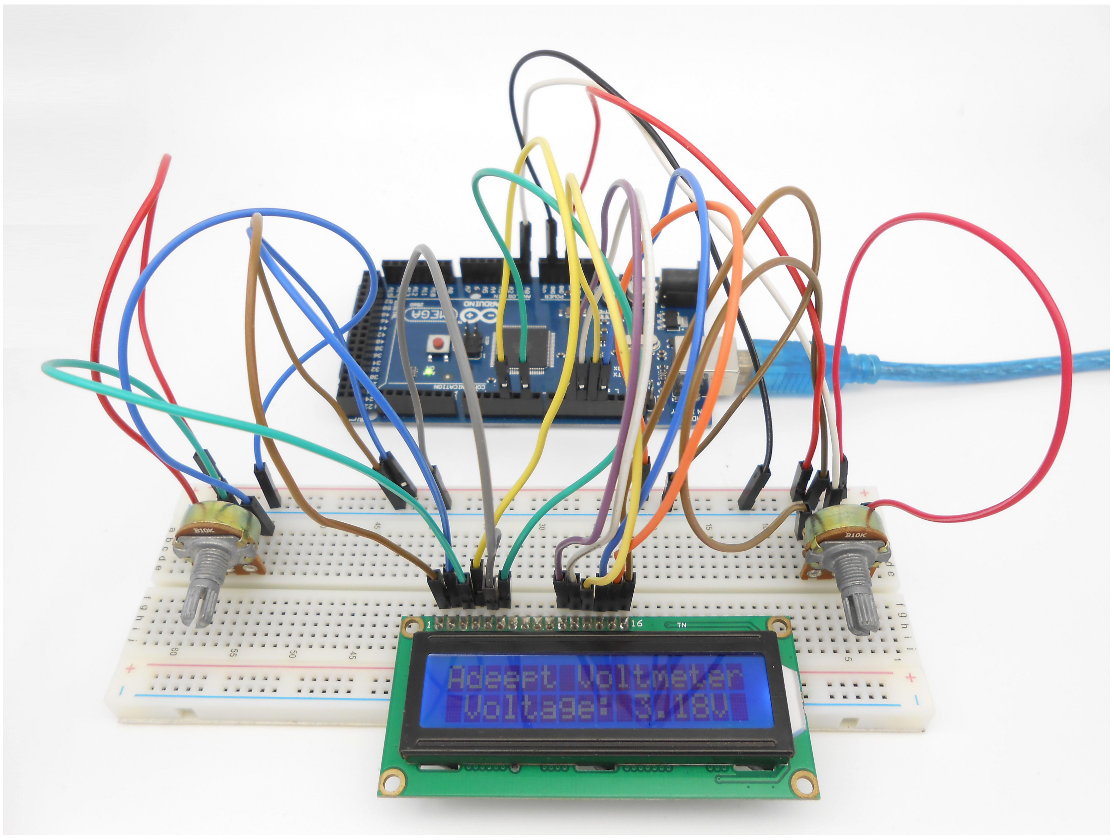
1. Build the circuit



2. Program

3. Compile the program and upload to Arduino MEGA 2560 board.

Now, when you turning the shaft of the potentiometer, you will see the voltage displayed on the LCD1602 will be changed.



Summary

The substance of voltmeter is reading analog voltage which input to ADC inside. Through this course, I believe that you have mastered how to read analog value and how to make a simple voltmeter with Arduino.

Adeept

Lesson 13 7-segment display

Overview

In this lesson, we will program the Arduino to achieve the controlling of segment display.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* 220Ω Resistor
- 1* 7-segment Display
- 1* Breadboard
- Several Jumper Wires

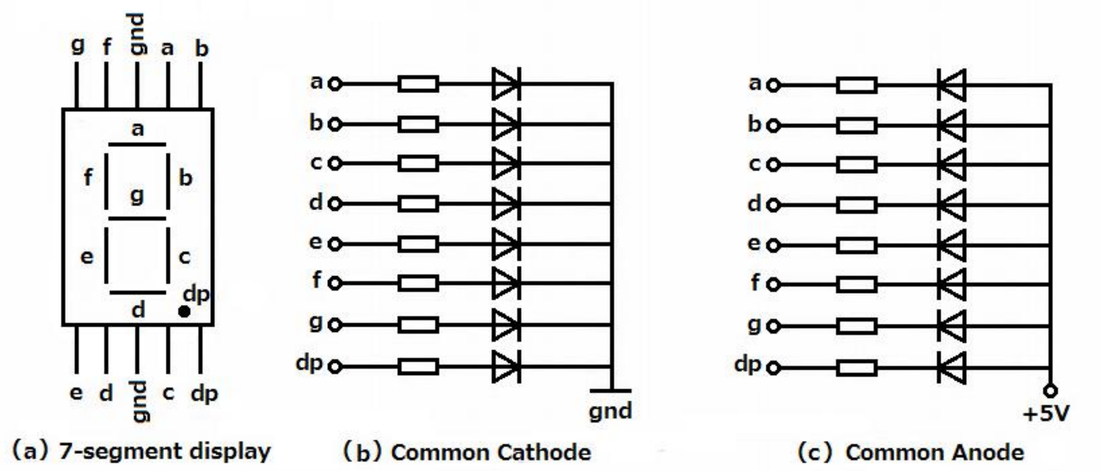
Principle

The seven-segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays.

Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information.

The seven-segment display is an 8-shaped LED display device composed of eight LEDs (including a decimal point), these segments respectively named a, b, c, d, e, f, g, dp.

The segment display can be divided into common anode and common cathode segment display by internal connections.

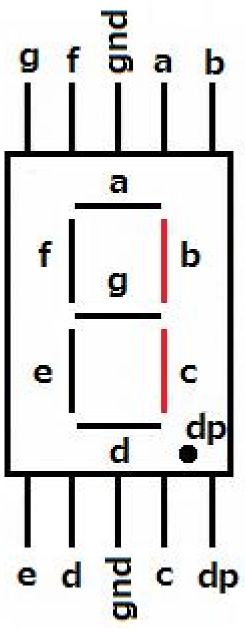


(a) 7-segment display (b) Common Cathode (c) Common Anode

When using a common anode LED, the common anode should to be connected to the power supply (VCC); when using a common cathode LED, the common cathode should be connected to the ground (GND).

Each segment of a segment display is composed of LED, so a resistor is needed for protecting the LED.

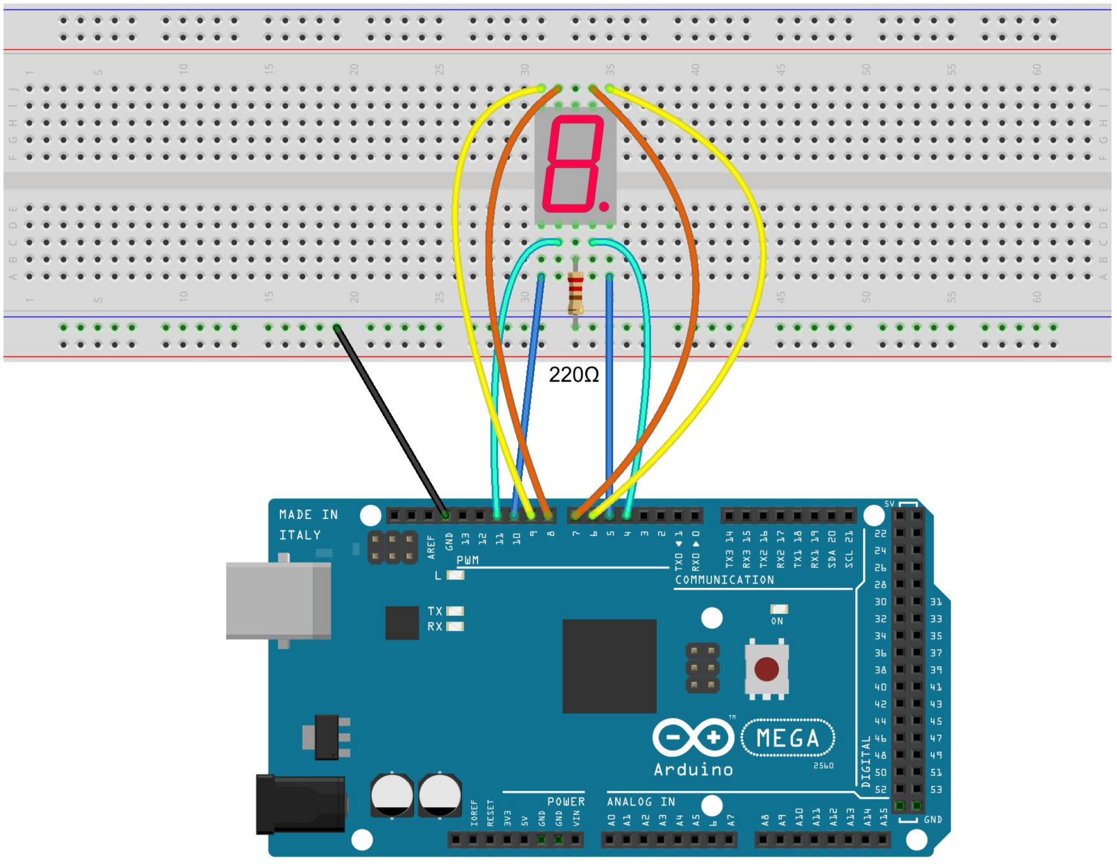
A 7-segment display has seven segments for displaying a figure and a segment for displaying a decimal point. If you want to display a number '1', you should only light the segment b and c.



Procedures

1. Build the circuit

Adeept

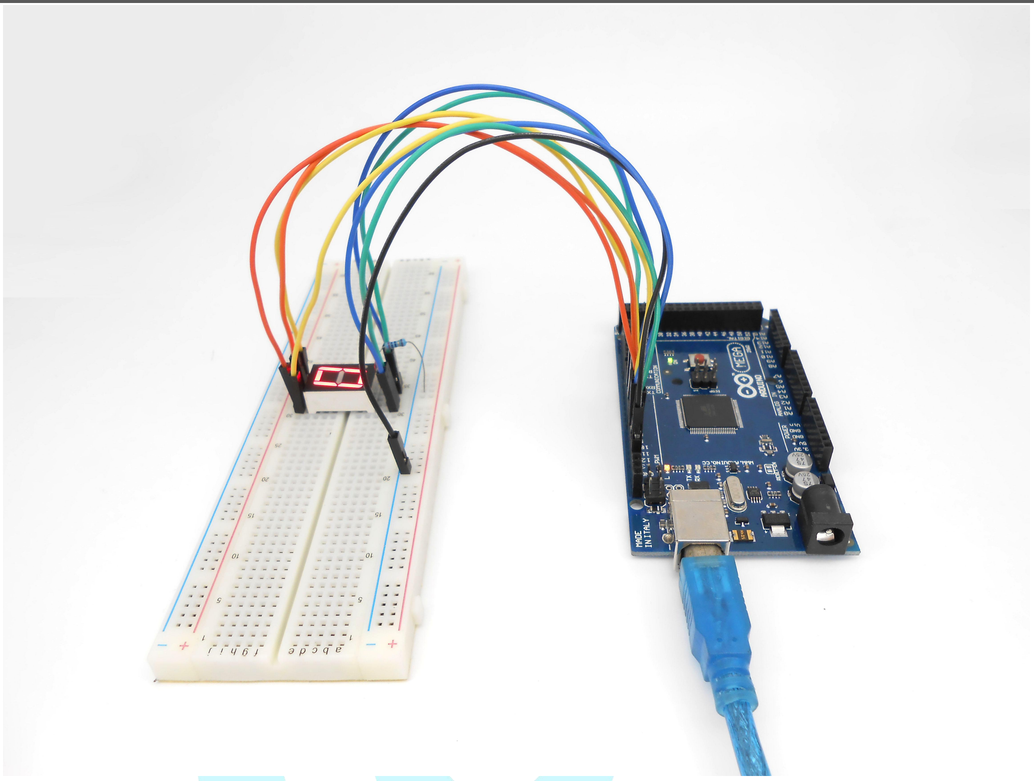


2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, you should see the number 0~9 are displayed on the segment display.

Adeept



Summary

Through this lesson, we have learned the principle and programming of segment display. I hope you can combine the former course to modify the code we provided in this lesson to achieve cooler originality.

Adeept

Lesson 14 A simple counter

Overview

In this lesson, we will program the Arduino MEGA 2560 to make a simple counter.

Requirement

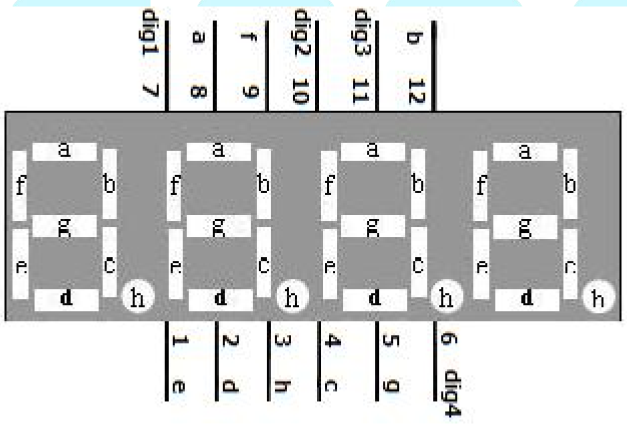
- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* 4-digit 7-segment Display
- 8* 220Ω Resistor
- 2* Button
- 1* Breadboard
- Several Jumper Wires

Principle

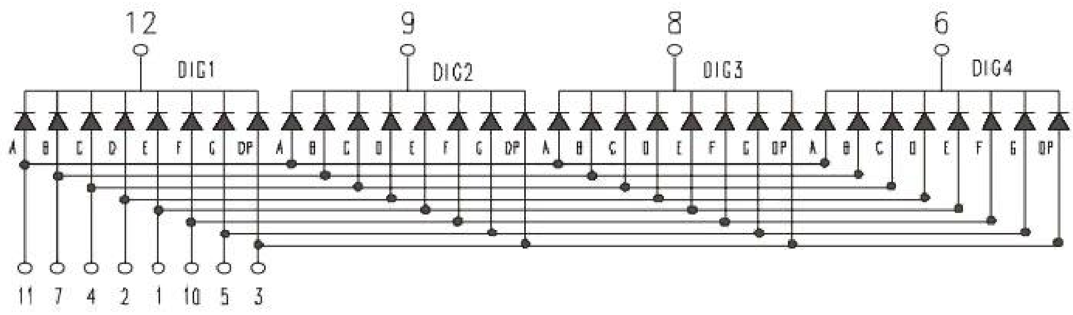
The 4-digit segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays.

4-digit segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information.

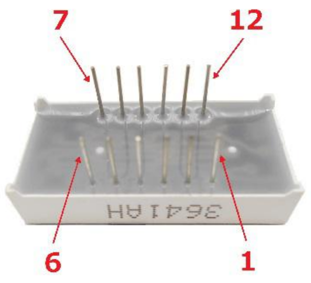
The 4-digit segment display is an 4*8-shaped LED display device composed of 32 LEDs (including four decimal points), these segments respectively named a, b, c, d, e, f, g, h, dig1, dig2, dig3, dig4.



What we used in this experiment is a common cathode 4-digit 7-segment display. Its internal structure is shown below:

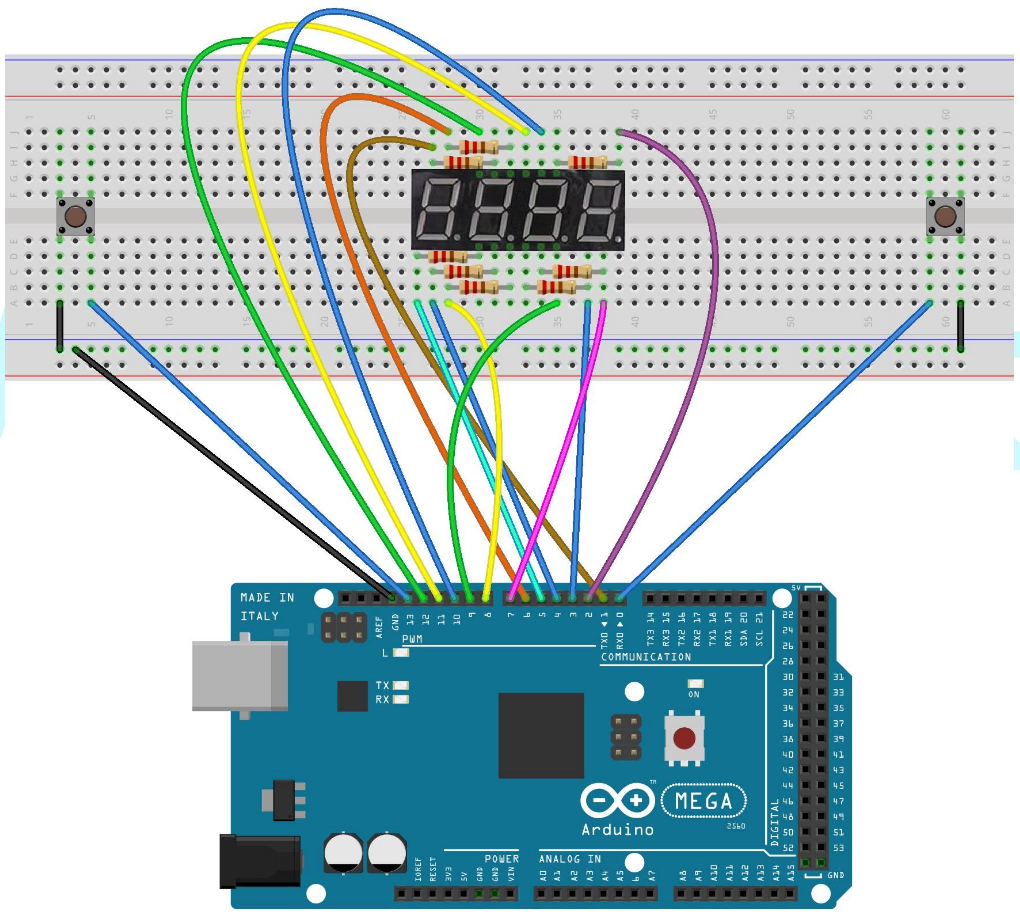


The pin number is showing below:



Procedures

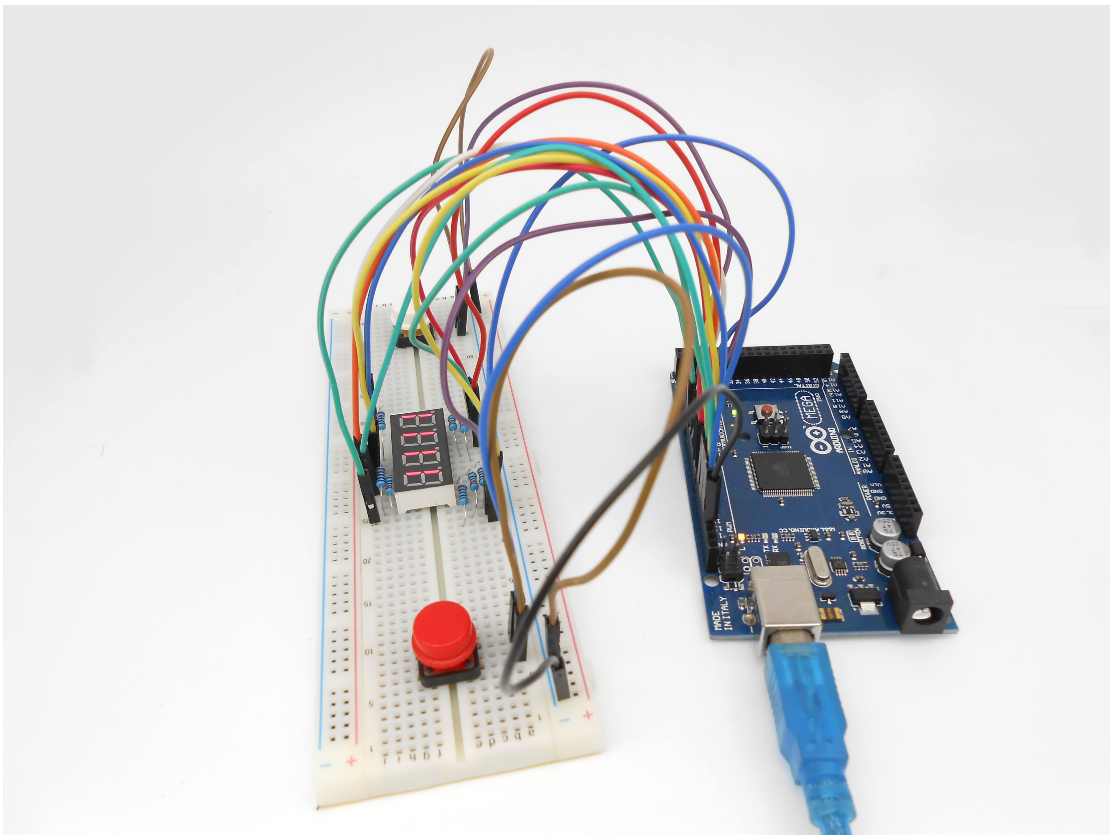
1. Build the circuit



2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, when you press one of the two buttons, the value displayed on the 4-digit 7-segment display will be changed.



Summary

By learning this lesson, you'll find that it is so easy to make a simple counter.

Adeept

Lesson 15 Controlling Servo motor

Overview

In this lesson, we will introduce a new electronic device (Servo) to you, and tell you how to control it with the Arduino MEGA 2560.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* Servo
- Several Jumper Wires

Principle

1. Servo motor

The servo motor have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the Arduino board. The signal pin is typically yellow, orange or white and should be connected to a digital pin on the Arduino board. Note the servo motor draw considerable power, so if you need to drive more than one or two, you'll probably need to power them from a separate supply (i.e. not the +5V pin on your Arduino). Be sure to connect the grounds of the Arduino and external power supply together.

2. Servo library

This library allows an Arduino board to control RC (hobby) servo motors. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

3. Key functions:

●attach()

Attach the Servo variable to a pin. Note that in Arduino 0016 and earlier, the Servo library supports only servos on only two pins: 9 and 10.

Syntax

servo.attach(pin)

servo.attach(pin, min, max)

Parameters

servo: a variable of type Servo

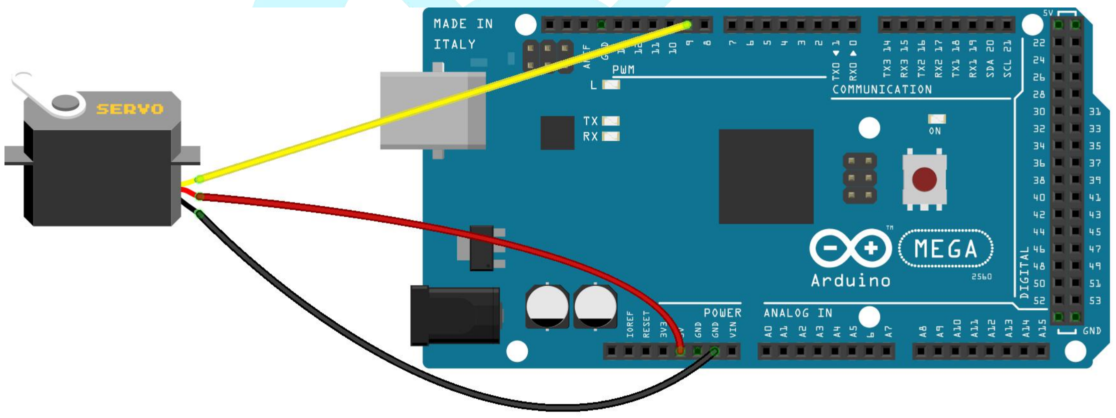
pin: the number of the pin that the servo is attached to

min (optional): the pulse width, in microseconds, corresponding to the minimum (0-degree) angle on the servo (defaults to 544)

max (optional): the pulse width, in microseconds, corresponding to the maximum (180-degree) angle on the servo (defaults to 2400)

Procedures

1. Build the circuit

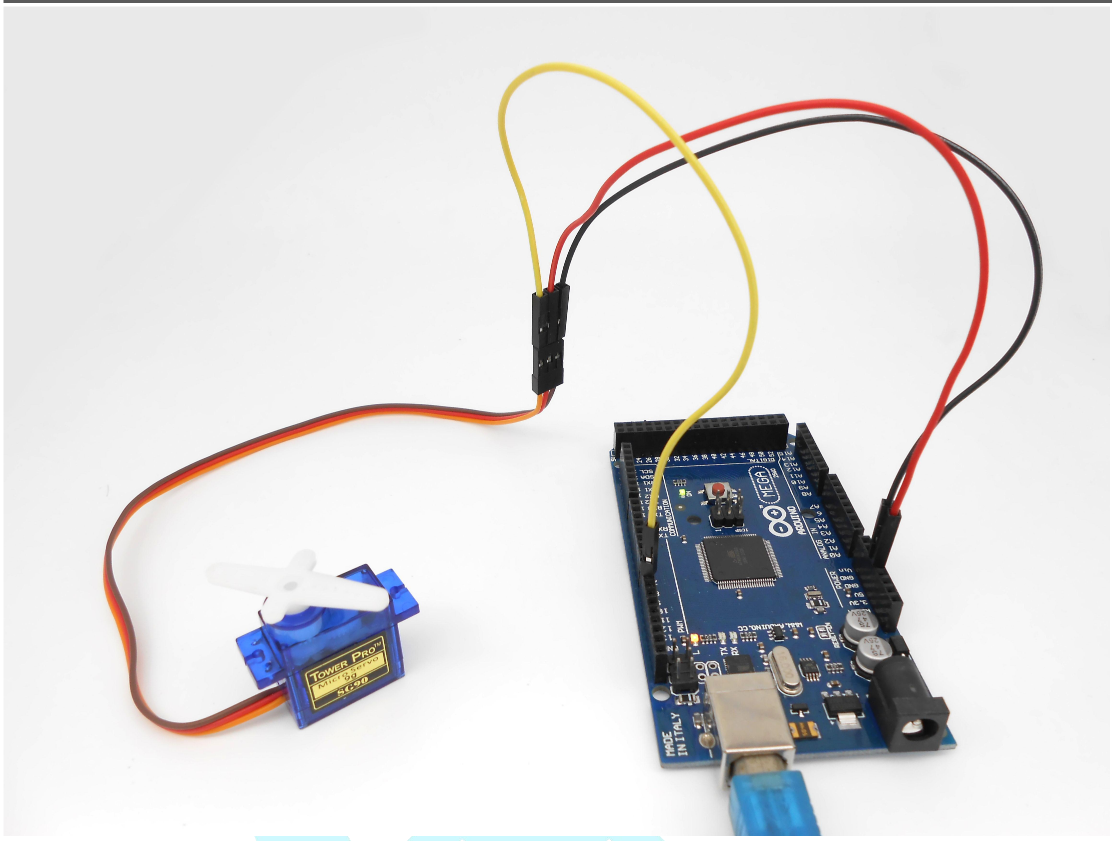


2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, you should see the servo rotate 180 degrees, and then rotate in opposite direction.

Adeept



Summary

By learning this lesson, you should have known that the Arduino provided a servo library to control a servo. By using the servo library, you can easily control a servo. Just enjoy your imagination and make some interesting applications.

Adeept

Lesson 16 Using a thermistor to measure the temperature

Overview

In this lesson, we will learn how to use a thermistor to collect temperature by programming Arduino. The information which a thermistor collects temperature is displayed on the LCD1602.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* LCD1602
- 1* 10KΩ Potentiometer
- 1* 10KΩ Resistor
- 1* Thermistor
- 1* Breadboard
- Several Jumper Wires

Principle

A thermistor is a type of resistor whose resistance varies significantly with temperature, more so than in standard resistors. We are using MF52 NTC thermistor type. BTC thermistor is usually used as a temperature sensor.

MF52 thermistor key parameters:

B-parameter : 3470.

25℃ resistor : 10KΩ.

The relationship between the resistance of thermistor and temperature is as follows:

$$R_{thermistor} = R * e^{\left(B*\left(\frac{1}{T_1}-\frac{1}{T_2}\right)\right)}$$

$R_{thermistor}$: The resistance of thermistor at temperature T1

R : The nominal resistance of thermistor at room temperature T2;

e : 2.718281828459 ;

B : It is one of the important parameters of thermistor;

T_1 : The Kelvin temperature that you want to measure.

T_2 : At the condition of room temperature 25 ℃ (298.15K), the standard resistance of MF52 thermistor is 10K;

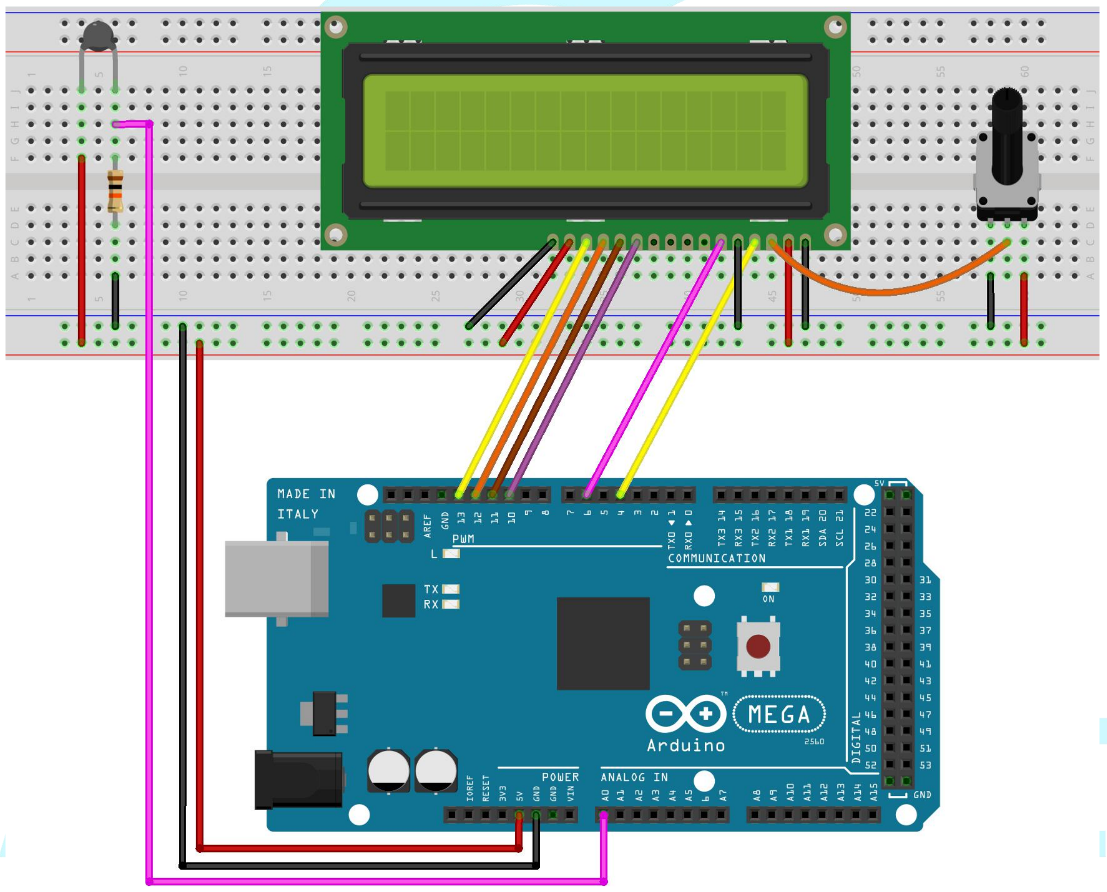
Kelvin temperature = 273.15 (absolute temperature) + degrees Celsius;

After transforming the above equation, we can get to the following formula:

$$T_1 = \frac{B}{\left(ln^{\left(\frac{R_{thermistor}}{R}\right)} + \frac{B}{T_2}\right)}$$

Procedures

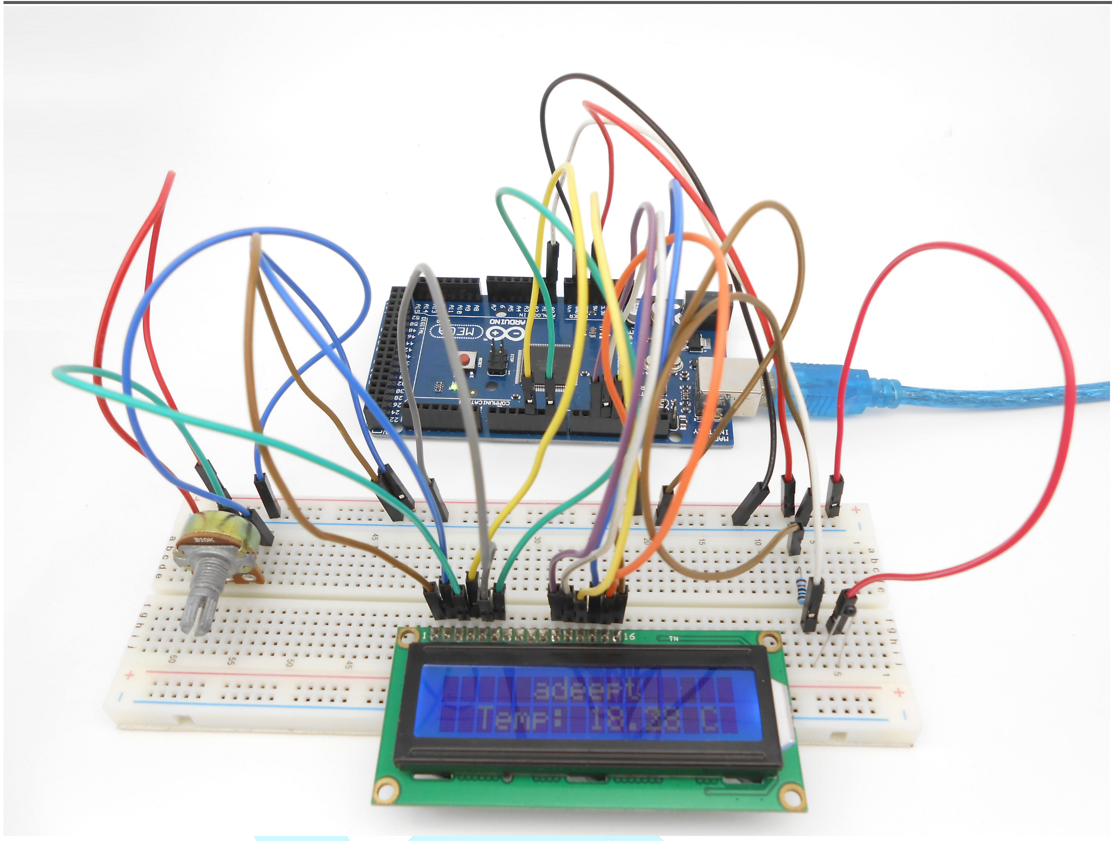
1. Build the circuit



2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, you can see the temperature which is collected by thermistor on the LCD1602.



Summary

By learning this lesson, I believe you have learned to use a thermistor to measure temperature. Next, you can use a thermistor to produce some interesting applications.

Adeept

Lesson 17 IR Remoter Controller

Overview

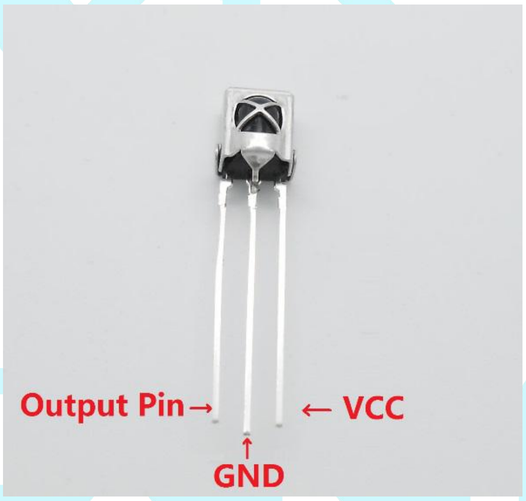
In this lesson, we will learn how to use an IR receiver to receive the remote controller signal.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* IR Receiver HX1838
- 1* Remote Controller
- 1* Breadboard
- Several Jumper Wires

Principle

The IR receiver HX1838 can receive the infrared remote controller signals. The IR receiver HX1838 has only three pins (a signal line, VCC and GND). It is easy to connect with the Arduino.



The following is an infrared remote controller:

In this experiment, we program the Arduino to receive the infrared signal, and then send the received data to the serial monitor. In the program, we used the Arduino-IRremote-master library (We provided).

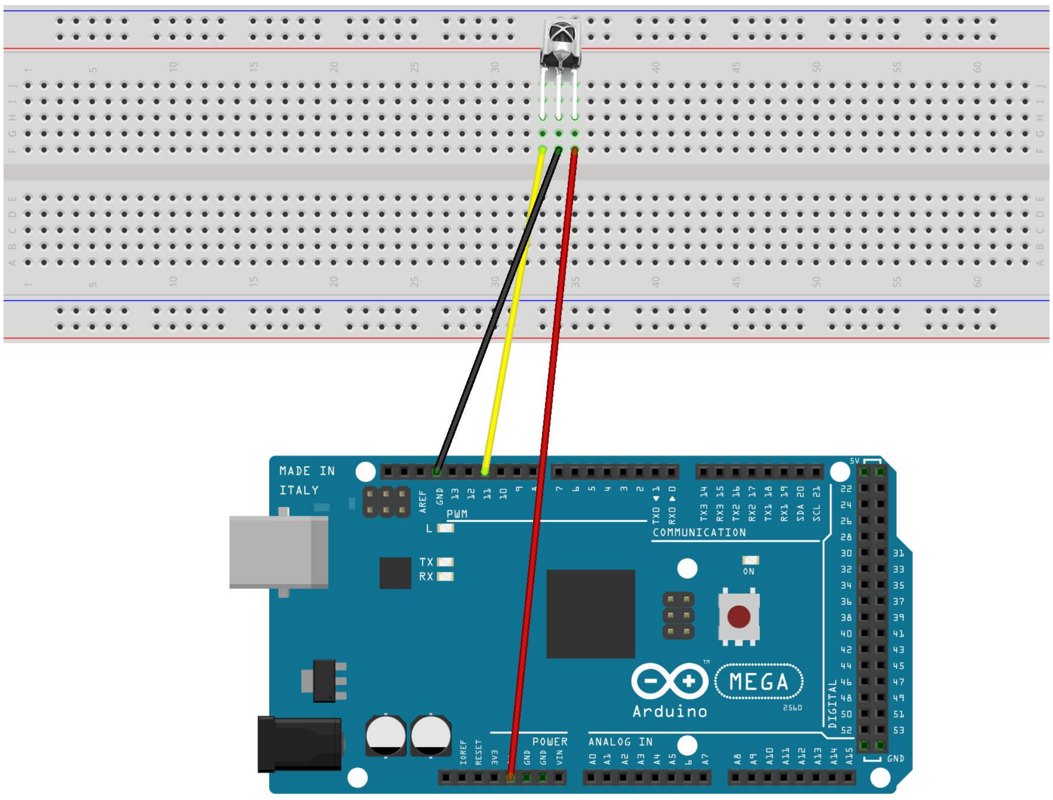
Note:

Before using this library, you have to delete the RobotIRremote directory in your Arduino IDE directory, and delete the RobotIRremote directory in system documents folder. For example, my system is windows 7, I need to delete the RobotIRremote directory in ***C:\Program Files (x86)\Arduino\libraries*** and ***C:\Users\SJG\Documents\Arduino\libraries***. Otherwise, when you compile the program, the compiler will complain.

Procedures

1. Build the circuit

Adeept

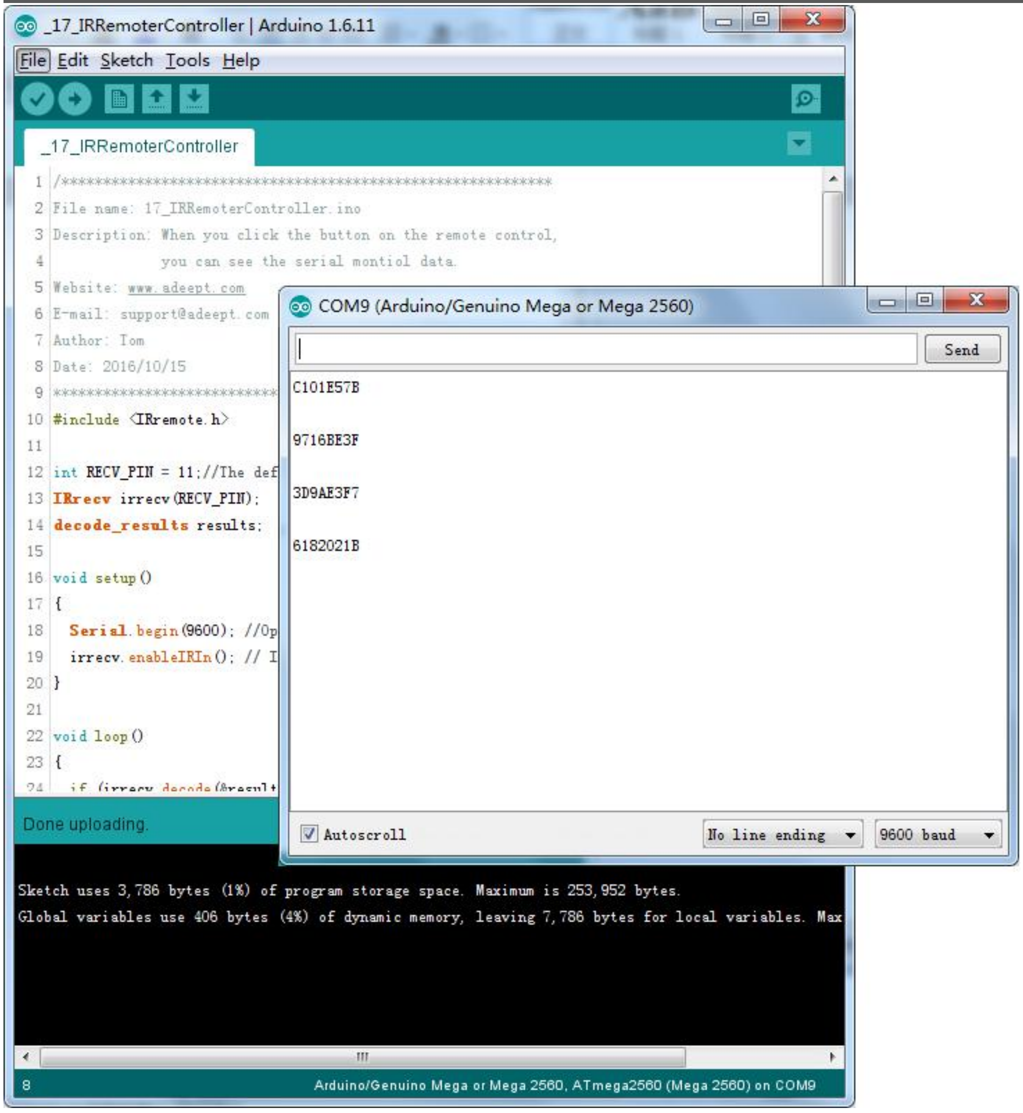


2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, when you click one of the buttons on the remote controller, you will see the button number of the remote controller is displayed on the serial monitor.

Adeept


_17_IRRemoterController | Arduino 1.6.11
File Edit Sketch Tools Help
_17_IRRemoterController
COM9 (Arduino/Genuino Mega or Mega 2560)
Send
Autoscroll
No line ending
9600 baud
Done uploading.

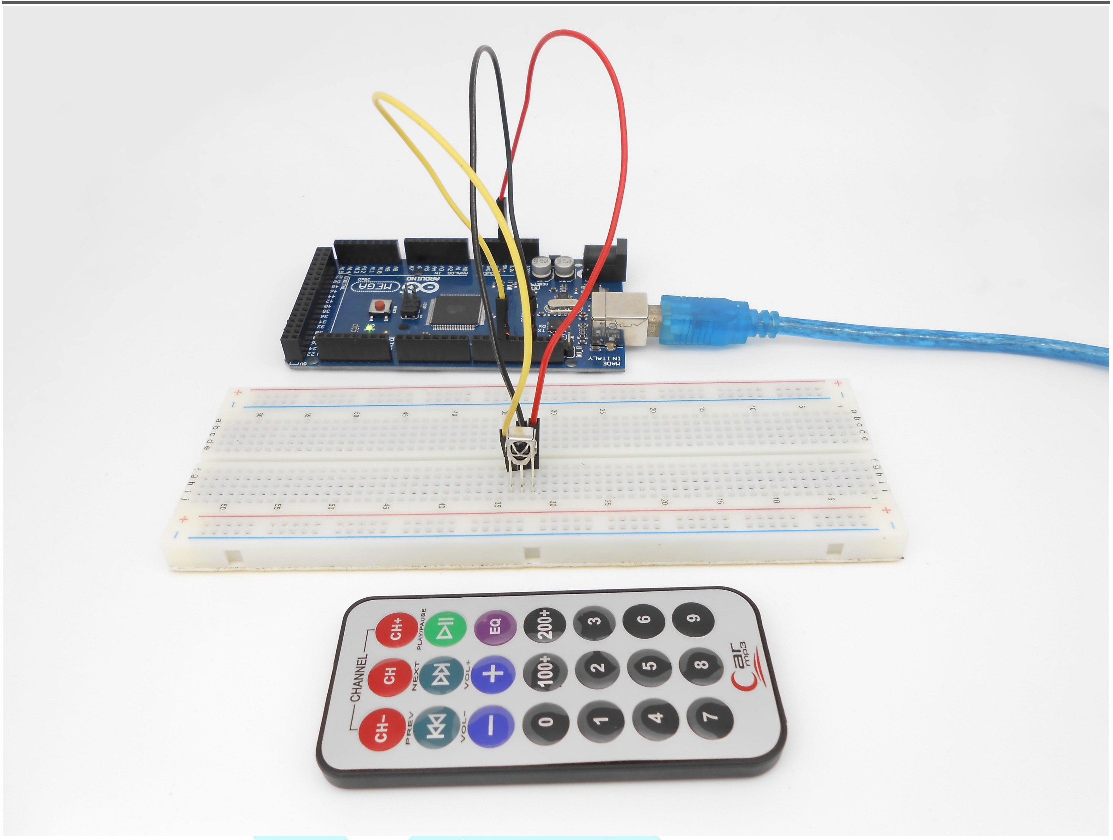
```
/*************************************************************
File name: 17_IRRemoterController.ino
Description: When you click the button on the remote control,
             you can see the serial montiol data.
Website: www.adeept.com
E-mail: support@adeept.com
Author: Tom
Date: 2016/10/15
*****************************
#include <IRremote.h>

int RECV_PIN = 11;//The def
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  Serial.begin(9600); //Op
  irrecv.enableIRIn(); // I
}

void loop()
{
  if (irrecv.decode(&result
```

```
C101E57B

9716BE3F

3D9AE3F7

6182021B
```

```
Sketch uses 3,786 bytes (1%) of program storage space. Maximum is 253,952 bytes.
Global variables use 406 bytes (4%) of dynamic memory, leaving 7,786 bytes for local variables. Max
```

8 Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Adeept



Summary

By learning this lesson, I believe you have mastered the basic principle of the infrared remote controlling.

Adeept

Lesson 18 DHT-11 Sensor Module

Introduction

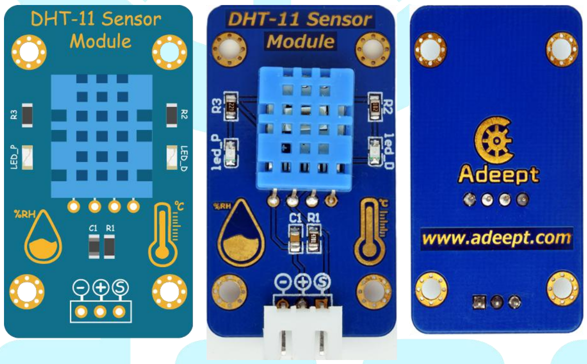
DHT11 is a composite digital thermal sensor that integrates temperature and humidity detection. It can convert the temperature and humidity analog values into digital values via corresponding sensitive components and built-in circuits, which can be directly read by computer or other data collecting devices.

Components

- 1 * Arduino MEGA 2560
- 1 * DHT-11 Sensor Module
- 1 * LCD1602
- 1 * 10KΩ Potentiometer
- 1 * USB Cable
- 1 * 3-Pin Wires
- Several Jumper Wires

Experimental Principle

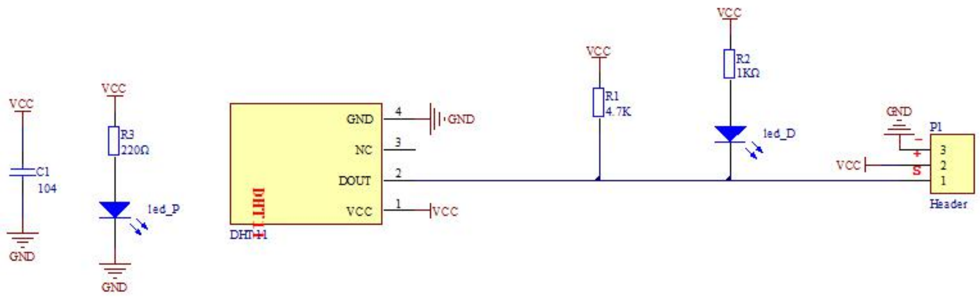
The Fritzing image:



Pin definition:

S	Digital output
+	VCC
-	GND

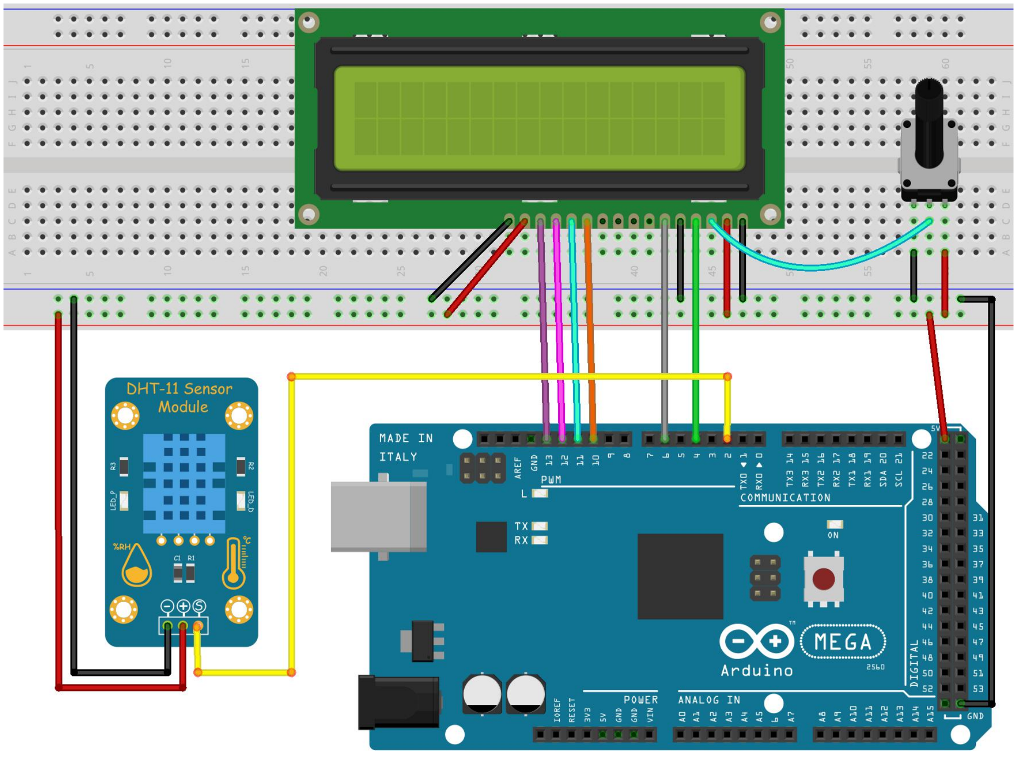
The schematic diagram:



In this experiment, by programming the Arduino, we read the temperature and humidity data collected by the DHT11 module by pin D2 of the Arduino board and display it on Serial Monitor via the serial port.

Experimental Procedures

Step 1: Build the circuit



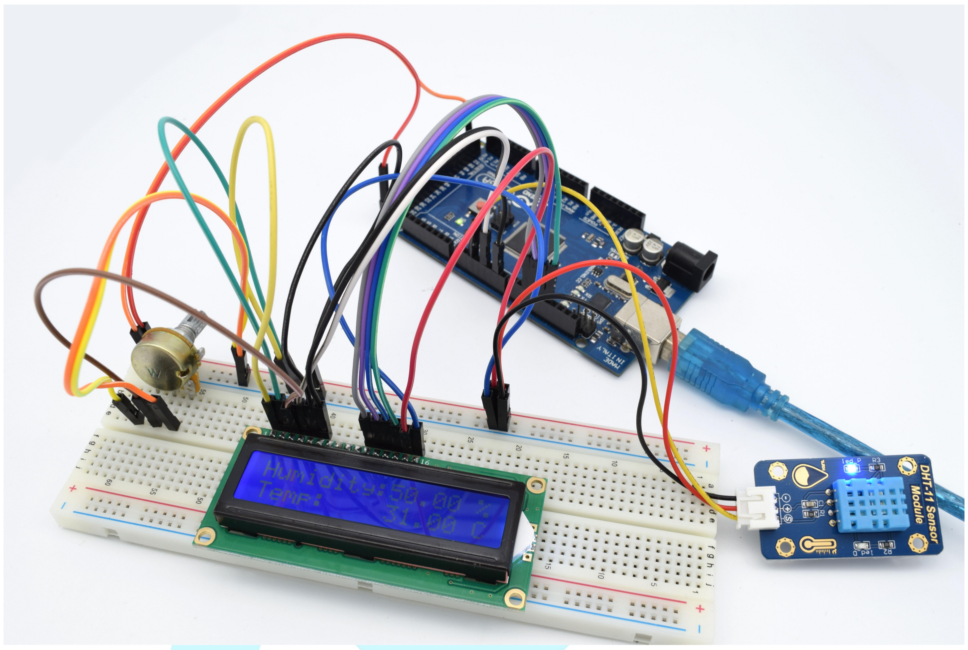
Adeept UNO R3 Board	DHT-11 Sensor Module
D2	S
5V	+
GND	-

Step 2: Program _18_DHT_11.ino

Step 3: Compile and download the sketch to the 2560 board.

Open the Serial Monitor in Arduino IDE and you will see the data of current temperature and humidity displayed on the window.

Adeept



Adeept

Lesson 19 Ultrasonic distance sensor

Overview

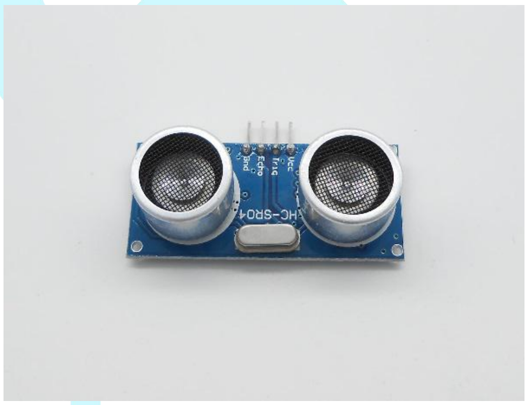
In this lesson, we will learn how to measure the distance by the ultrasonic distance sensor.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* Ultrasonic Distance Sensor
- 1* LCD1602
- 1* 10KΩ Potentiometer
- Several Jumper Wires

Principle

This recipe uses the popular Parallax PING ultrasonic distance sensor to measure the distance of an object ranging from 2 cm to around 3 m.

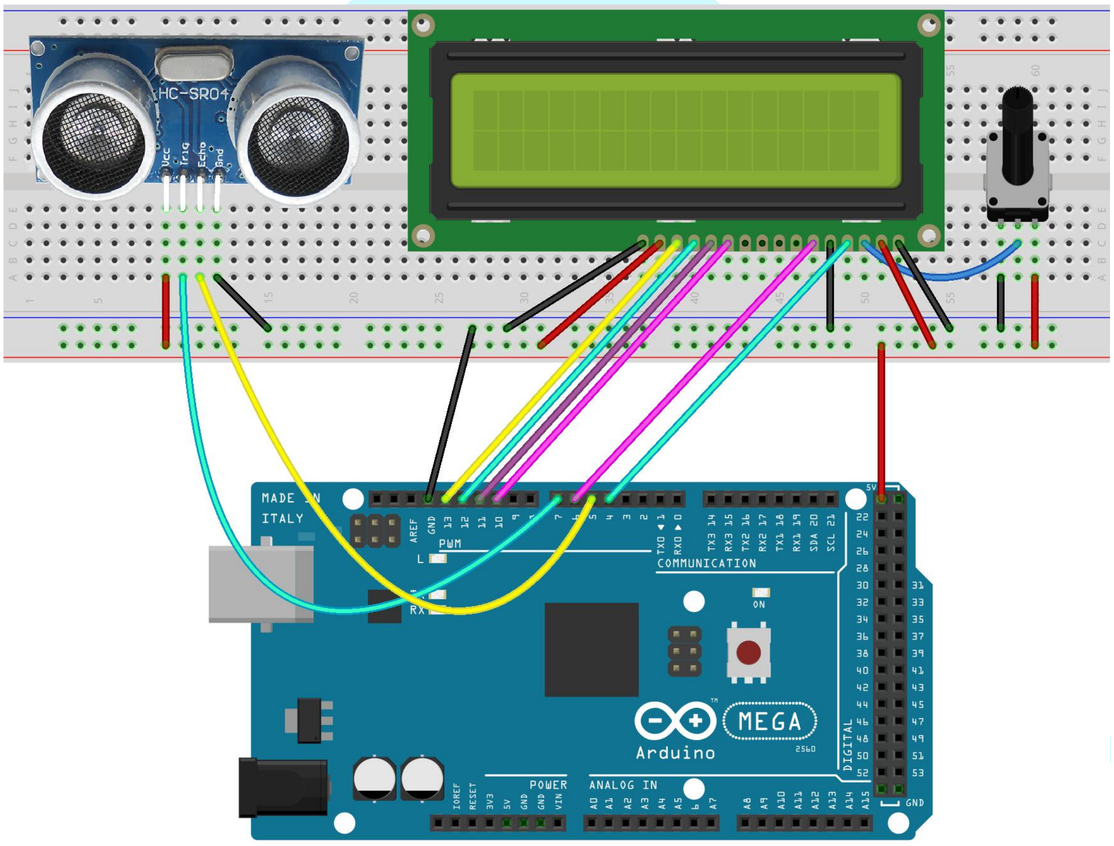


Ultrasonic sensors provide a measurement of the time it takes for sound to bounce off an object and return to the sensor. The “ping” sound pulse is generated when the pingPin level goes HIGH for two micro-seconds. The sensor will then generate a pulse that terminates when the sound returns. The width of the pulse is proportional to the distance the sound traveled and the sketch then uses the pulseIn function to measure that duration. The speed of sound is 340 meters per second, which is 29 microseconds per centimeter. The formula for the distance of the round trip is: RoundTrip = microseconds / 29.

So, the formula for the one-way distance in centimeters is: microseconds / 29 / 2

Procedures

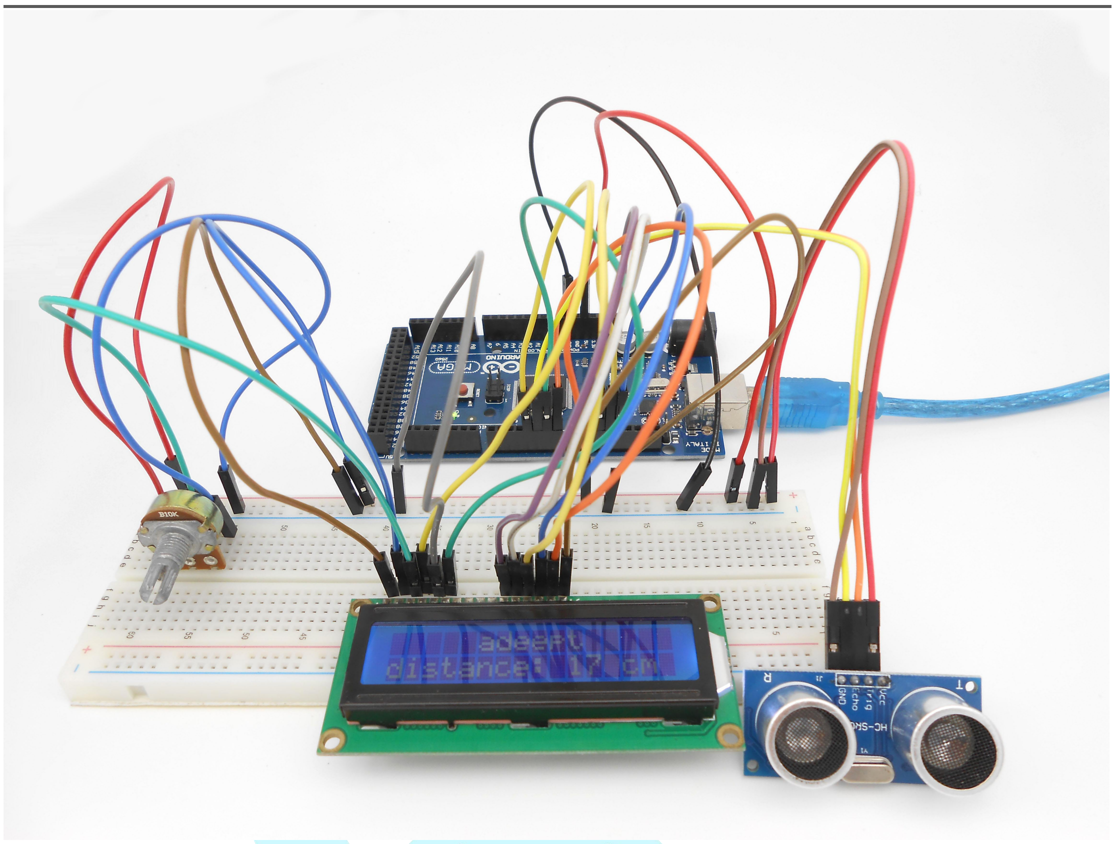
1. Build the circuit



2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, when you try to change the distance between the ultrasonic module and the obstacles, you will find the distance value displayed on the LCD1602 will be changed.



Adeept

Lesson 20 4x4 matrix keyboard

Overview

In this lesson, we will learn how to use the matrix keyboard.

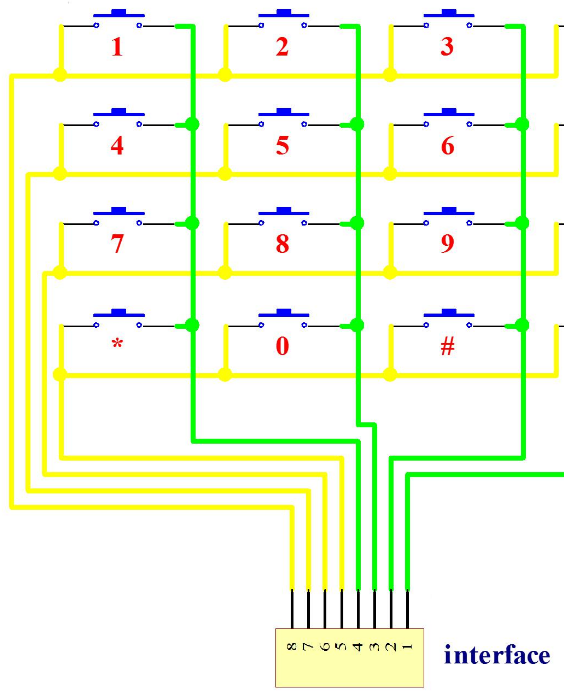
Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* 4x4 Matrix Keyboard
- Several Jumper Wires

Principle

In order to save the resources of the microcontroller port, we usually connect the buttons in a matrix in an actual project.

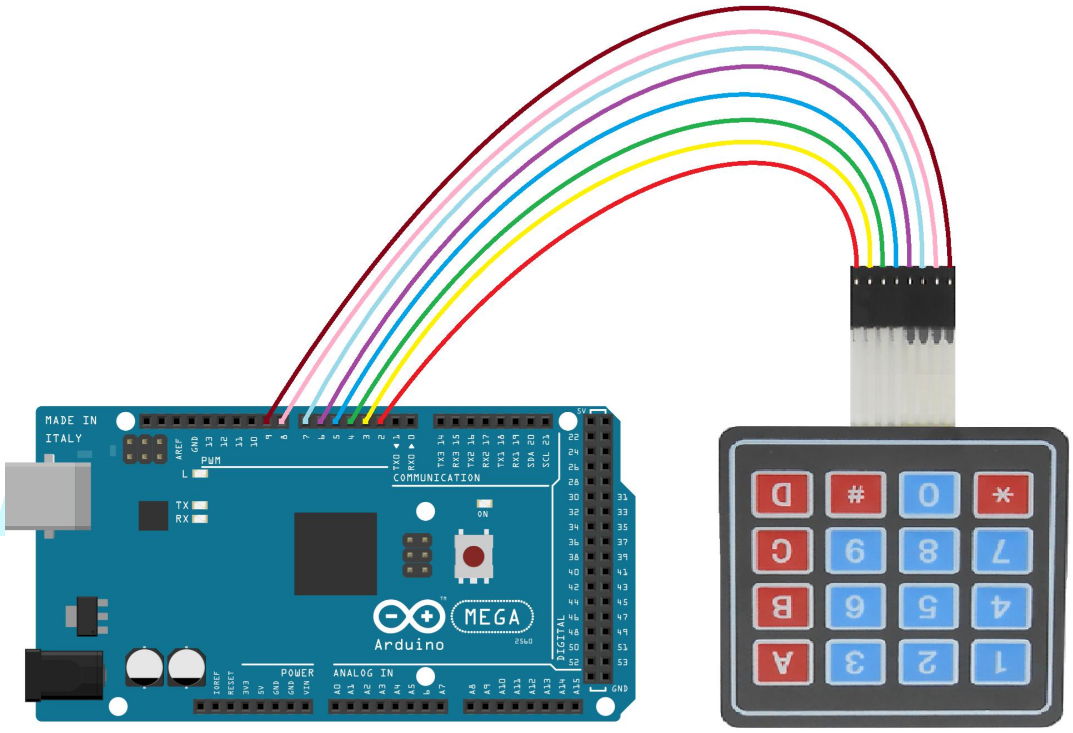
The following is the schematics of 4x4 matrix keyboard:



In this tutorial, we use the ‘Keypad’ function library. Before programming, please install the library.

Procedures

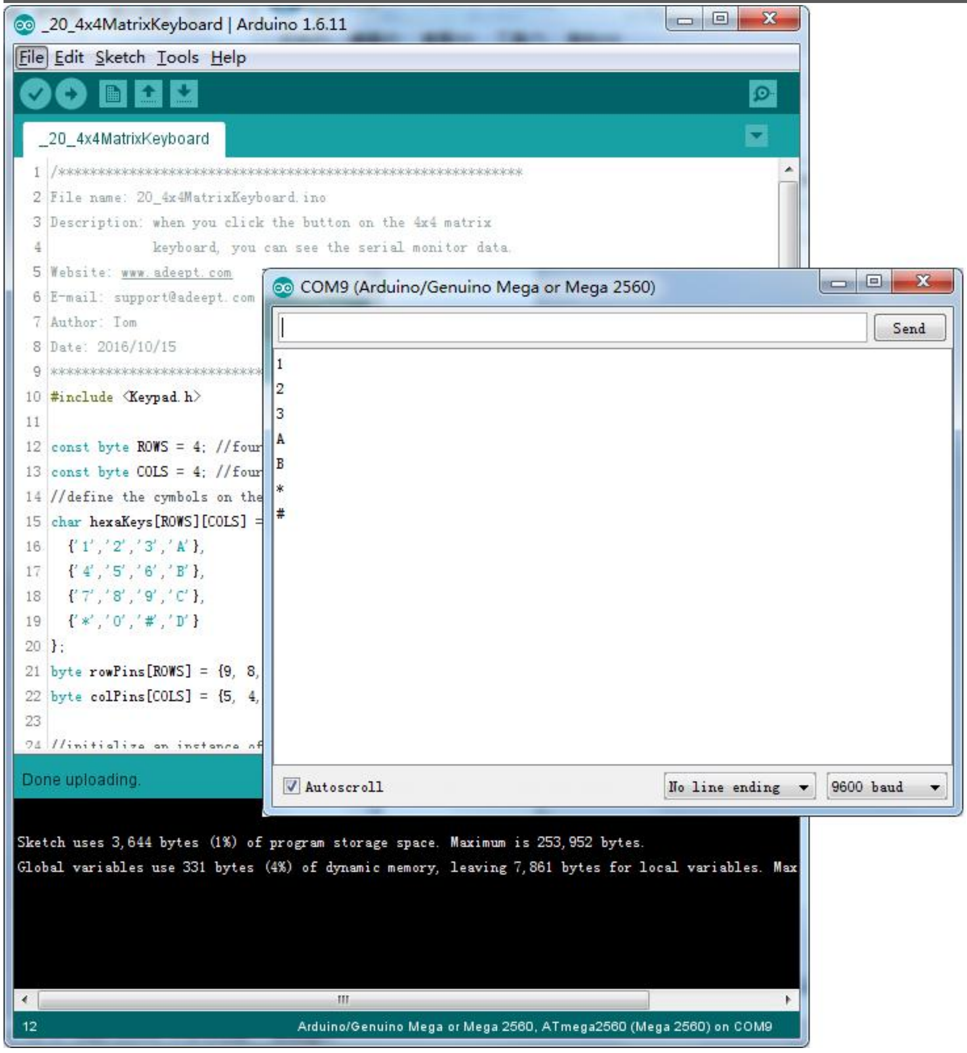
1. Build the circuit



2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, when you click one of the button on the 4x4 matrix keyboard, you will see the corresponding key value will be displayed on the serial monitor.

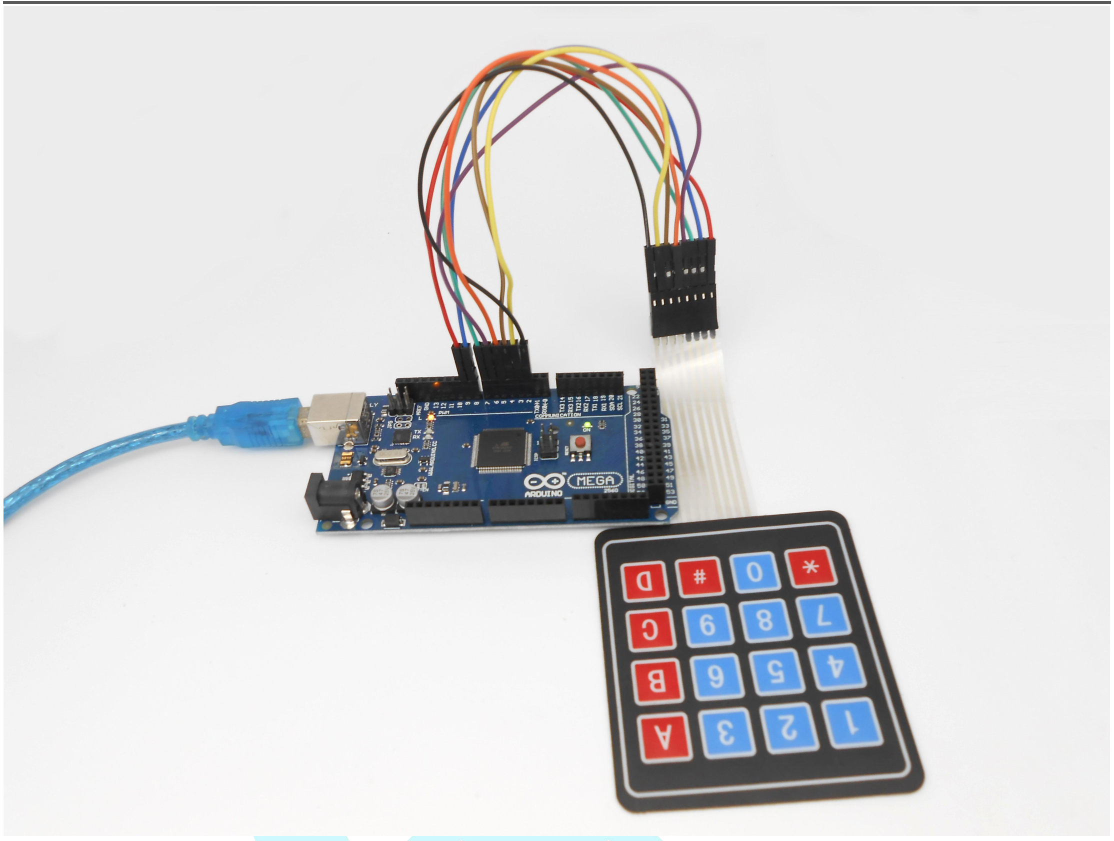

_20_4x4MatrixKeyboard | Arduino 1.6.11
File Edit Sketch Tools Help
_20_4x4MatrixKeyboard
COM9 (Arduino/Genuino Mega or Mega 2560)
Send
Autoscroll
No line ending
9600 baud
Done uploading.
12
Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

```
/*************************************************************
File name: 20_4x4MatrixKeyboard.ino
Description: when you click the button on the 4x4 matrix
             keyboard, you can see the serial monitor data.
Website: www.adeept.com
E-mail: support@adeept.com
Author: Tom
Date: 2016/10/15
****************************
#include <Keypad.h>

const byte ROWS = 4; //four
const byte COLS = 4; //four
//define the cymbols on the
char hexaKeys[ROWS][COLS] =
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {9, 8,
byte colPins[COLS] = {5, 4,

//initialize an instance of
```

```
1
2
3
A
B
*
#
```

```
Sketch uses 3,644 bytes (1%) of program storage space. Maximum is 253,952 bytes.
Global variables use 331 bytes (4%) of dynamic memory, leaving 7,861 bytes for local variables. Max
```

Adeept



Adeept

Lesson 21 Controlling DC motor

Overview

In this comprehensive experiment, we will learn how to control the state of DC motor with Arduino, and the state will be displayed through the LED at the same time. The state of DC motor includes its forward, reverse, acceleration, deceleration and stop.

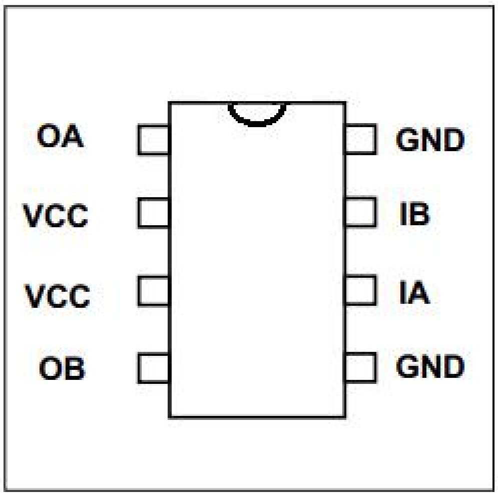
Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* L9110 DC Motor Driver
- 1* DC Motor
- 4* Button
- 4* LED
- 4* 220Ω Resistor
- 1* 9V Battery Clip
- 1* Breadboard
- Several Jumper Wires

Principle

1. L9110

L9110 is a driver chip which is used to control and drive motor. The chip has two TTL/CMOS compatible input terminals, and possesses the property of anti-interference: it has high current driving capability, two output terminals that can directly drive DC motor, each output port can provide 750~800mA dynamic current, and its peak current can reach 1.5~2.0A; L9110 is widely applied to various motor drives, such as toy cars, stepper motor, power switches and other electric circuits.



OA, OB: These are used to connect the DC motor.

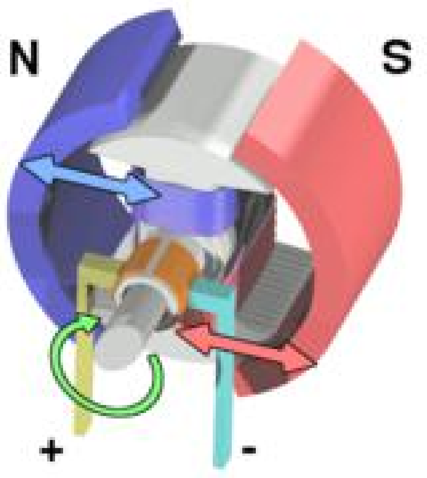
VCC: Power supply (+5V)

GND: The cathode of the power supply (Ground).

IA, IB: The input terminal of drive signal.

2. DC motor

A DC motor is any of a class of electrical machines that converts direct current electrical power into mechanical power. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor. Most types produce rotary motion; a linear motor directly produces force and motion in a straight line.



DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight motor used for portable power tools and appliances.

3. Key functions

●switch / case statements

Like if statements, switch…case controls the flow of programs by allowing programmers to specify different code that should be executed in various conditions. In particular, a switch statement compares the value of a variable to the values specified in case statements. When a case statement is found whose value matches that of the variable, the code in that case statement is run.

The break keyword exits the switch statement, and is typically used at the end of each case. Without a break statement, the switch statement will continue executing the following expressions (“falling-through”) until a break, or the end of the switch statement is reached.

Example

```
switch (var) {
    case 1:
      //do something when var equals 1
      break;
    case 2:
      //do something when var equals 2
```

```
     break;
   default:
     // if nothing else matches, do the default
     // default is optional
  }
```

Syntax

```
switch (var) {
  case label:
    // statements
    break;
  case label:
    // statements
    break;
  default:
    // statements
}
```

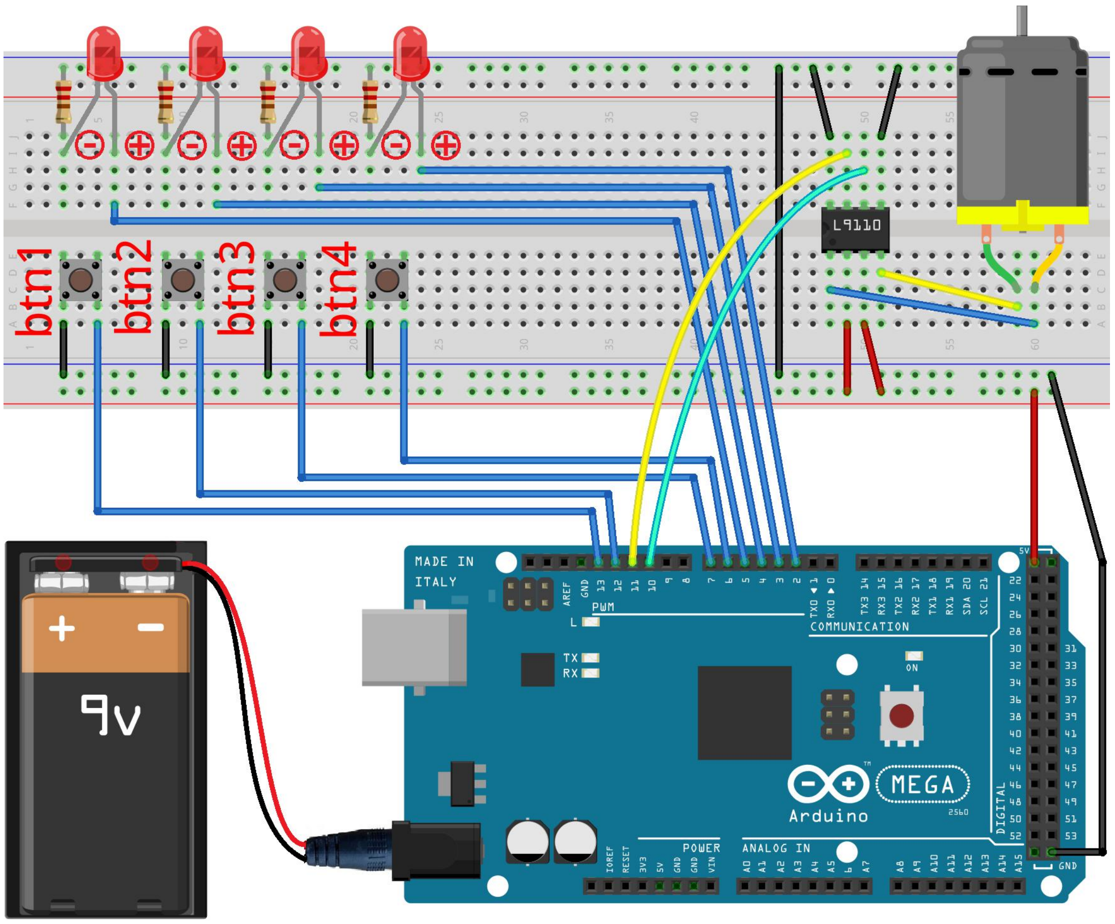
Parameters

var: the variable whose value to compare to the various cases label: a value to compare the variable to

Procedures

1. Build the circuit (Make sure that the circuit connection is correct and then power, otherwise it may cause the chips to burn.)

Adeept

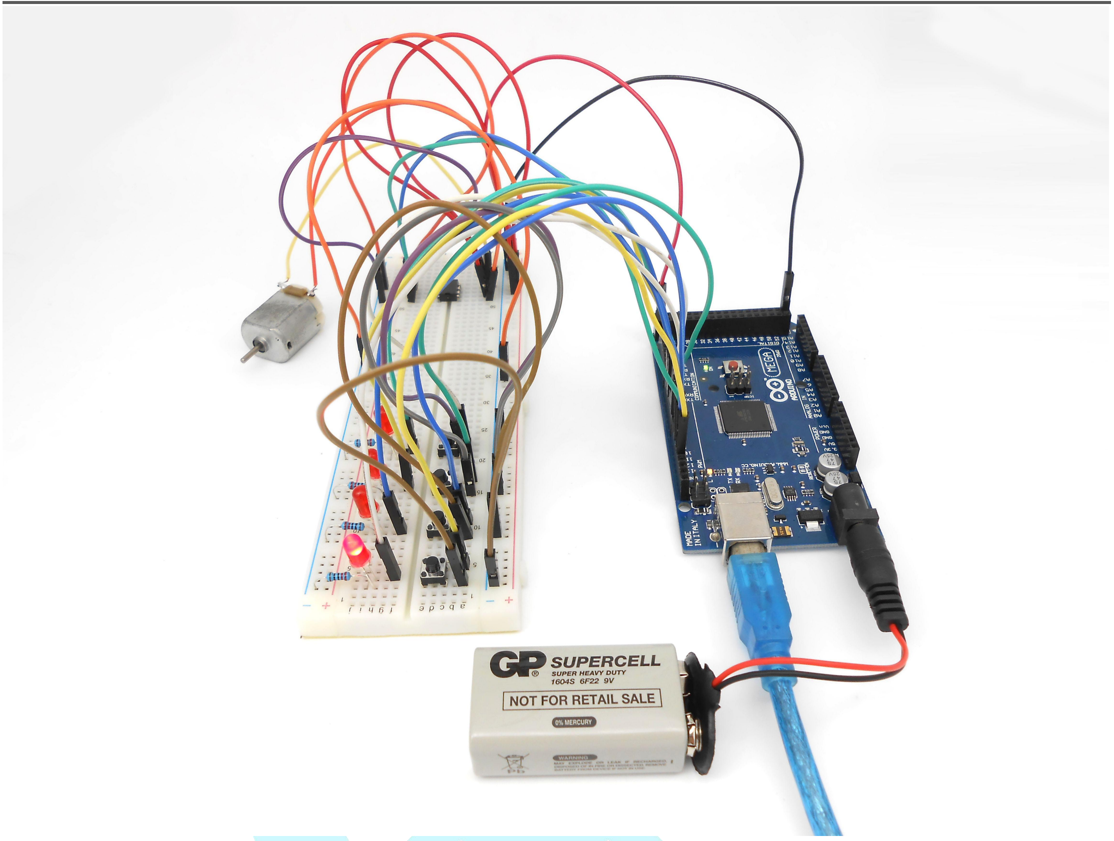


2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Press the btn1 button to stop or run the DC motor; press the btn2 button to forward or reverse the DC motor; Press the btn3 button to accelerate the DC motor; Press the btn4 button to decelerate the DC motor. When one of the four buttons is pressed, their corresponding LED will be flashing which prompts that the current button is clicked.

Adeept



Summary

I think you must have grasped the basic theory and programming of the DC motor after studying this experiment. You not only can forward and reverse it, but also can regulate its speed. Besides, you can do some interesting applications with the combination of this course and your prior knowledge.

Adeept

Lesson 22 Joystick Module

Introduction

The PS2 Joystick Module is an input device. It consists of a station and the control knob onside. It functions by sending angle or direction signals to the device controlled. The button on the module can also be recognized by the microcontroller. The module supports two-channel analog output, namely, x- and y-axis offset, and one-channel digital output which indicates whether the user has pressed the button at z-axis or not. The Joystick Module can be used to easily control the object to move in a three-dimensional space. For example, it can be applied to control crane, truck, electronic games, robots, etc.

Components

- 1 * Arduino MEGA 2560
- 1 * Joystick Module
- 1 * USB Cable
- 1 * 5-Pin Wires
- 1 * LCD1602
- 1 * 10KΩ Potentiometer
- Several Jumper Wires

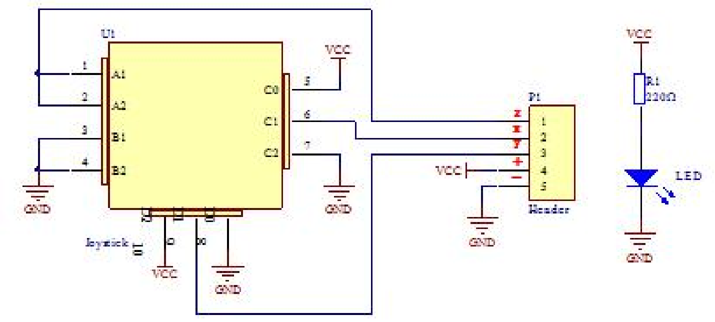
Experimental Principle

The Fritzing image:

Pin definition:

z	Digital key output
x	Analog output(X)
y	Analog output(Y)
+	VCC
-	GND

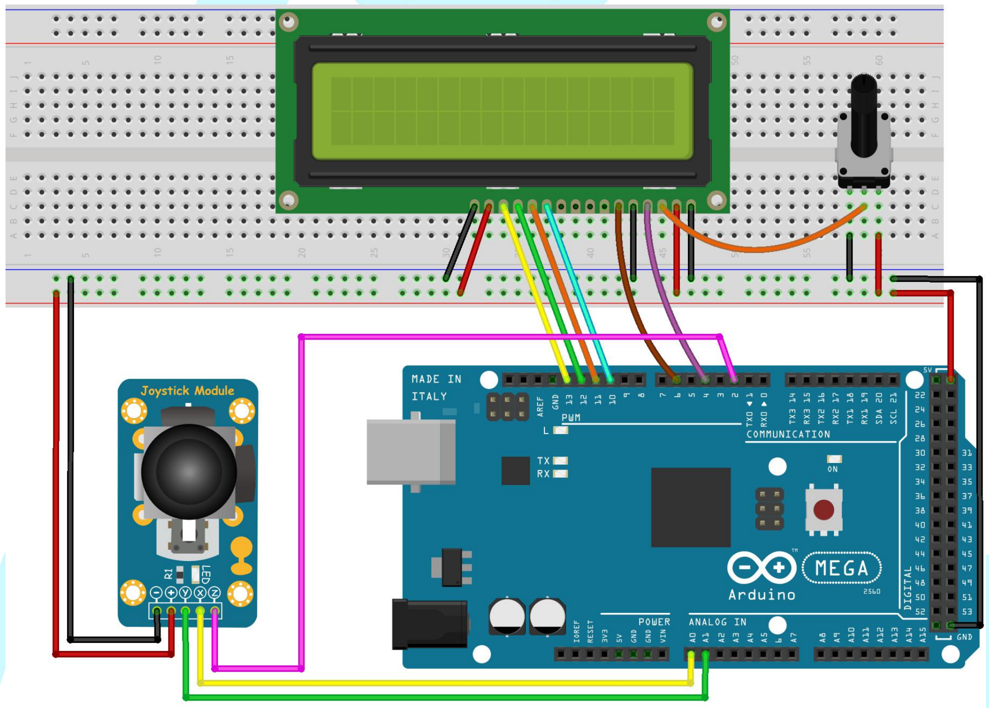
The schematic diagram:



The experiment reads the status of the PS2 Joystick Module, send the data to and display it on Serial Monitor.

Experimental Procedures

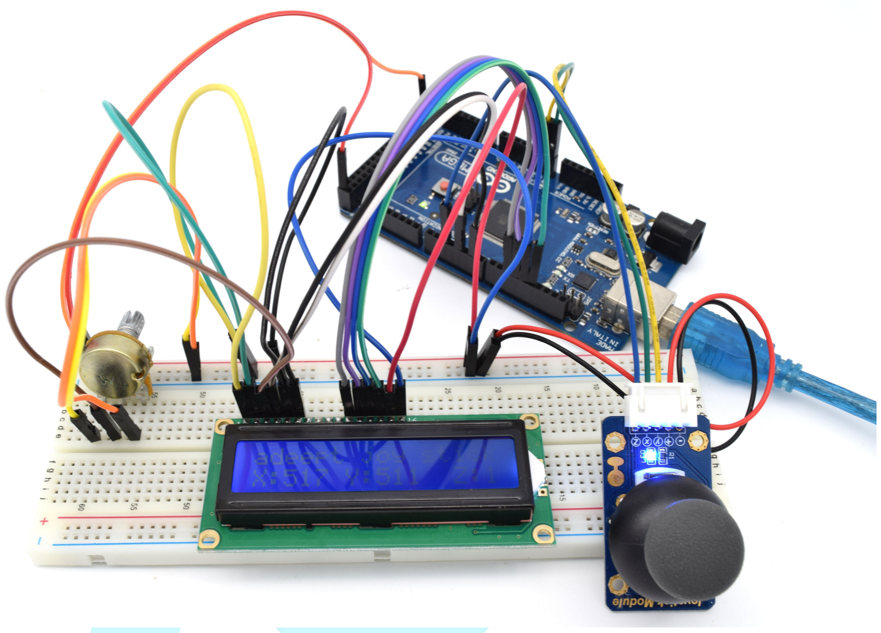
Step 1: Build the circuit



Step 2: Program _22_Joystick.ino

Step 3: Compile and download the sketch to the 2560 board.

Open Serial Monitor of the Arduino IDE. Press or pull the knob and you will see the value of current status displayed on the window.



Adeept

Lesson 23 Slide Potentiometer Module

Introduction

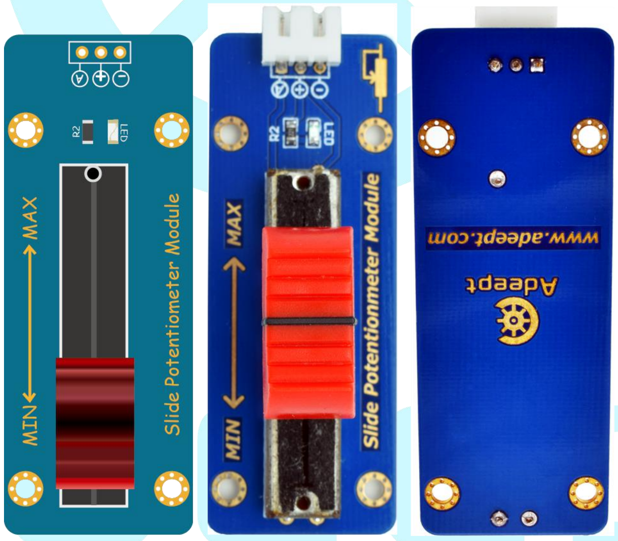
The similarity of the Slide Potentiometer Module and Potentiometer Module lies in: holding three terminals; changing the resistance between the changeable terminal and one end by changing the position of the slider. When the difference is: the Slide Potentiometer usually has a larger power (and size) and can be used directly as a load or connected in serial in the circuit of the load for current limiting. The potentiometer has a smaller power and size, and generally used for voltage sampling in signal circuit.

Components

- 1 * Arduino MEGA 2560
- 1 * Slide Potentiometer Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

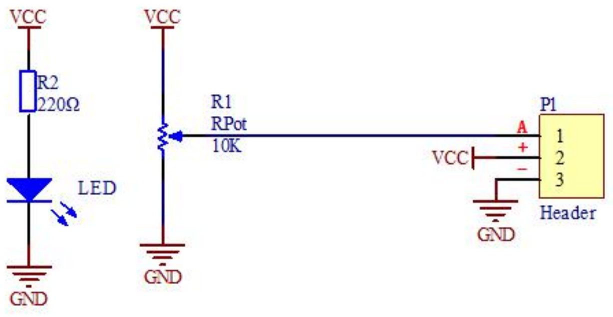
The Fritzing image:



Pin definition:

A	Analog output
+	VCC
-	GND

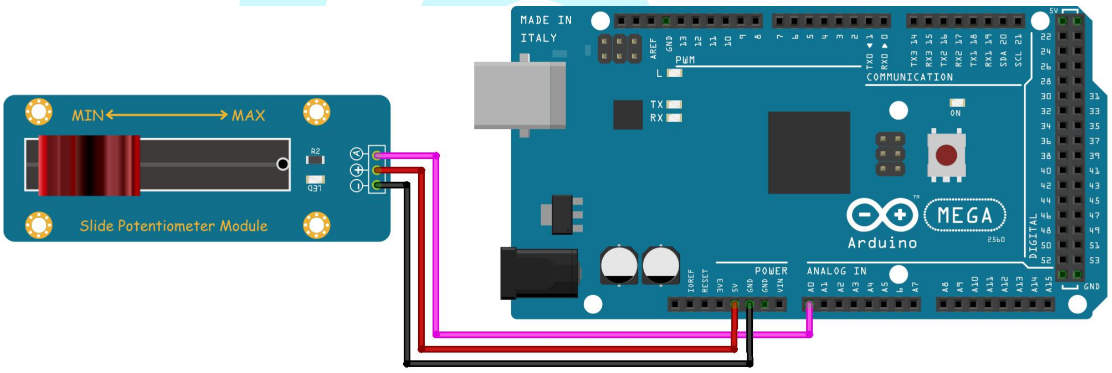
The schematic diagrams:



This experiment programs the Arduino board and collects analog quantities output from the Slide Potentiometer module via pin A0 of the board, and converts them into digital ones and display the value on the computer by serial port.

Experimental Procedures

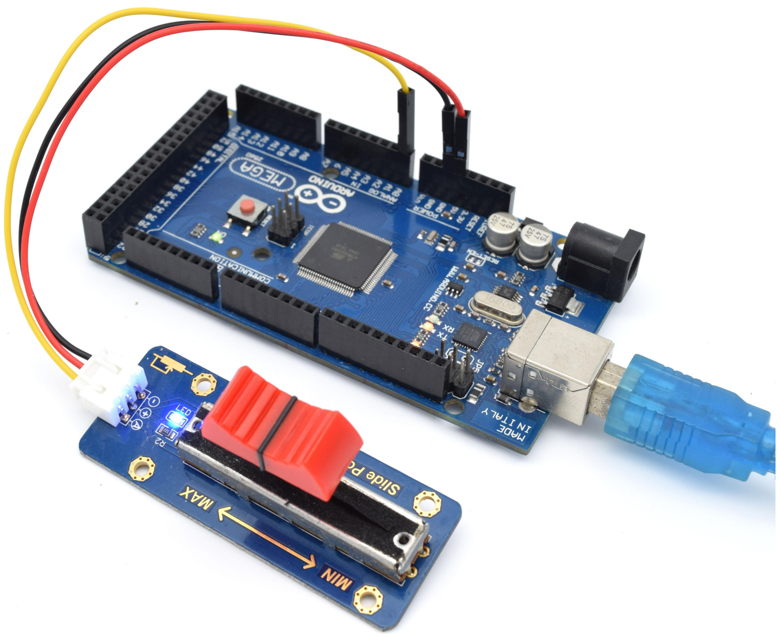
Step 1: Build the circuit

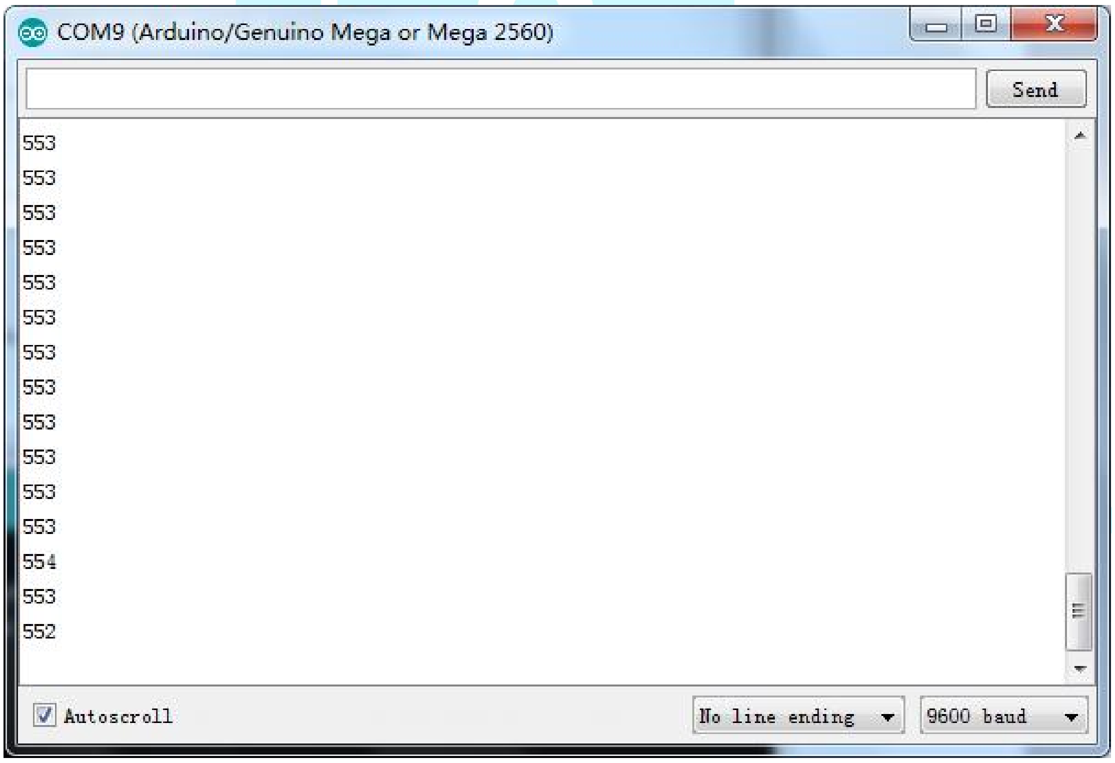


Step 2: Program _23_SlidePotentiometerModule.ino

Step 3: Compile and download the sketch to the 2560 board.

Open the Serial Monitor in Arduino IDE. Move the slide of the Slide Potentiometer module. Then the value output by port A of the module will be displayed on the Serial Monitor. Slide it toward MIN and the value on the window will decrease; slide it toward MAX, it will increase.





```
553
553
553
553
553
553
553
553
553
553
553
553
554
553
552
```

Autoscroll ✓ | No line ending | 9600 baud

Lesson 24 8*8 LED Matrix Module

Introduction

The module drives the 8*8 LED Matrix Module by cascading two 74HC595 chips. The module communicates with the microcontroller through SPI. It only occupies three I/Os of the Arduino board and save precious ones for connecting other devices.

Components

- 1 * Arduino MEGA 2560
- 1 * 8*8 LED Matrix Module
- 1 * USB Cable
- 1 * 5-Pin Wires

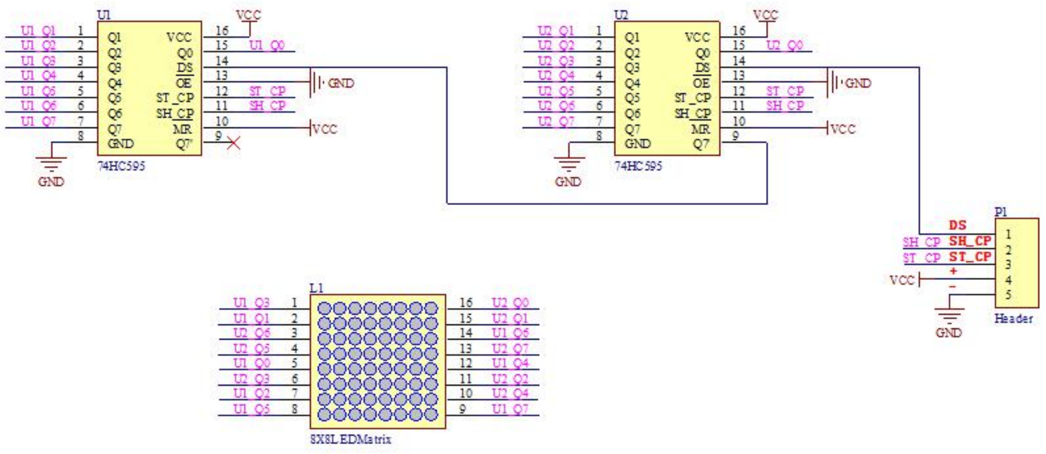
Experimental Principle

The Fritzing image:

Pin definition:

DS	Digital input
SH_CP	Digital input
ST_CP	Digital input
+	VCC
-	GND

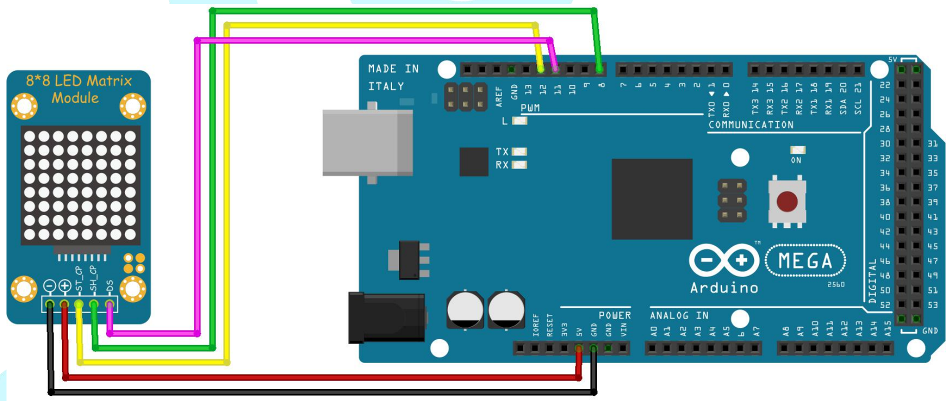
The schematic diagram:



In this experiment, by programming the Arduino board, we send the data to the dot matrix module via the SPI interface and make the display scroll the characters “Adeept”.

Experimental Procedures

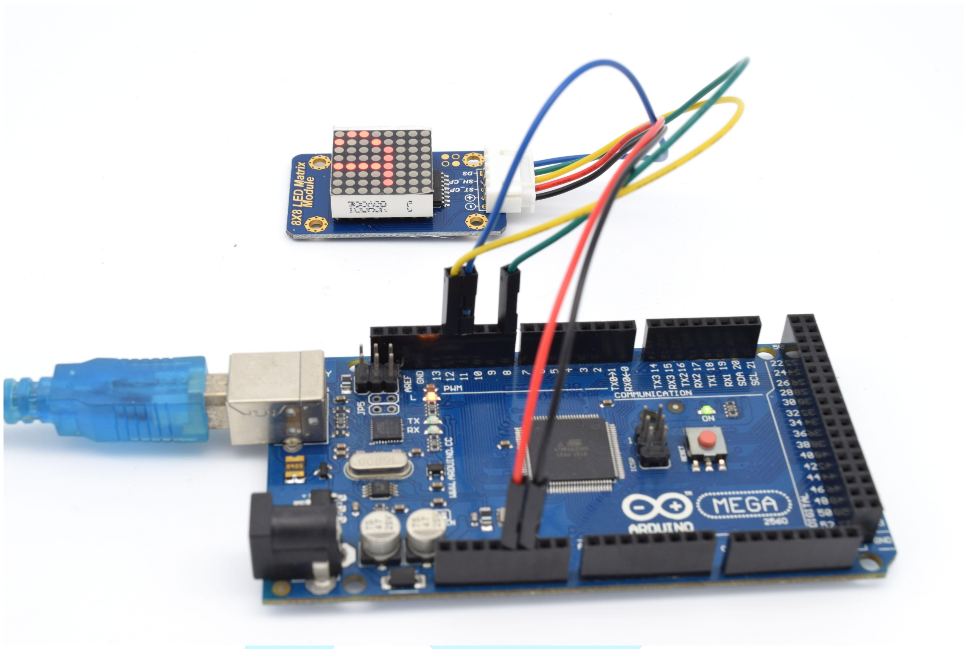
Step 1: Build the circuit



Step 2: Program _24_LEDMatrix.ino

Step 3: Compile and download the sketch to the 2560 board.

Now you can see on the dot matrix module, “Adeept” is displayed in the way of scrolling.



Adeept

Lesson 25 Photoresistor

Overview

In this lesson, we will learn how to measure the light intensity by photoresistor and make the measurement result displayed on the LCD1602.

Requirement

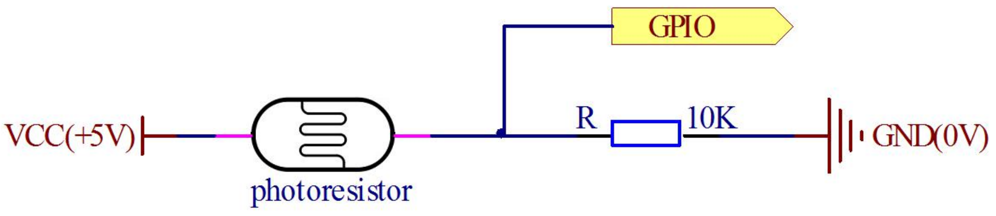
- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* LCD1602
- 1* Photoresistor
- 1* 10KΩ Resistor
- 1* 10KΩ Potentiometer
- 1* Breadboard
- Several Jumper Wires

Principle

A photoresistor is a light-controlled variable resistor. The resistance of a photoresistor decreases with the increasing incident light intensity; in other words, it exhibits photoconductivity. A photoresistor can be applied in light-sensitive detector circuits.

A photoresistor is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as a few megohms (MΩ), while in the light, a photoresistor can have a resistance as low as a few hundred ohms. If incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons (and their hole partners) conduct electricity, thereby lowering resistance. The resistance range and sensitivity of a photoresistor can substantially differ among dissimilar devices. Moreover, unique photoresistors may react substantially differently to photons within certain wavelength bands.

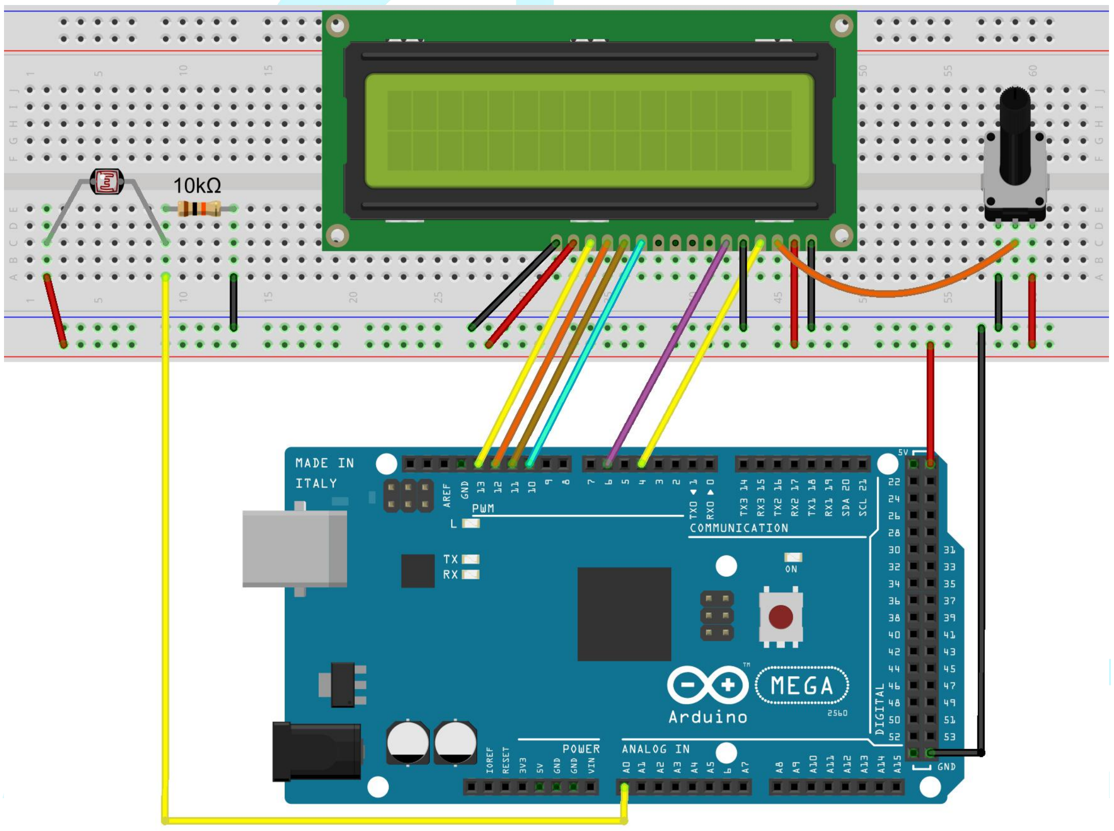
The schematic diagram of this experiment is shown below:



With the increase of the light intensity, the resistance of photoresistor will be decreased. The voltage of GPIO port in the above figure will become high.

Procedures

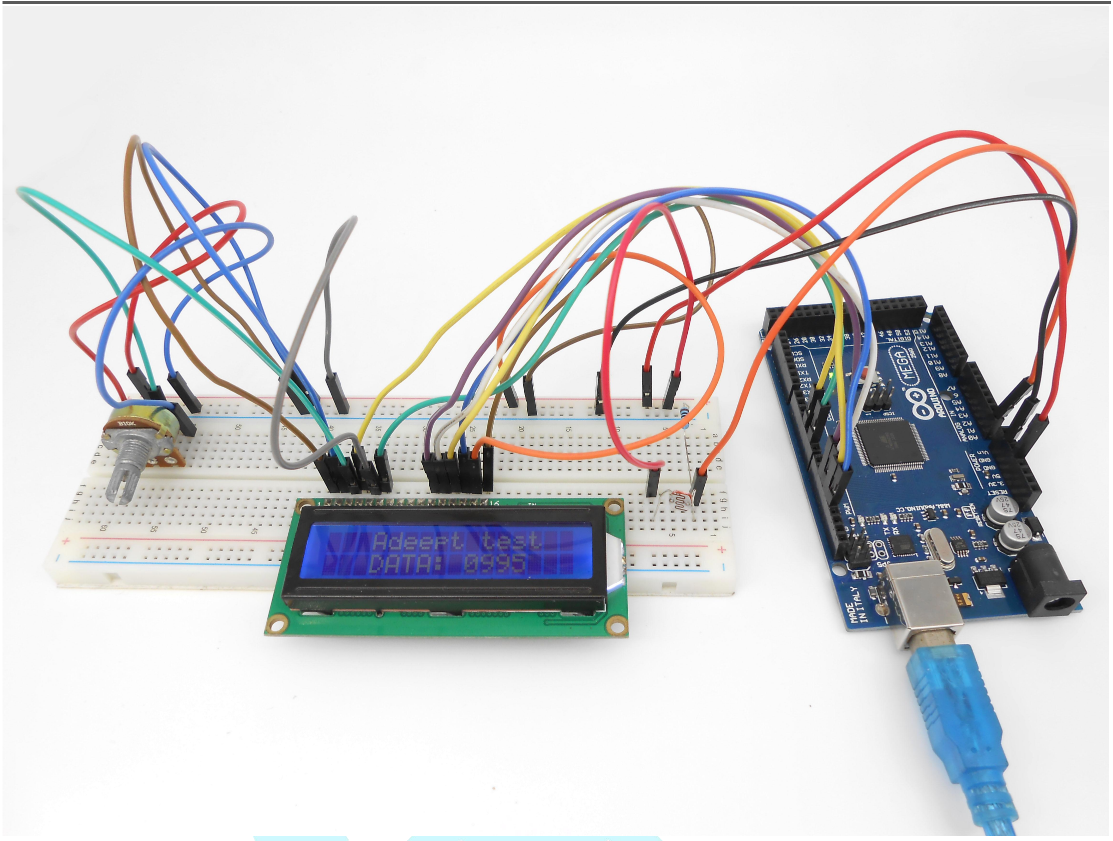
1. Build the circuit



2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, when you try to block the light towards the photoresistor, you will find that the value displayed on the LCD1602 will be reduced. Otherwise, when you use a powerful light to irradiate the photoresistor, the value displayed on the LCD1602 will be increased.



Summary

By learning this lesson, we have learned how to detect surrounding light intensity with the photoresistor. You can play your own wisdom, and make more originality based on this experiment and the former experiment.

Adeept

Lesson 26 Soil Moisture Sensor Module

Introduction

The Soil Moisture Sensor module is a simple sensor that measures the soil moisture. When the soil moisture is insufficient, the output value of the sensor will decrease; on the other hand, the value will increase when there's enough water. The surface of the sensor is gilded to prolong its life.

The CM Module consists of a comparator LM393 and extremely simple external circuits. When using the module, you can set a threshold via the blue potentiometer beforehand. When the input analog value reaches the threshold, the digital pin S will output a Low level.

Components

- 1 * Arduino MEGA 2560
- 1 * Soil Moisture Sensor Module
- 1 * CM Module
- 1 * USB Cable
- 1 * 4-Pin Wires
- 1 * 2-Pin Female to Female Wires

Experimental Principle

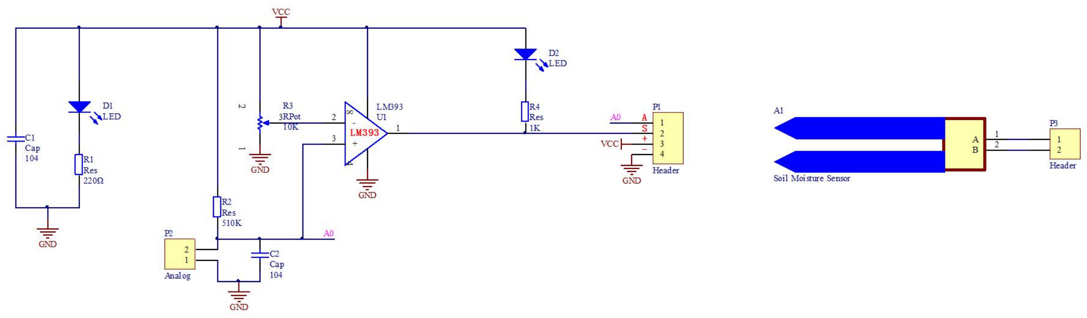
The Fritzing images:

Adeept

Pin definition:

Soil Moisture Sensor Module	
1	Analog output
2	Analog output
CM Module	
1	Analog output
2	Analog output
S	Digital output
A	Analog output
+	VCC
-	GND

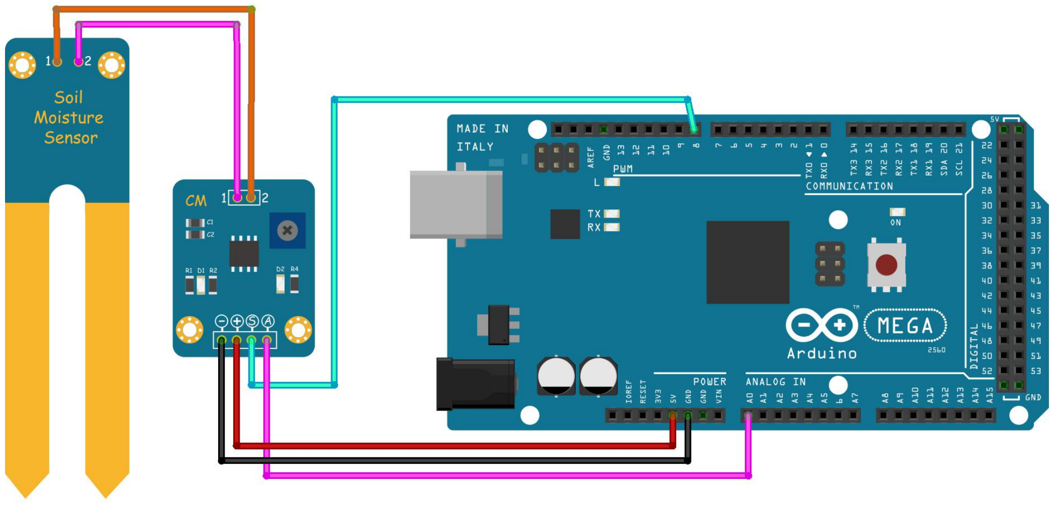
The schematic diagram:



The experiment uses the Soil Moisture Sensor module to collect data of soil moisture and display it on Serial Monitor.

Experimental Procedures

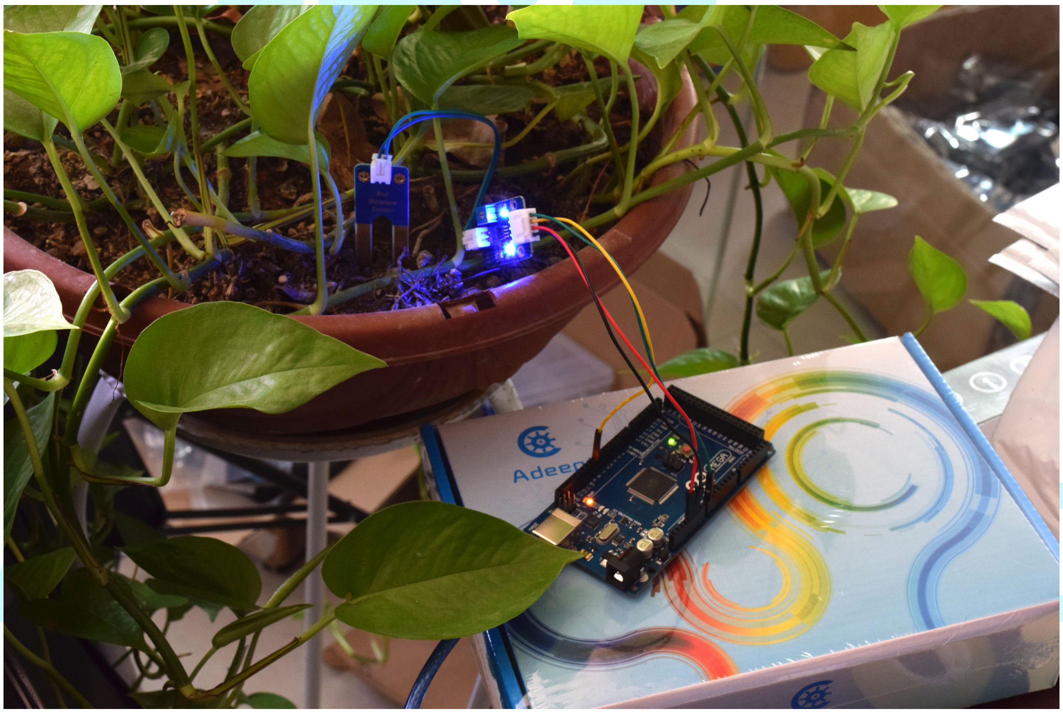
Step 1: Build the circuit

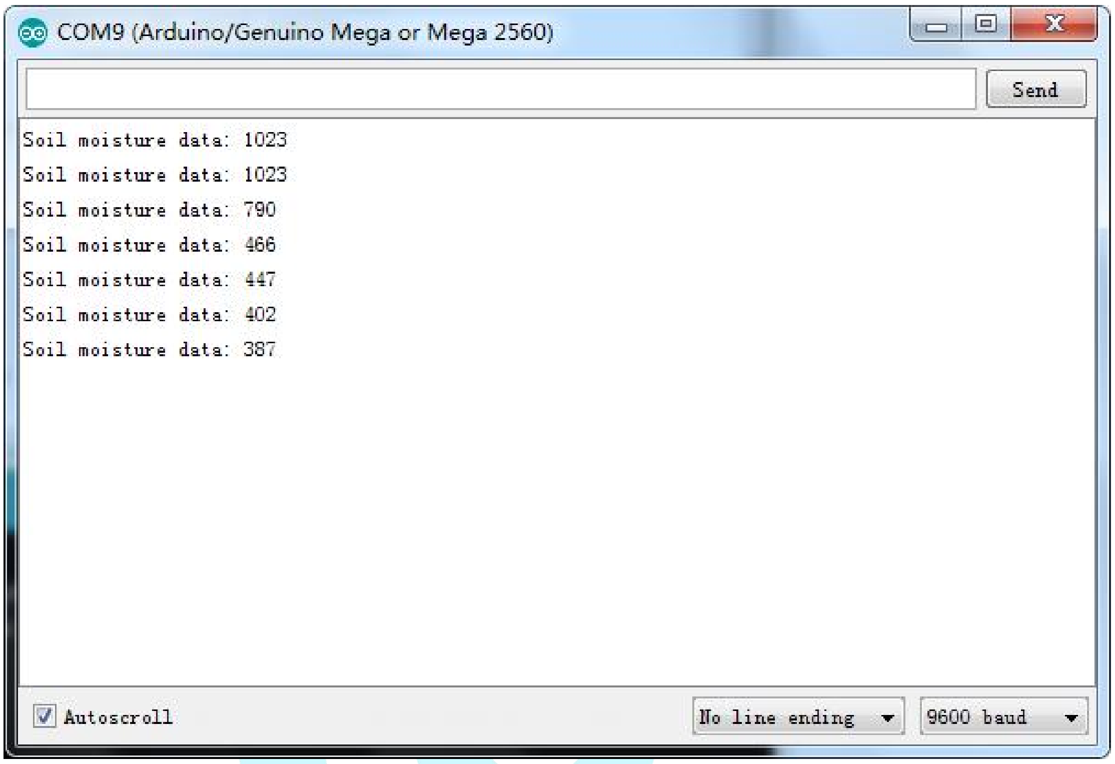


Step 2: Program _26_SoilMoistureModule.ino

Step 3: Compile and download the sketch to the 2560 board.

Open Serial Monitor of the Arduino IDE. You will see the value of soil moisture collected by the module displayed on the window.





```
Soil moisture data: 1023
Soil moisture data: 1023
Soil moisture data: 790
Soil moisture data: 466
Soil moisture data: 447
Soil moisture data: 402
Soil moisture data: 387
```

Adeept

Lesson 27 Rotary Encoder

Introduction

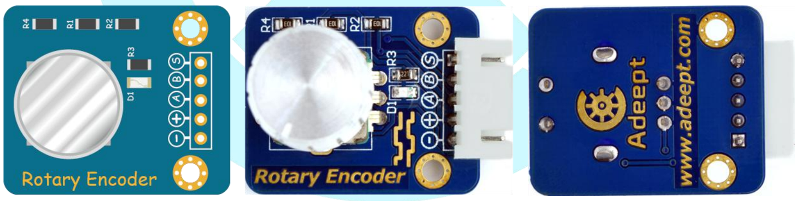
Rotary encoder switch, or small rotary encoder, is a switch electronic component that has a set of regular and strictly-sequenced pulses. The module supports functions such as increase, decrease, turn pages, etc., by collaboration with a microcontroller. For example, in daily life you can see page turning of the mouse, menu selection, volume adjustment of speakers, temperature adjustment of toaster, frequency adjustment of medical equipment, etc.

Components

- 1 * Arduino MEGA 2560
- 1 * Rotary Encoder Module
- 1 * USB Cable
- 1 * 5-Pin Wires

Experimental Principle

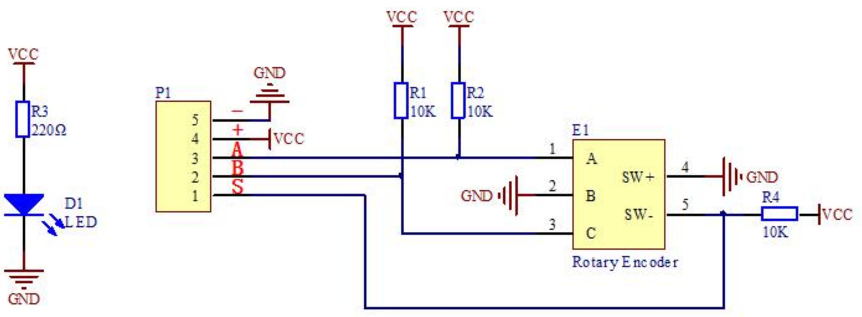
The Fritzing image:



Pin definition:

S	Digital output
B	Digital output
A	Digital output
+	VCC
-	GND

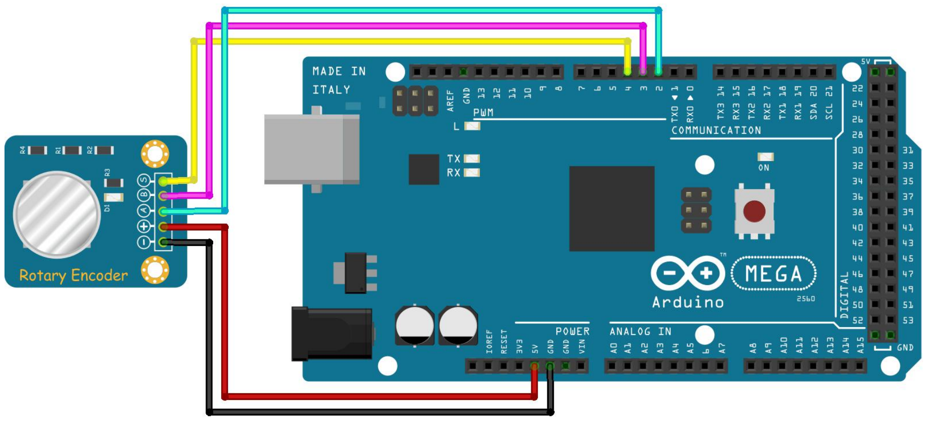
The schematic diagram:



In this experiment, by programming the Arduino, we change a value by reading the status of the Rotary Encoder. When we turn the knob of the Rotary Encoder clockwise, the value on the window will increase; when we turn the knob counterclockwise, the value will decrease. When we press down the switch, the value will be zeroed out.

Experimental Procedures

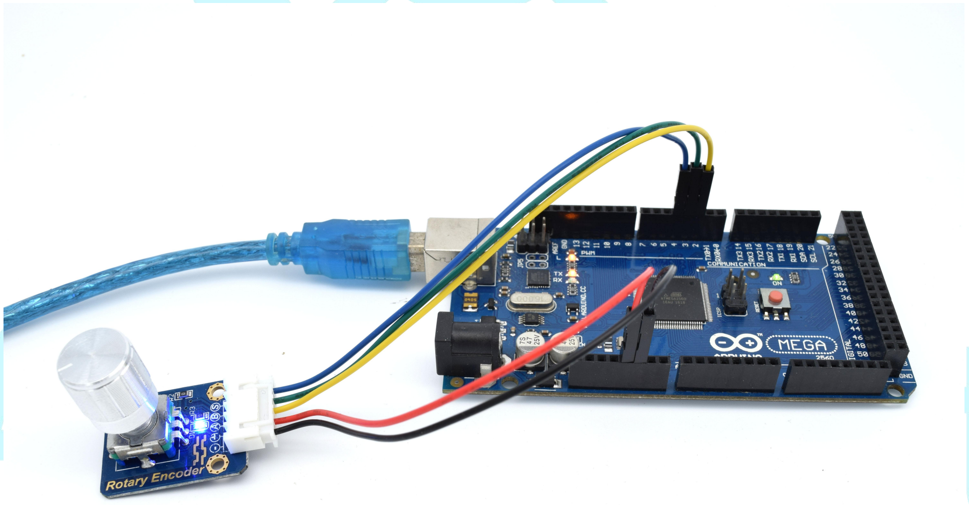
Step 1: Build the circuit

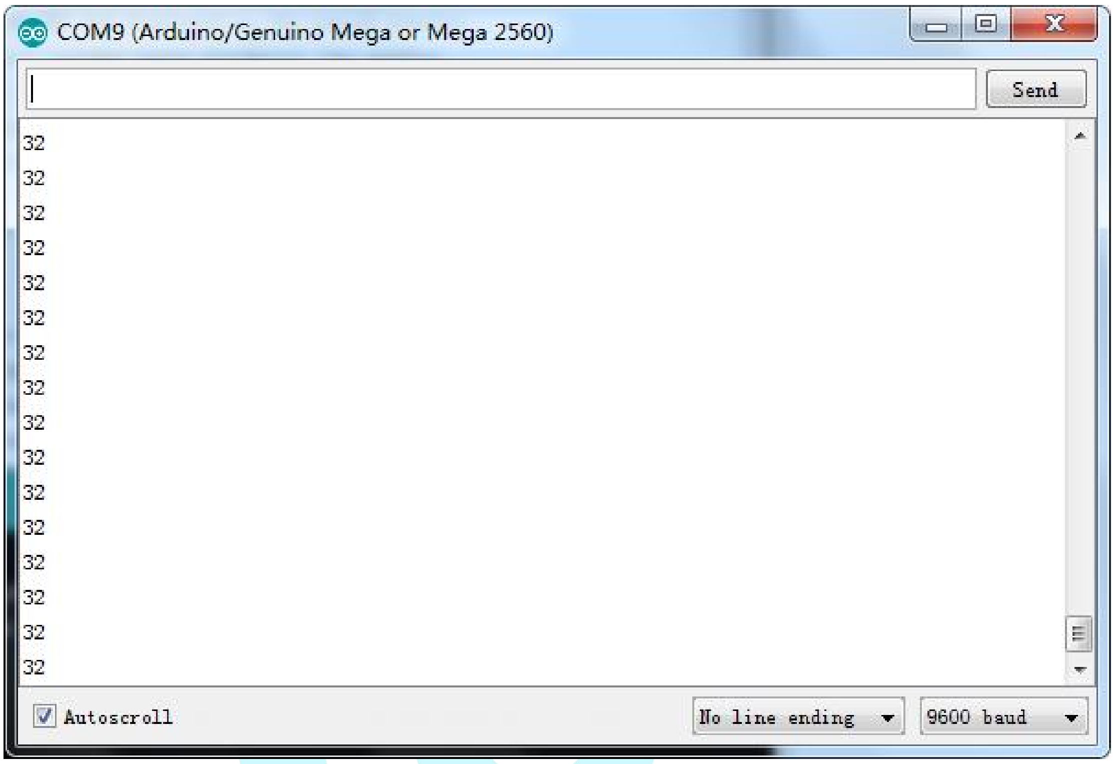


Step 2: Program _27_RotaryEncoderModule.ino

Step 3: Compile and download the sketch to the 2560 board.

Open the Serial Monitor in Arduino IDE. Turn or press down the knob of the Rotary Encoder, the value on the window will increase, decrease, or be cleared.





```
32
32
32
32
32
32
32
32
32
32
32
32
32
32
32
32
```

Adeept

Lesson 28 Control a relay with IR remoter controller

Overview

In this experiment, we will program the Arduino MEGA 2560 to control a relay by remote controller.

Requirement

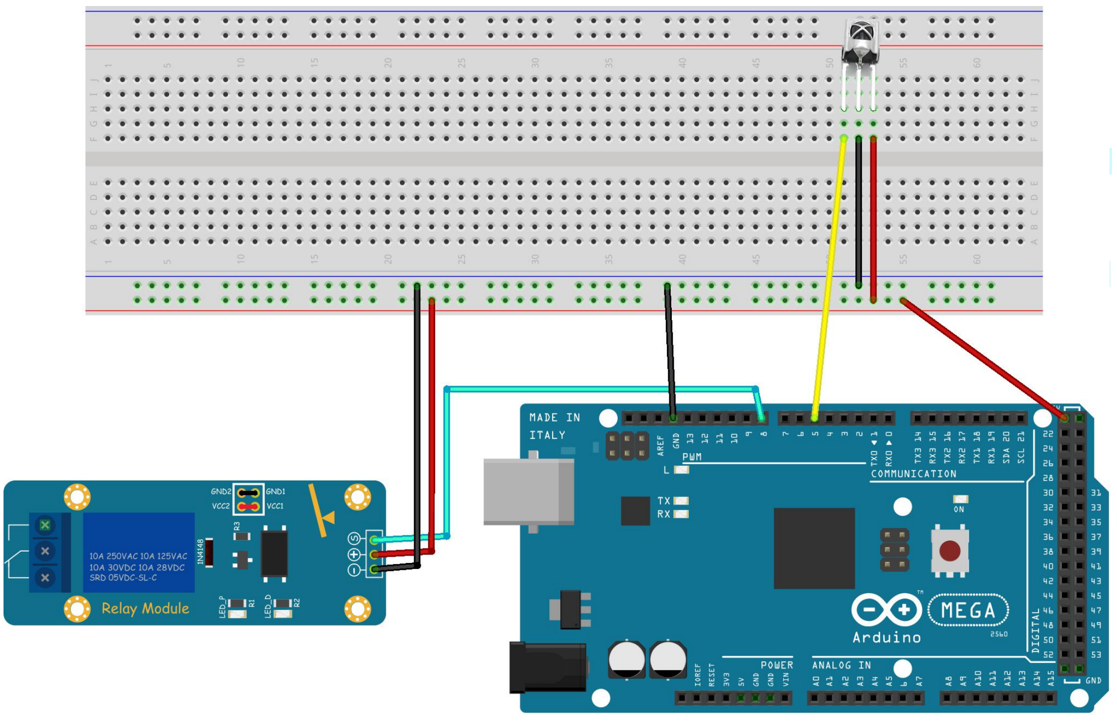
- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* IR Receiver HX1838
- 1* Remote Controller
- 1* Relay Module
- 1* Breadboard
- Several Jumper Wires

Principle

The remote IR receiver connected to the Arduino MEGA 2560 is used to receive IR signal from remote controller. If you press the different keys(key 0 or key 1) on the remote controller, the relay state will be toggled.

Procedures

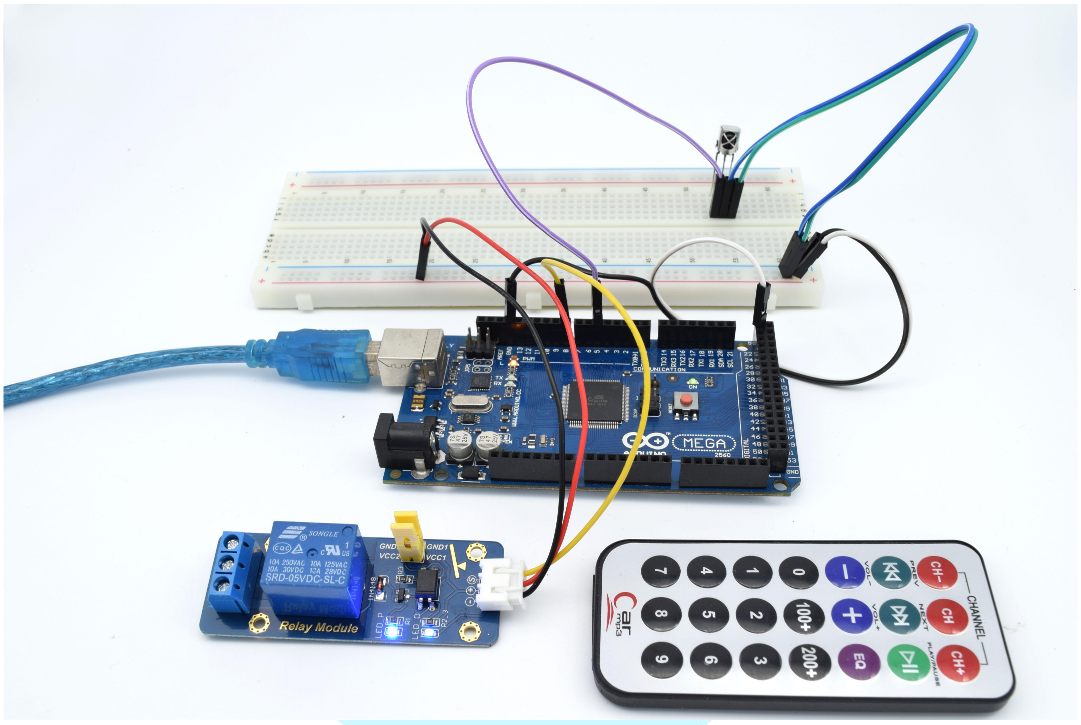
1. Build the circuit



2. Program

3. Compile the program and upload to Arduino MEGA 2560 board

Now, when you press the button ‘0’ on the remote controller, the LED is off. When you press the button ‘1’ on the remote controller, the LED is on. At the same time, you will hear the sound of relay toggling.



Adeept

Lesson 29 Control a RGB LED with IR remoter controller

Overview

In this lesson, we will use the remote IR receiver and the RGB to do an experiment that the infrared remote controls the RGB.

Requirement

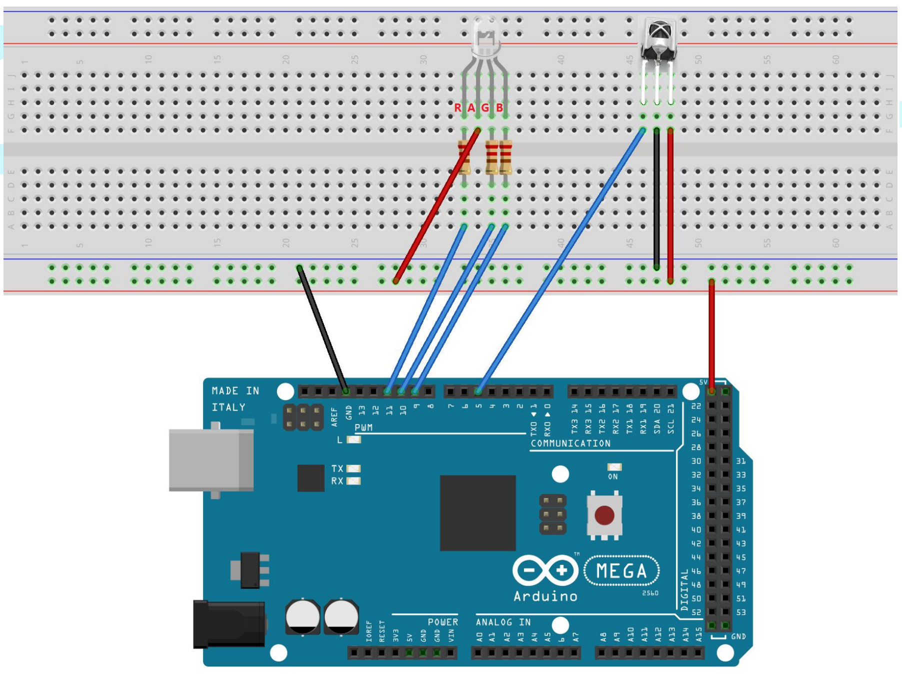
- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* IR Receiver HX1838
- 1* Remote Controller
- 1* RGB LED
- 3* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

Principle

If you press the different key with different number on the remote controller, you will find the color of RGB LED will be changed. When you press the key with number 0, the RGB LED will be off.

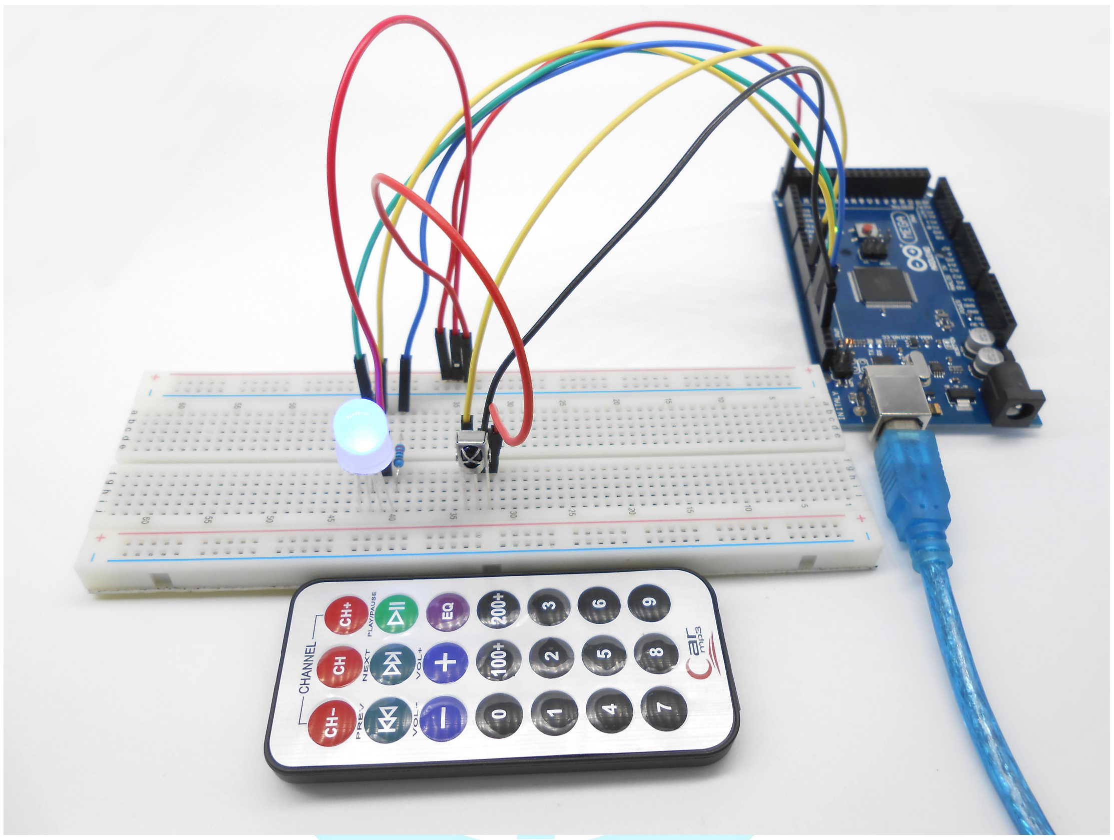
Procedures

1. Build the circuit



2. Program

3. Compile the program and upload to Arduino MEGA 2560 board



Adeept

Lesson 30 RFID module

Overview

In this lesson, we will learn how to use RFID Module. We programmed the Arduino 2560 to read the data which is acquired by the RFID module, and then make the ID data displayed on the serial monitor.

Requirement

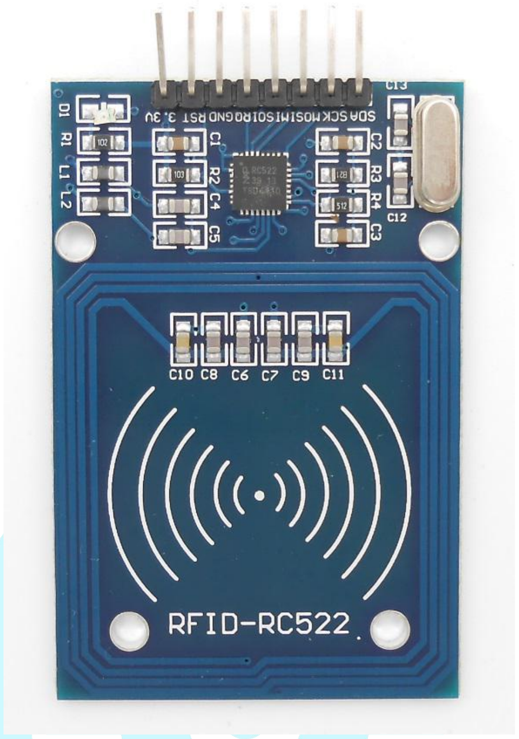
- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* RFID-RC522 Module
- 1* RFID ID Round Tag
- 1* RFID ID Card
- 1* Battery holder
- Several Jumper Wires

Principle

RFID technology is used for a wide variety of applications including access control, package identification, warehouse stock control, point-of-sale scanning, retail antitheft systems, toll-road passes, surgical instrument inventory, and even for identifying individual sheets of paper placed on a desk. RFID tags are embedded in name badges, shipping labels, library books, product tags and boxes; installed in aircraft; hidden inside car keys; and implanted under the skin of animals or even people. RFID systems work on a wide range of frequencies, have a variety of modulation and encoding schemes, and vary from low-power passive devices with range of only a few millimeters to active systems that work for hundreds of kilometers.

However, all RFID systems have the same basic two-part architecture: a reader and a transponder. The reader is an active device that sends out a signal and listens for responses, and the transponder (the part generally called the "tag") detects the signal from a reader and automatically sends back a response containing its identity code.

A reader is shown in the following:



A transponder is shown in the following:

Different types of RFID tags fall into one of three broad categories: active, passive, and battery-assisted passive.

Active tags are physically large because they require their own power supply such as a battery. They can also have a very long range because the availability of local power allows them to send high-powered responses that can travel from tens of meters to hundreds of kilometers. An active tag is essentially a combination of a radio receiver to detect the challenge, some logic to formulate a response, and a radio transmitter to send back the response. They can even have the challenge and response signals operate on totally different frequencies. The downsides are the size of the tag, a high manufacturing cost due to the number of parts required, and the reliance on a battery that will go flat eventually.

Passive tags can be much smaller and cheaper than active tags because they don't require a local power supply and have much simpler circuitry. Instead of supplying their own power, they leach all the power they need from the signal sent by the reader. Early passive tags operated on the "Wiegand effect," which uses a specially formed wire to convert received electromagnetic energy into radio-wave pulses. Some early passive RFID tags actually consisted of nothing more than a number of very carefully formed wires made from a combination of cobalt, iron, and vanadium, with no other parts at all.

Modern passive tags use a clever technique that uses current induced in their antenna coil to power the electronics required to generate the response. The response is then sent by modulating the reader's own field, and the reader detects the modulation as a tiny fluctuation in the voltage across the transmitter coil. The result is that passive tags can be incredibly small and extremely inexpensive: the antenna can be a simple piece of metal foil, and the microchips are produced in such large quantities that a complete RFID-enabled product label could cost only a few cents and be no thicker than a normal paper label. Passive tags can theoretically last indefinitely because they don't contain a battery to go flat, but their disadvantage is a very short operational range due to the requirement to leach power from the reader's signal, and lack of an actively powered transmitter to send back the response.

Passive tags typically operate over a range of a few millimeters up to a few meters.

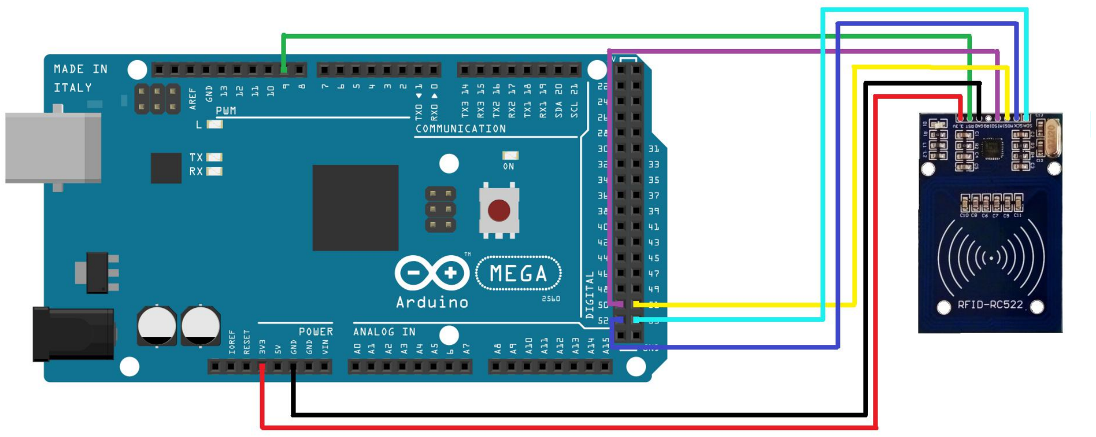
Tags can also have a variety of different modulation schemes, including AM, PSK, and ASK, and different encoding systems. With so many incompatible variations, it's sometimes hard to know if specific tags and readers are compatible. Generally speaking, each type of tag will only function on one specific frequency, modulation scheme, and communications protocol. Readers, on the other hand, are far more flexible and will often support a range of modulation schemes and comms protocols, but are usually still limited to just one frequency due to the tuning requirements of the coil.

Apart from the specific requirements for communicating with them, tags can also have a number of different features. The most common passive tags simply contain a hard-coded unique serial number and when interrogated by a reader they automatically respond with their ID code. Most tags are read-only so you can't change the value they return, but some types of tags are read/write and contain a tiny amount of rewritable storage so you can insert data into them using a reader and retrieve it later. However, most uses of RFID don't rely on any storage within the tag, and merely use the ID code of the tag as a reference number to look up information about it in an external database or other system.

RFID tags are produced in a wide variety of physical form factors to suit different deployment requirements. The most commonly seen form factor is a flat plastic card the same size as a credit card, often used as an access control pass to gain access to office buildings or other secure areas. The most common form by sheer number produced, even though you might not notice them, is RFID-enabled stickers that are commonly placed on boxes, packages, and products. Key fob tags are also quite common, designed to be attached to a keyring so they're always handy for operating access control systems.

Procedures

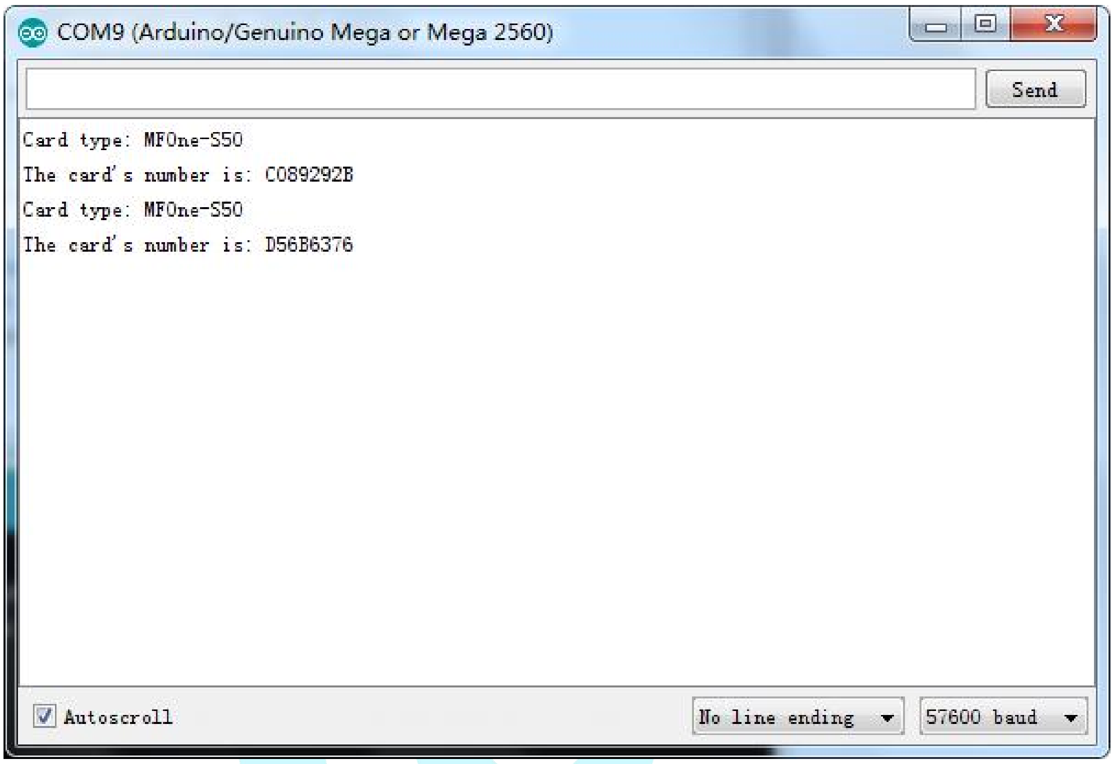
1. Build the circuit



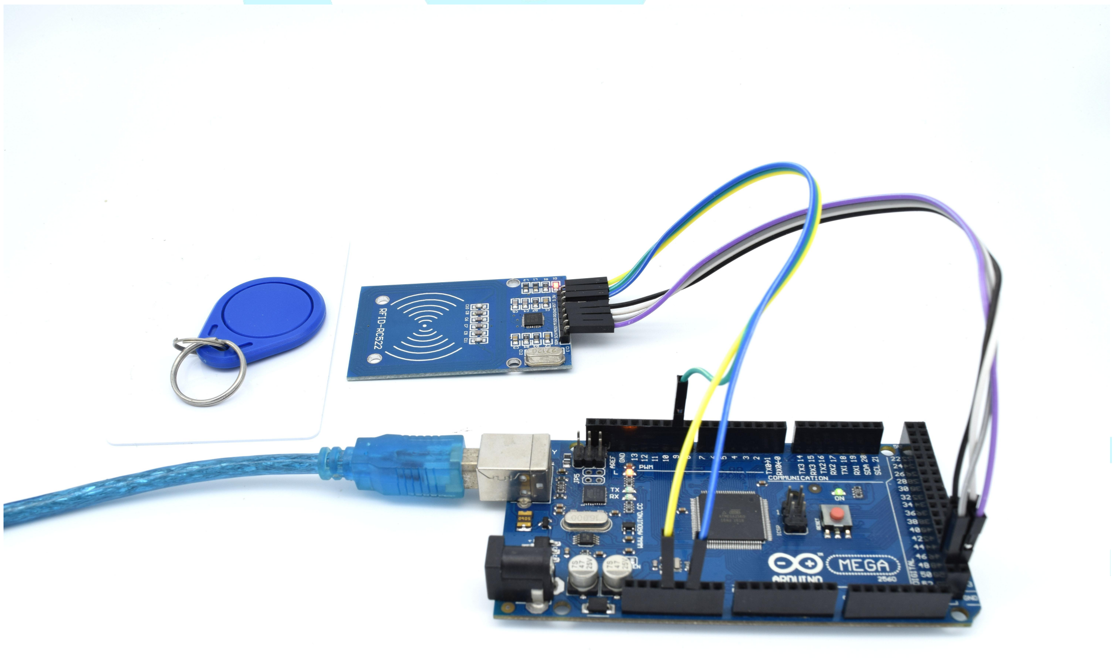
2. Program

3. Compile the program and upload to Adeept 2560 board

Now, when you close the ID card to the RFID reader and the ID number will be sent to the serial monitor.



```
Card type: MFOne-S50
The card's number is: C089292B
Card type: MFOne-S50
The card's number is: D56B6376
```



Lesson 31 Control relay module with RFID module

Overview

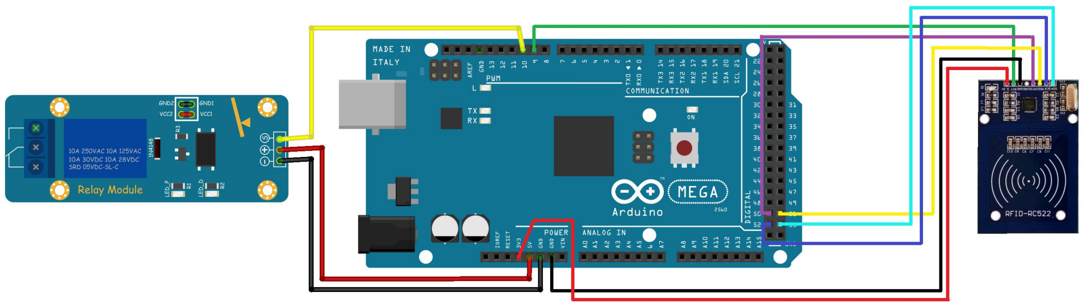
In this lesson, we will learn how to use RFID Module. We programmed the Arduino 2560 to read the data which is acquired by the RFID module, and then make the ID data displayed on the serial monitor. We can control the relay module with RFID module.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* RFID-RC522 Module
- 1* RFID ID Round Tag
- 1* RFID ID Card
- 1* Relay Module
- Several Jumper Wires

Procedures

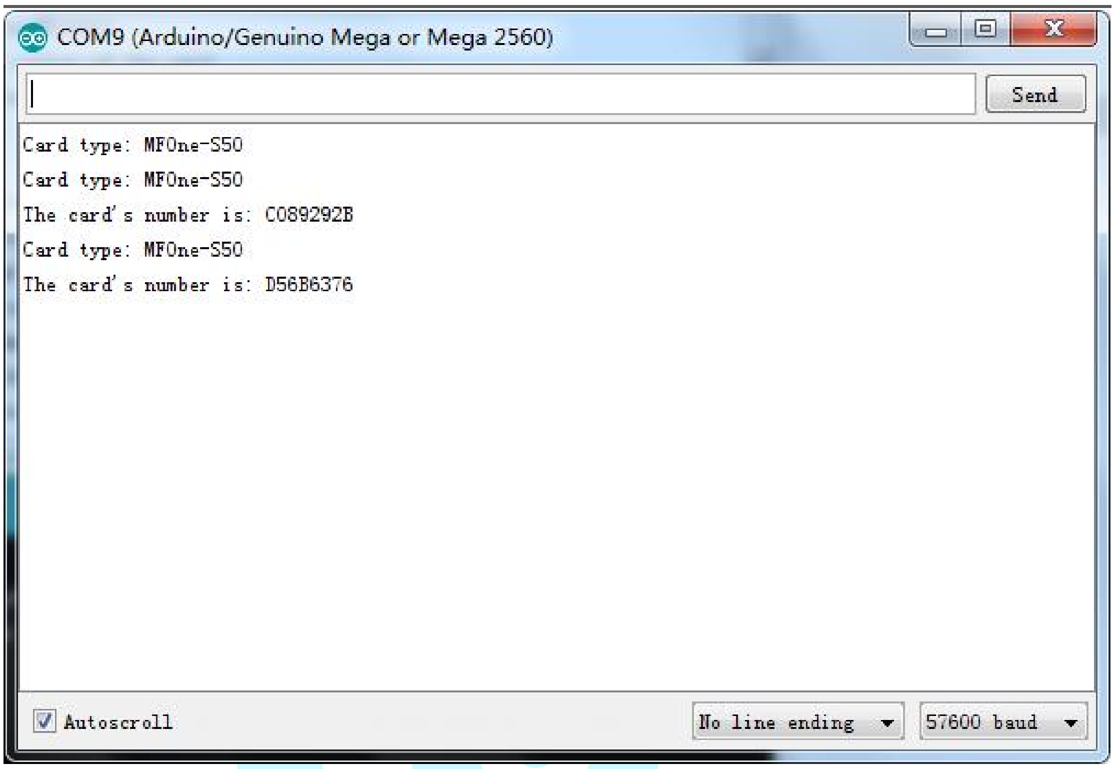
1. Build the circuit



2. Program

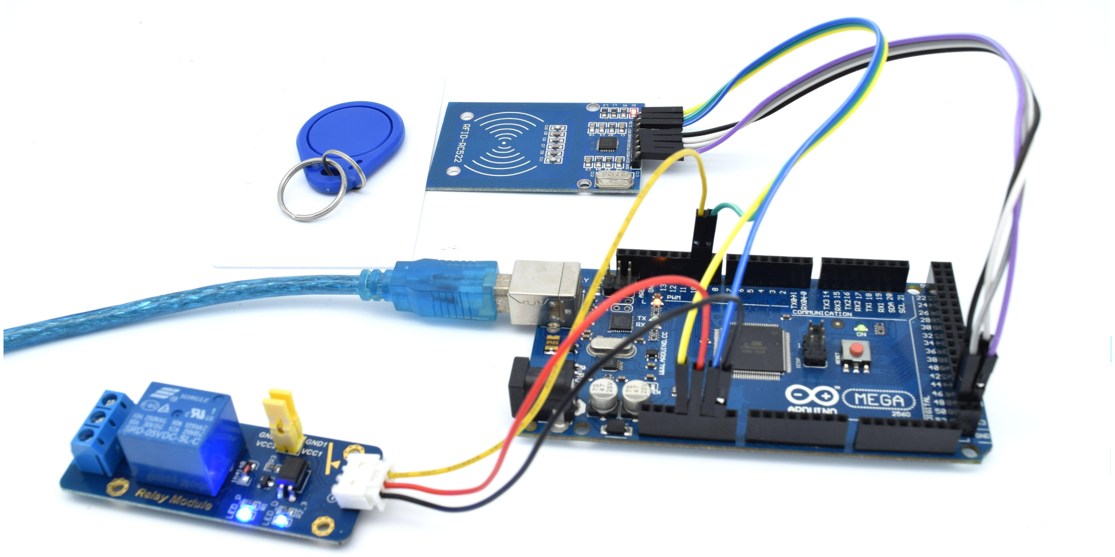
3. Compile the program and upload to Adeept 2560 board

Now, when you close the ID card to the RFID reader, the ID number will be sent to the serial monitor, and it will contral the relay module with RFID module.



```
Card type: MFOne-S50
Card type: MFOne-S50
The card's number is: C089292B
Card type: MFOne-S50
The card's number is: D56B6376
```

Autoscroll [✓] — No line ending — 57600 baud



Lesson 32 RFID Identification System

Overview

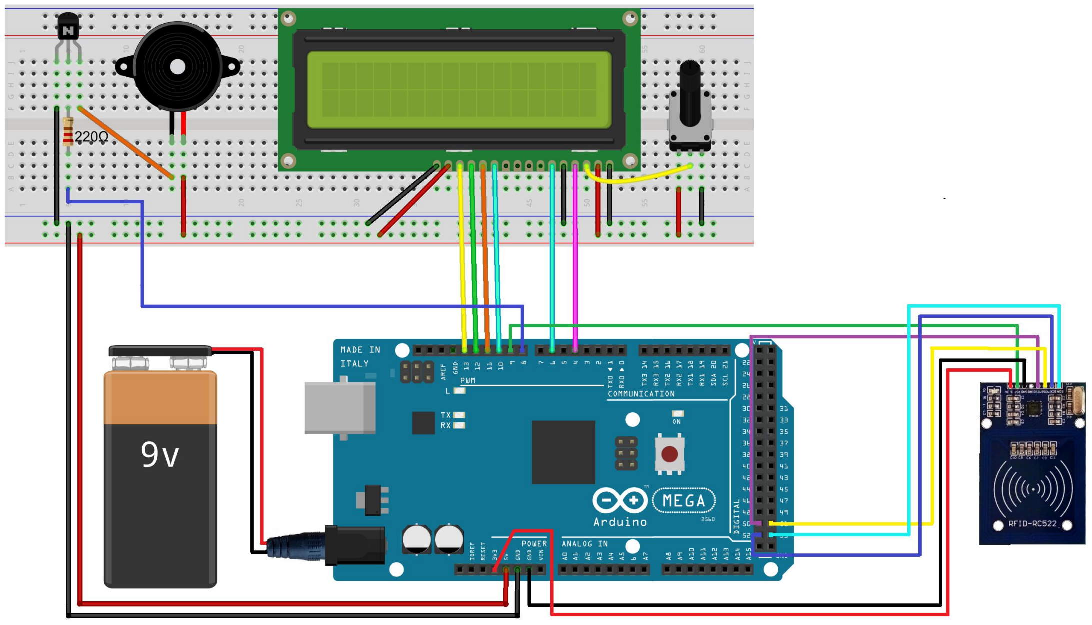
In this lesson, we will learn how to use RFID Module. We programmed the Arduino 2560 to read the data which is acquired by the RFID module, and then make the ID data displayed on the LCD1602 and the serial monitor.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* RFID-RC522 Module
- 1* RFID ID Round Tag
- 1* RFID ID Card
- 1* LCD1602
- 1* 10KΩ Potentiometer
- 1* NPN Transistor (8050)
- 1* 1KΩ Resistor
- 1* Passive buzzer
- 1* Battery holder
- 1* Breadboard
- Several Jumper Wires

Procedures

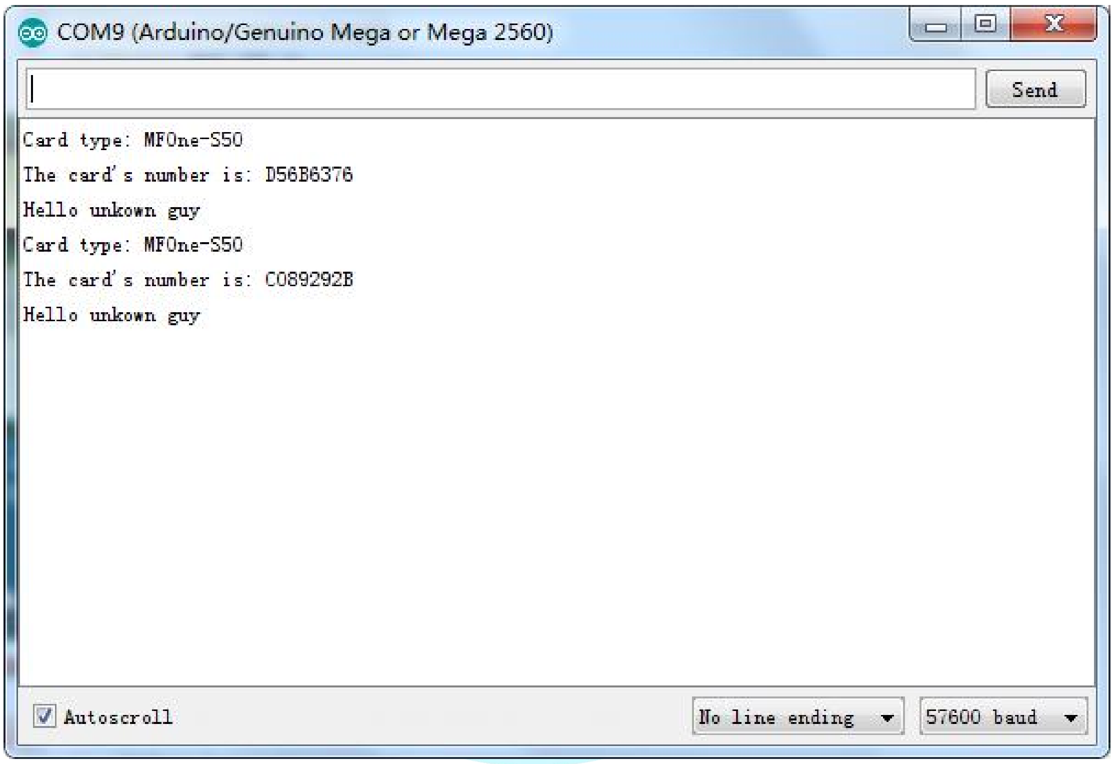
1. Build the circuit



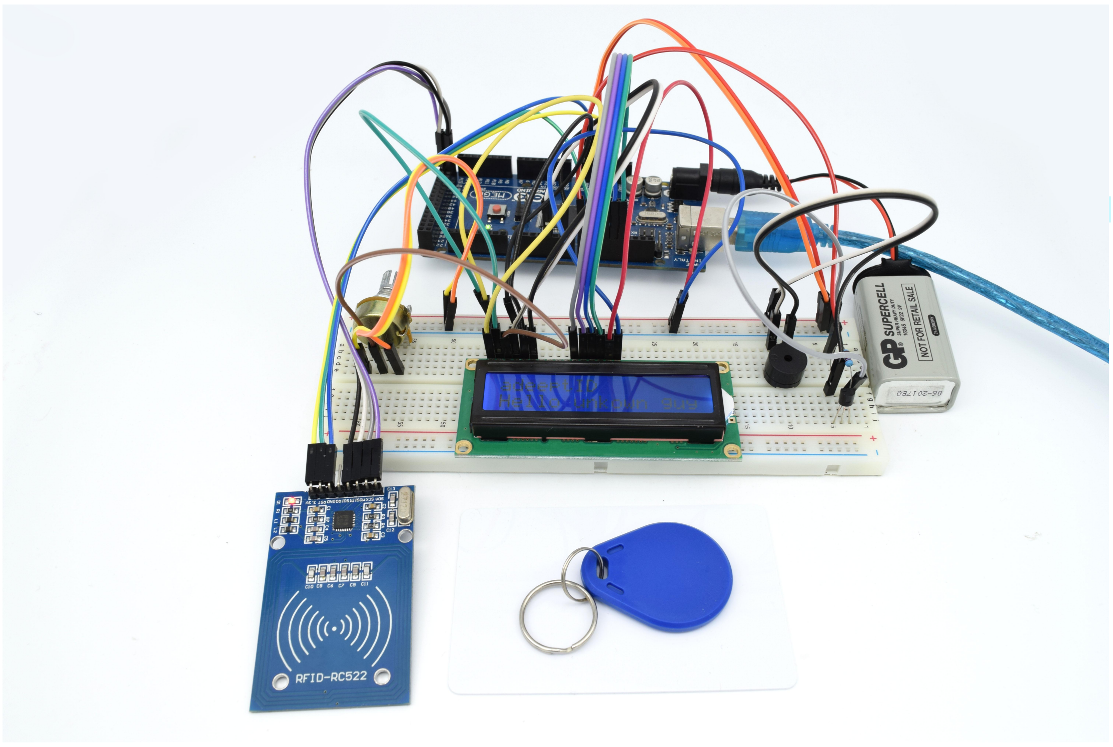
2. Program

3. Compile the program and upload to Adeept 2560 board

Now, when you close the ID card to the RFID reader, the buzzer will sound, and the ID number will be sent to the serial monitor, and it will be also displayed on the LCD1602.



```
Card type: MFOne-S50
The card's number is: D56B6376
Hello unkown guy
Card type: MFOne-S50
The card's number is: C089292B
Hello unkown guy
```



Lesson 33 Move a cat

Overview

This is a simple interaction experiment for Arduino and Processing. We collect the distance data by programming the Arduino 2560, and send the data to the Processing via serial port, and then make a cat move according to the distance data.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* Ultrasonic Distance Sensor
- 1* Breadboard
- Several Jumper Wires

Principle

The experiment is divided into two parts. The first is used to acquire the data from ultrasonic module, another is used to process the data.

The distance data will be displayed on the screen with the form of visualization. When the distance decreases, the cat close to the robot. On the contrary, the cat move away from the robot

Note:

1. In this experiment, my Arduino 2560 board is connected to my computer COM26, please adjust according to actual situation.

2. If the Processing has not running normally, you need to install the related function libraries.

Arduino key function:

●write()

Writes binary data to the serial port. This data is sent as a byte or series of bytes; to send the characters representing the digits of a number use the print() function instead.

Syntax

Serial.write(val) Serial.write(str) Serial.write(buf, len)

Parameters

val: a value to send as a single byte

str: a string to send as a series of bytes

buf: an array to send as a series of bytes

len: the length of the buffer

Returns

byte

write() will return the number of bytes written, though reading that number is optional

Processing key function:

●Name: size()

Description

Defines the dimension of the display window in units of pixels. The size() function must be the first line of code, or the first code inside setup(). Any code that appears before the size() command may run more than once, which can lead to confusing results.

The system variables width and height are set by the parameters passed to this function. If size() is not used, the window will be given a default size of 100x100 pixels.

Syntax

size(w, h)

size(w, h, renderer)

Parameters

w int: width of the display window in units of pixels

h int: height of the display window in units of pixels

renderer

String: Either P2D, P3D, or PDF

Returns

void

●Name: background()

Description

The background() function sets the color used for the background of the Processing window. The default background is light gray. This function is typically used within draw() to clear the display window at the beginning of each frame, but it can be used inside setup() to set the background on the first frame of animation or if the backgound need only be set once.

An image can also be used as the background for a sketch, although the image's width and height must match that of the sketch window. Images used with background() will ignore the current tint() setting. To resize an image to the size of the sketch window, use image.resize(width, height).

It is not possible to use the transparency alpha parameter with background colors on the main drawing surface. It can only be used along with a PGraphics object and createGraphics().

Syntax

background(rgb)

background(rgb, alpha)

background(gray)

background(gray, alpha)

background(v1, v2, v3)

background(v1, v2, v3, alpha)

background(image)

Parameters

rgb	int: any value of the color datatype
alpha	float: opacity of the background
gray	float: specifies a value between white and black
v1	float: red or hue value (depending on the current color mode)
v2	float: green or saturation value (depending on the current color mode)

v3 float: blue or brightness value (depending on the current color mode)

image PImage: PImage to set as background (must be same size as the sketch window)

Returns

Void

●Name: loadImage()

Description

Loads an image into a variable of type PImage. Four types of images (.gif, .jpg, .tga, .png) images may be loaded. To load correctly, images must be located in the data directory of the current sketch.

Syntax

loadImage(filename)

loadImage(filename, extension)

Parameters

filename String: name of file to load, can be .gif, .jpg, .tga, or a handful of other image types depending on your platform

extension String: type of image to load, for example "png", "gif", "jpg"

Returns

PImage

●Name: createFont()

Description

Dynamically converts a font to the format used by Processing from a .ttf or .otf file inside the sketch's "data" folder or a font that's installed elsewhere on the computer. If you want to use a font installed on your computer, use the PFont.list() method to first determine the names for the fonts recognized by the computer and are compatible with this function. Not all fonts can be used and some might work with one operating system and not others. When sharing a sketch with other people or posting it on the web, you may need to include a .ttf or .otf version of your font in the data directory of the sketch because other people might not have the font installed on their computer. Only fonts that can legally be distributed should be included with a sketch.

The size parameter states the font size you want to generate. The smooth parameter specifies if the font should be antialiased or not. The charset parameter is an array of chars that specifies the characters to generate.

Syntax

createFont(name, size)

createFont(name, size, smooth)

createFont(name, size, smooth, charset)

Parameters

name	String: name of the font to load
size	float: point size of the font
smooth	boolean: true for an antialiased font, false for aliased
charset	char[]: array containing characters to be generated

Returns

PFont

●Name: fill()

Description

Sets the color used to fill shapes. For example, if you run fill(204, 102, 0), all subsequent shapes will be filled with orange. This color is either specified in terms of the RGB or HSB color depending on the current colorMode(). (The default color space is RGB, with each value in the range from 0 to 255.)

When using hexadecimal notation to specify a color, use "#" or "0x" before the values (e.g., #CCFFAA or 0xFFCCFFAA). The # syntax uses six digits to specify a color (just as colors are typically specified in HTML and CSS). When using the hexadecimal notation starting with "0x", the hexadecimal value must be specified with eight characters; the first two characters define the alpha component, and the remainder define the red, green, and blue components.

The value for the "gray" parameter must be less than or equal to the current maximum value as specified by colorMode(). The default maximum value is 255.

Syntax

fill(rgb)

fill(rgb, alpha)

fill(gray)

fill(gray, alpha)

fill(v1, v2, v3)

fill(v1, v2, v3, alpha)

Parameters

rgb int: color variable or hex value

alpha float: opacity of the fill

gray float: number specifying value between white and black

v1 float: red or hue value (depending on current color mode)

v2 float: green or saturation value (depending on current color mode)

v3 float: blue or brightness value (depending on current color mode)

Returns

void

●Name: textFont()

Description

Sets the current font that will be drawn with the text() function. Fonts must be created for Processing with createFont() or loaded with loadFont() before they can be used. The font set through textFont() will be used in all subsequent calls to the text() function. If no size parameter is input, the font will appear at its original size (the size in which it was created with the "Create Font..." tool) until it is changed with textSize().

Because fonts are usually bitmapped, you should create fonts at the sizes that will be used most commonly. Using textFont() without the size parameter will result in the cleanest type.

With the default and PDF renderers, it's also possible to enable the use of native fonts via the command hint(ENABLE_NATIVE_FONTS). This will produce vector text in both on-screen sketches and PDF output when the vector data is available, such as when the font is still installed, or the font is created dynamically via the createFont() function (rather than with the "Create Font..." tool).

Syntax

textFont(which)

textFont(which, size)

Parameters

which PFont: any variable of the type PFont

size float: the size of the letters in units of pixels

Returns

void

●Name: text()

Description

Draws text to the screen. Displays the information specified in the first parameter on the screen in the position specified by the additional parameters. A default font will be used unless a font is set with the textFont() function and a default size will be used unless a font is set with textSize(). Change the color of the text with the fill() function. The text displays in relation to the textAlign() function, which gives the option to draw to the left, right, and center of the coordinates.

The x2 and y2 parameters define a rectangular area to display within and may only be used with string data. When these parameters are specified, they are interpreted based on the current rectMode() setting. Text that does not fit completely within the rectangle specified will not be drawn to the screen.

Note that Processing now lets you call text() without first specifying a PFont with textFont(). In that case, a generic sans-serif font will be used instead.

Syntax

text(c, x, y)

text(c, x, y, z)

text(str, x, y)

text(chars, start, stop, x, y)

text(str, x, y, z)

text(chars, start, stop, x, y, z)

text(str, x1, y1, x2, y2)

text(num, x, y)

text(num, x, y, z)

Parameters

c char: the alphanumeric character to be displayed

x float: x-coordinate of text

y float: y-coordinate of text

z float: z-coordinate of text

chars char[]: the alphanumberic symbols to be displayed

start int: array index at which to start writing characters

stop int: array index at which to stop writing characters

x1 float: by default, the x-coordinate of text, see rectMode() for more info

y1 float: by default, the x-coordinate of text, see rectMode() for more info

x2 float: by default, the width of the text box, see rectMode() for more info

y2 float: by default, the height of the text box, see rectMode() for more info

num int, or float: the numeric value to be displayed

Returns

void

●Name: Serial

Description

Class for sending and receivinag data using the serial communication protocol.

●Name: available()

Description

Returns the number of bytes available.

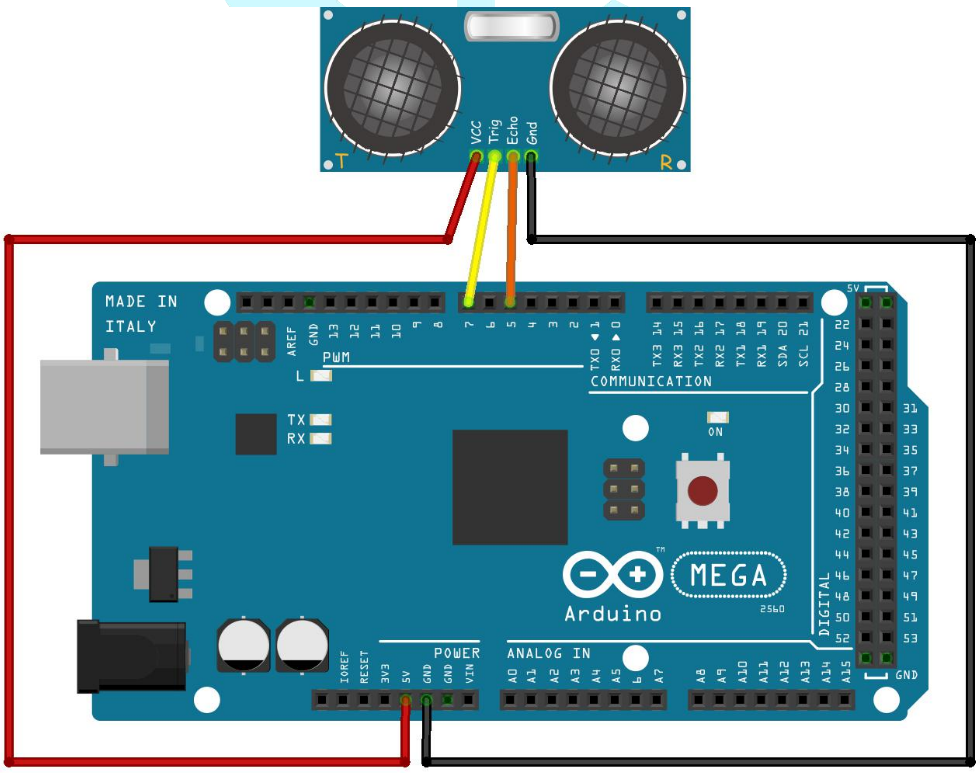
●Name: read()

Description

Returns a number between 0 and 255 for the next byte that's waiting in the buffer. Returns -1 if there is no byte, although this should be avoided by first cheacking available() to see if data is available.

Procedures

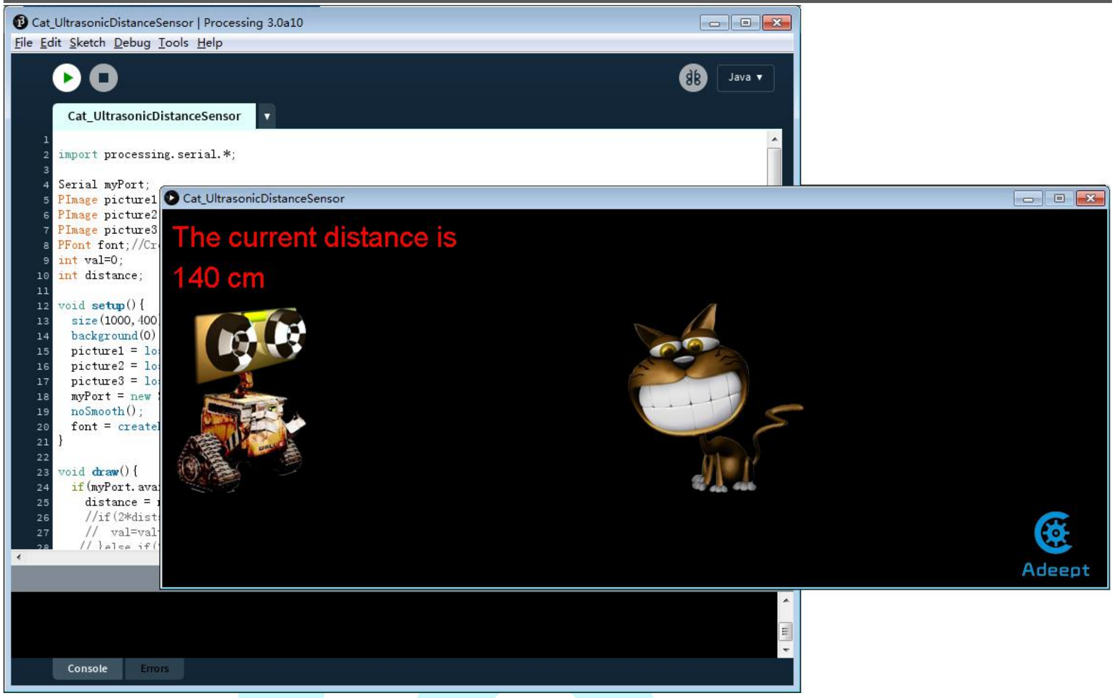
1. Build the circuit



2. Program

3. Compile the program and upload to Adeept 2560 board

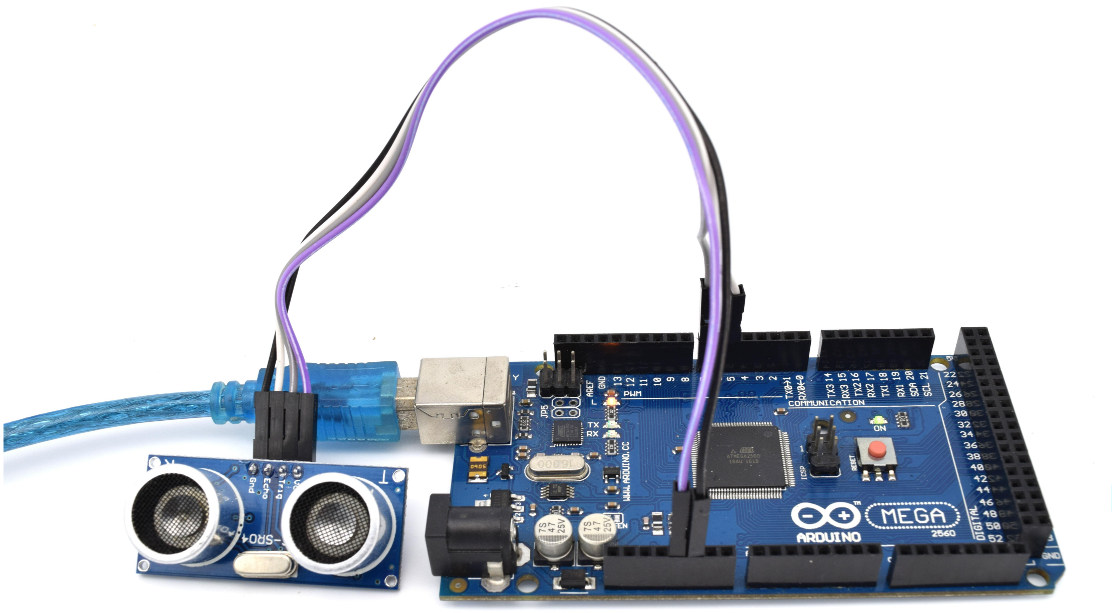
4. Run processing software (Cat_UltrasonicDistanceSensor.pde)


Cat_UltrasonicDistanceSensor | Processing 3.0a10
File Edit Sketch Debug Tools Help
Java
Cat_UltrasonicDistanceSensor
Console
Errors

```
import processing.serial.*;

Serial myPort;
PImage picture1
PImage picture2
PImage picture3
PFont font;//Cr
int val=0;
int distance;

void setup(){
  size(1000,400
  background(0)
  picture1 = lo
  picture2 = lo
  picture3 = lo
  myPort = new
  noSmooth();
  font = create
}

void draw(){
  if(myPort.ava
    distance =
    //if(2*dist
    //  val=val
  // }else if(
```

Cat_UltrasonicDistanceSensor

The current distance is
140 cm



Lesson 34 Control the brightness of a photo with a photoresistor

Overview

This is an interesting interaction experiment for the Arduino and Processing. We acquire the brightness by programming the Arduino 2560, and then change the brightness of a photo.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* Light Sensor (Photoresistor)
- 1* 10KΩ Resistor
- 1* Breadboard
- Several Jumper Wires

Principle

The experiment is divided into two parts. The first is used to acquire the data from Arduino, another is the used to process the data.

The Arduino 2560 board sends the brightness data to the Processing software via serial port, and then the Processing software changes the brightness of a image according to the data. When the photoresistor in a dark environment, the brightness of the image will be decreased. In contrast, the brightness of the image will be increased.

Note:

1. In this experiment, my Arduino 2560 board is connected to my computer COM26, please adjust according to actual situation.

2. If the Processing has not running normally, you need to install the related function libraries.

Arduino key function:

●map(value, fromLow, fromHigh, toLow, toHigh)

Description

Re-maps a number from one range to another. That is, a value of fromLow would get mapped to toLow, a value of fromHigh to toHigh, values in-between to values in-between, etc.

Adeept

Does not constrain values to within the range, because out-of-range values are sometimes intended and useful. The constrain() function may be used either before or after this function, if limits to the ranges are desired.

Note that the “lower bounds” of either range may be larger or smaller than the “upper bounds” so the map() function may be used to reverse a range of numbers, for example

y = map(x, 1, 50, 50, 1);

The function also handles negative numbers well, so that this example

y = map(x, 1, 50, 50, -100);

is also valid and works well. The map() function uses integer math so will not generate fractions, when the math might indicate that it should do so. Fractional remainders are truncated, and are not rounded or averaged.

Parameters

value: the number to map

fromLow: the lower bound of the value's current range

fromHigh: the upper bound of the value's current range

toLow: the lower bound of the value's target range

toHigh: the upper bound of the value's target range

Returns

The mapped value.

Processing key function:

●Name: tint()

Description

Sets the fill value for displaying images. Images can be tinted to specified colors or made transparent by including an alpha value.

To apply transparency to an image without affecting its color, use white as the tint color and specify an alpha value. For instance, tint(255, 128) will make an image 50% transparent (assuming the default alpha range of 0-255, which can be changed with colorMode()).

When using hexadecimal notation to specify a color, use "#" or "0x" before the values (e.g., #CCFFAA or 0xFFCCFFAA). The # syntax uses six digits to specify a color (just as colors are typically specified in HTML and CSS). When using the hexadecimal notation starting with "0x", the hexadecimal value must be specified with eight characters; the first two characters define the alpha component, and the remainder define the red, green, and blue components.

The value for the gray parameter must be less than or equal to the current maximum value as specified by colorMode(). The default maximum value is 255.

The tint() function is also used to control the coloring of textures in 3D.

Syntax

tint(rgb)

tint(rgb, alpha)

tint(gray)

tint(gray, alpha)

tint(v1, v2, v3)

tint(v1, v2, v3, alpha)

Parameters

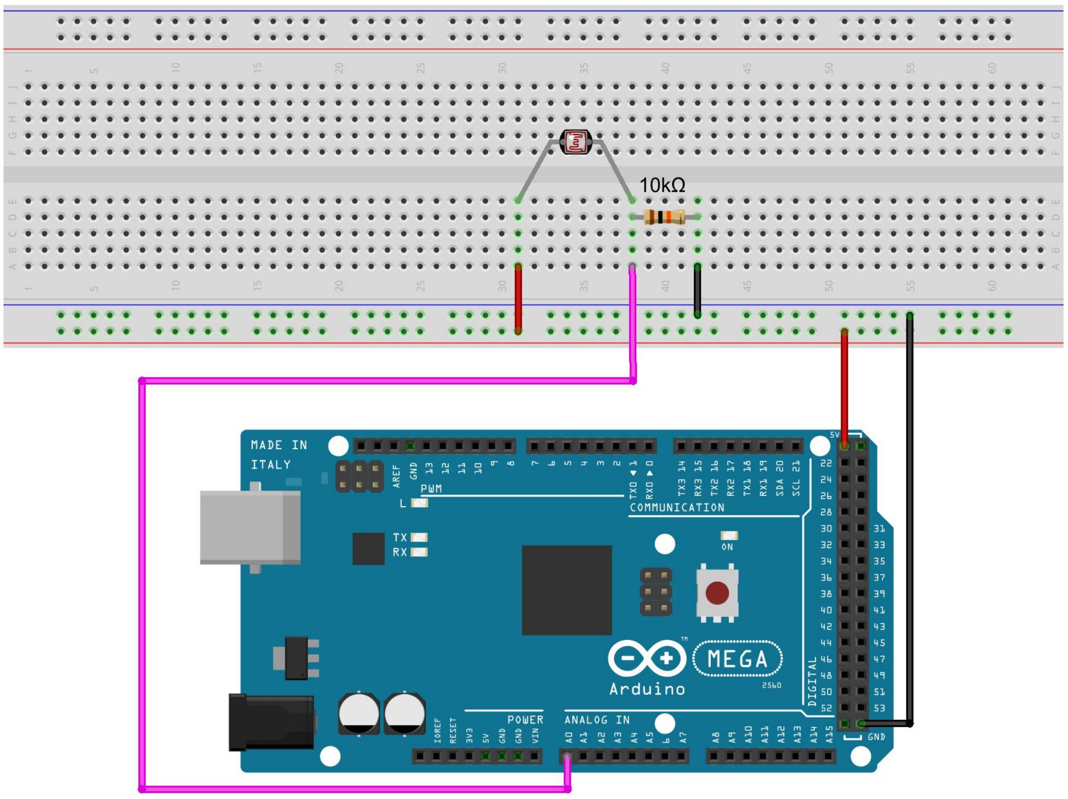
rgb	int: color value in hexadecimal notation
alpha	float: opacity of the image
gray	float: specifies a value between white and black
v1	float: red or hue value (depending on current color mode)
v2	float: green or saturation value (depending on current color mode)
v3	float: blue or brightness value (depending on current color mode)

Returns

void

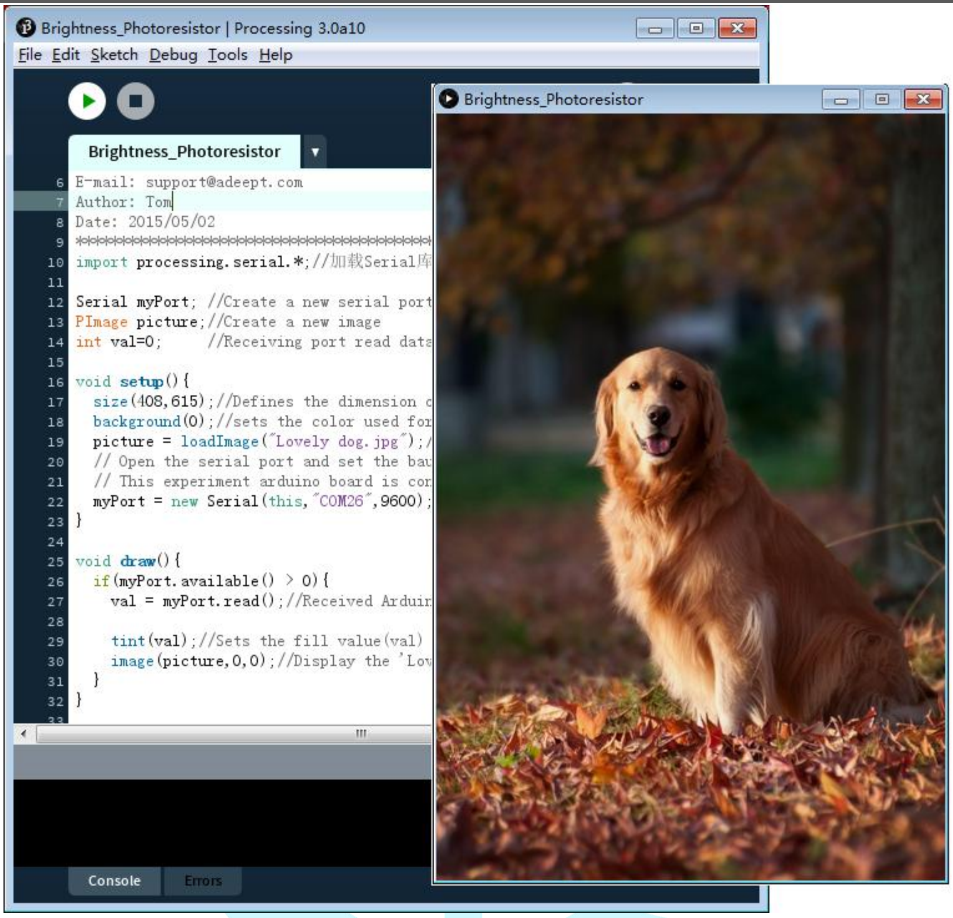
Procedures

1. Build the circuit

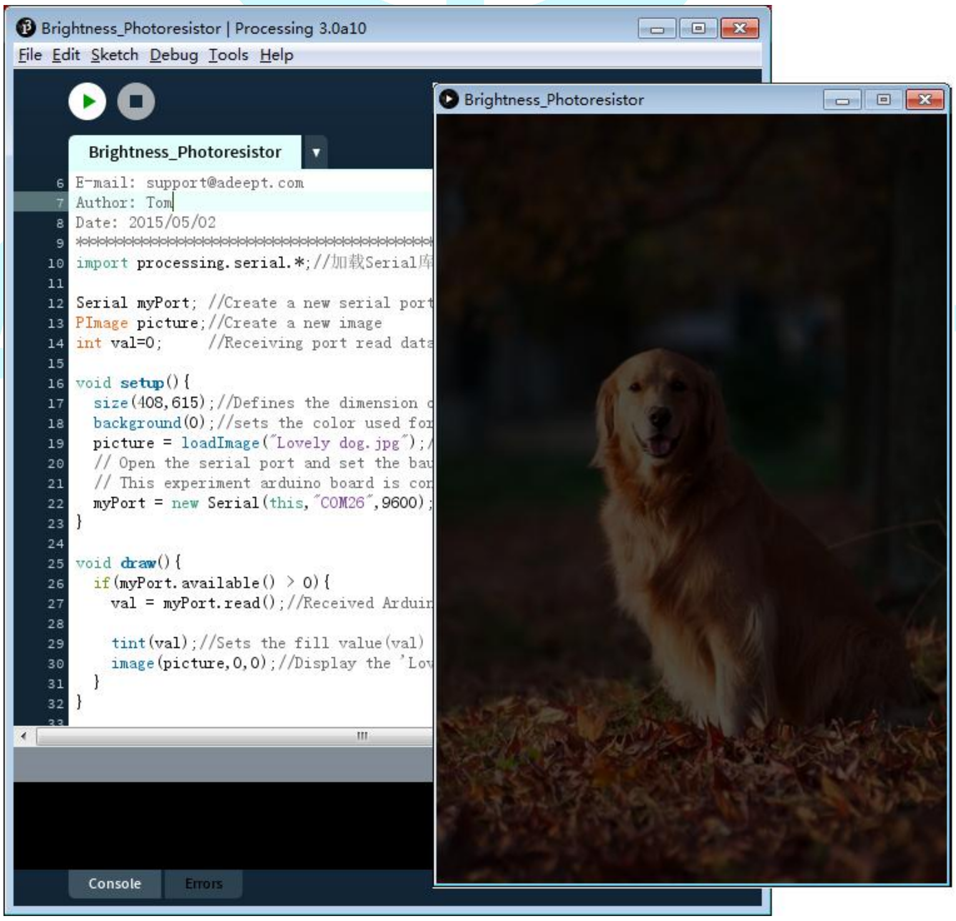


2. Program
3. Compile the program and upload to Adeept 2560 board
4. Run processing software (Brightness_Photoresistor.pde)

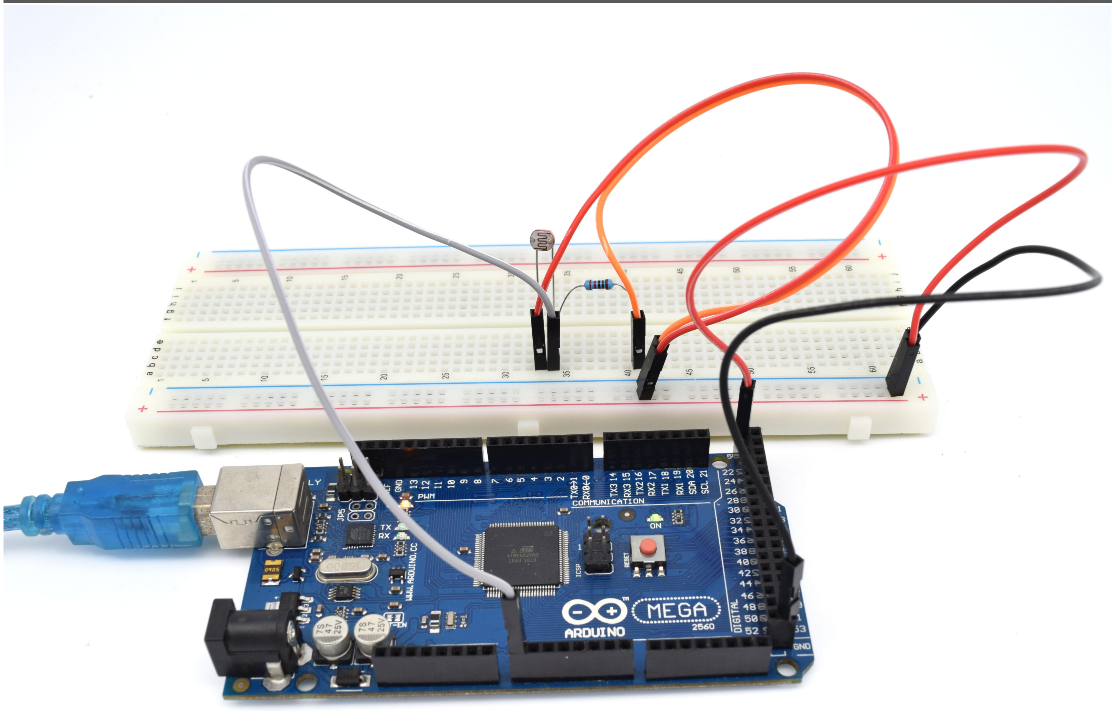
Adeept


Brightness_Photoresistor | Processing 3.0a10
File Edit Sketch Debug Tools Help
Brightness_Photoresistor
Brightness_Photoresistor
Console
Errors

```
E-mail: support@adeept.com
Author: Tom
Date: 2015/05/02
*******************************************
import processing.serial.*;//加载Serial库

Serial myPort; //Create a new serial port
PImage picture;//Create a new image
int val=0;      //Receiving port read data

void setup(){
  size(408,615);//Defines the dimension c
  background(0);//sets the color used for
  picture = loadImage("Lovely dog.jpg");/
  // Open the serial port and set the bau
  // This experiment arduino board is cor
  myPort = new Serial(this,"COM26",9600);
}

void draw(){
  if(myPort.available() > 0){
    val = myPort.read();//Received Arduin

    tint(val);//Sets the fill value(val)
    image(picture,0,0);//Display the 'Lov
  }
}
```


Brightness_Photoresistor | Processing 3.0a10
File Edit Sketch Debug Tools Help
Brightness_Photoresistor
Brightness_Photoresistor
Console
Errors

```
E-mail: support@adeept.com
Author: Tom
Date: 2015/05/02
*******************************************
import processing.serial.*;//加载Serial库

Serial myPort; //Create a new serial port
PImage picture;//Create a new image
int val=0;      //Receiving port read data

void setup(){
  size(408,615);//Defines the dimension c
  background(0);//sets the color used for
  picture = loadImage("Lovely dog.jpg");/
  // Open the serial port and set the bau
  // This experiment arduino board is cor
  myPort = new Serial(this,"COM26",9600);
}

void draw(){
  if(myPort.available() > 0){
    val = myPort.read();//Received Arduin

    tint(val);//Sets the fill value(val)
    image(picture,0,0);//Display the 'Lov
  }
}
```



Adeept

Lesson 35 Controlling the 3D Model by PS2 Joystick

Overview

In this lesson, we will collect the state of a joystick by programming the Arduino 2560 Board, and then send the data to the Processing through the serial communication.

Components

- 1 * Arduino MEGA 2560
- 1 * USB Cable
- 1 * PS2 Joystick
- 1 * Breadboard
- Several jumper wires

Principle

The experiment consists of two parts: first, acquire the data from Arduino; second, process the data.

Here use the Arduino 2560 board to collect data of the joystick state, and upload the data to the computer through the serial port. The data will be processed by Processing and shown with 3D image.

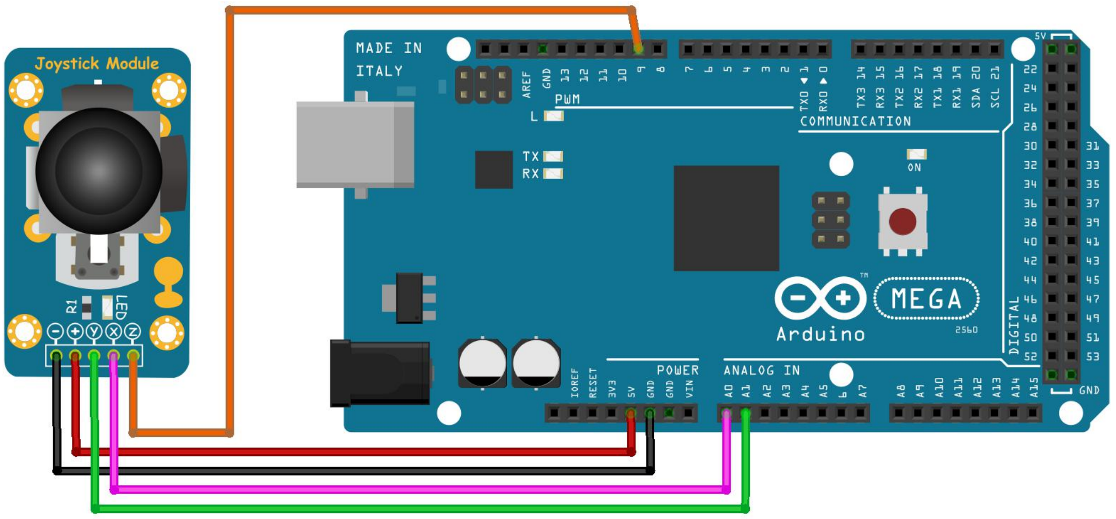
Note:

1. In this experiment, my Arduino 2560 board is connected to my computer port COM26. But it may differ in your case. So please adjust it according to your actual situation.

2. If the Processing does not run normally, you may need to install the related function libraries.

Procedures

Step 1: Build the circuit

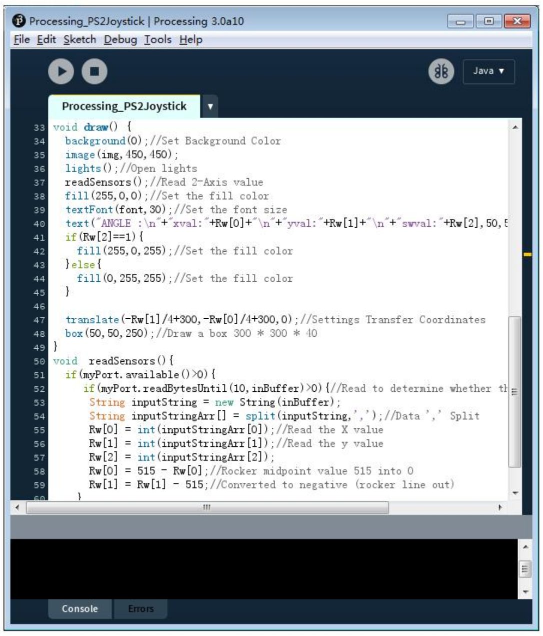


Step 2: Program

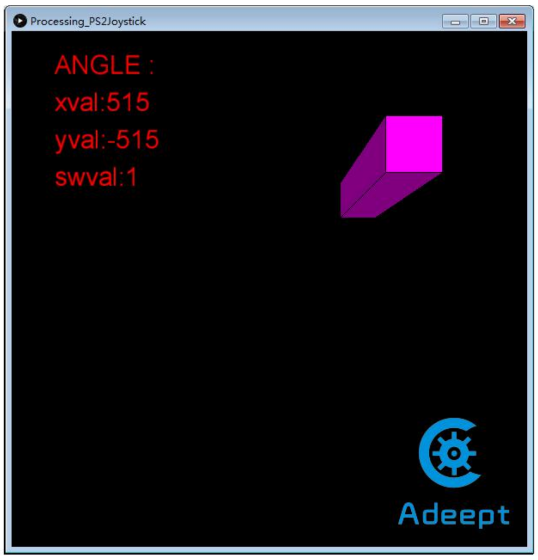
Step 3: Compile the program and upload to arduino 2560 board

Step 4: Run the Processing software (Processing_PS2Joystick.pde)

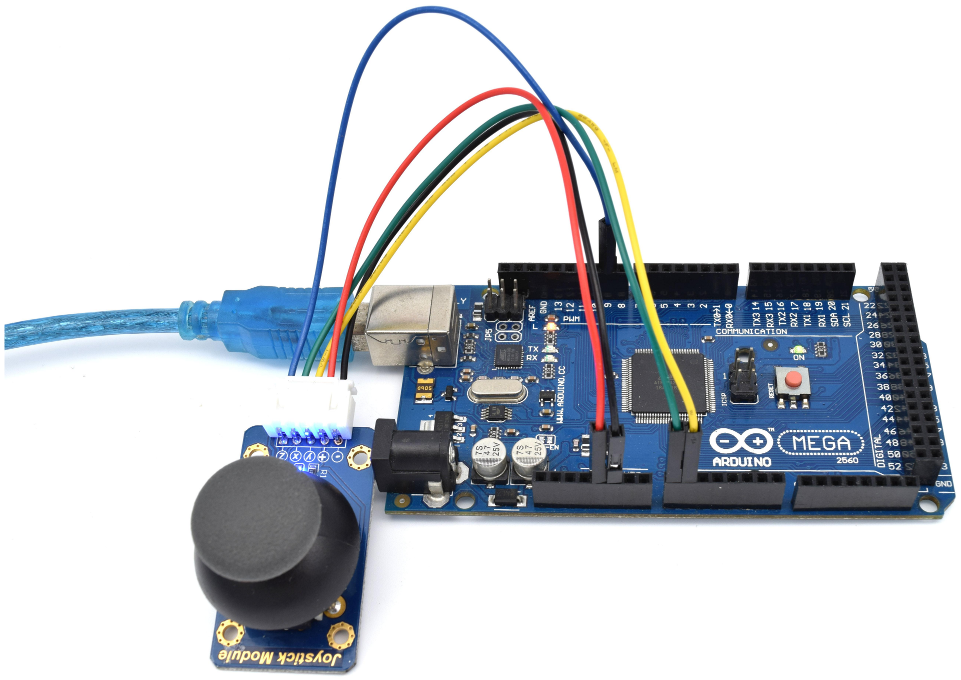
Move the joystick, and the 3D model will follow movement changes accordingly on your computer.


Processing_PS2Joystick | Processing 3.0a10
File Edit Sketch Debug Tools Help
Java
Processing_PS2Joystick
Console
Errors

```
void draw()  {
  background(0);//Set Background Color
  image(img,450,450);
  lights();//Open lights
  readSensors();//Read 2-Axis value
  fill(255,0,0);//Set the fill color
  textFont(font,30);//Set the font size
  text("ANGLE :\n"+"xval:"+Rw[0]+"\n"+"yval:"+Rw[1]+"\n"+"swval:"+Rw[2],50,5
  if(Rw[2]==1){
    fill(255,0,255);//Set the fill color
  }else{
    fill(0,255,255);//Set the fill color
  }

  translate(-Rw[1]/4+300,-Rw[0]/4+300,0);//Settings Transfer Coordinates
  box(50,50,250);//Draw a box 300 * 300 * 40
}
void  readSensors(){
  if(myPort.available()>0){
     if(myPort.readBytesUntil(10,inBuffer)>0){//Read to determine whether th
      String inputString = new String(inBuffer);
      String inputStringArr[] = split(inputString,',');//Data ',' Split
      Rw[0] = int(inputStringArr[0]);//Read the X value
      Rw[1] = int(inputStringArr[1]);//Read the y value
      Rw[2] = int(inputStringArr[2]);
      Rw[0] = 515 - Rw[0];//Rocker midpoint value 515 into 0
      Rw[1] = Rw[1] - 515;//Converted to negative (rocker line out)
    }
```



```
ANGLE :
xval:515
yval:-515
swval:1
```



Adeept

Lesson 36 Adeept ardublock blinking LED

Overview

In this tutorial, we will start the journey of learning Arduino ardublock. In the first lesson, we will learn how to make a LED blinking.

Requirement

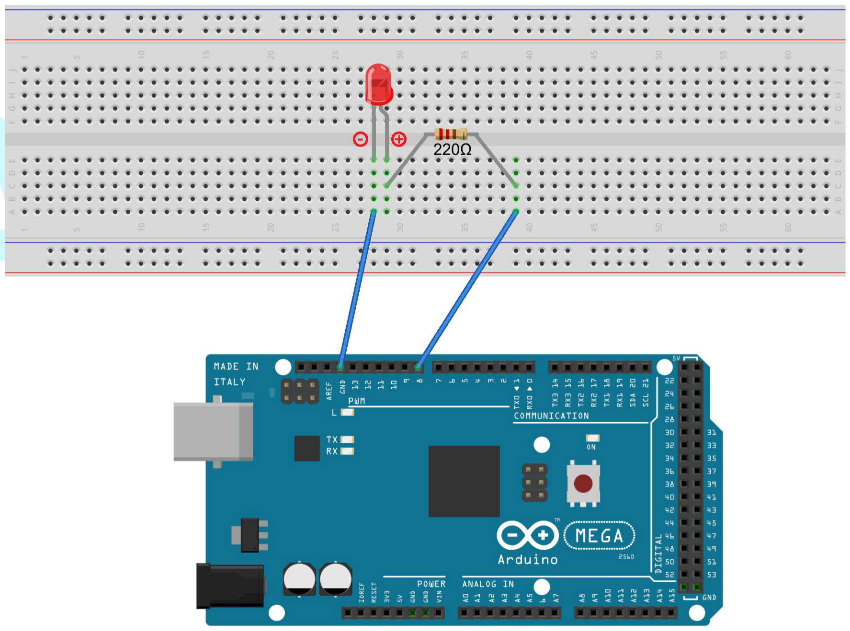
- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* 220Ω Resistor
- 1* LED
- 1* Breadboard
- 2* Jumper Wires

Principle

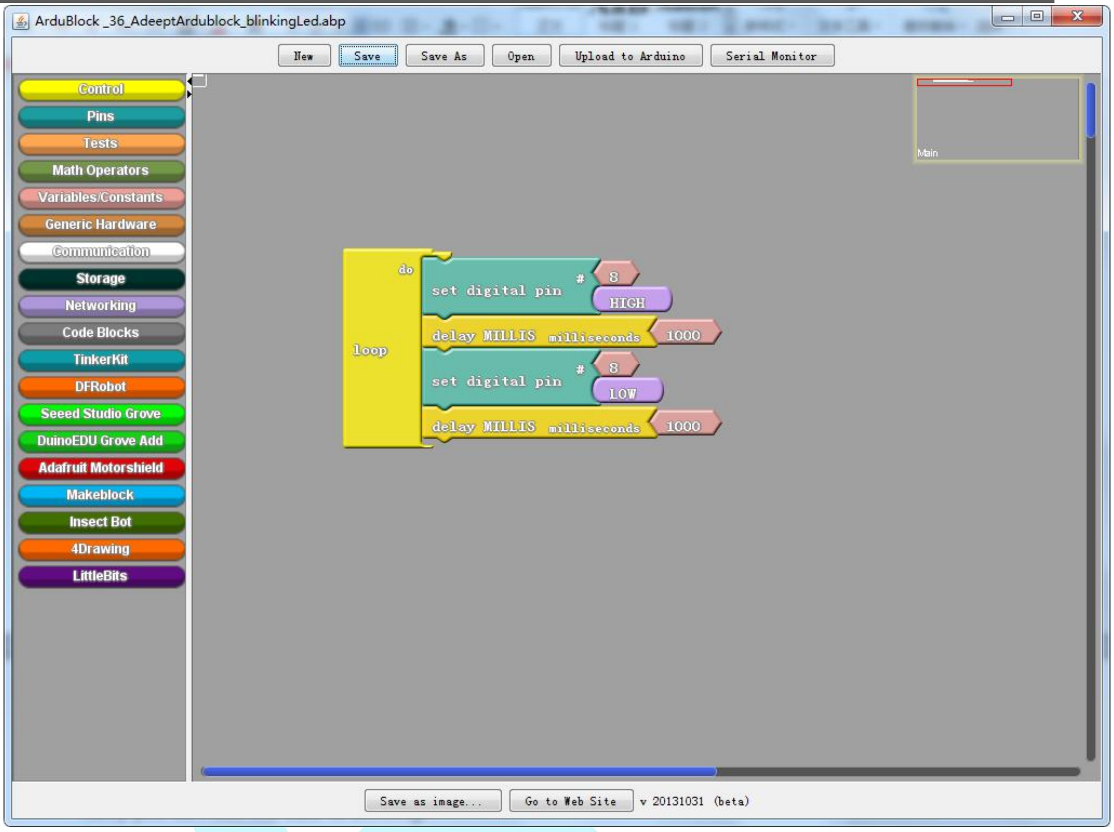
In this lesson, we will program the Arduino's GPIO output high(+5V) and low level(0V), and then make the LED which is connected to the Arduino’s GPIO flicker with a certain frequency.

Procedures

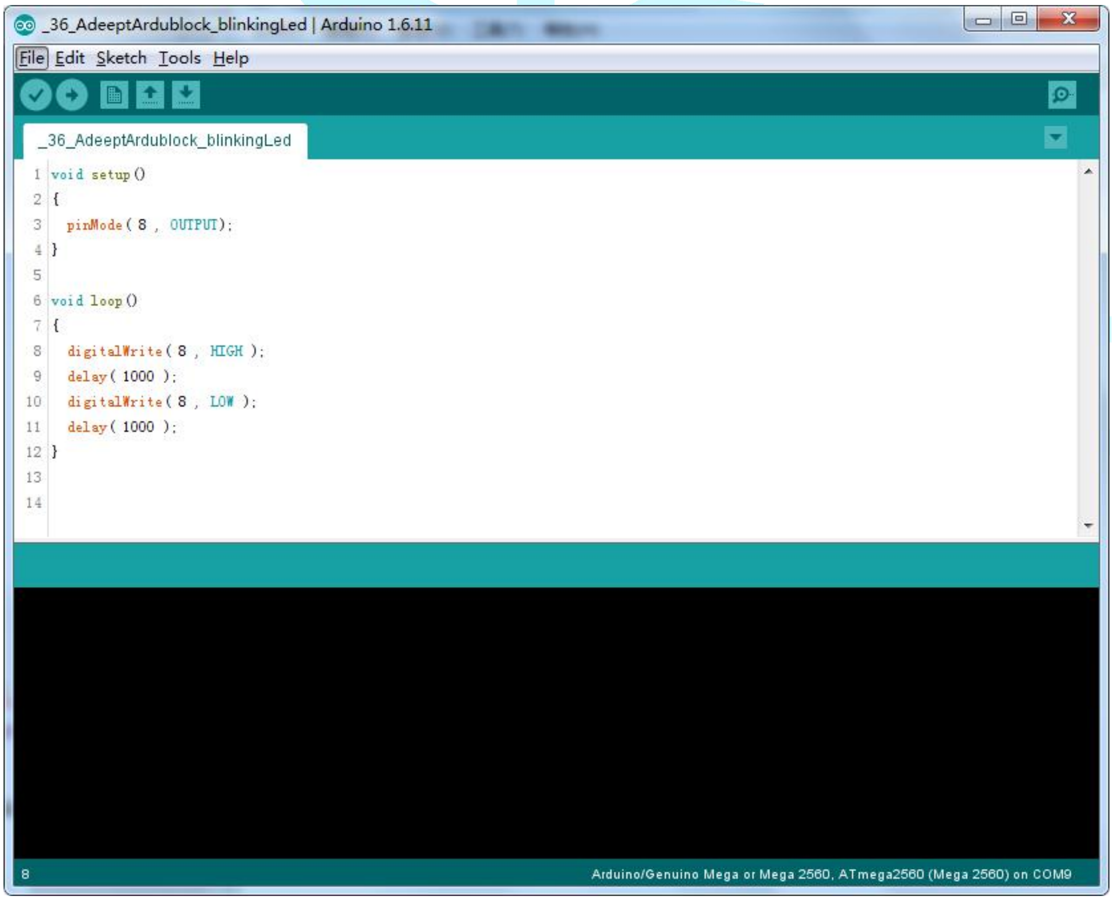
1. Build the circuit



2. Program



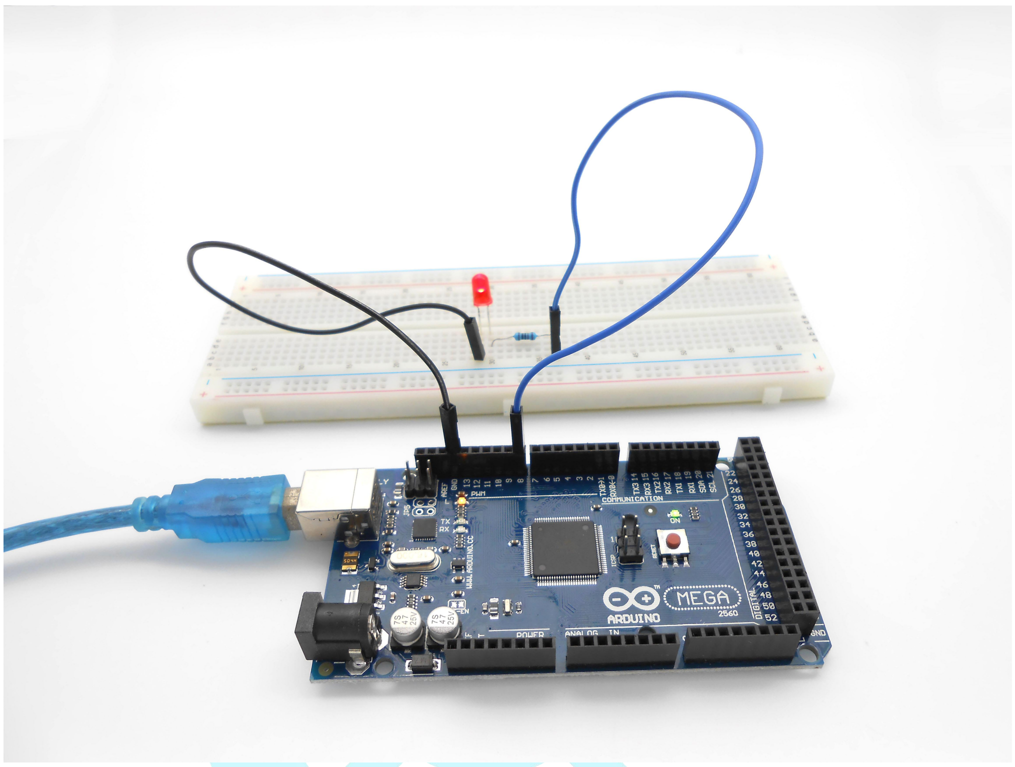
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_36_AdeeptArdublock_blinkingLed | Arduino 1.6.11
File Edit Sketch Tools Help
_36_AdeeptArdublock_blinkingLed

```
void setup()
{
  pinMode( 8 , OUTPUT);
}

void loop()
{
  digitalWrite( 8 , HIGH );
  delay( 1000 );
  digitalWrite( 8 , LOW );
  delay( 1000 );
}
```

8
Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Now, you can see the LED is blinking.



Lesson 37 Adeept ardublock active buzzer

Overview

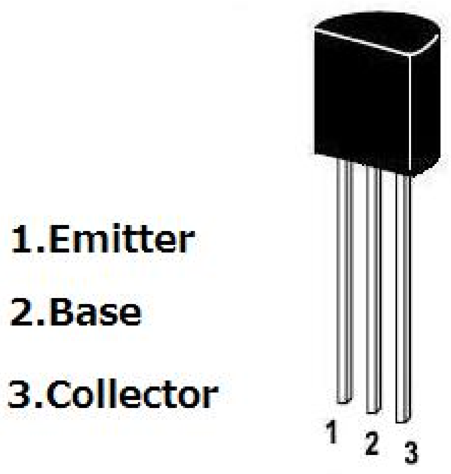
In this lesson, we will learn how to program the ardublock to make an active buzzer sound.

Requirement

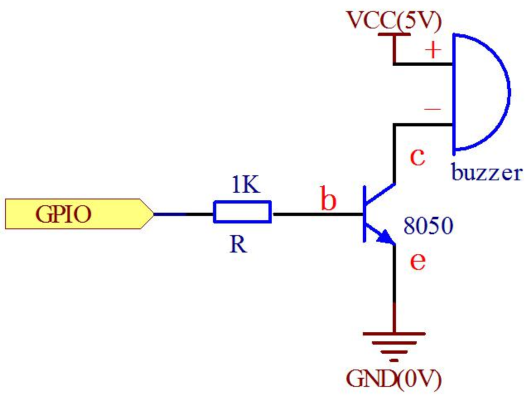
- 1* Arduino MEGA 2560
- 1* USB cable
- 1* Active buzzer
- 1* 1 kΩ Resistor
- 1* NPN Transistor (S8050)
- 1* Breadboard
- Several Jumper Wires

Principle

The main function of transistor is blowing up the voltage or current. The transistor can also be used to control the circuit conduction or deadline. And the transistor is divided into two kinds, one kind is NPN, for instance, the S8050 we provided; another kind is PNP transistor such as the S8550 we provided. The transistor we used is as shown in below:



There are two driving circuit for the buzzer:



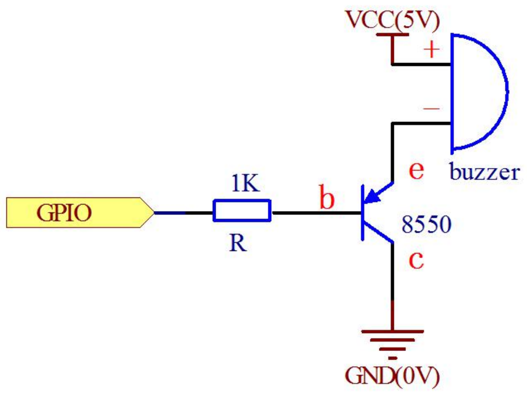


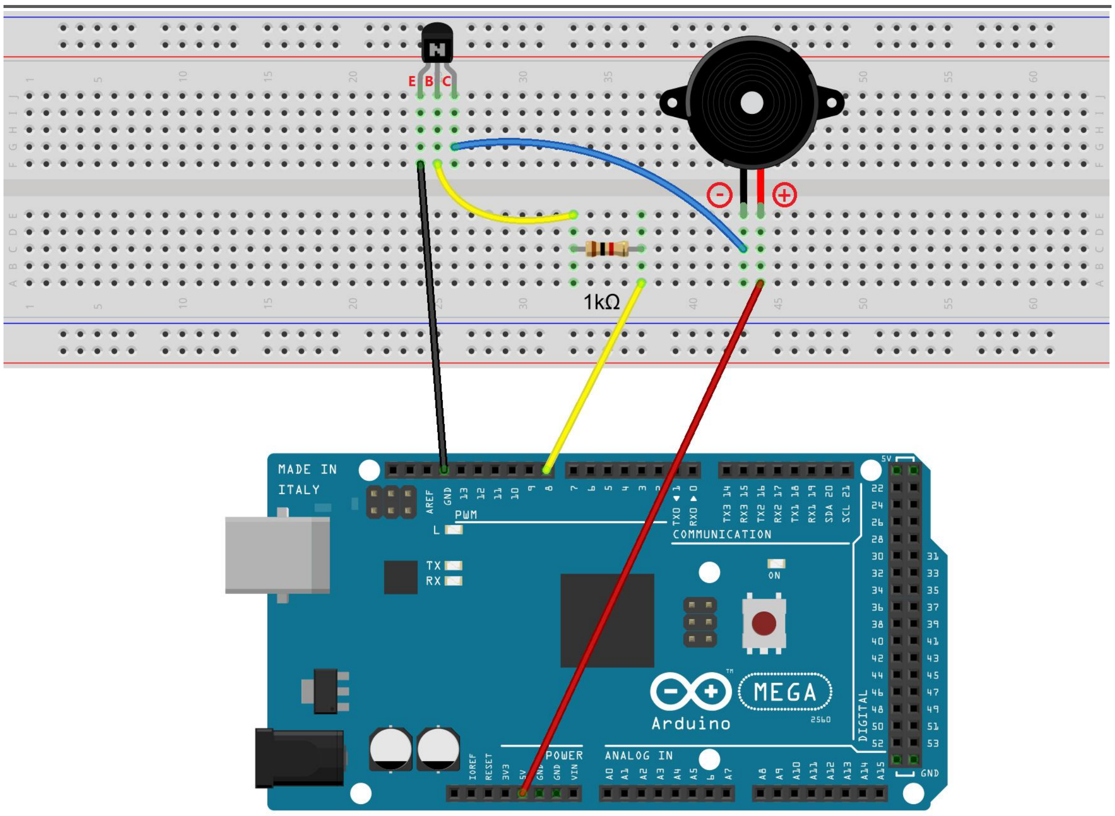
Figure1 Figure2

Figure 1: Set the Arduino GPIO as a high level, the transistor S8050 will conduct, and then the buzzer will sound; set the Arduino GPIO as low level, the transistor S8050 will cut off, then the buzzer will stop.

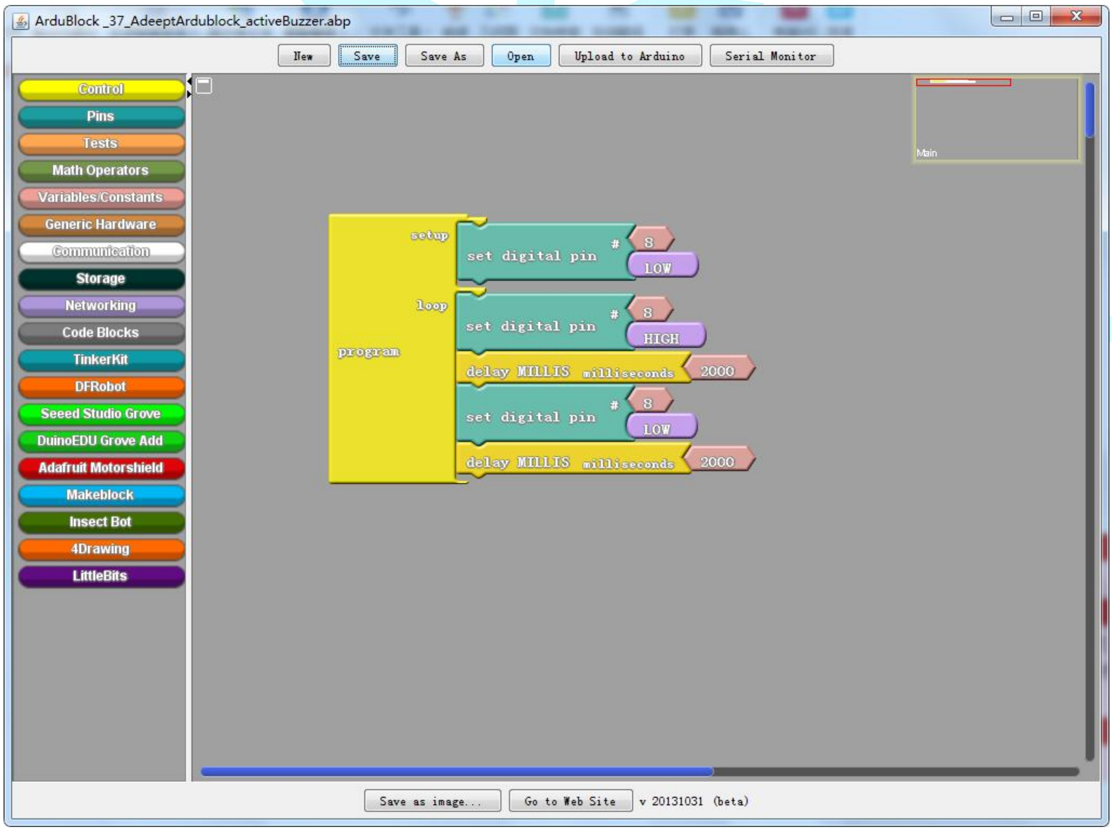
Figure 2: Set the Arduino GPIO as low level, the transistor S8550 will conduct, and the buzzer will sound; set the Arduino GPIO as a high level, the transistor S8550 will cut off, then the buzzer will stop.

Procedures

1. Build the circuit

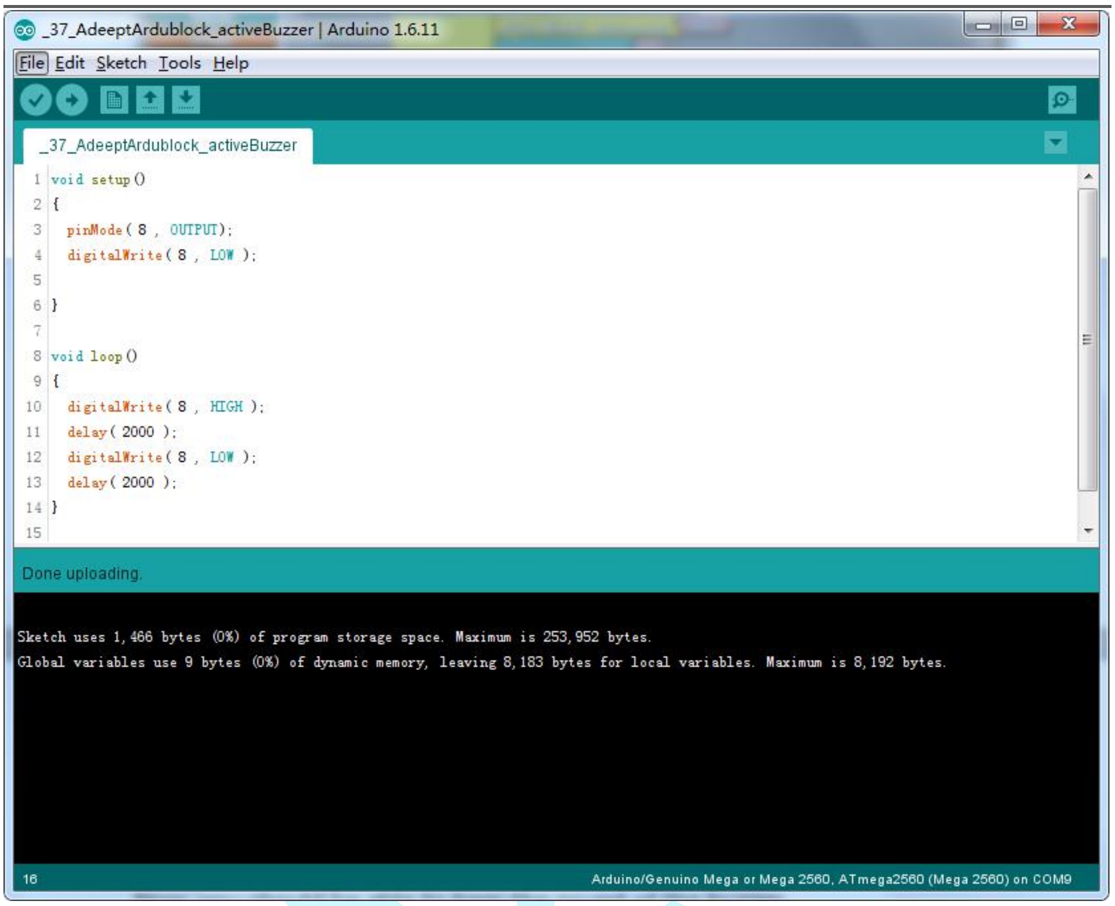


2. Program



```
program
  setup
    set digital pin # 8  LOW
  loop
    set digital pin # 8  HIGH
    delay MILLIS milliseconds 2000
    set digital pin # 8  LOW
    delay MILLIS milliseconds 2000
```

3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_37_AdeeptArdublock_activeBuzzer | Arduino 1.6.11
File Edit Sketch Tools Help
_37_AdeeptArdublock_activeBuzzer

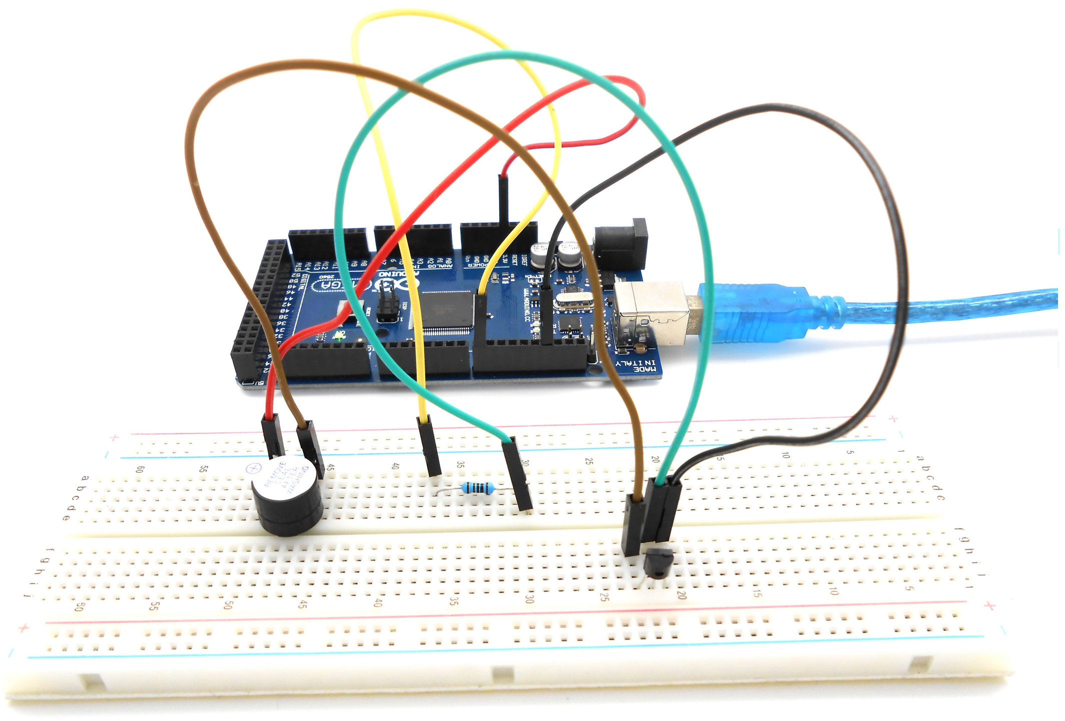
```
void setup()
{
  pinMode( 8 , OUTPUT);
  digitalWrite( 8 , LOW );

}

void loop()
{
  digitalWrite( 8 , HIGH );
  delay( 2000 );
  digitalWrite( 8 , LOW );
  delay( 2000 );
}

```

Done uploading.

```
Sketch uses 1,466 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 8,183 bytes for local variables. Maximum is 8,192 bytes.
```

16
Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Now, you should be able to hear the sound of the buzzer.



Lesson 38 Adeept ardublock controlling an LED with a button

Overview

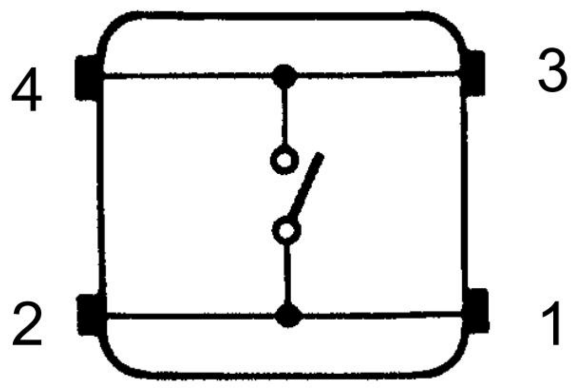
In this lesson, we will learn how to detect the state of a button, and then toggle the state of LED based on the state of the button.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* Button
- 1* LED
- 1* 10KΩ Resistor
- 1* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

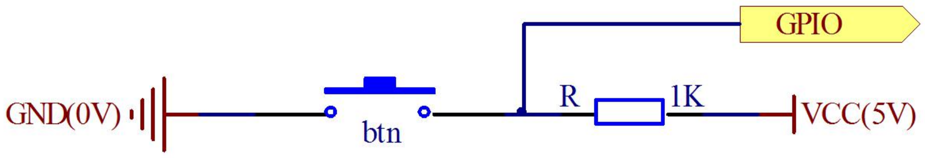
Principle

Buttons are a common component used to control electronic devices. They are usually used as switches to connect or disconnect circuits. Although buttons come in a variety of sizes and shapes, the one used in this experiment will be a 12mm button as shown in the following pictures. Pins pointed out by the arrows of same color are meant to be connected.



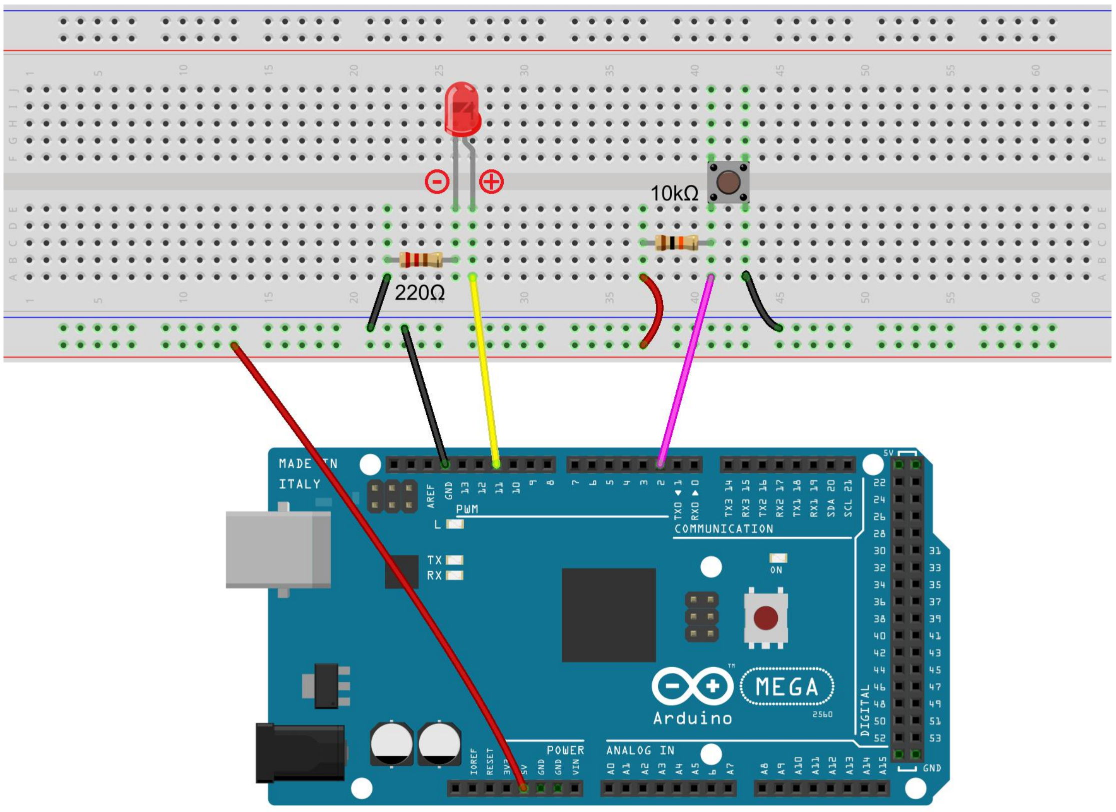
The button we used is a normally open type button. The two contacts of a button is in the off state under the normal conditions, only when the button is pressed they are closed.

The schematic diagram we used is as follows:



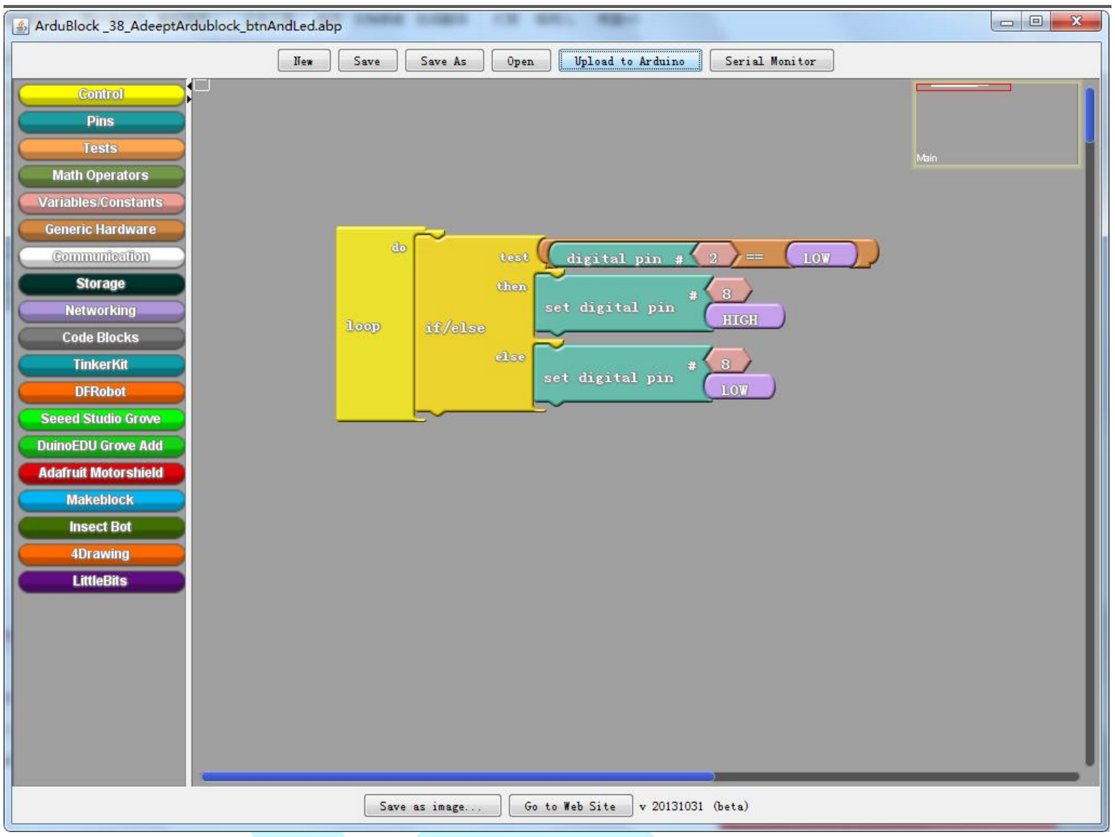
Procedures

1. Build the circuit

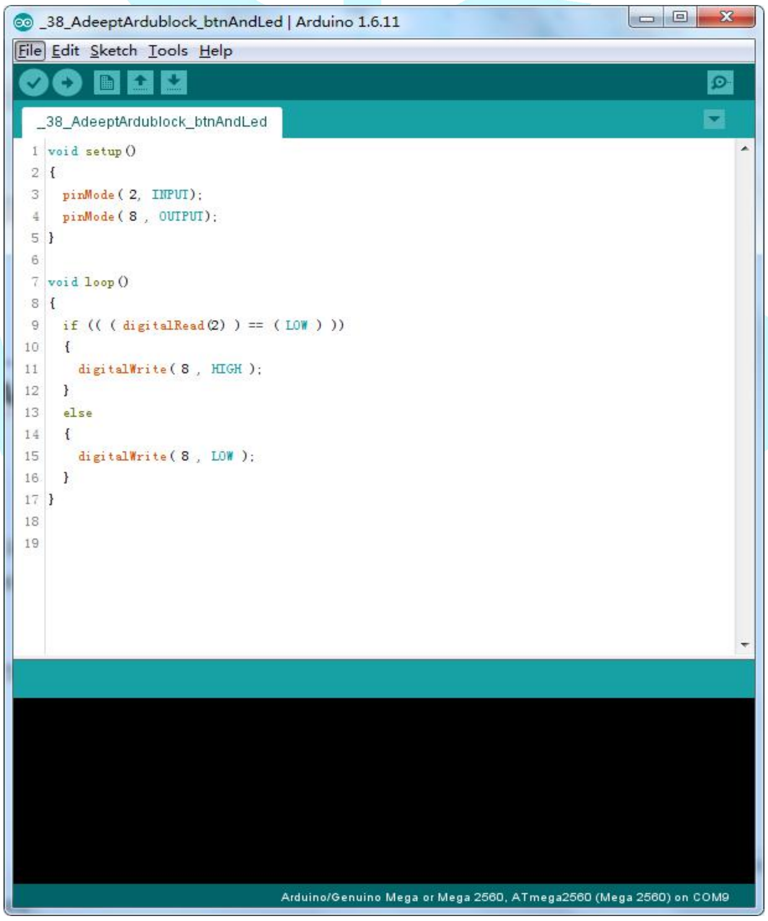


2. Program

Adeept

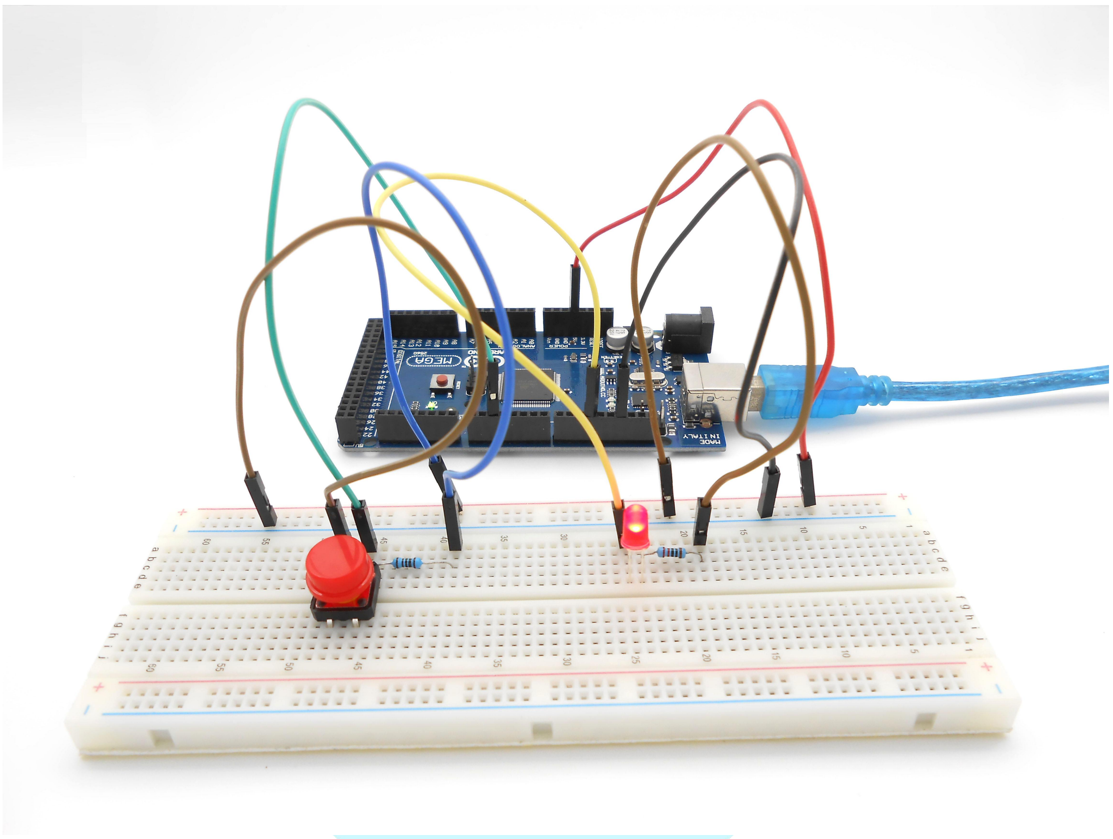


3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_38_AdeeptArdublock_btnAndLed | Arduino 1.6.11
File Edit Sketch Tools Help
_38_AdeeptArdublock_btnAndLed
Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

```
void setup()
{
  pinMode( 2, INPUT);
  pinMode( 8 , OUTPUT);
}

void loop()
{
  if (( ( digitalRead(2) ) == ( LOW ) ))
  {
    digitalWrite( 8 , HIGH );
  }
  else
  {
    digitalWrite( 8 , LOW );
  }
}
```

When you press the button, you can see the state of the LED will be toggled. (ON->OFF, OFF->ON).



Summary

Through this lesson, you should have learned how to use the Arduino MEGA 2560 detects an external button state, and then toggle the state of LED relying on the state of the button detected before.

Adeept

Lesson 39 Adeept ardublock relay module

Introduction

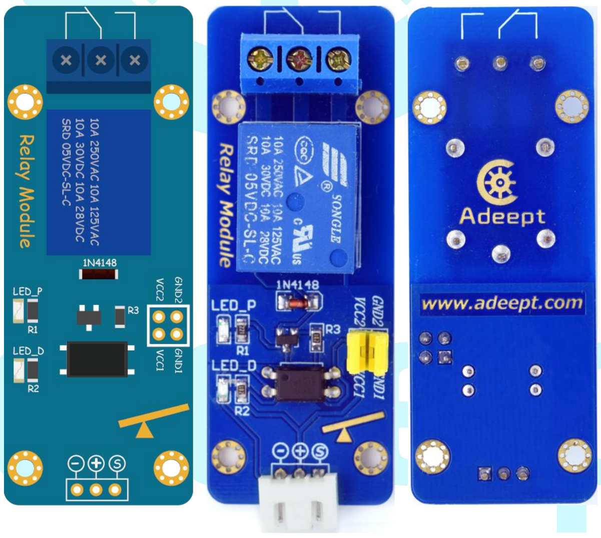
The relay is an electronic and electrical component that controls large currents by small currents. In the course of building an Arduino project, generally many large current or high volume devices like solenoid valve, lamp and motor cannot be connected directly to digital I/Os of the Arduino board. At this moment, a relay can save your project.

Components

- 1 * Arduino MEGA 2560
- 1 * Relay Module
- 1 * LED Module
- 1 * USB Cable
- 2 * 3-Pin Wires
- 3 * Hookup Wire Set
- 1 * Breadboard

Experimental Principle

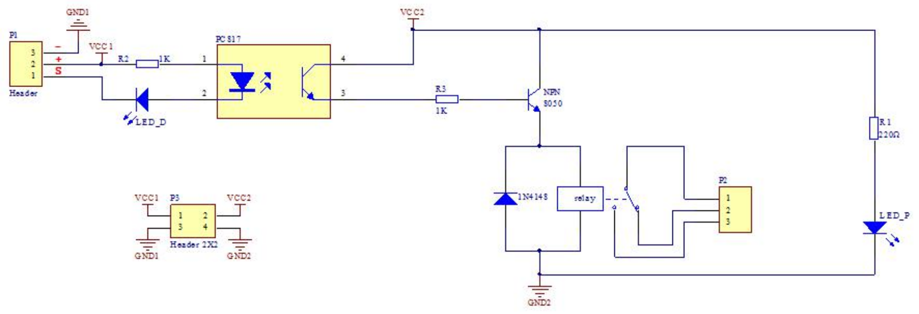
The Fritzing image:



Pin definition:

S	Digital Data Input
+	VCC1
-	GND1
VCC1	VCC1
GND1	GND1
VCC2	VCC1
GND2	GND2

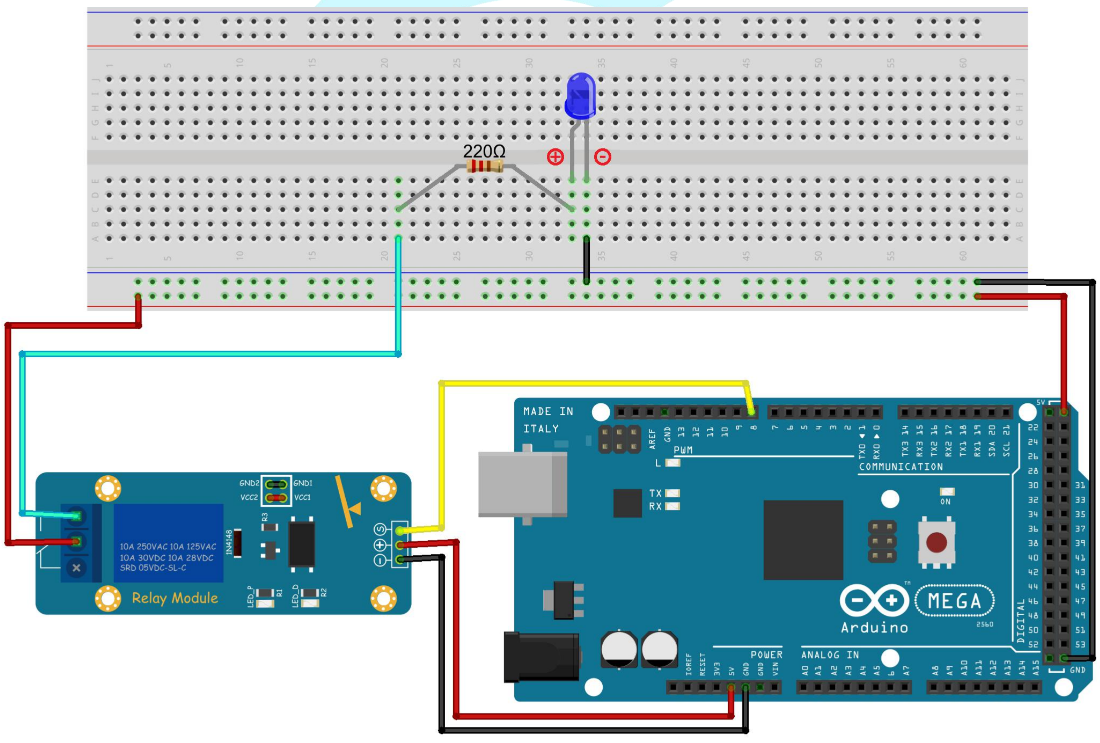
The schematic diagram:



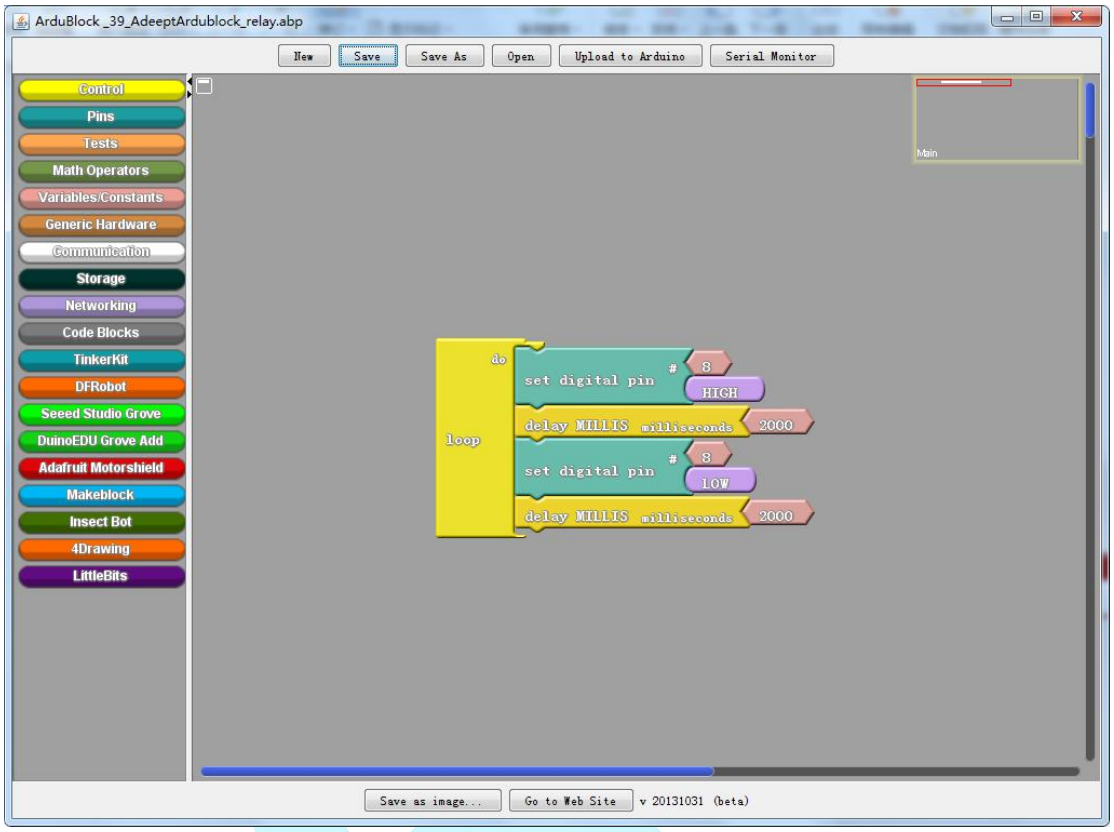
This experiment is to control an LED to brighten and dim by a relay.

Experimental Procedures

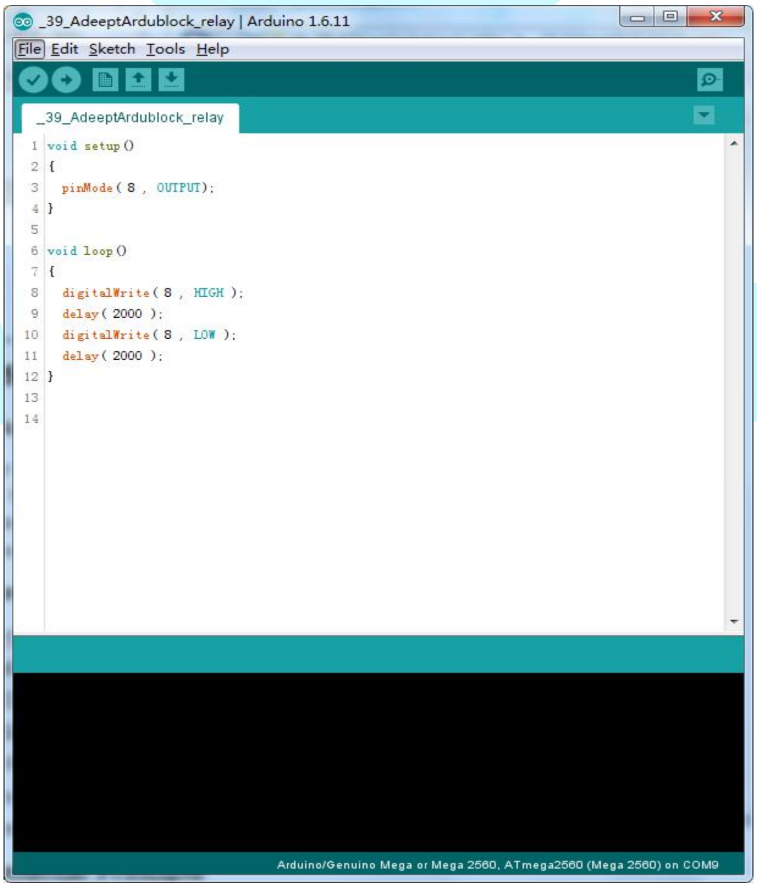
Step 1: Build the circuit



Step 2: Program



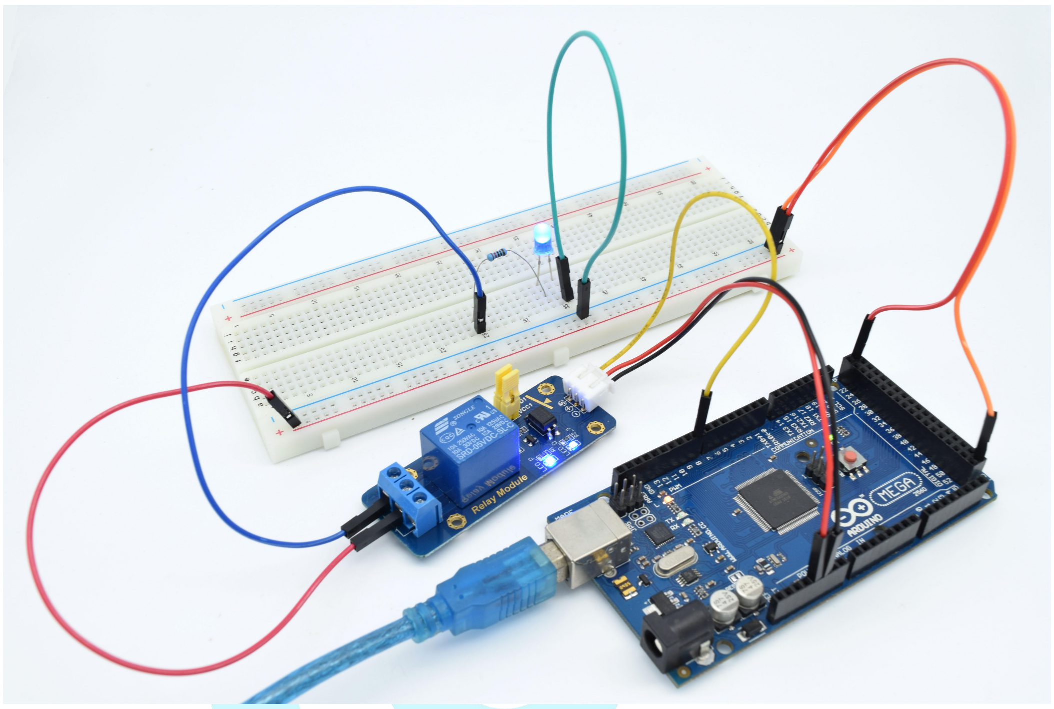
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_39_AdeeptArdublock_relay | Arduino 1.6.11
File Edit Sketch Tools Help
_39_AdeeptArdublock_relay

```
void setup()
{
  pinMode( 8 , OUTPUT);
}

void loop()
{
  digitalWrite( 8 , HIGH );
  delay( 2000 );
  digitalWrite( 8 , LOW );
  delay( 2000 );
}
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Now you can see the LED flickers every 2s and can hear the sound of relay closing and opening.



Adeept

Lesson 40 Adeept ardublock serial port

Overview

In this lesson, we will program the Arduino MEGA 2560 to achieve function of send and receive data through the serial port. The Arduino receiving data which send from PC, and then controlling an LED according to the received data, then return the state of LED to the PC's serial port monitor.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* LED
- 1* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

Principle

Serial ports

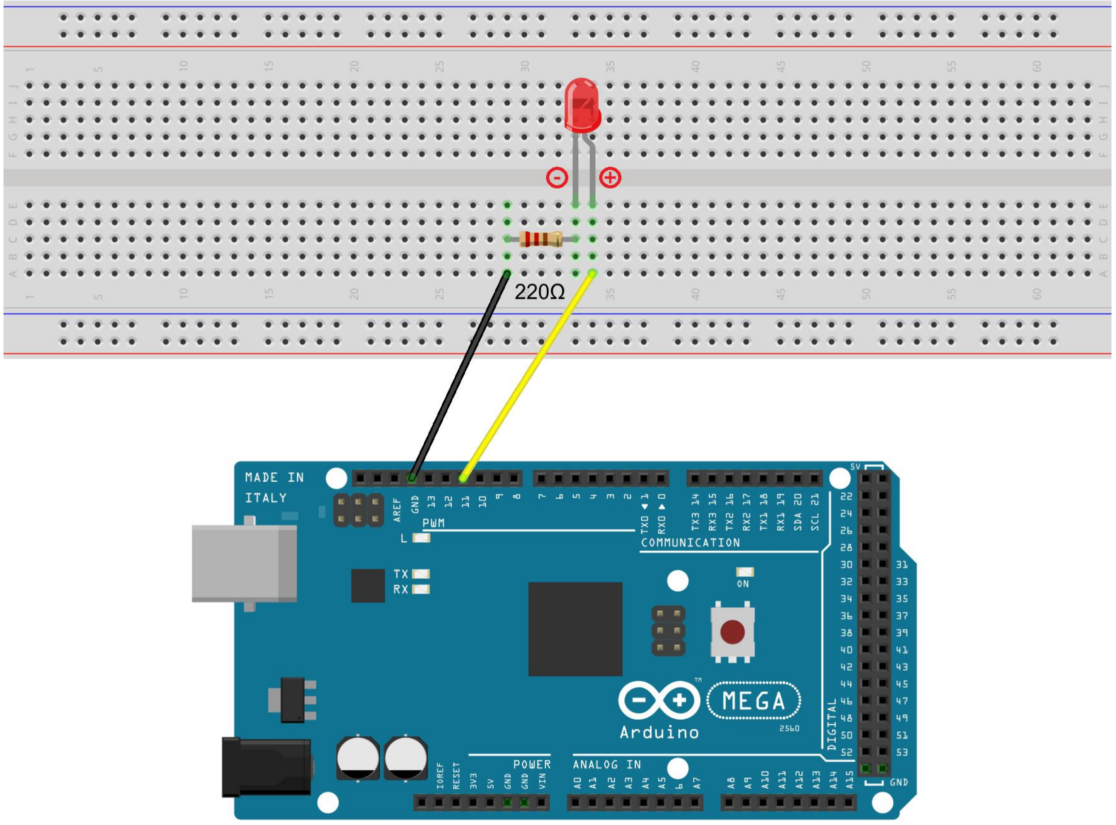
Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART). It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.

You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to begin().

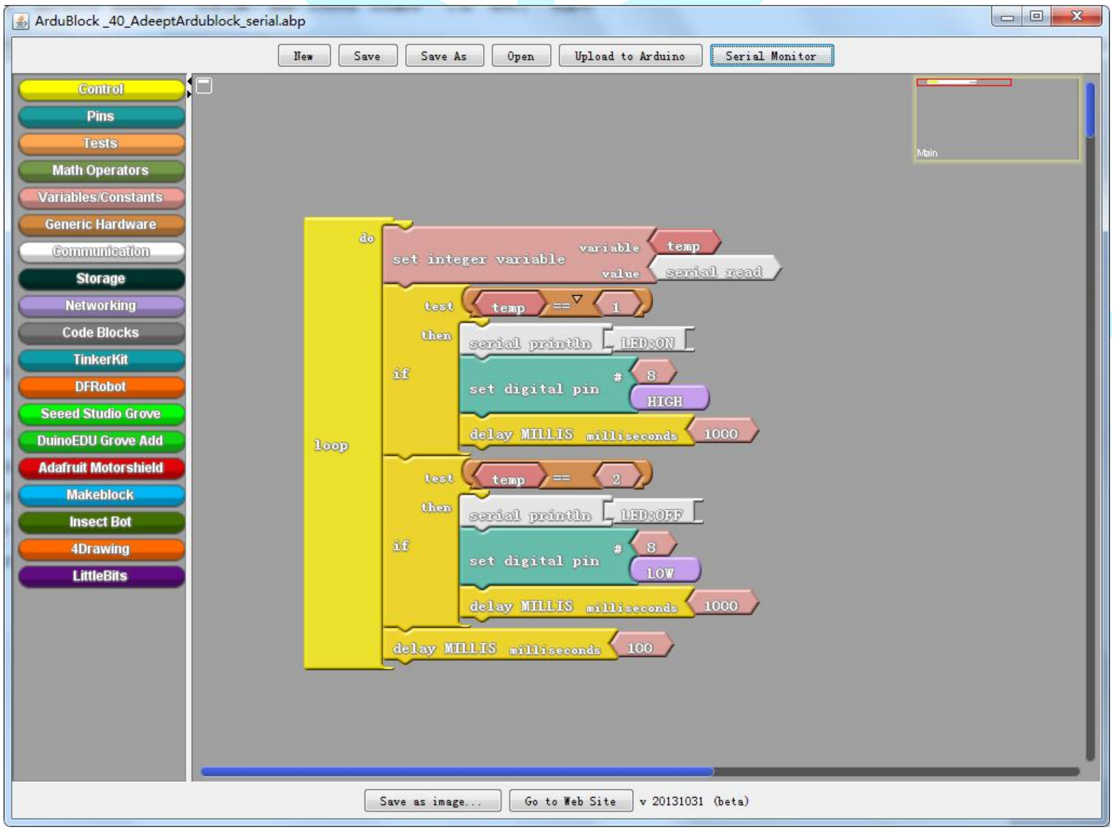
To use these pins to communicate with your personal computer, you will need an additional USB-to-serial adaptor, as they are not connected to the MEGA 2560's USB-to-serial adaptor. To use them to communicate with an external TTL serial device, connect the TX pin to your device's RX pin, the RX to your device's TX pin, and the ground of your MEGA 2560 to your device's ground. (Don't connect these pins directly to an RS232 serial port; they operate at +/- 12V and can damage your Arduino board.)

Procedures

1. Build the circuit

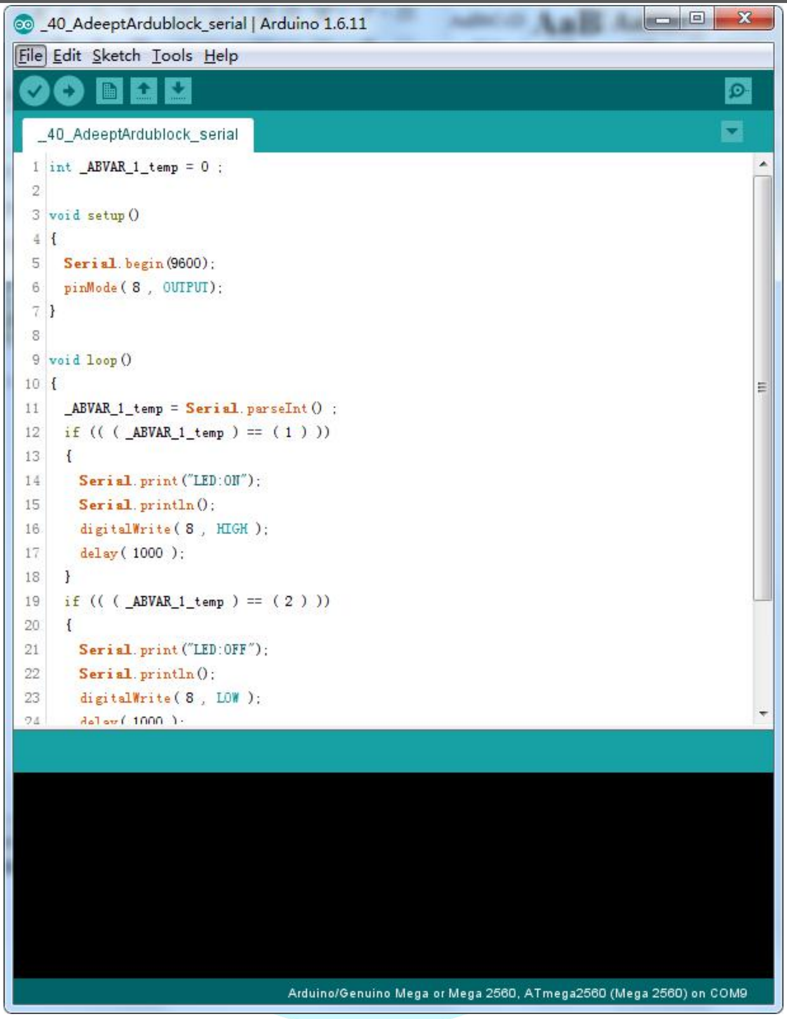


2. Program



```
loop
do
  set integer variable   variable temp
                         value    serial read
  if
    test  temp == 1
    then  serial println  LED:ON
          set digital pin  # 8
                           HIGH
          delay MILLIS milliseconds 1000
  if
    test  temp == 2
    then  serial println  LED:OFF
          set digital pin  # 8
                           LOW
          delay MILLIS milliseconds 1000
  delay MILLIS milliseconds 100
```

3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:

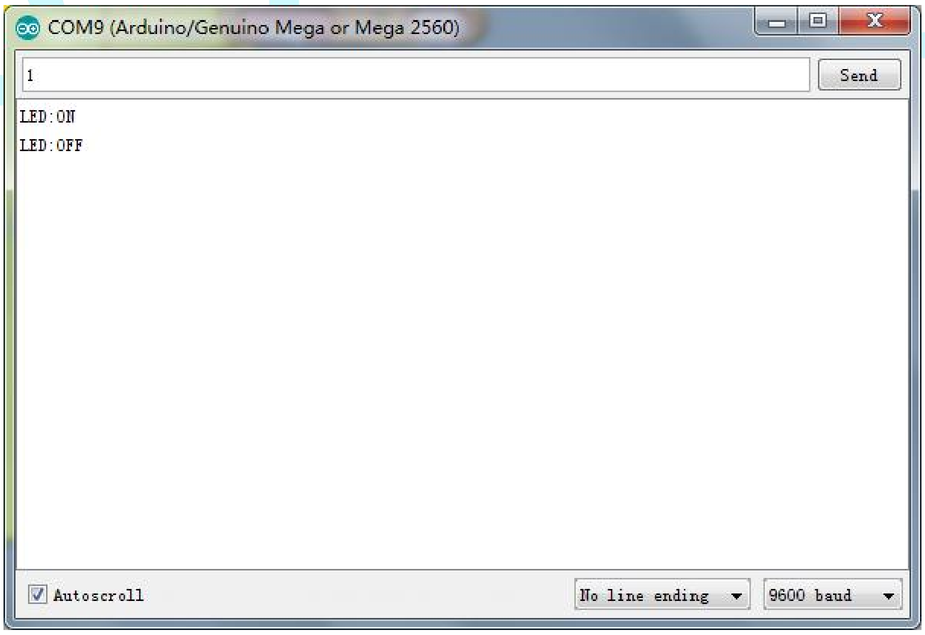

_40_AdeeptArdublock_serial | Arduino 1.6.11
File Edit Sketch Tools Help
_40_AdeeptArdublock_serial

```
int _ABVAR_1_temp = 0 ;

void setup()
{
  Serial.begin(9600);
  pinMode( 8 , OUTPUT);
}

void loop()
{
  _ABVAR_1_temp = Serial.parseInt() ;
  if (( ( _ABVAR_1_temp ) == ( 1 ) ))
  {
    Serial.print("LED:ON");
    Serial.println();
    digitalWrite( 8 , HIGH );
    delay( 1000 );
  }
  if (( ( _ABVAR_1_temp ) == ( 2 ) ))
  {
    Serial.print("LED:OFF");
    Serial.println();
    digitalWrite( 8 , LOW );
    delay( 1000 );
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

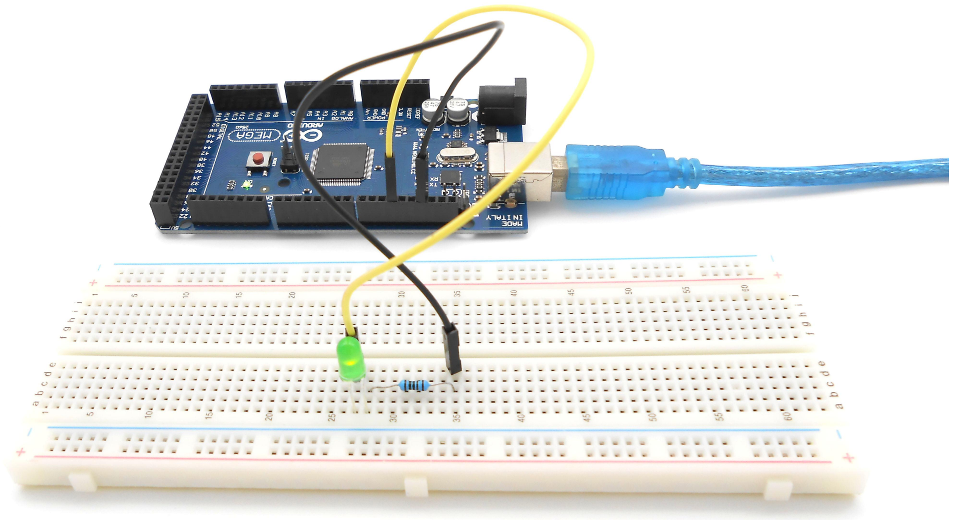
Open the port monitor, and then select the appropriate baud rate according to the program.

Now, if you send a character'1'or'2'on the serial monitor, the state of LED will be lit or gone out.



```
1
LED:ON
LED:OFF
```

☑ Autoscroll   No line ending   9600 baud



Summary

Through this lesson, you should have understood that the computer can send data to Arduino MEGA 2560 via the serial port, and then control the state of LED. I hope you can use your head to make more interesting things based on this lesson.

Adeept

Lesson 41 Adeept ardublock LED flowing lights

Overview

In the first class, we have learned how to make an LED blink by programming the Arduino 2560. Today, we will use the Arduino to control 8 LEDs, so that 8 LEDs showing the result of flowing.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 8* LED
- 8* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

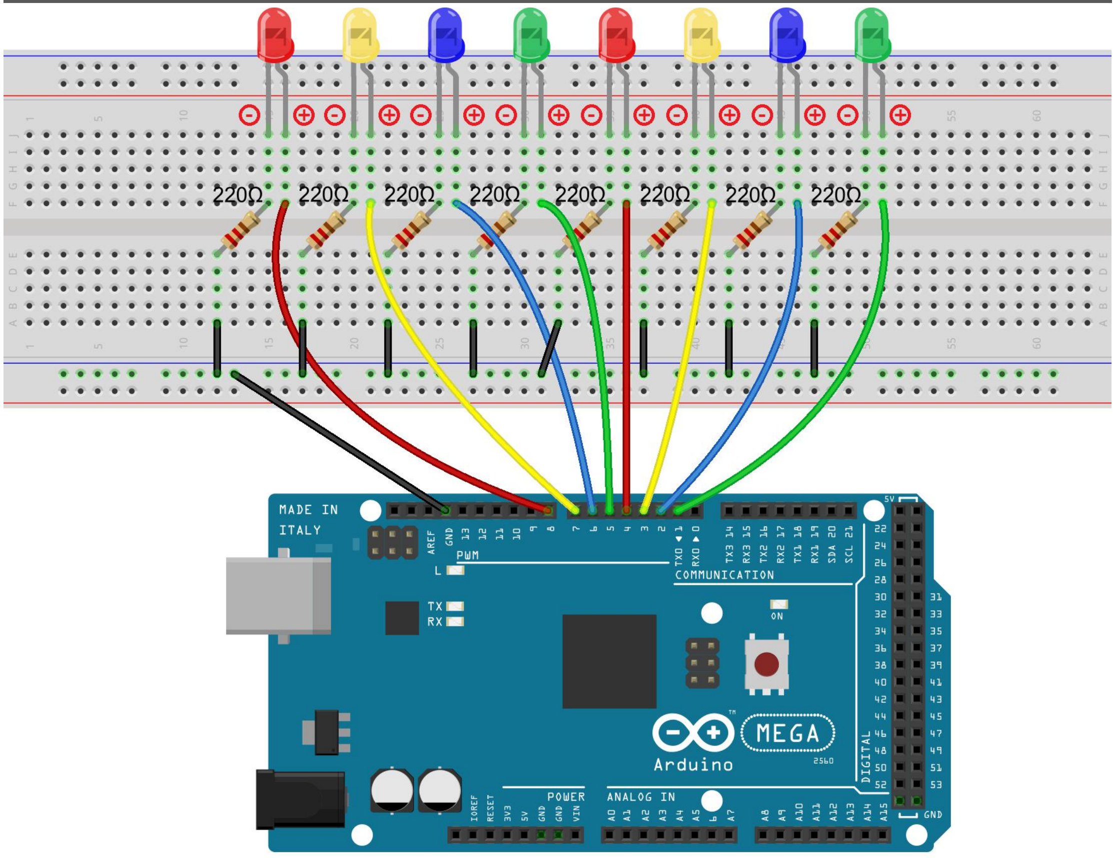
Principle

The principle of this experiment is very simple. It is very similar with the 36 class.

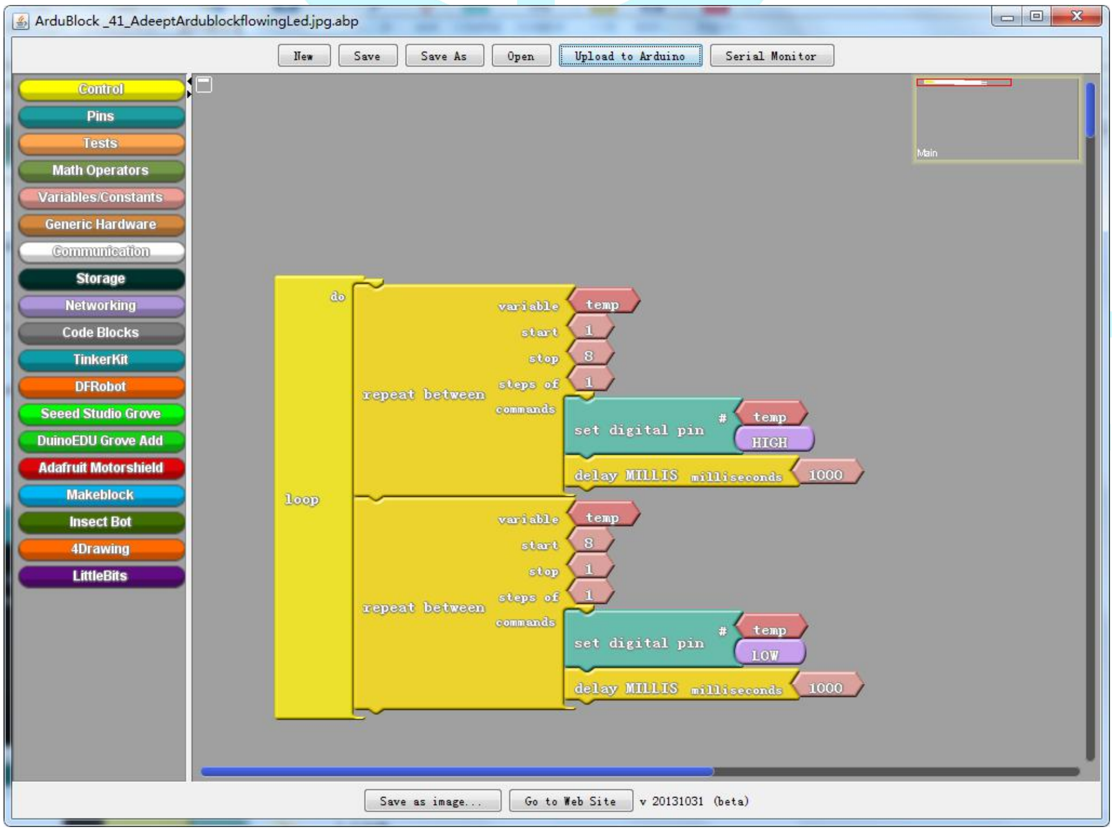
Procedures

1. Build the circuit

Adeept

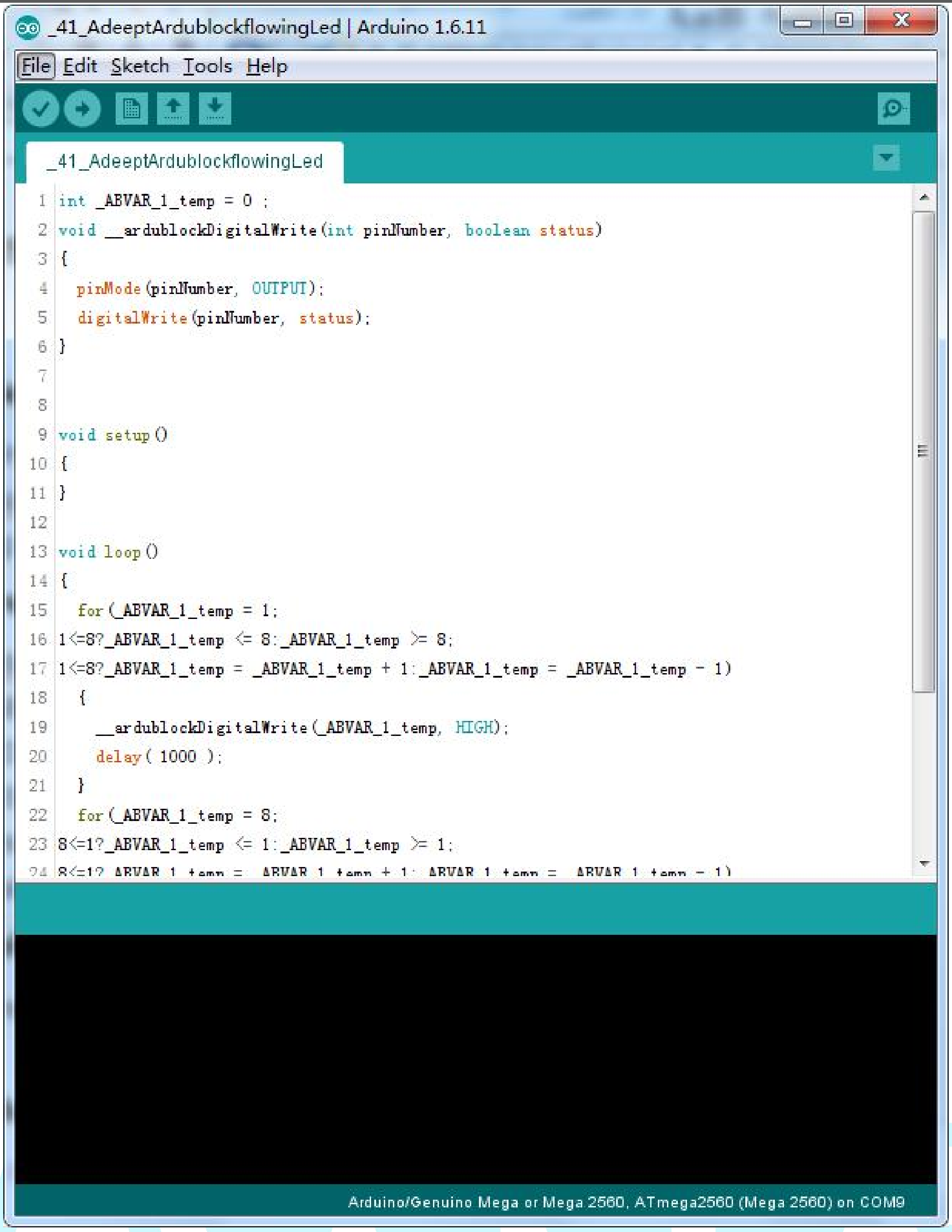


2. Program



```
loop
  do
    repeat between
      variable temp
      start 1
      stop 8
      steps of 1
      commands
        set digital pin # temp
                          HIGH
        delay MILLIS milliseconds 1000
    repeat between
      variable temp
      start 8
      stop 1
      steps of 1
      commands
        set digital pin # temp
                          LOW
        delay MILLIS milliseconds 1000
```

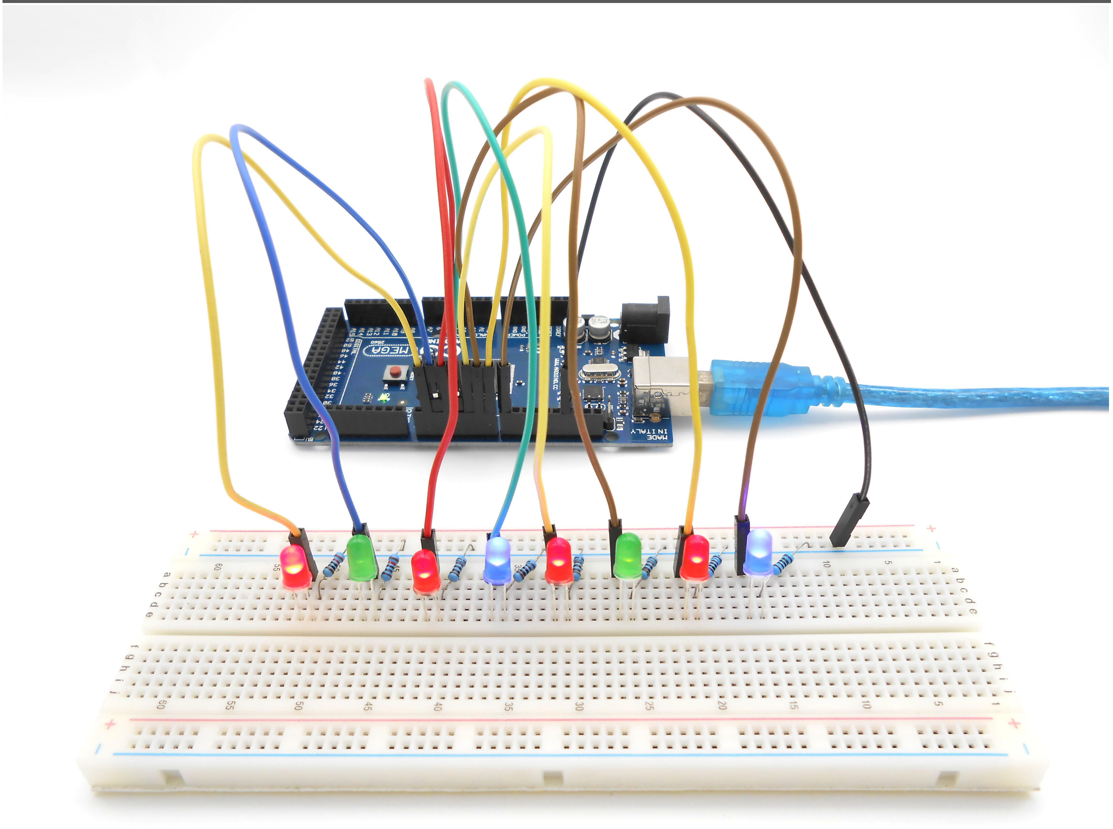
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_41_AdeeptArdublockflowingLed | Arduino 1.6.11
File Edit Sketch Tools Help
_41_AdeeptArdublockflowingLed

```
int _ABVAR_1_temp = 0 ;
void __ardublockDigitalWrite(int pinNumber, boolean status)
{
  pinMode(pinNumber, OUTPUT);
  digitalWrite(pinNumber, status);
}


void setup()
{
}

void loop()
{
  for (_ABVAR_1_temp = 1;
1<=8?_ABVAR_1_temp <= 8:_ABVAR_1_temp >= 8;
1<=8?_ABVAR_1_temp = _ABVAR_1_temp + 1:_ABVAR_1_temp = _ABVAR_1_temp - 1)
  {
    __ardublockDigitalWrite(_ABVAR_1_temp, HIGH);
    delay( 1000 );
  }
  for (_ABVAR_1_temp = 8;
8<=1?_ABVAR_1_temp <= 1:_ABVAR_1_temp >= 1;
8<=1?_ABVAR_1_temp = _ABVAR_1_temp + 1:_ABVAR_1_temp = _ABVAR_1_temp - 1)
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Now, you should see 8 LEDs are lit in sequence from the right green one to the left, next from the left to the right one. And then repeat the above phenomenon.



Summary

Through this simple and fun experiment, we have learned more skilled programming about the Arduino. In addition, you can also modify the circuit and code we provided to achieve even more dazzling effect.

Adeept

Lesson 42 Adeept ardublock slide potentiometer module

Introduction

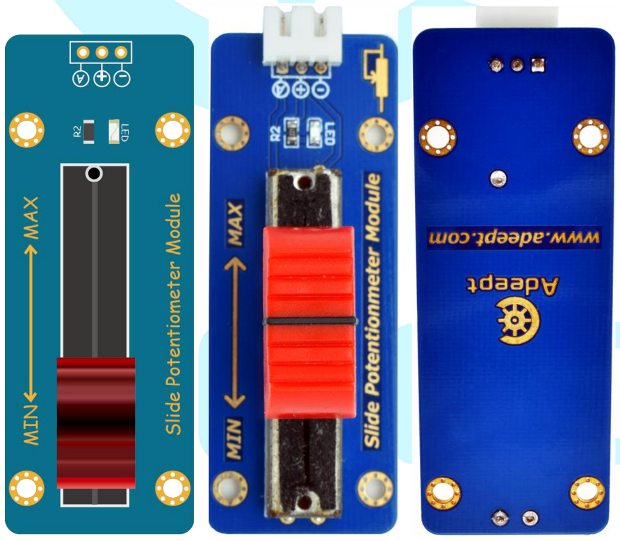
The similarity of the Slide Potentiometer Module and Potentiometer Module lies in: holding three terminals; changing the resistance between the changeable terminal and one end by changing the position of the slider. When the difference is: the Slide Potentiometer usually has a larger power (and size) and can be used directly as a load or connected in serial in the circuit of the load for current limiting. The potentiometer has a smaller power and size, and generally used for voltage sampling in signal circuit.

Components

- 1 * Arduino MEGA 2560
- 1 * Slide Potentiometer Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

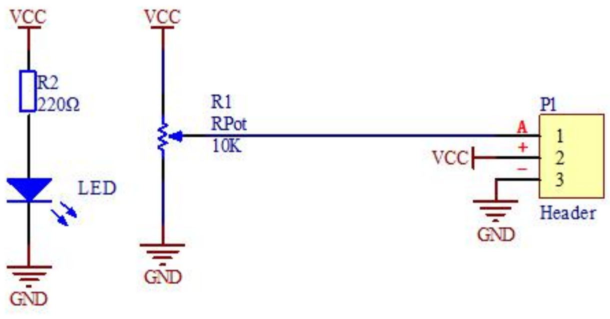
The Fritzing image:



Pin definition:

A	Analog output
+	VCC
-	GND

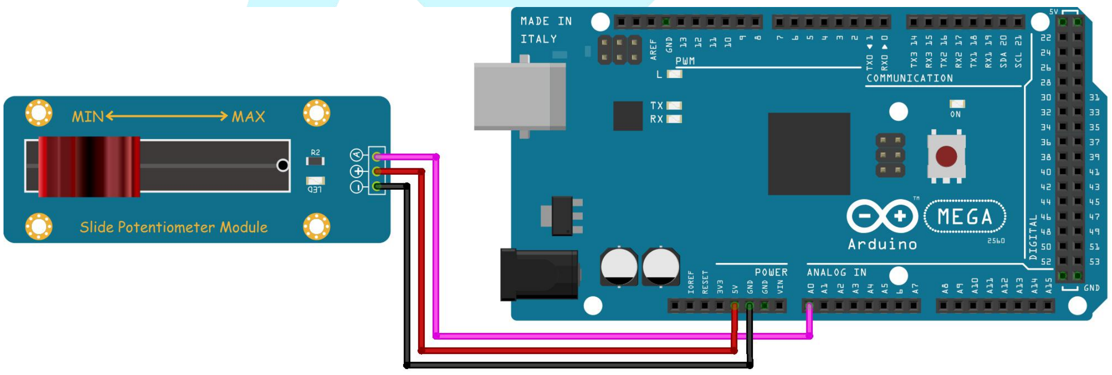
The schematic diagrams:



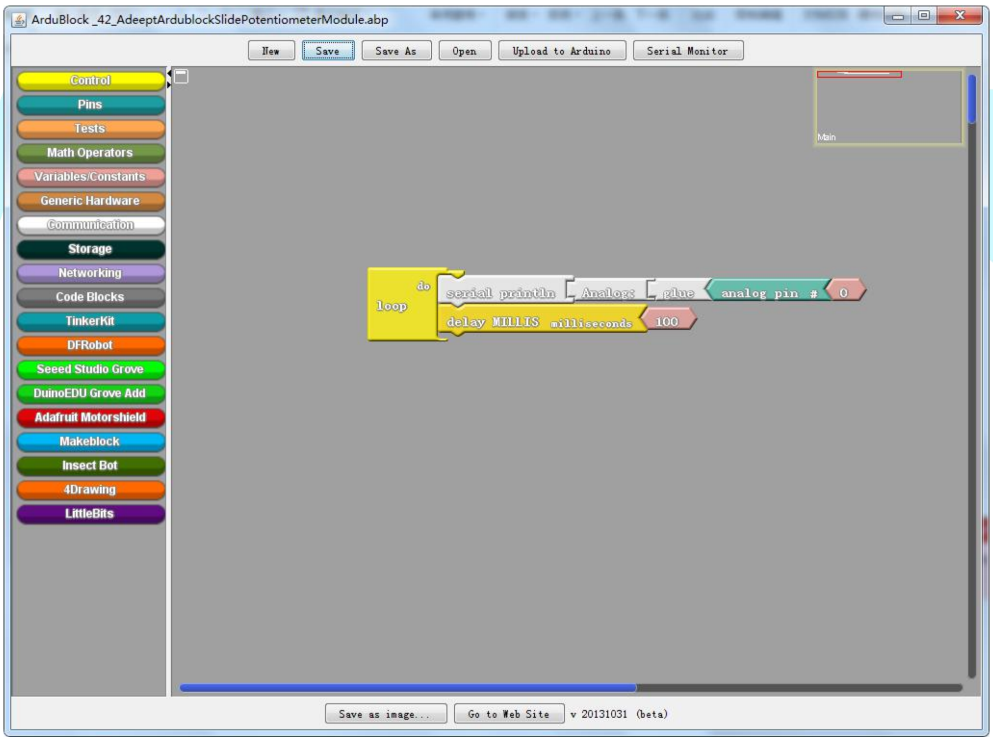
This experiment programs the Arduino board and collects analog quantities output from the Slide Potentiometer module via pin A0 of the board, and converts them into digital ones and display the value on the computer by serial port.

Experimental Procedures

Step 1: Build the circuit

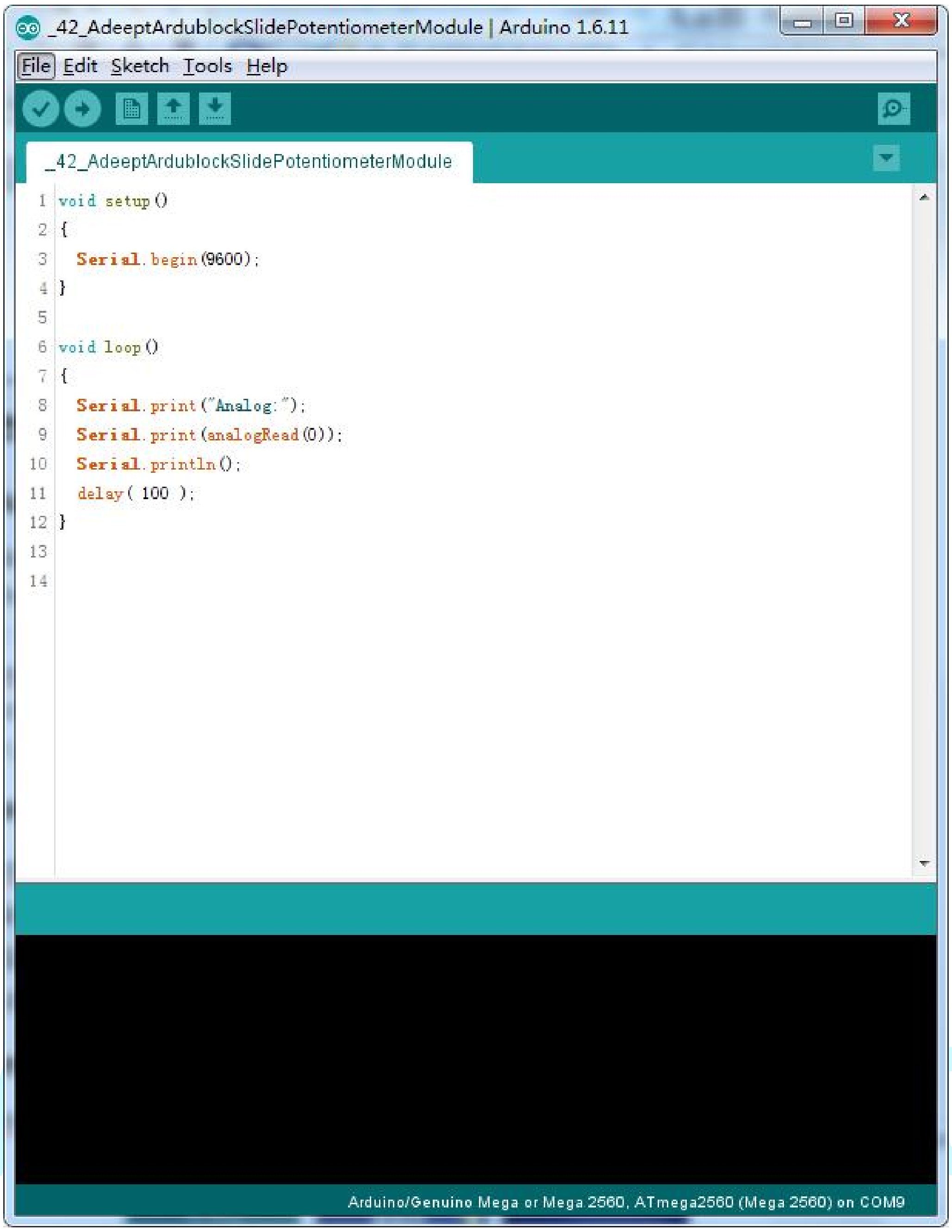


Step 2: Program



```
loop do
  serial println  Analog:  glue  analog pin # 0
  delay MILLIS milliseconds 100
```

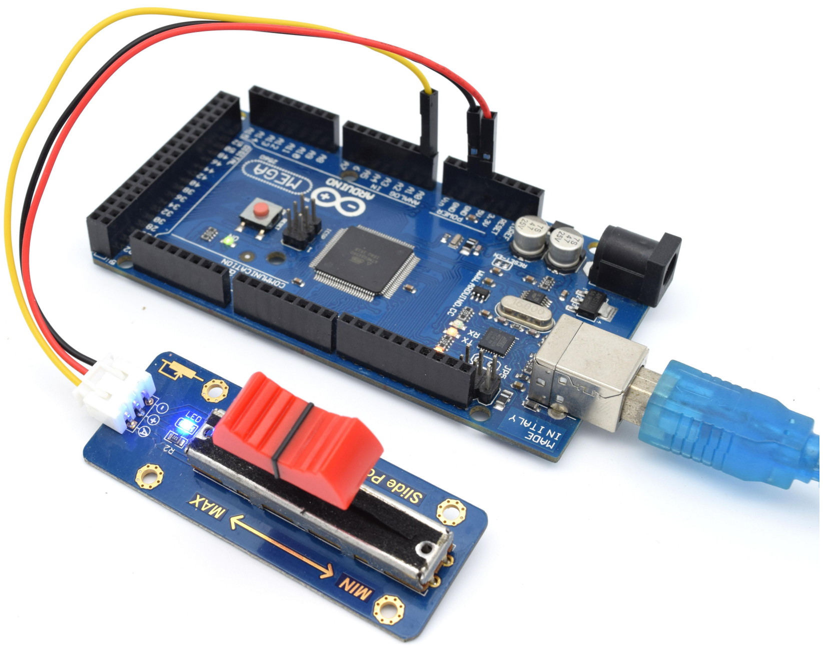
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:

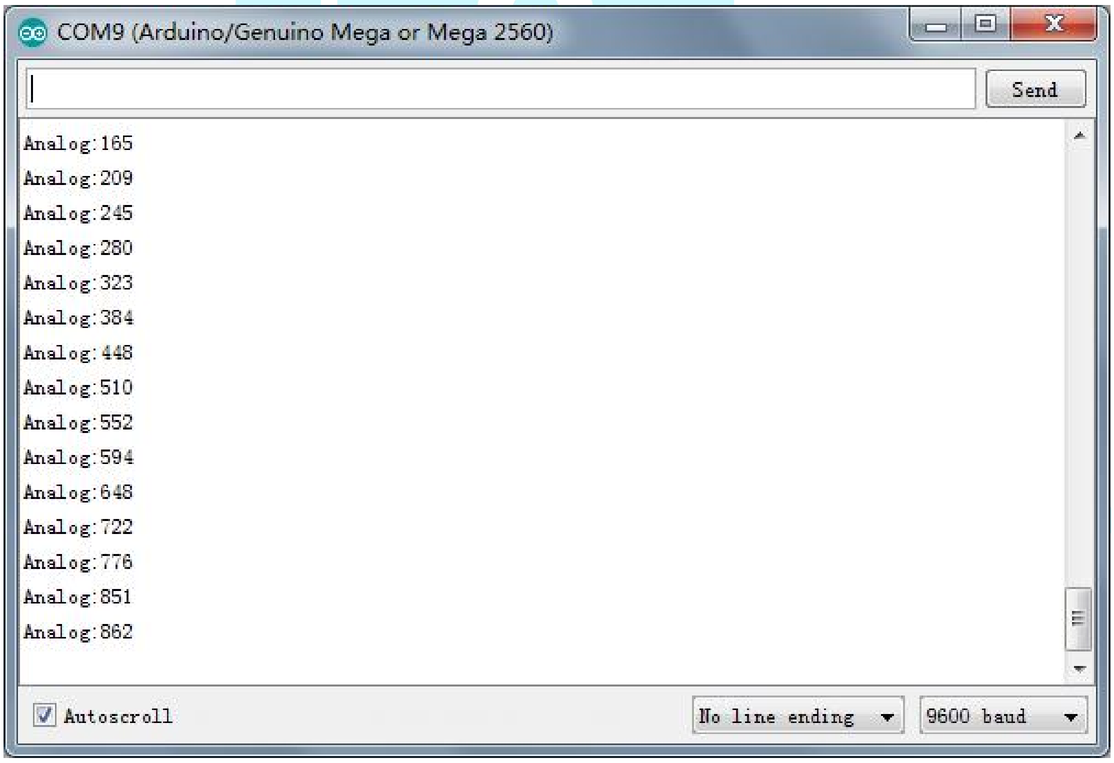

_42_AdeeptArdublockSlidePotentiometerModule | Arduino 1.6.11
File Edit Sketch Tools Help
_42_AdeeptArdublockSlidePotentiometerModule

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.print("Analog:");
  Serial.print(analogRead(0));
  Serial.println();
  delay( 100 );
}
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Open the Serial Monitor in Arduino IDE. Move the slide of the Slide Potentiometer module. Then the value output by port A of the module will be displayed on the Serial Monitor. Slide it toward MIN and the value on the window will decrease; slide it toward MAX, it will increase.





```
Analog:165
Analog:209
Analog:245
Analog:280
Analog:323
Analog:384
Analog:448
Analog:510
Analog:552
Analog:594
Analog:648
Analog:722
Analog:776
Analog:851
Analog:862
```

[✓] Autoscroll   No line ending   9600 baud

Lesson 43 Adeept ardublock LED bar graph display

Overview

In this lesson, we will learn how to control a LED bar graph by programming the Arduino MEGA 2560.

Requirement

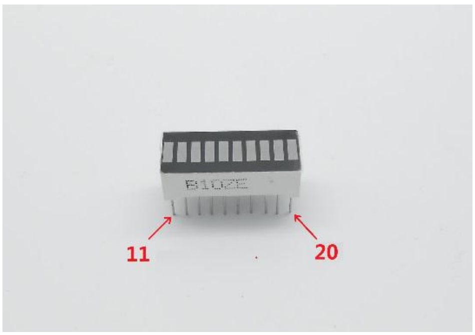
- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* Slide Potentiometer Module
- 10* 220Ω Resistor
- 1* LED Bar Graph
- 1* Breadboard
- Several Jumper Wires

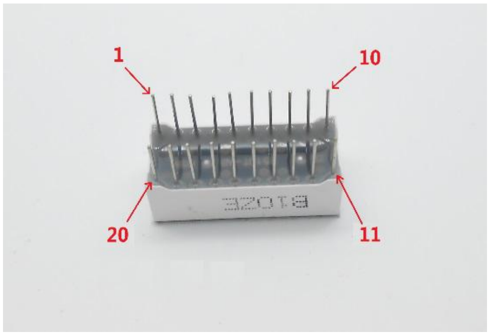
Principle

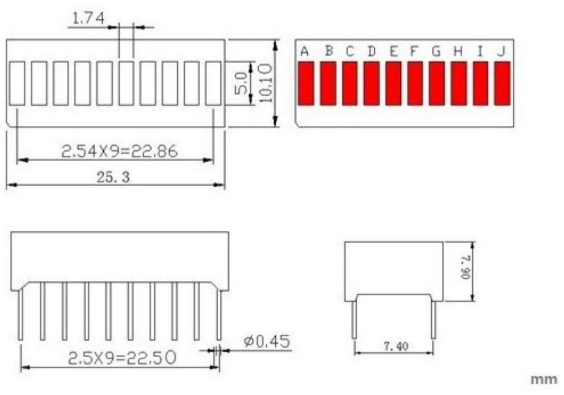
The bar graph - a series of LEDs in a line, such as you see on an audio display is a common hardware display for analog sensors. It's made up of a series of LEDs in a row, an analog input like a potentiometer, and a little code in between. You can buy multi-LED bar graph displays fairly cheaply. This tutorial demonstrates how to control a series of LEDs in a row, but can be applied to any series of digital outputs.

This tutorial borrows from the For Loop and Arrays tutorial as well as the Analog Input tutorial.

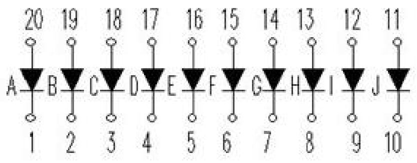
The sketch works like this: first you read the input. You map the input value to the output range, in this case ten LEDs. Then you set up a ***for*** loop to iterate over the outputs. If the output's number in the series is lower than the mapped input range, you turn it on. If not, you turn it off.







The internal schematic diagram for the LED bar graph is shown below:

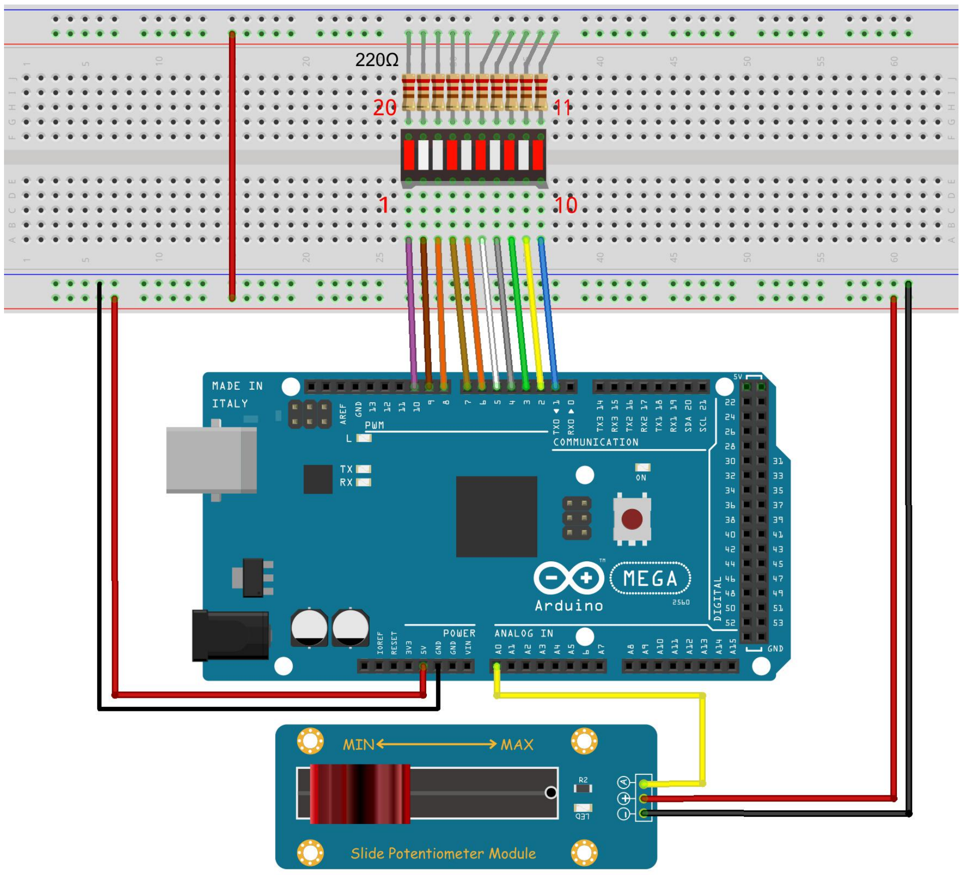


A potentiometer, informally a pot, is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat.

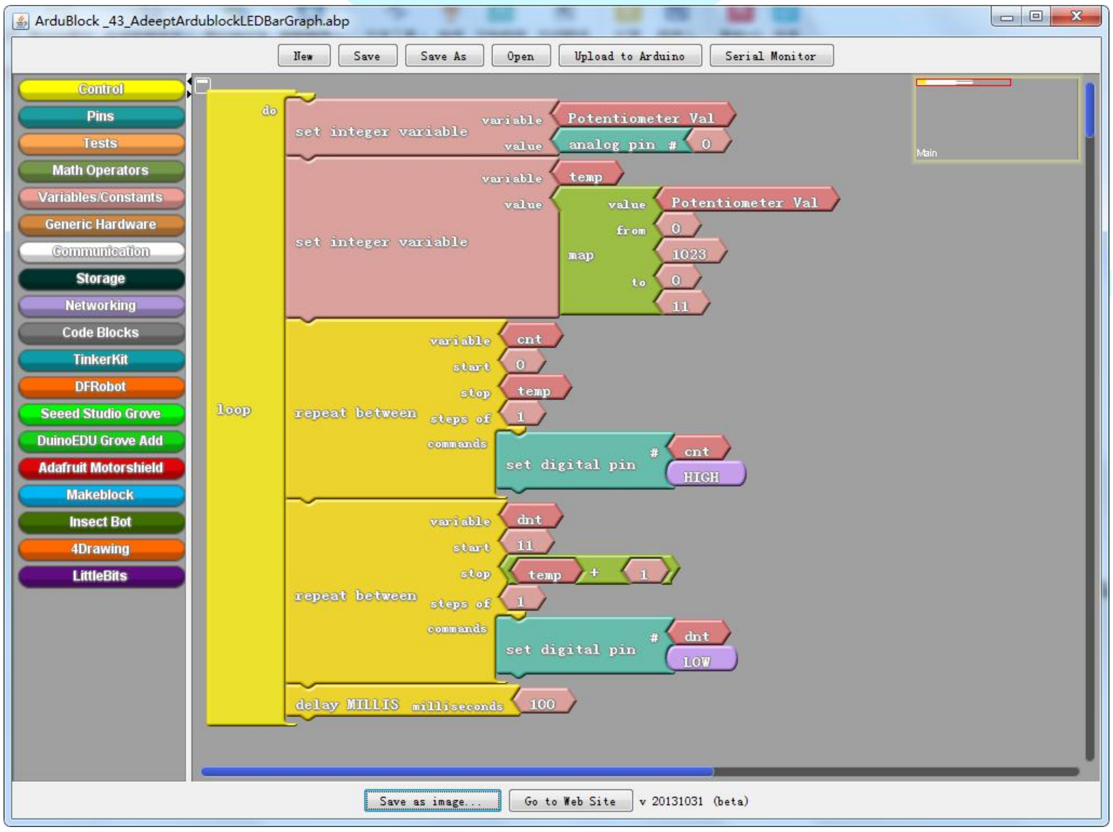
Procedures

1. Build the circuit

Adeept

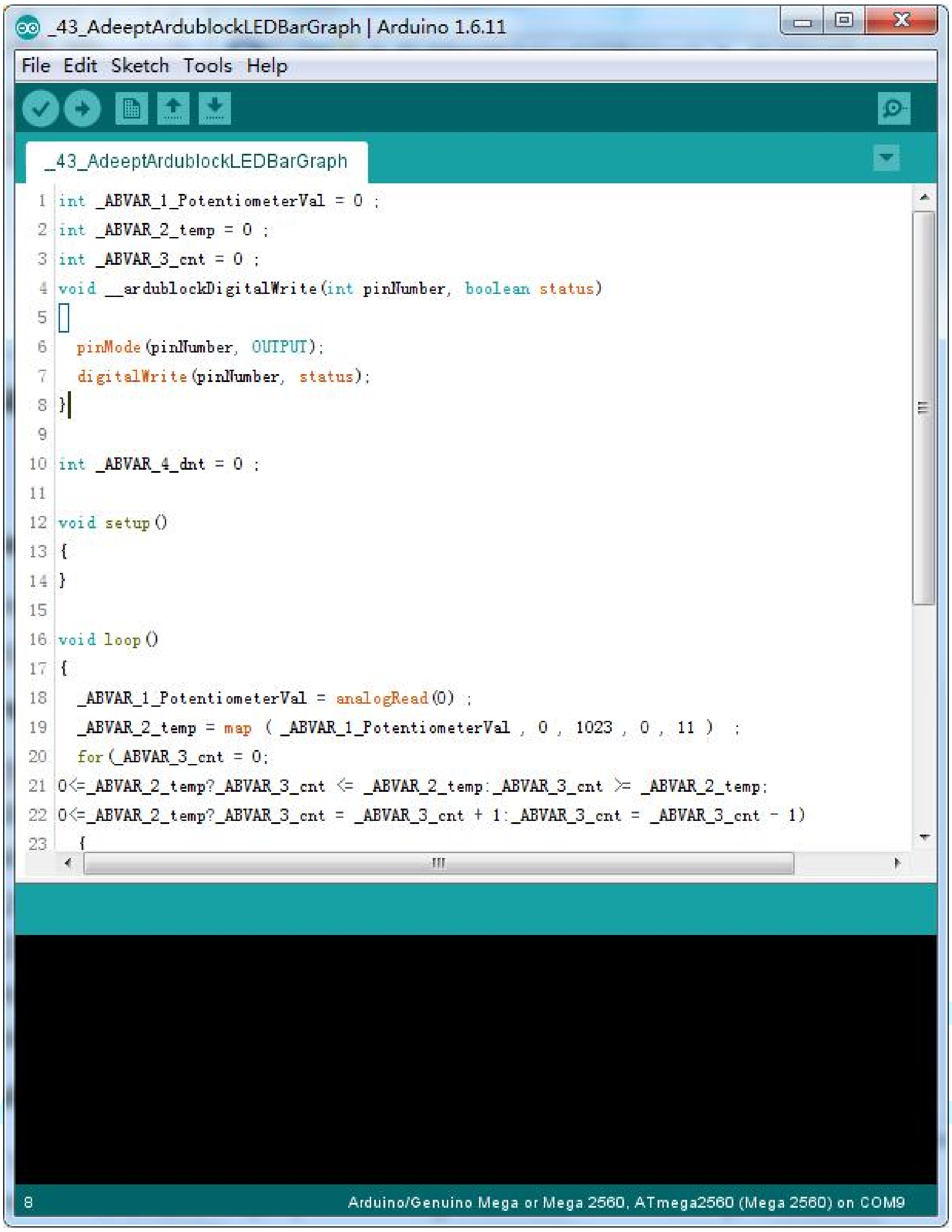


2. Program



```
loop
do
  set integer variable  variable  Potentiometer Val
                        value     analog pin # 0
  set integer variable  variable  temp
                        value     map  value  Potentiometer Val
                                       from   0
                                              1023
                                       to     0
                                              11
  repeat between  variable  cnt
                  start     0
                  stop      temp
                  steps of  1
                  commands  set digital pin  #  cnt
                                                HIGH
  repeat between  variable  dnt
                  start     11
                  stop      temp + 1
                  steps of  1
                  commands  set digital pin  #  dnt
                                                LOW
  delay MILLIS milliseconds  100
```

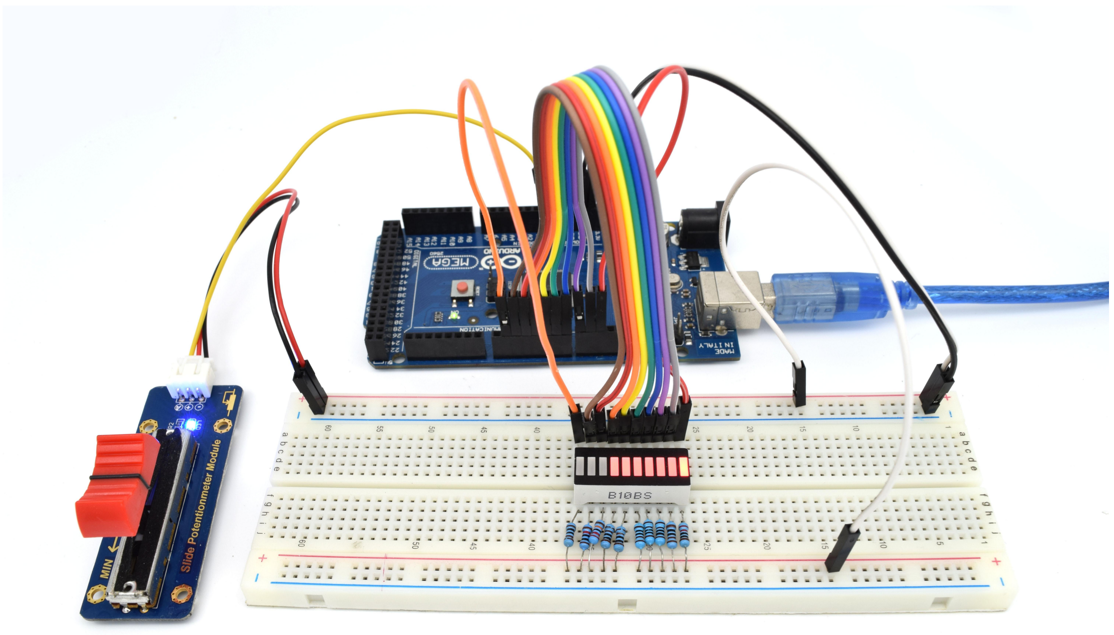
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_43_AdeeptArdublockLEDBarGraph | Arduino 1.6.11
File Edit Sketch Tools Help
_43_AdeeptArdublockLEDBarGraph

```
int _ABVAR_1_PotentiometerVal = 0 ;
int _ABVAR_2_temp = 0 ;
int _ABVAR_3_cnt = 0 ;
void __ardublockDigitalWrite(int pinNumber, boolean status)
{
  pinMode(pinNumber, OUTPUT);
  digitalWrite(pinNumber, status);
}

int _ABVAR_4_dnt = 0 ;

void setup()
{
}

void loop()
{
  _ABVAR_1_PotentiometerVal = analogRead(0) ;
  _ABVAR_2_temp = map ( _ABVAR_1_PotentiometerVal , 0 , 1023 , 0 , 11 )  ;
  for (_ABVAR_3_cnt = 0;
0<=_ABVAR_2_temp?_ABVAR_3_cnt <= _ABVAR_2_temp:_ABVAR_3_cnt >= _ABVAR_2_temp;
0<=_ABVAR_2_temp?_ABVAR_3_cnt = _ABVAR_3_cnt + 1:_ABVAR_3_cnt = _ABVAR_3_cnt - 1)
  {
```

```
8                          Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9
```

Now, when you turn the knob of the potentiometer, you will see that the number of LED in the LED bar graph will be changed.



Adeept

Lesson 44 Adeept ardublock breathing LED

Overview

In this lesson, we will learn how to program the Arduino to generate PWM signal. And use the PWM square-wave signal control an LED gradually becomes brighter and then gradually becomes dark like the animal’s breathing.

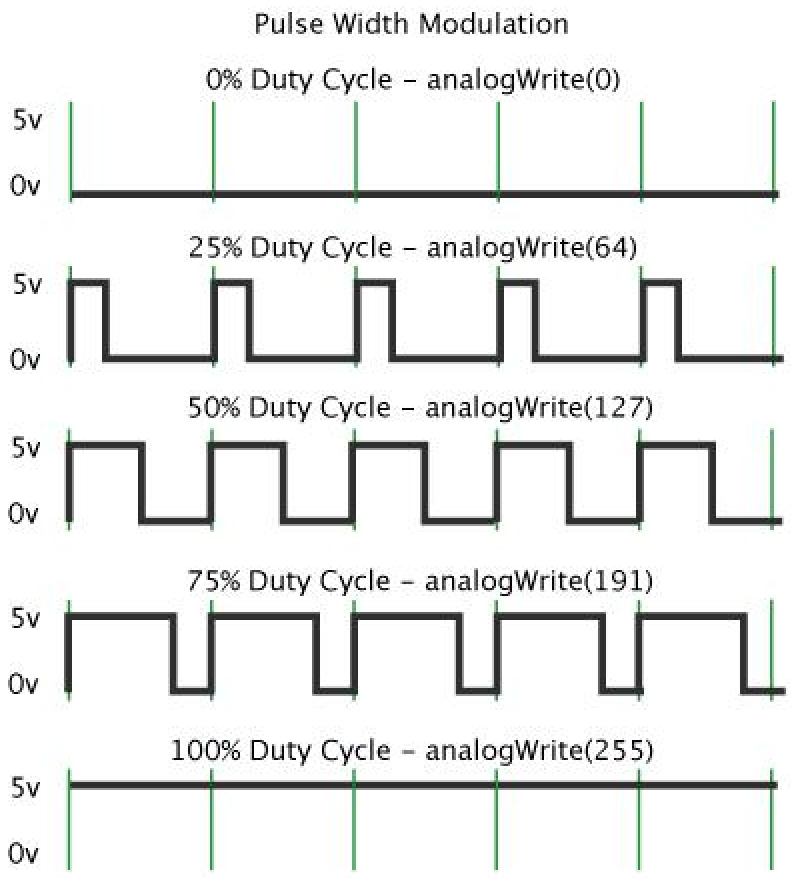
Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* LED
- 1* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

Principle

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

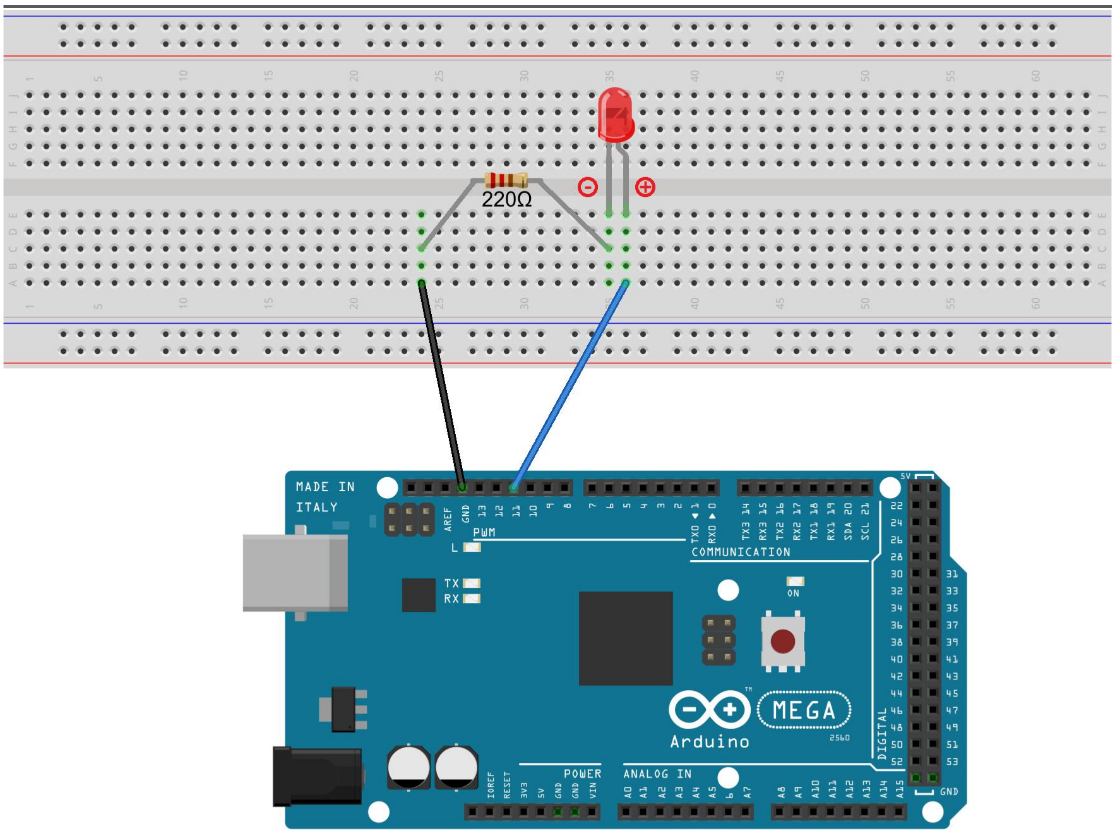
In the graphic below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to analogWrite() is on a scale of 0 - 255, such that analogWrite(255) requests a 100% duty cycle (always on), and analogWrite(127) is a 50% duty cycle (on half the time) for example.



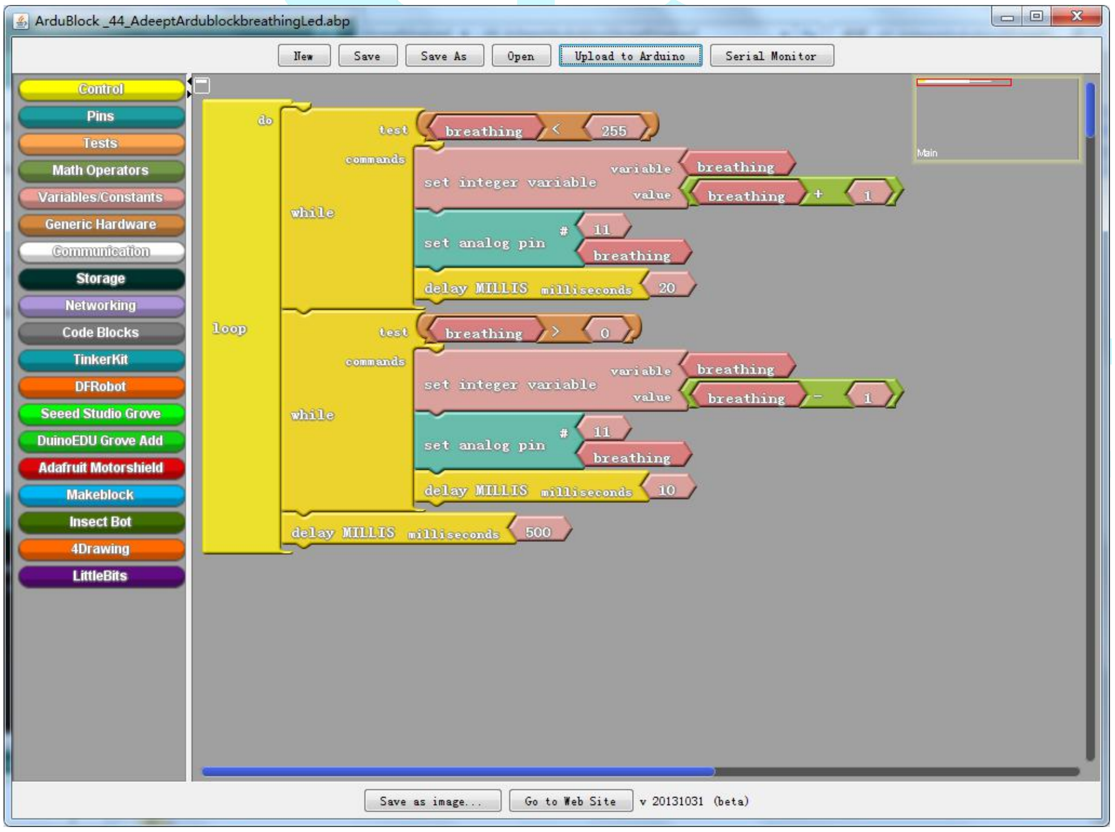
Procedures

1. Build the circuit

Adeept

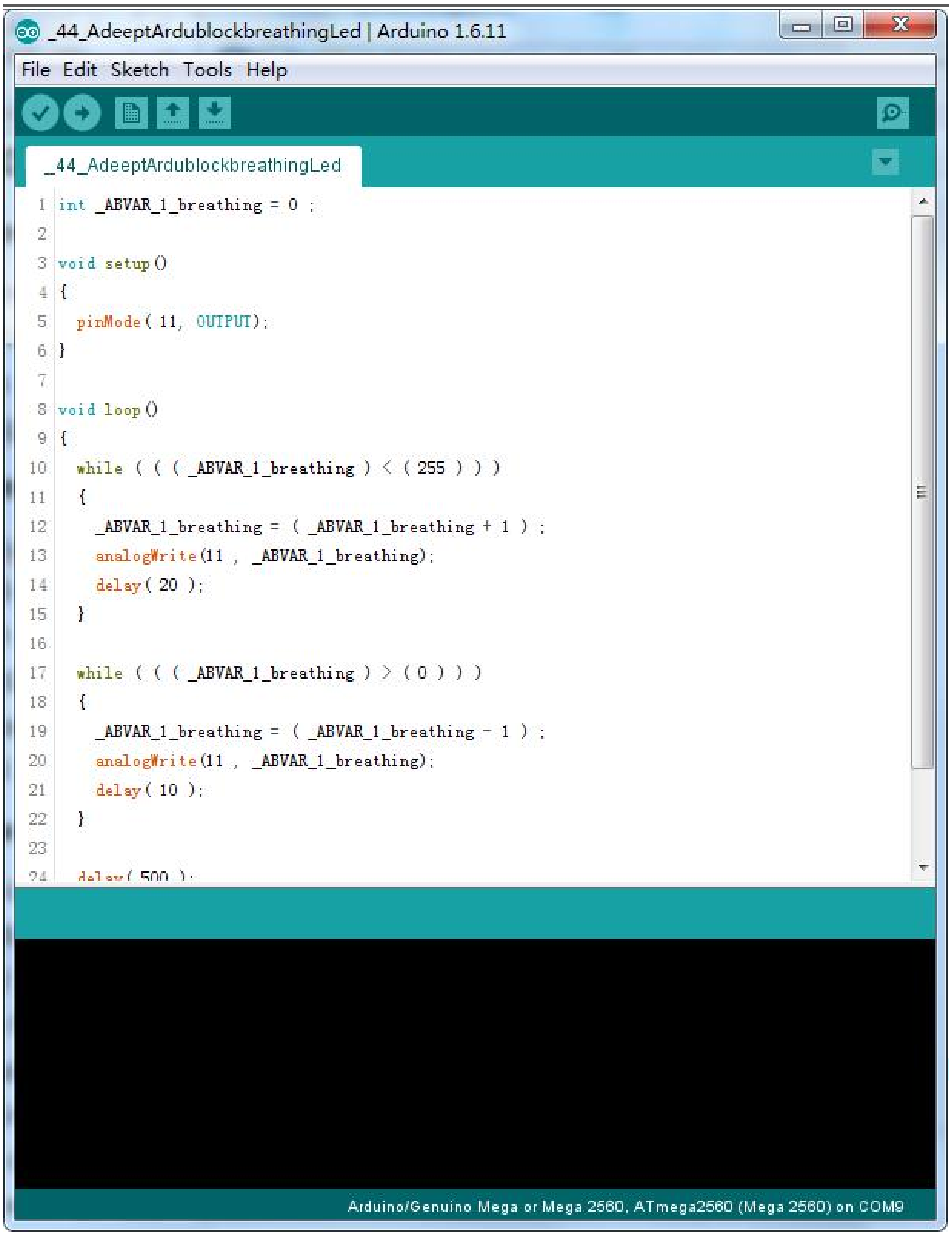


2. Program



```
loop
  do
    while  test      breathing < 255
           commands  set integer variable  variable breathing
                                           value    breathing + 1
                     set analog pin  # 11
                                     breathing
                     delay MILLIS milliseconds 20
    while  test      breathing > 0
           commands  set integer variable  variable breathing
                                           value    breathing - 1
                     set analog pin  # 11
                                     breathing
                     delay MILLIS milliseconds 10
    delay MILLIS milliseconds 500
```

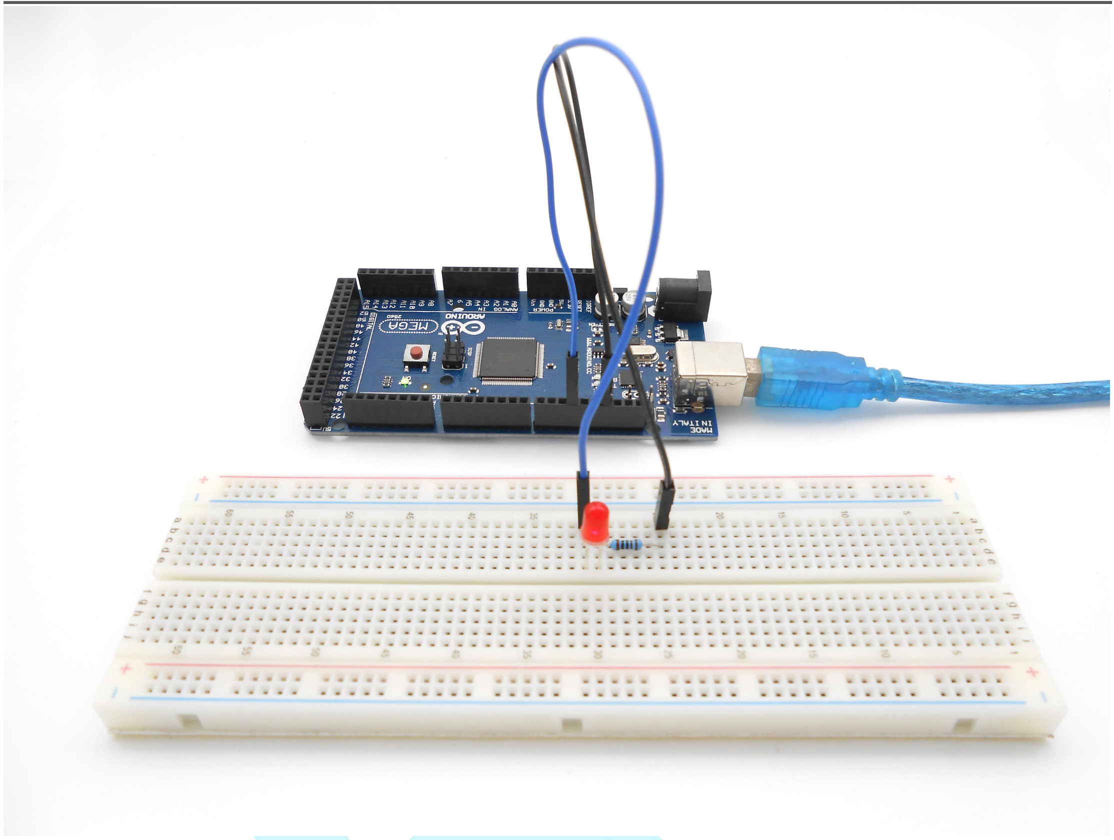
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_44_AdeeptArdublockbreathingLed | Arduino 1.6.11
File Edit Sketch Tools Help
_44_AdeeptArdublockbreathingLed

```
int _ABVAR_1_breathing = 0 ;

void setup()
{
  pinMode( 11, OUTPUT);
}

void loop()
{
  while ( ( ( _ABVAR_1_breathing ) < ( 255 ) ) )
  {
    _ABVAR_1_breathing = ( _ABVAR_1_breathing + 1 ) ;
    analogWrite(11 , _ABVAR_1_breathing);
    delay( 20 );
  }

  while ( ( ( _ABVAR_1_breathing ) > ( 0 ) ) )
  {
    _ABVAR_1_breathing = ( _ABVAR_1_breathing - 1 ) ;
    analogWrite(11 , _ABVAR_1_breathing);
    delay( 10 );
  }

  delay( 500 );
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Now, you should see the LED gradually from dark to brighter, and then from brighter to dark, continuing to repeat the process, its rhythm like the animal's breathing.



Summary

By learning this lesson, I believe that you have understood the basic principles of the PWM, and mastered the PWM programming on the Arduino platform.

Adeept

Lesson 45 Adeept ardublock controlling a RGB LED by PWM

Overview

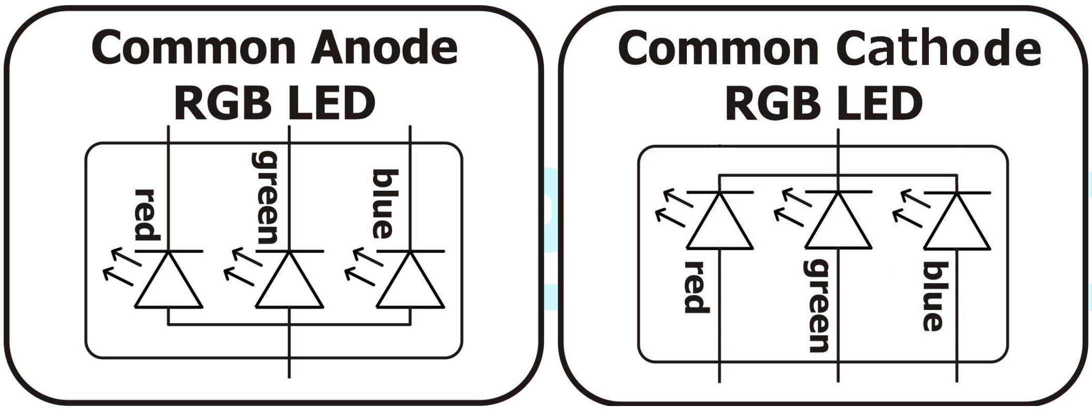
In this lesson, we will program the Arduino for RGB LED control, and make RGB LED emits a various of colors of light.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* RGB LED
- 3* 220Ω Resistor
- 1* Breadboard
- Several Jumper Wires

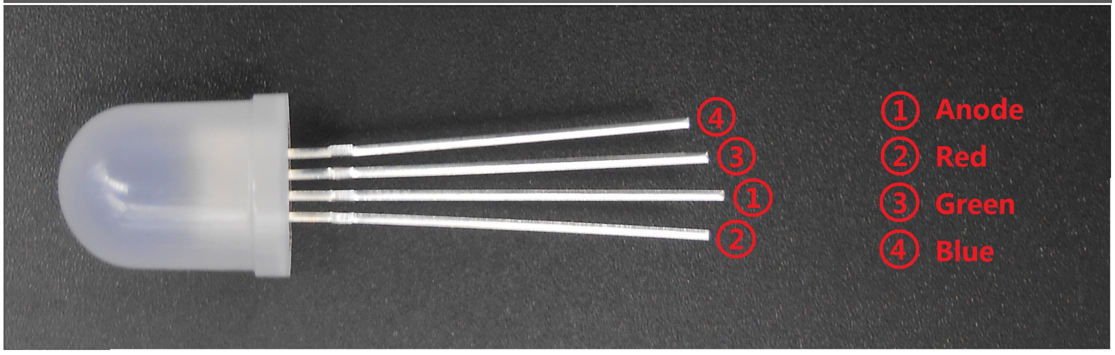
Principle

RGB LEDs consist of three LEDs. Each LED actually has one red, one green and one blue light. These three colored LEDs are capable of producing any color. Tri-color LEDs with red, green, and blue emitters, in general using a four-wire connection with one common lead (anode or cathode). These LEDs can have either common anode or common cathode leads.



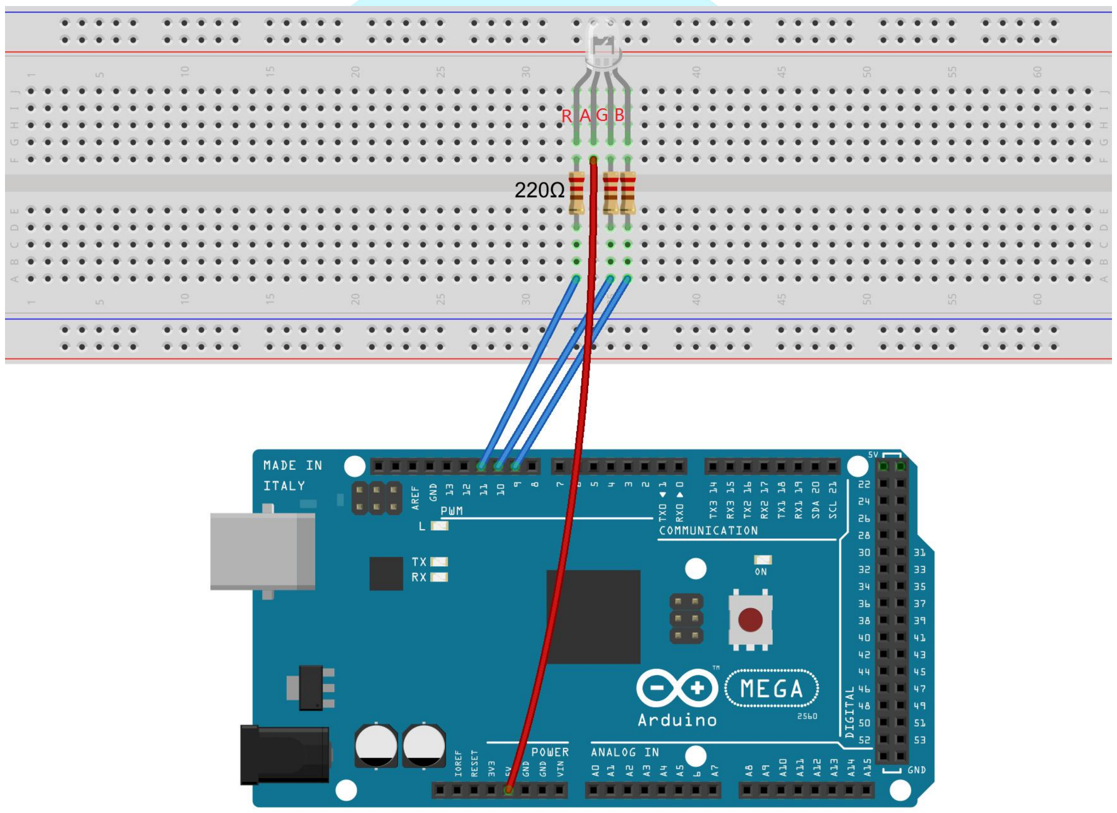
What we used in this experiment is the common anode RGB LED. The longest pin is the common anode of three LEDs. The pin is connected to the +5V pin of the Arduino, and the three remaining pins are connected to the Arduino's D9, D10, D11 pins through a current limiting resistor.

In this way, we can control the color of RGB LED by 3-channel PWM signal.

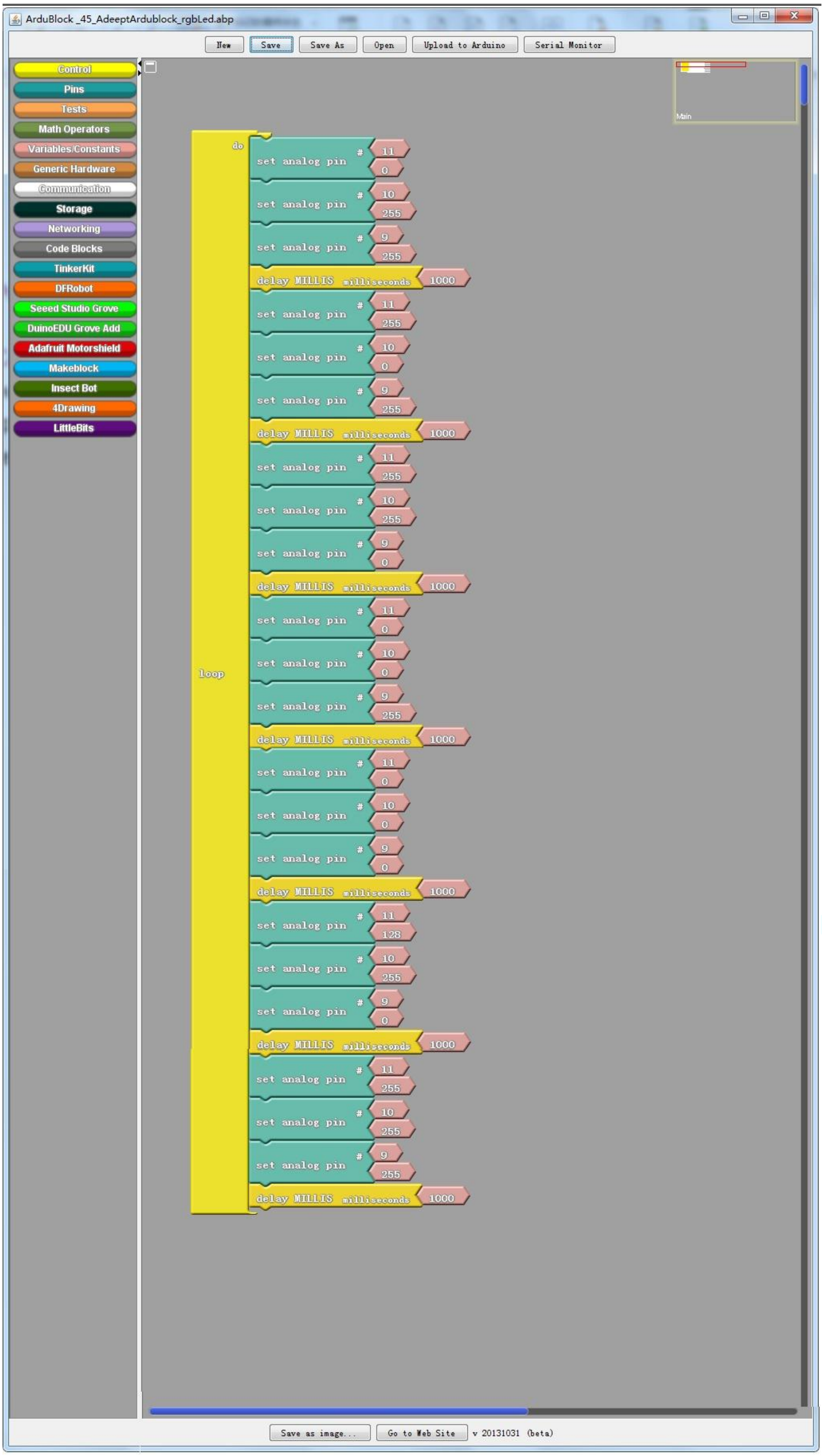


Procedures

1. Build the circuit



2. Program

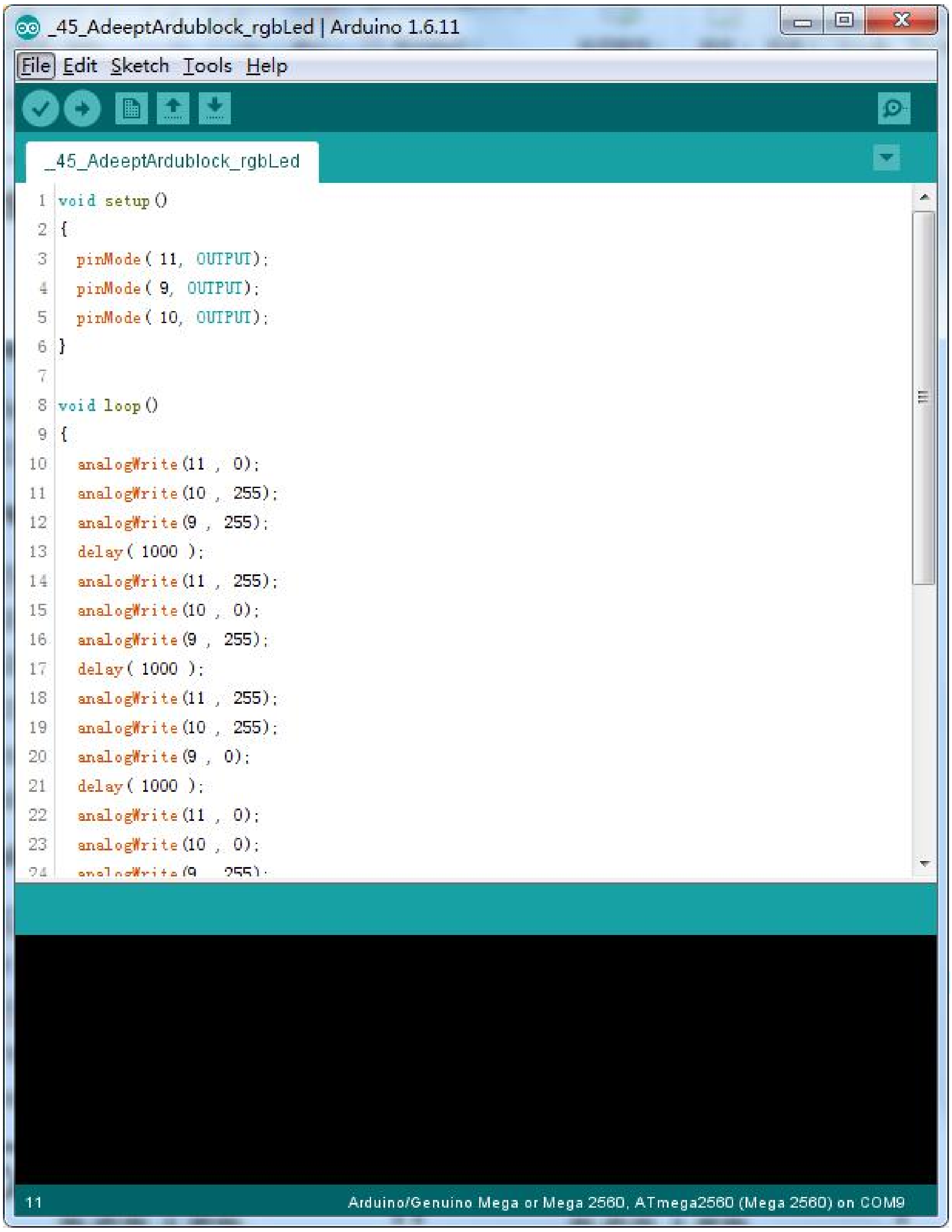


ArduBlock _45_AdeeptArdublock_rgbLed.abp

```
loop
do
  set analog pin # 11   0
  set analog pin # 10   255
  set analog pin # 9    255
  delay MILLIS milliseconds 1000
  set analog pin # 11   255
  set analog pin # 10   0
  set analog pin # 9    255
  delay MILLIS milliseconds 1000
  set analog pin # 11   255
  set analog pin # 10   255
  set analog pin # 9    0
  delay MILLIS milliseconds 1000
  set analog pin # 11   0
  set analog pin # 10   0
  set analog pin # 9    255
  delay MILLIS milliseconds 1000
  set analog pin # 11   0
  set analog pin # 10   0
  set analog pin # 9    0
  delay MILLIS milliseconds 1000
  set analog pin # 11   128
  set analog pin # 10   255
  set analog pin # 9    0
  delay MILLIS milliseconds 1000
  set analog pin # 11   255
  set analog pin # 10   255
  set analog pin # 9    255
  delay MILLIS milliseconds 1000
```

v 20131031 (beta)

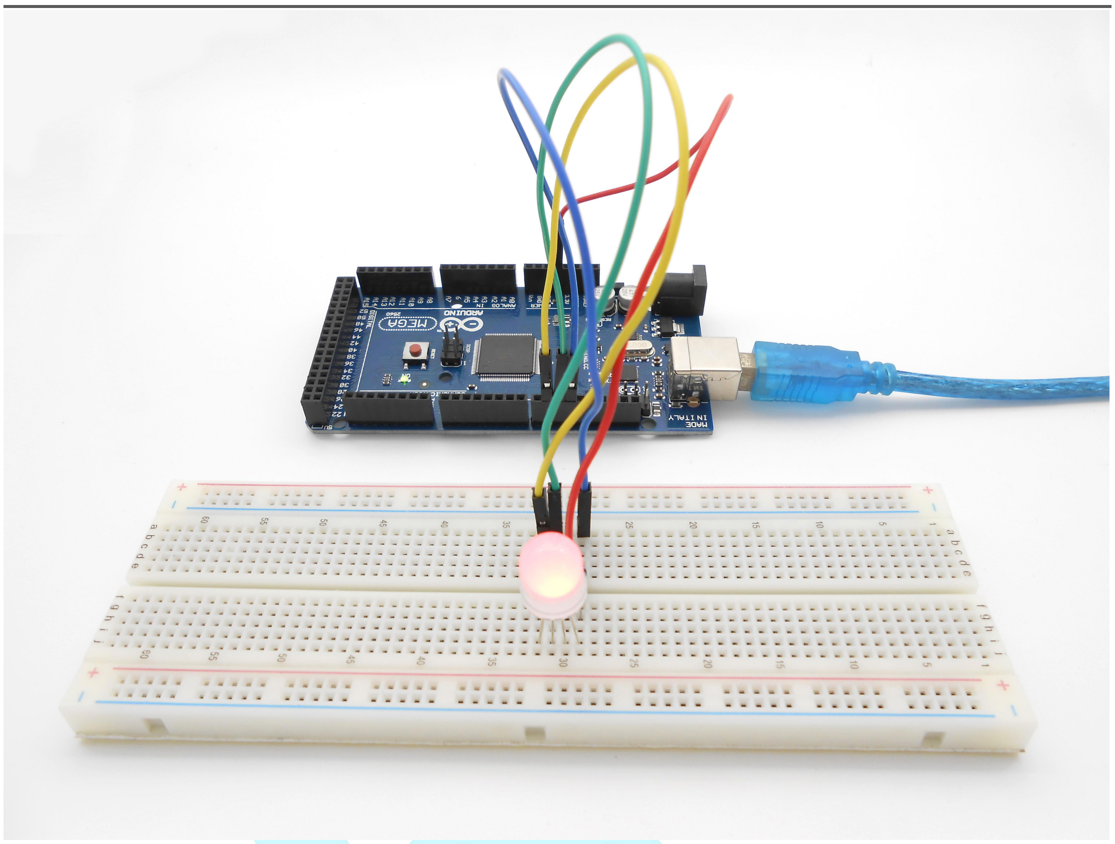
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_45_AdeeptArdublock_rgbLed | Arduino 1.6.11
File Edit Sketch Tools Help
_45_AdeeptArdublock_rgbLed

```
void setup()
{
  pinMode( 11, OUTPUT);
  pinMode( 9, OUTPUT);
  pinMode( 10, OUTPUT);
}

void loop()
{
  analogWrite(11 , 0);
  analogWrite(10 , 255);
  analogWrite(9 , 255);
  delay( 1000 );
  analogWrite(11 , 255);
  analogWrite(10 , 0);
  analogWrite(9 , 255);
  delay( 1000 );
  analogWrite(11 , 255);
  analogWrite(10 , 255);
  analogWrite(9 , 0);
  delay( 1000 );
  analogWrite(11 , 0);
  analogWrite(10 , 0);
  analogWrite(9 , 255);
```

11 Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Now, you can see the RGB LED emitting red, green, blue, yellow, white and purple light, then the RGB LED will be off, each state continues 1s, after repeating the above procedure.



Summary

By learning this lesson, I believe you have already known the principle and the programming of RGB LED. I hope you can use your imagination to achieve even more cool ideas based on this lesson.

Adeept

Lesson 46 Adeept ardublock LCD1602 display

Overview

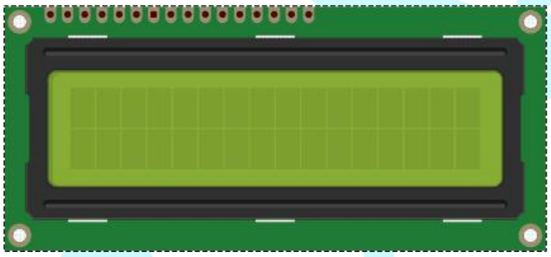
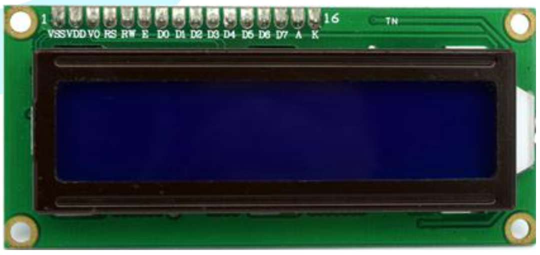
In this lesson, we will learn how to use a character display device—LCD1602 on the Arduino platform. First, we make the LCD1602 display a string "**Hello Geeks!**" , then display " *****Adeept***** " and "*www.adeept.com*" static.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* LCD1602
- 1* 10KΩ Potentiometer
- 1* Breadboard
- Several Jumper Wires

Principle

The LCD1602 image:

Pin definition:

VSS	GND
VDD	VCC
VO	Analog input
RS	Digital input
RW	Digital input
E	Digital input
D0	Digital input
D1	Digital input
D2	Digital input
D3	Digital input
D4	Digital input
D5	Digital input
D6	Digital input
D7	Digital input

A	VCC
K	GND

LCD1602 is a kind of character LCD display. The LCD has a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

● A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

● A Read/Write (R/W) pin that selects reading mode or writing mode

● An Enable pin that enables writing to the registers

● 8 data pins (D0-D7). The state of these pins (high or low) are the bits that you're writing to a register when you write, or the values when you read.

● There's also a display contrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on or off the LED backlight respectively.

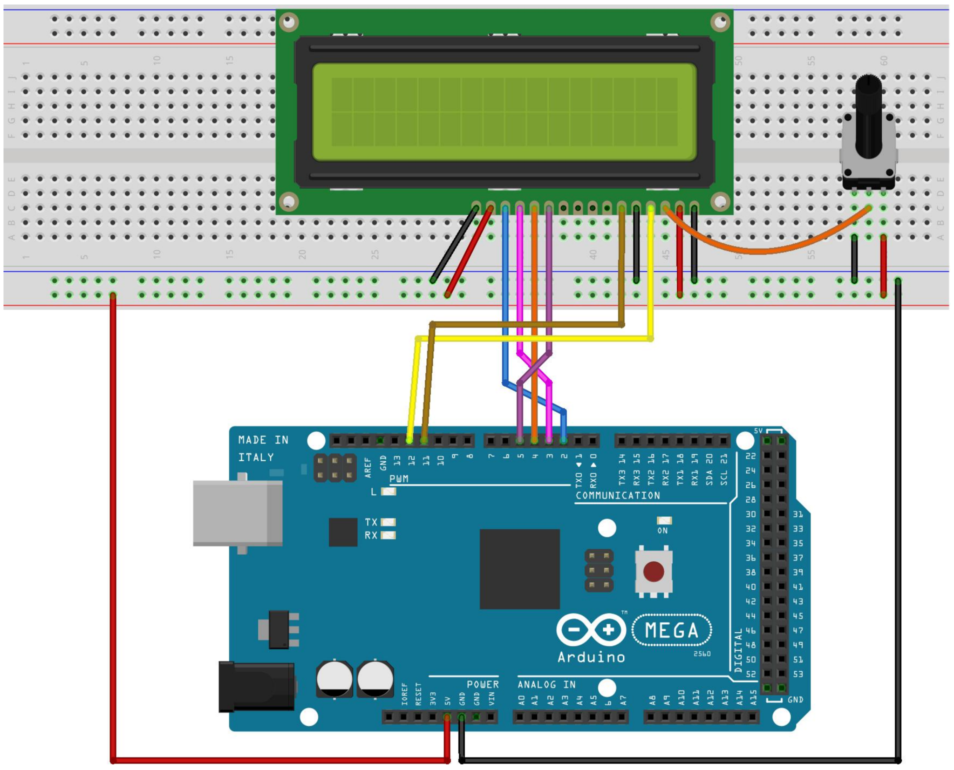
The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The LiquidCrystal Library simplifies this for you so you don't need to know the low-level instructions.

The Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode, so example shows how to control a 2x16 LCD in 4-bit mode.

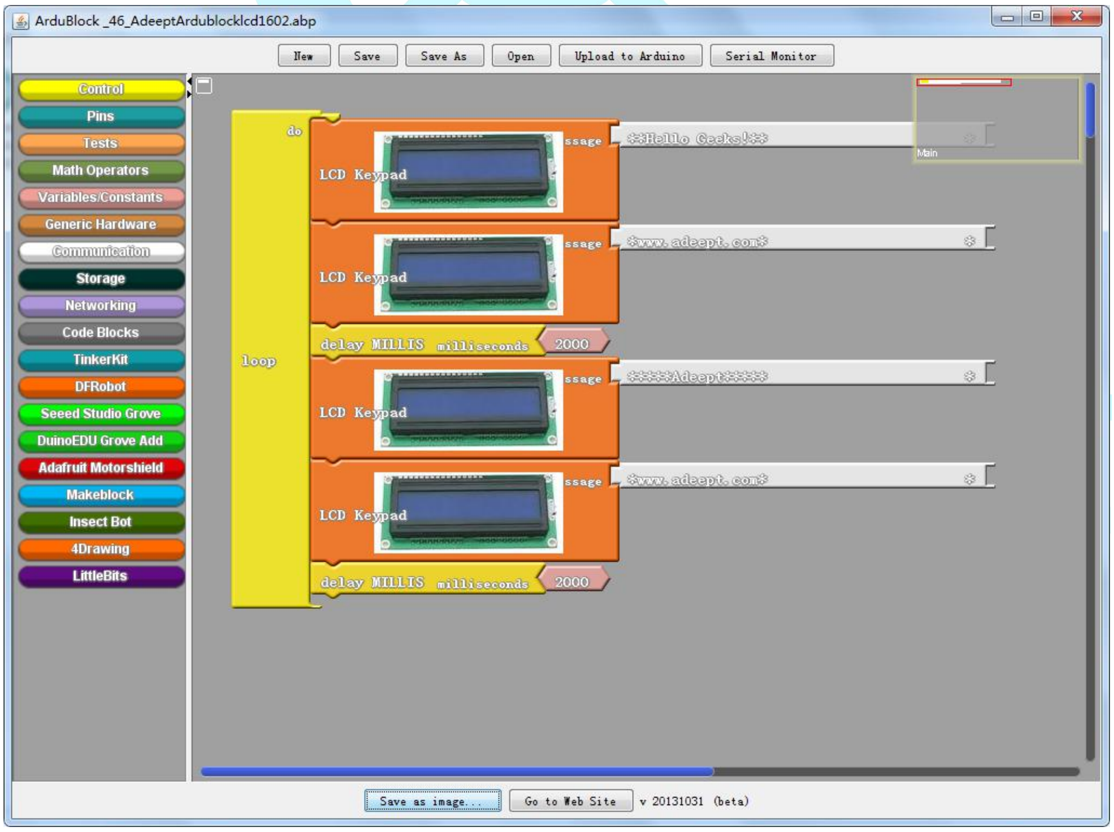
A potentiometer , informally a pot, is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat.

Procedures

1. Build the circuit

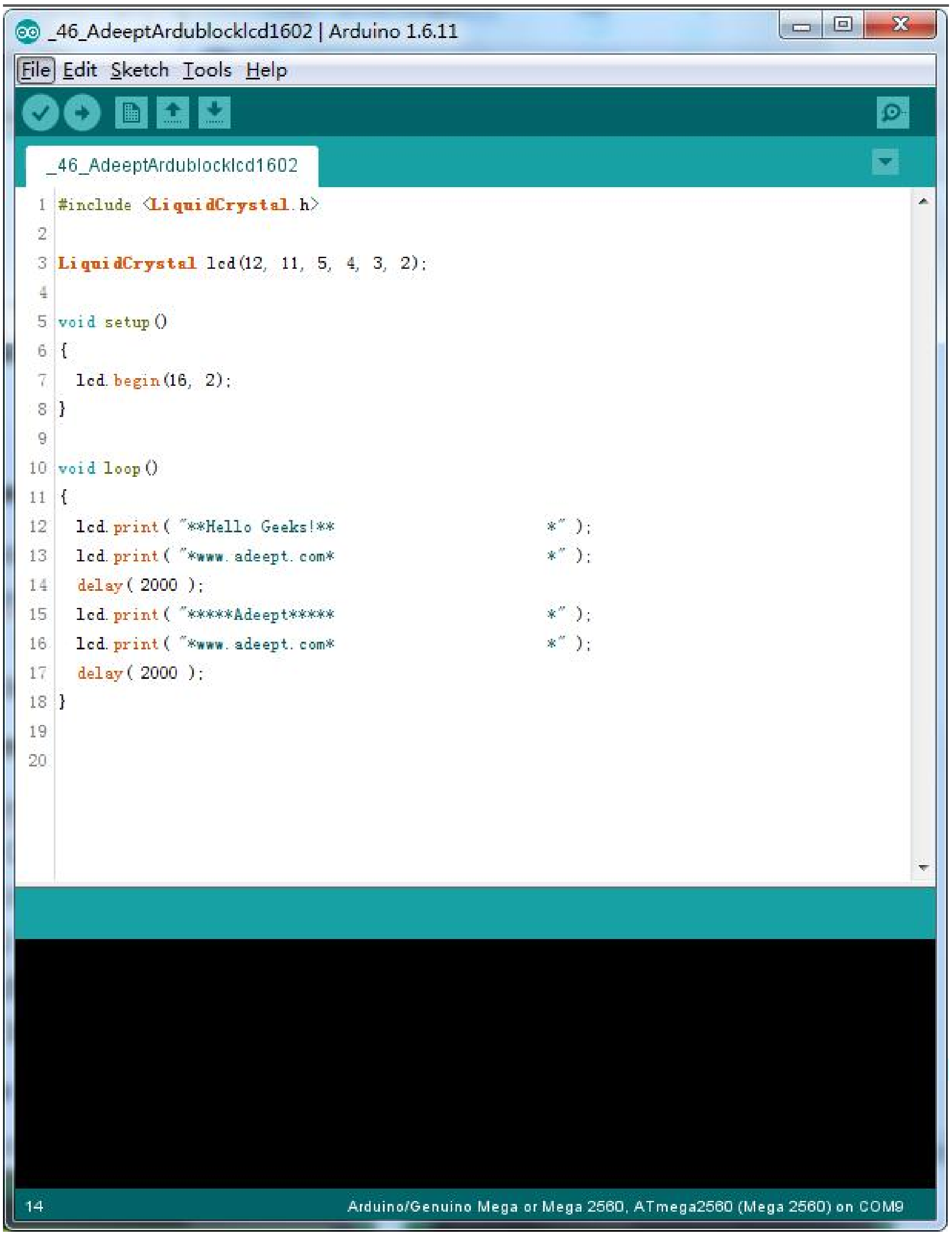


2. Program



```
loop
do
  LCD Keypad  message  **Hello Geeks!**
  LCD Keypad  message  *www.adeept.com*
  delay MILLIS milliseconds 2000
  LCD Keypad  message  *****Adeept*****
  LCD Keypad  message  *www.adeept.com*
  delay MILLIS milliseconds 2000
```

3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:



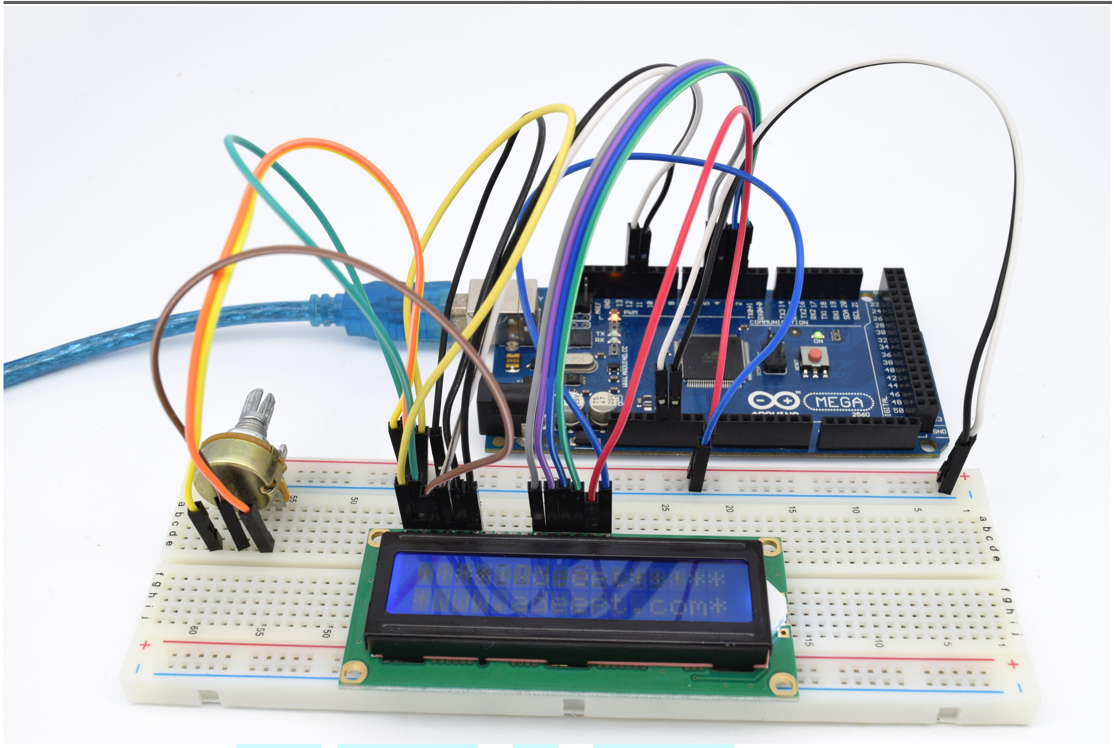
_46_AdeeptArdublocklcd1602 | Arduino 1.6.11

_46_AdeeptArdublocklcd1602

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  lcd.begin(16, 2);
}

void loop()
{
  lcd.print( "**Hello Geeks!**                          *" );
  lcd.print( "*www.adeept.com*                          *" );
  delay( 2000 );
  lcd.print( "*****Adeept*****                          *" );
  lcd.print( "*www.adeept.com*                          *" );
  delay( 2000 );
}
```

14   Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Now, you can see the string "**Hello Geeks!**" is shown on the LCD1602, and then the string "*****Adeept*****" and "*www.adeept.com*" is displayed on the LCD1602 static.



Summary

I believe that you have already mastered the driver of LCD1602 through this lesson. I hope you can make something more interesting base on this lesson and the previous lesson learned.

Adeept

Lesson 47 Adeept ardublock a simple voltmeter

Overview

In this lesson, we will make a simple voltmeter with Arduino MEGA 2560 and LCD1602, the range of this voltmeter is 0~500. Then, we will measure the voltage of the potentiometer's adjustment end with the simple voltmeter and display it on the LCD1602.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* LCD1602
- 2* Potentiometer
- 1* Breadboard
- Several Jumper Wires

Principle

The basic principle of this experiment: Converting the analog voltage that the Arduino collected to digital quantity by the ADC(analog-to-digital converter) through programming, then display the voltage on the LCD1602.

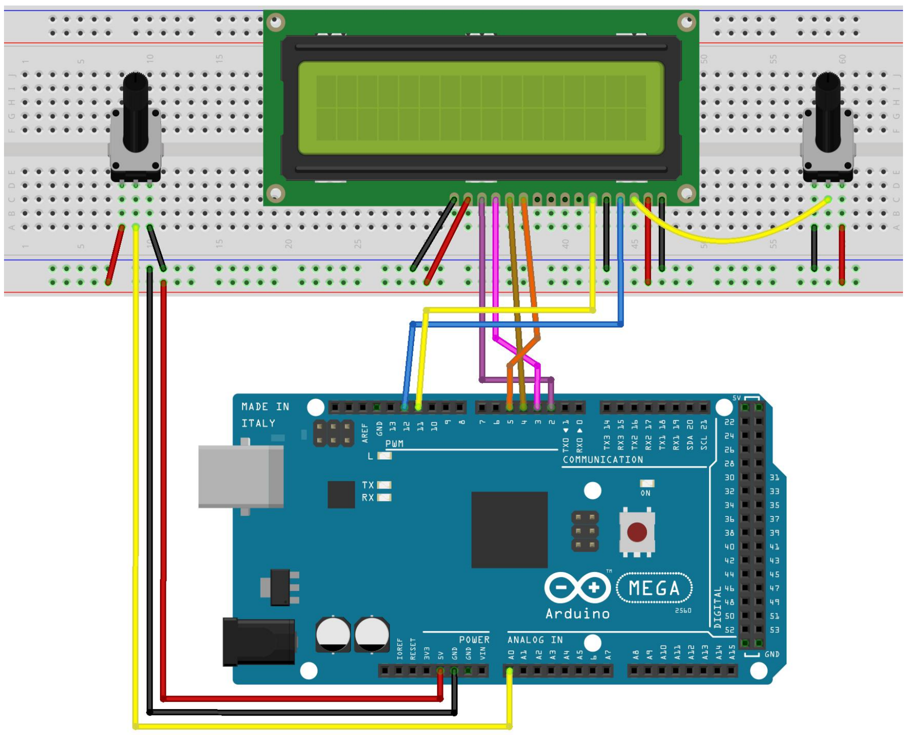
Connect the three wires from the potentiometer to your Arduino board. The first goes to ground from one of the outer pins of the potentiometer. The second goes from analog input 0 to the middle pin of the potentiometer. The third goes from 5 volts to the other outer pin of the potentiometer.

By turning the shaft of the potentiometer, you change the amount of resistance on either side of the wiper which is connected to the center pin of the potentiometer. This changes the voltage at the center pin. When the resistance between the center and the side connected to 5 volts is close to zero (and the resistance on the other side is close to 10 kilohms), the voltage at the center pin nears 5 volts. When the resistances are reversed, the voltage at the center pin nears 0 volts, or ground. This voltage is the analog voltage that you're reading as an input.

The Arduino has a circuit inside called an analog-to-digital converter that reads this changing voltage and converts it to a number between 0 and 1023. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and the input value is 0. When the shaft is turned all the way in the opposite direction, there are 5 volts going to the pin and the input value is 1023. In between, analogRead( ) returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

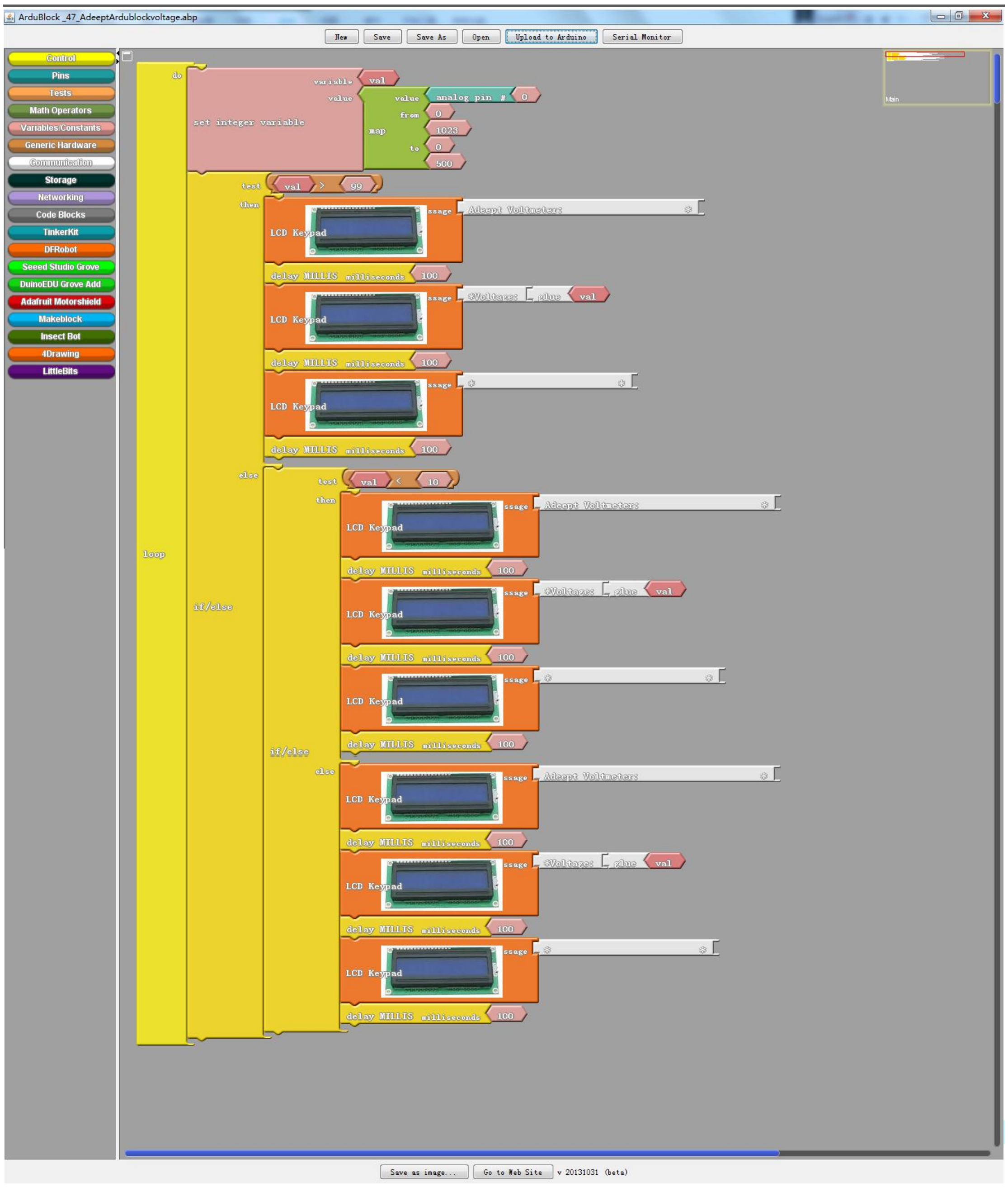
Procedures

1. Build the circuit

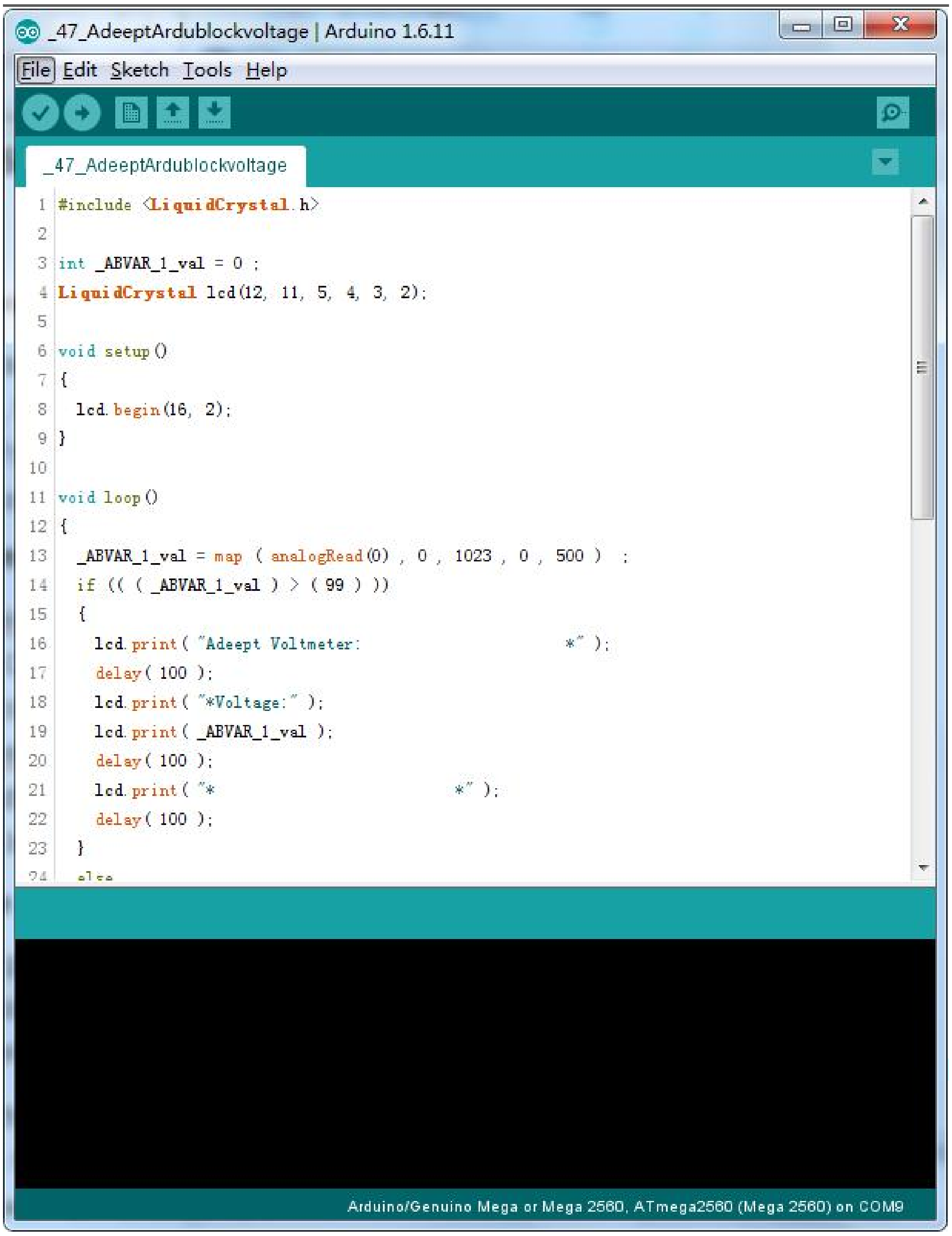


2. Program

Adeept



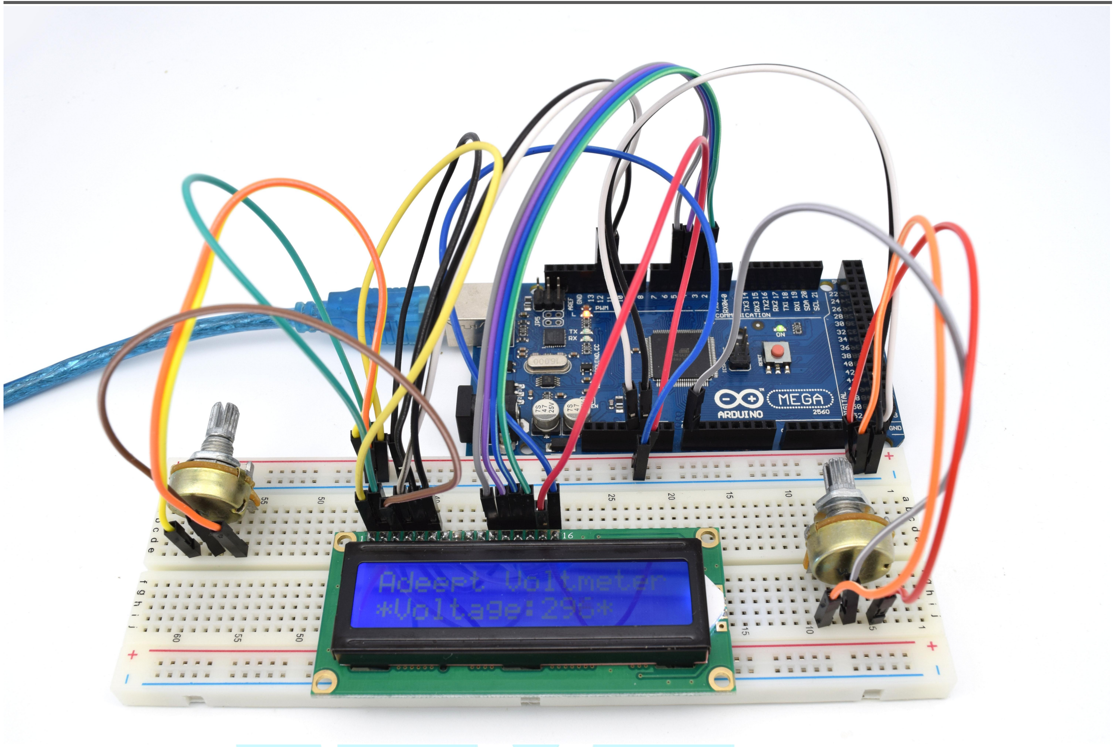
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_47_AdeeptArdublockvoltage | Arduino 1.6.11
File Edit Sketch Tools Help
_47_AdeeptArdublockvoltage

```
#include <LiquidCrystal.h>

int _ABVAR_1_val = 0 ;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  lcd.begin(16, 2);
}

void loop()
{
  _ABVAR_1_val = map ( analogRead(0) , 0 , 1023 , 0 , 500 )  ;
  if (( ( _ABVAR_1_val ) > ( 99 ) ))
  {
    lcd.print( "Adeept Voltmeter:                  *" );
    delay( 100 );
    lcd.print( "*Voltage:" );
    lcd.print( _ABVAR_1_val );
    delay( 100 );
    lcd.print( "*                  *" );
    delay( 100 );
  }
  else
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Now, when you turning the shaft of the potentiometer, you will see the voltage displayed on the LCD1602 will be changed.



Summary

The substance of voltmeter is reading analog voltage which input to ADC inside. Through this course, I believe that you have mastered how to read analog value and how to make a simple voltmeter with Arduino.

Adeept

Lesson 48 Adeept ardublock 7-segment display

Overview

In this lesson, we will program the Arduino to achieve the controlling of segment display.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* 220Ω Resistor
- 1* 7-segment Display
- 1* Breadboard
- Several Jumper Wires

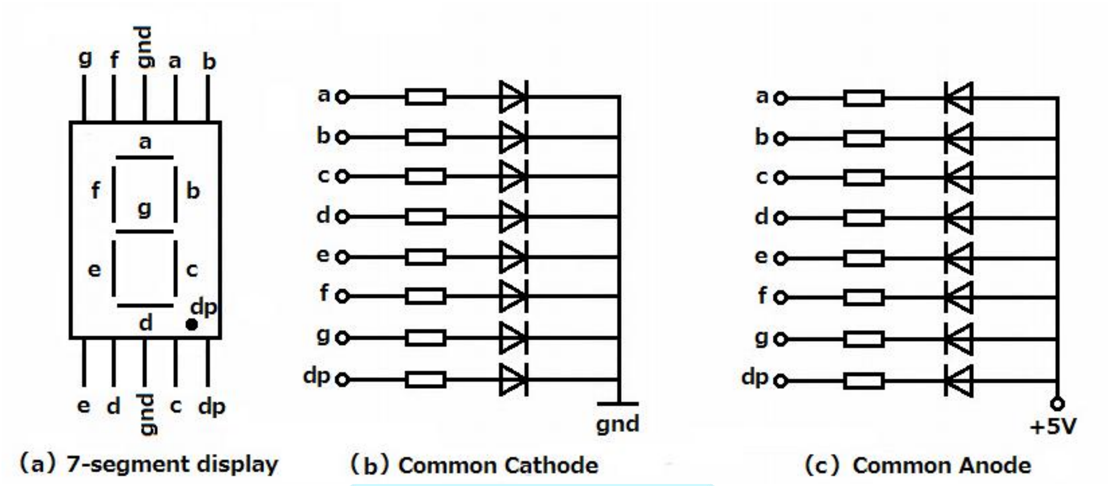
Principle

The seven-segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays.

Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information.

The seven-segment display is an 8-shaped LED display device composed of eight LEDs (including a decimal point), these segments respectively named a, b, c, d, e, f, g, dp.

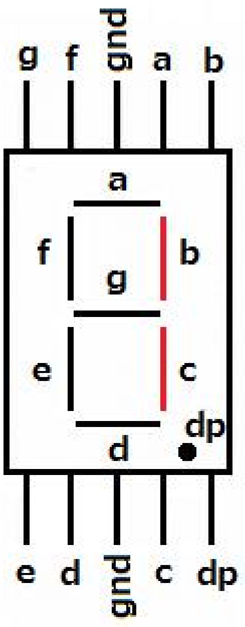
The segment display can be divided into common anode and common cathode segment display by internal connections.



When using a common anode LED, the common anode should to be connected to the power supply (VCC); when using a common cathode LED, the common cathode should be connected to the ground (GND).

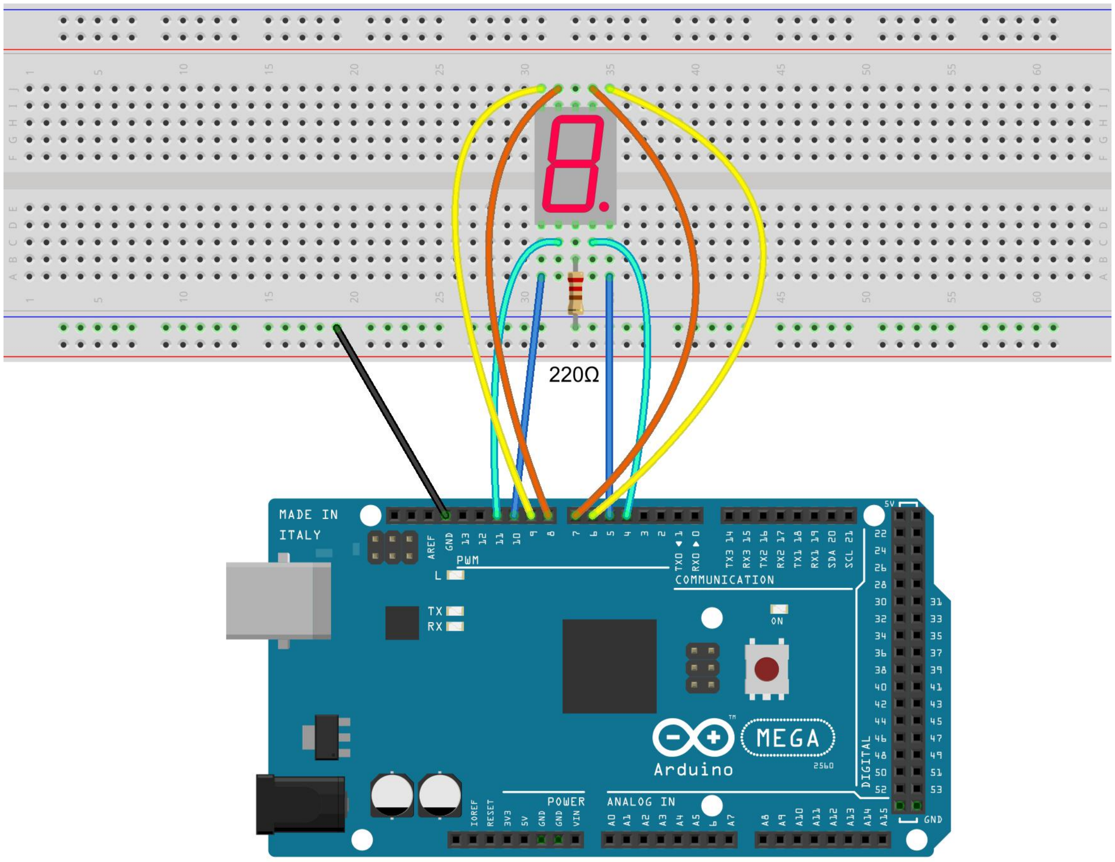
Each segment of a segment display is composed of LED, so a resistor is needed for protecting the LED.

A 7-segment display has seven segments for displaying a figure and a segment for displaying a decimal point. If you want to display a number ‘1’, you should only light the segment b and c.

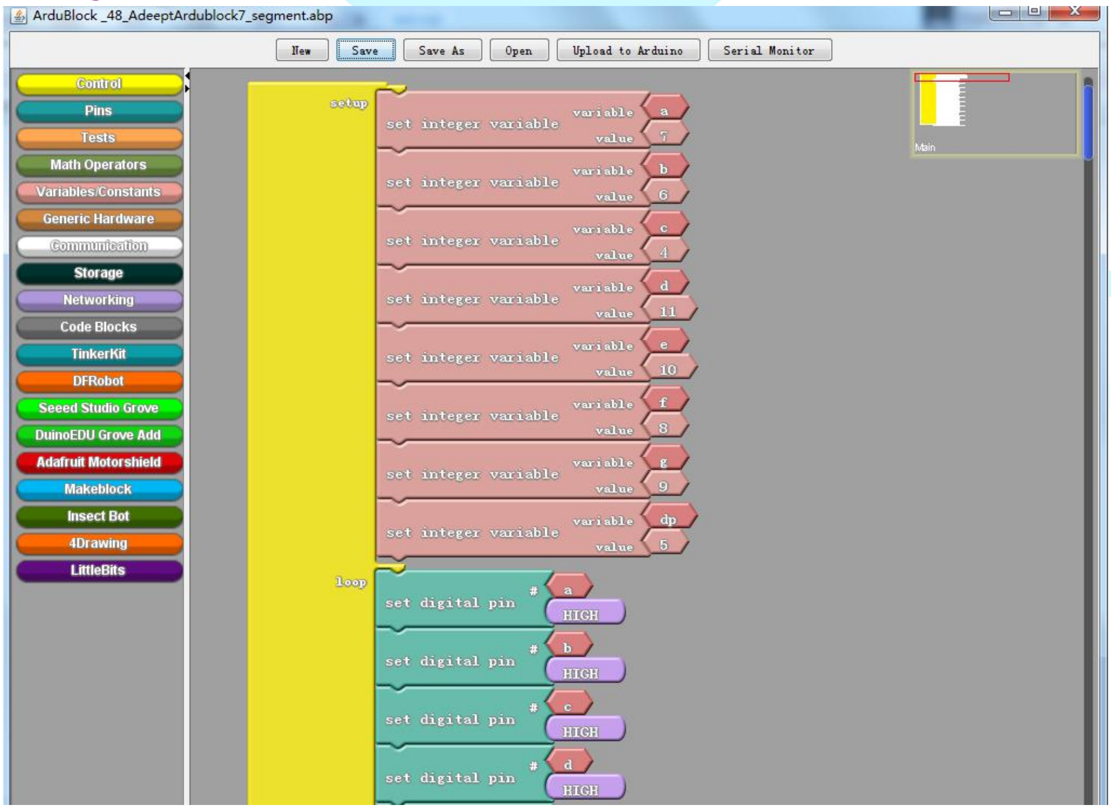


Procedures

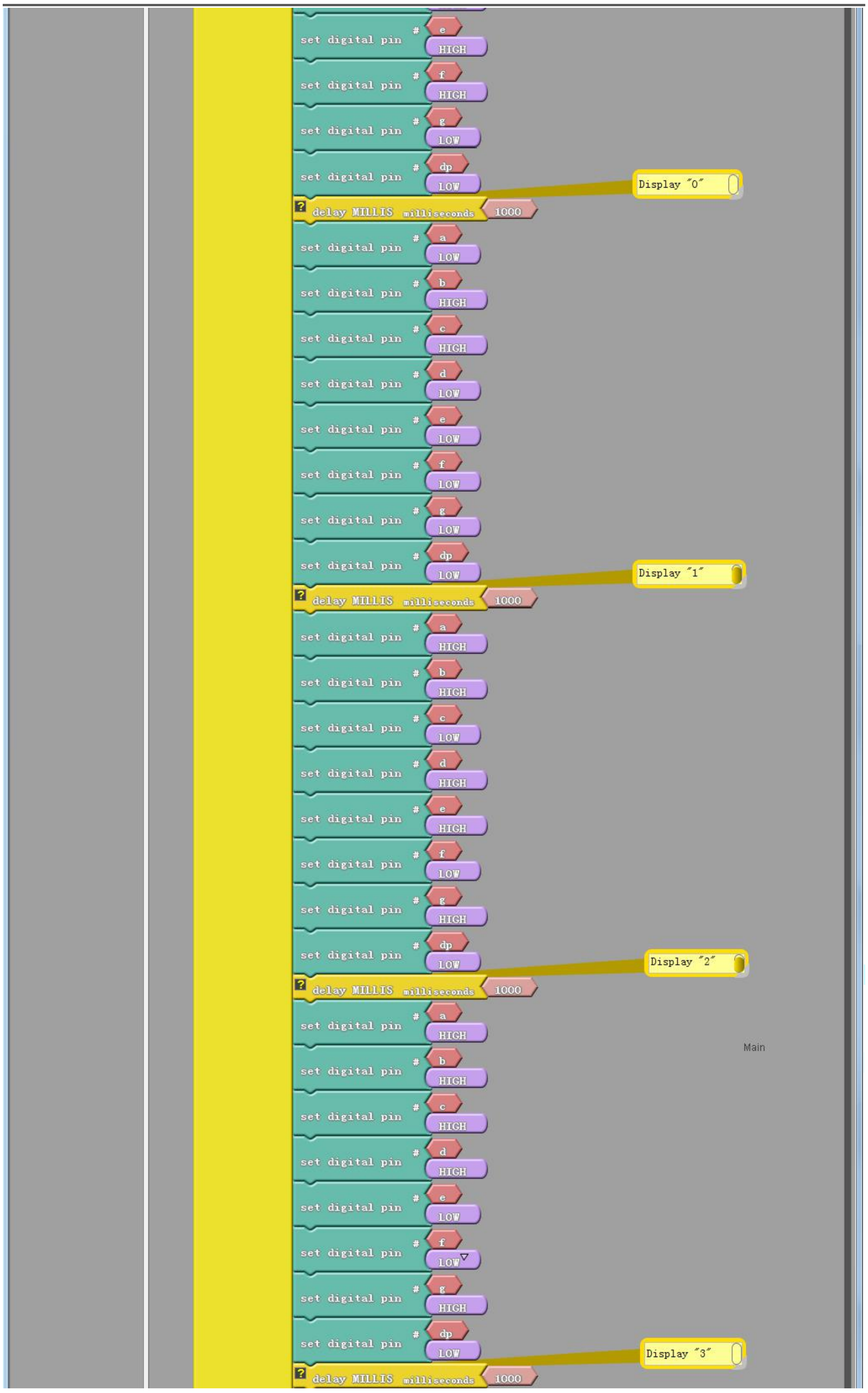
1. Build the circuit



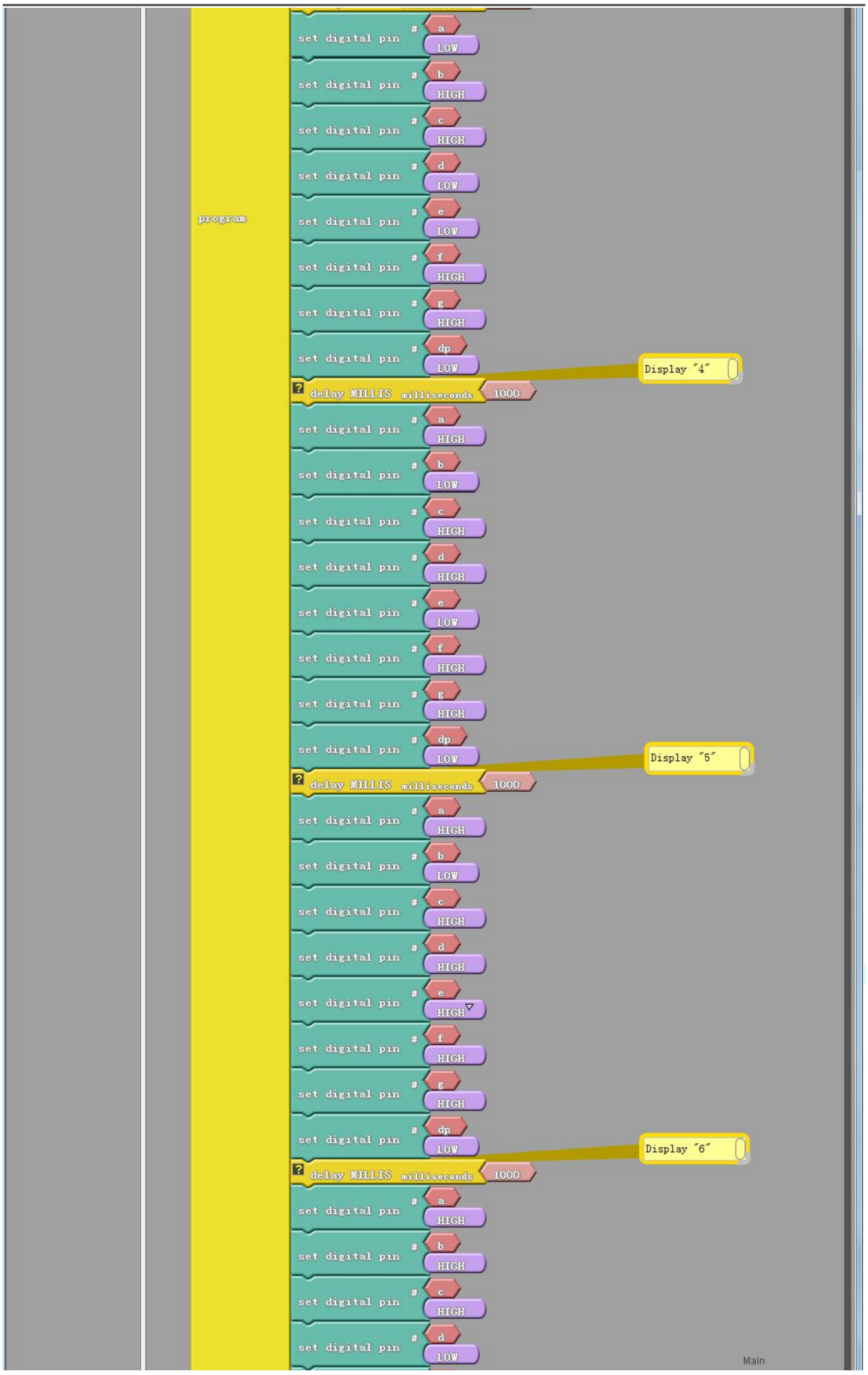
2. Program



```
setup
  set integer variable  variable a   value 7
  set integer variable  variable b   value 6
  set integer variable  variable c   value 4
  set integer variable  variable d   value 11
  set integer variable  variable e   value 10
  set integer variable  variable f   value 8
  set integer variable  variable g   value 9
  set integer variable  variable dp  value 5
loop
  set digital pin  # a  HIGH
  set digital pin  # b  HIGH
  set digital pin  # c  HIGH
  set digital pin  # d  HIGH
```



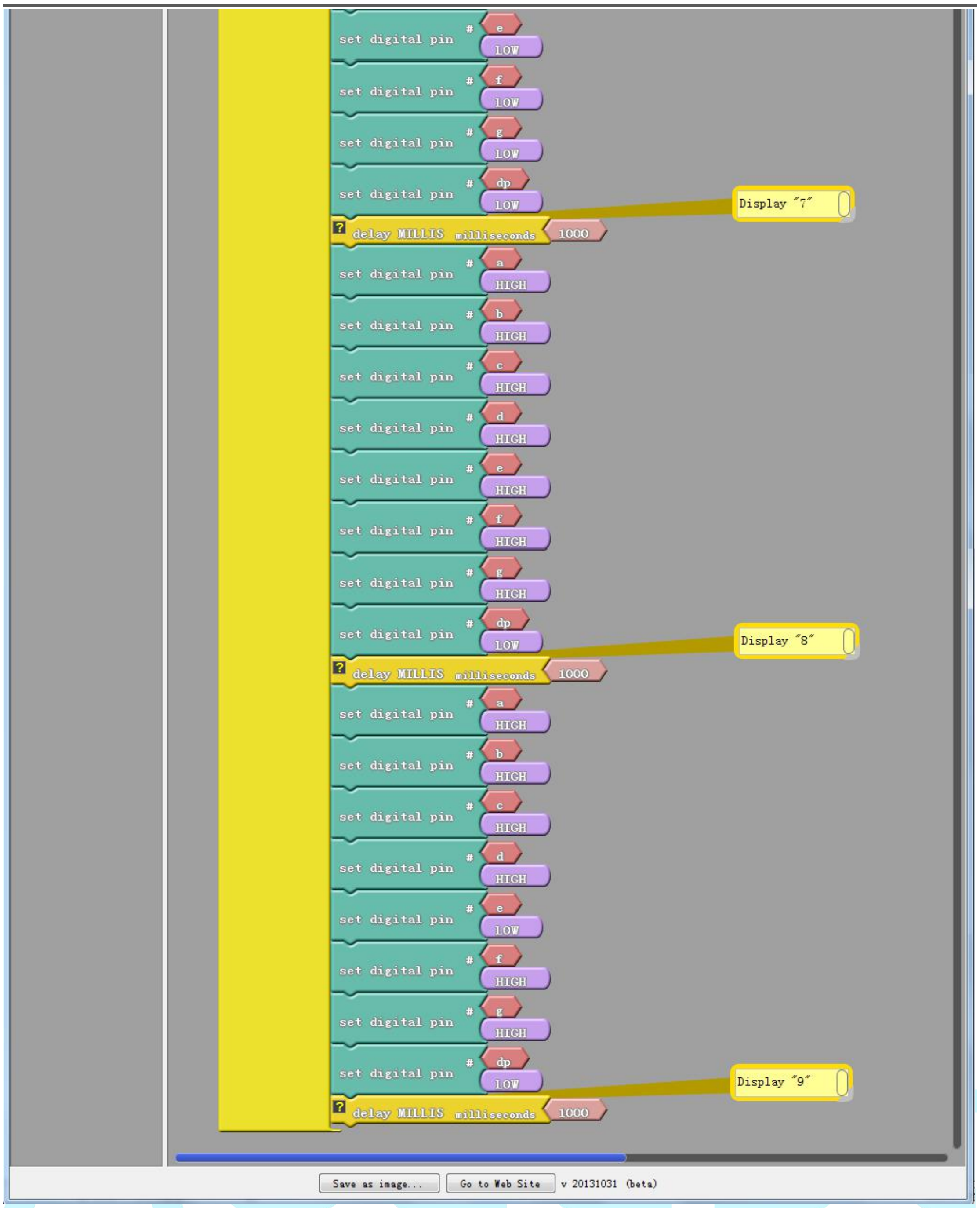
```
set digital pin # e HIGH
set digital pin # f HIGH
set digital pin # g LOW
set digital pin # dp LOW
delay MILLIS milliseconds 1000        Display "0"
set digital pin # a LOW
set digital pin # b HIGH
set digital pin # c HIGH
set digital pin # d LOW
set digital pin # e LOW
set digital pin # f LOW
set digital pin # g LOW
set digital pin # dp LOW
delay MILLIS milliseconds 1000        Display "1"
set digital pin # a HIGH
set digital pin # b HIGH
set digital pin # c LOW
set digital pin # d HIGH
set digital pin # e HIGH
set digital pin # f LOW
set digital pin # g HIGH
set digital pin # dp LOW
delay MILLIS milliseconds 1000        Display "2"
set digital pin # a HIGH
set digital pin # b HIGH
set digital pin # c HIGH
set digital pin # d HIGH
set digital pin # e LOW
set digital pin # f LOW
set digital pin # g HIGH
set digital pin # dp LOW
delay MILLIS milliseconds 1000        Display "3"
```



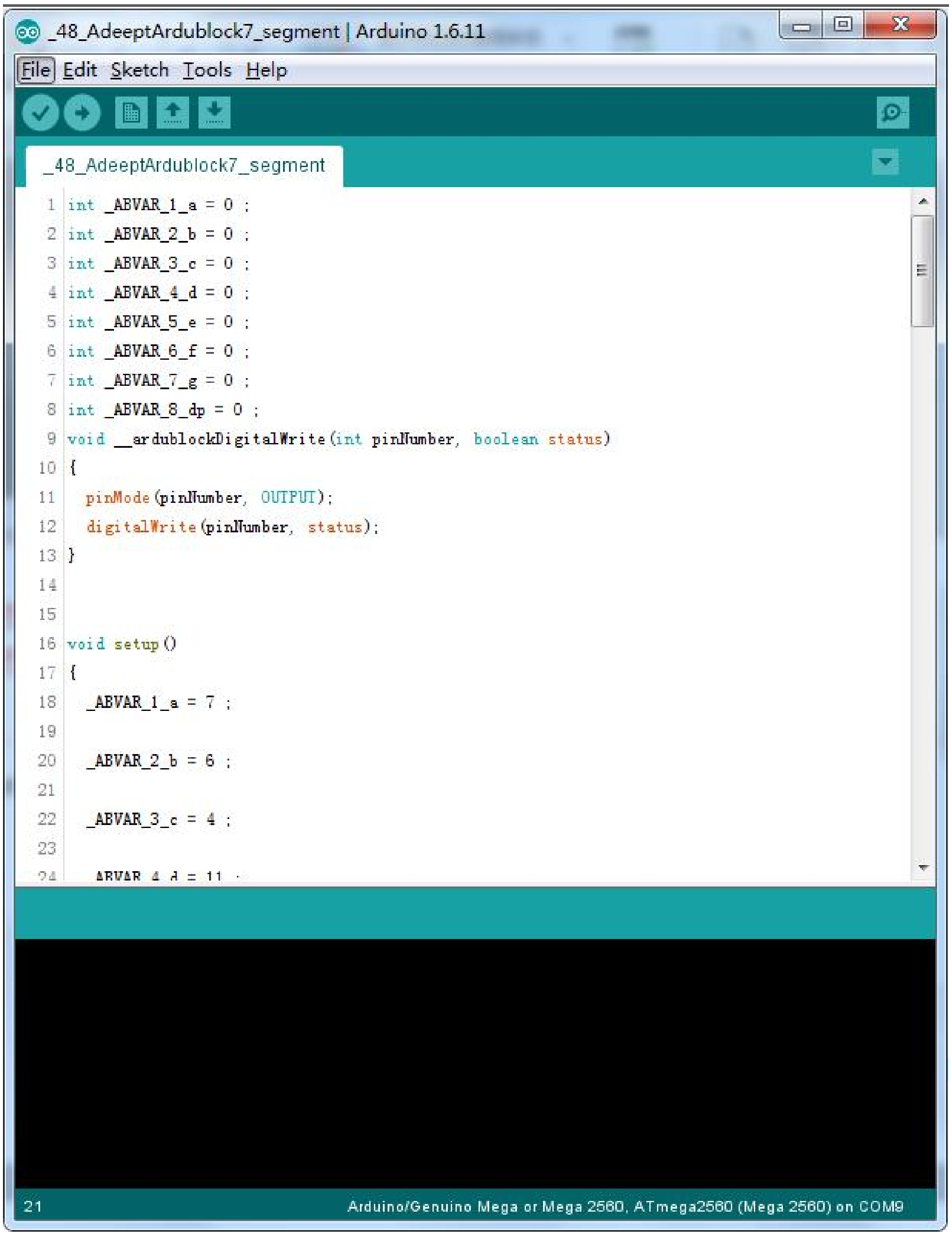
program

set digital pin # a LOW
set digital pin # b HIGH
set digital pin # c HIGH
set digital pin # d LOW
set digital pin # e LOW
set digital pin # f HIGH
set digital pin # g HIGH
set digital pin # dp LOW
Display "4"
delay MILLIS milliseconds 1000
set digital pin # a HIGH
set digital pin # b LOW
set digital pin # c HIGH
set digital pin # d HIGH
set digital pin # e LOW
set digital pin # f HIGH
set digital pin # g HIGH
set digital pin # dp LOW
Display "5"
delay MILLIS milliseconds 1000
set digital pin # a HIGH
set digital pin # b LOW
set digital pin # c HIGH
set digital pin # d HIGH
set digital pin # e HIGH
set digital pin # f HIGH
set digital pin # g HIGH
set digital pin # dp LOW
Display "6"
delay MILLIS milliseconds 1000
set digital pin # a HIGH
set digital pin # b HIGH
set digital pin # c HIGH
set digital pin # d LOW

Main


set digital pin # e LOW
set digital pin # f LOW
set digital pin # g LOW
set digital pin # dp LOW
Display "7"
delay MILLIS milliseconds 1000
set digital pin # a HIGH
set digital pin # b HIGH
set digital pin # c HIGH
set digital pin # d HIGH
set digital pin # e HIGH
set digital pin # f HIGH
set digital pin # g HIGH
set digital pin # dp LOW
Display "8"
delay MILLIS milliseconds 1000
set digital pin # a HIGH
set digital pin # b HIGH
set digital pin # c HIGH
set digital pin # d HIGH
set digital pin # e LOW
set digital pin # f HIGH
set digital pin # g HIGH
set digital pin # dp LOW
Display "9"
delay MILLIS milliseconds 1000

3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_48_AdeeptArdublock7_segment | Arduino 1.6.11

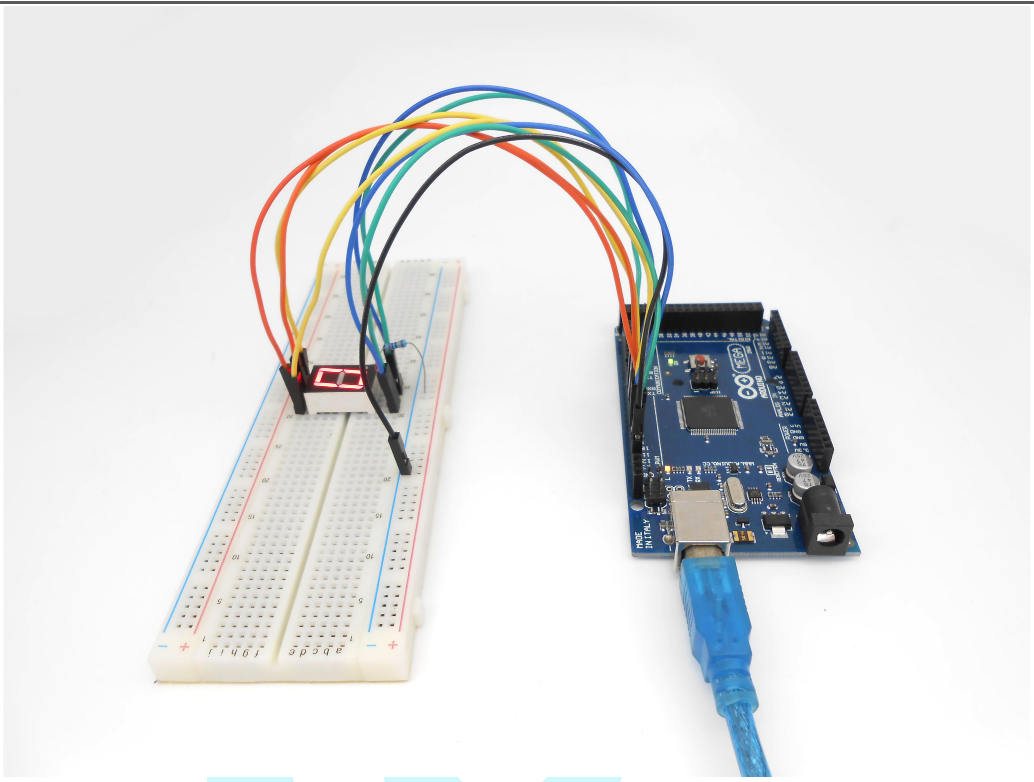
File Edit Sketch Tools Help

_48_AdeeptArdublock7_segment

```
int _ABVAR_1_a = 0 ;
int _ABVAR_2_b = 0 ;
int _ABVAR_3_c = 0 ;
int _ABVAR_4_d = 0 ;
int _ABVAR_5_e = 0 ;
int _ABVAR_6_f = 0 ;
int _ABVAR_7_g = 0 ;
int _ABVAR_8_dp = 0 ;
void __ardublockDigitalWrite(int pinNumber, boolean status)
{
  pinMode(pinNumber, OUTPUT);
  digitalWrite(pinNumber, status);
}


void setup()
{
  _ABVAR_1_a = 7 ;

  _ABVAR_2_b = 6 ;

  _ABVAR_3_c = 4 ;

  _ABVAR_4_d = 11 ;
```

21   Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Now, you should see the number 0~9 are displayed on the segment display.



Summary

Through this lesson, we have learned the principle and programming of segment display. I hope you can combine the former course to modify the code we provided in this lesson to achieve cooler originality.

Adeept

Lesson 49 Adeept ardublock controlling servo motor

Overview

In this lesson, we will introduce a new electronic device (Servo) to you, and tell you how to control it with the Arduino MEGA 2560.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* Servo
- Several Jumper Wires

Principle

1. Servo motor

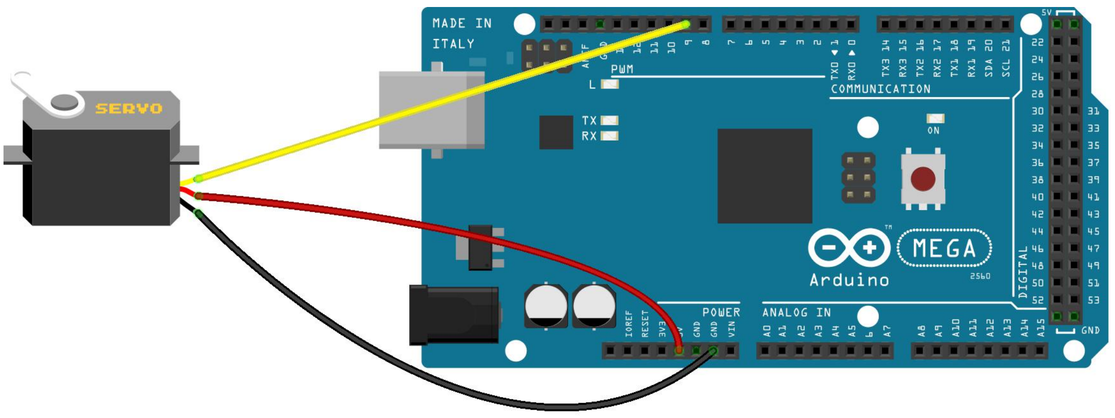
The servo motor have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the Arduino board. The signal pin is typically yellow, orange or white and should be connected to a digital pin on the Arduino board. Note the servo motor draw considerable power, so if you need to drive more than one or two, you'll probably need to power them from a separate supply (i.e. not the +5V pin on your Arduino). Be sure to connect the grounds of the Arduino and external power supply together.

2. Servo library

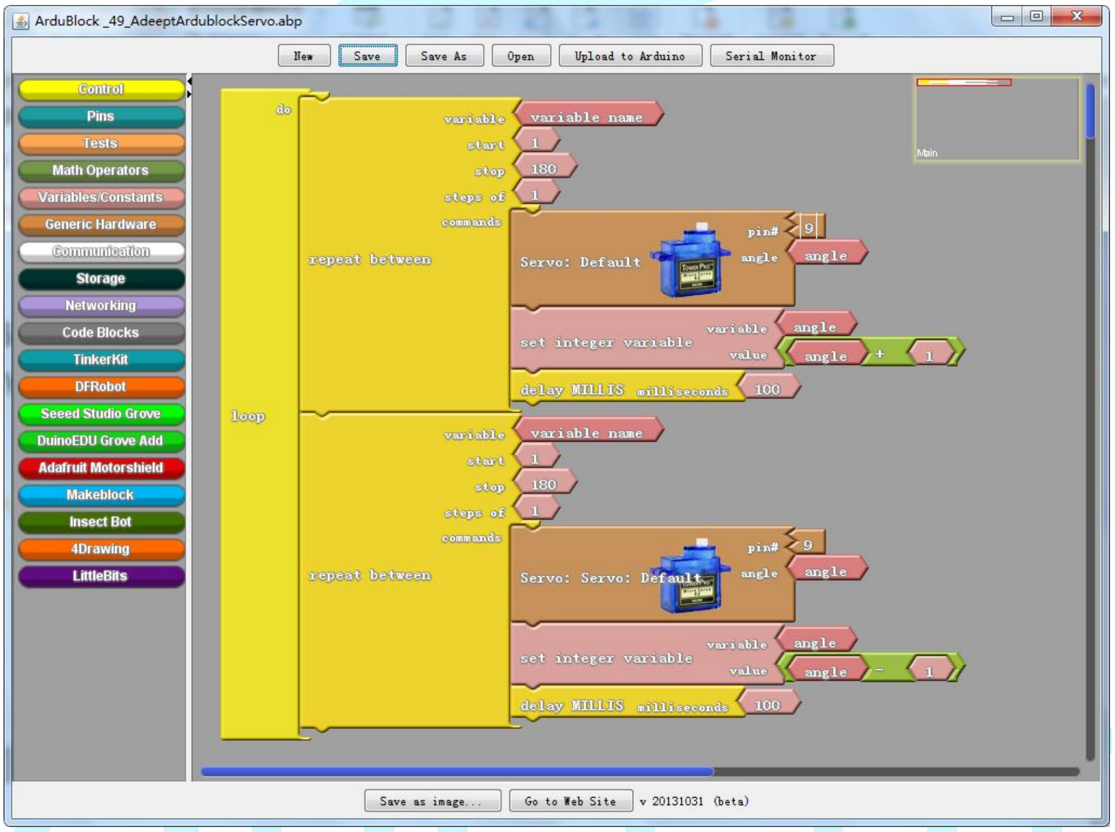
This library allows an Arduino board to control RC (hobby) servo motors. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

Procedures

1. Build the circuit

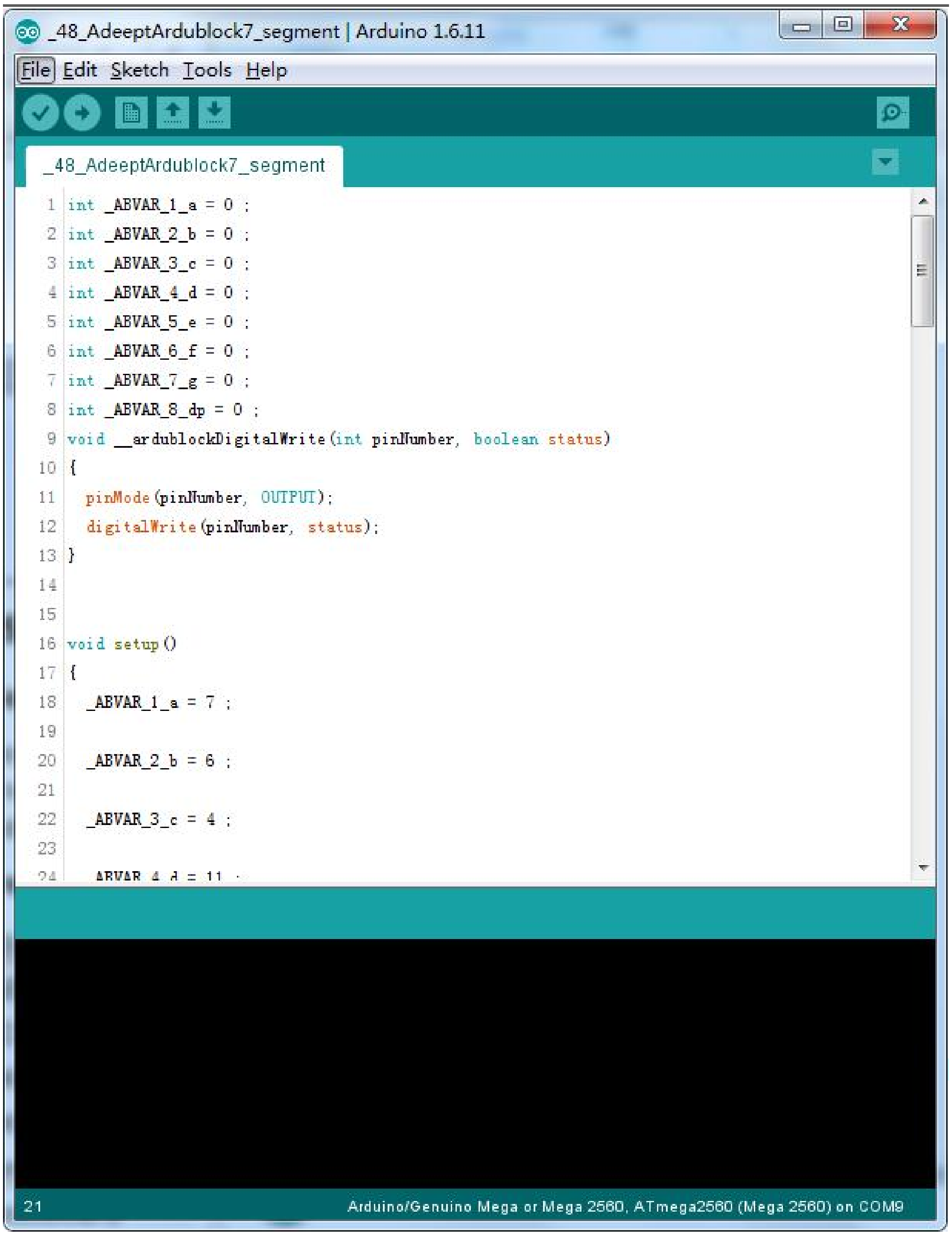


2. Program



```
loop
  do
    repeat between
      variable   variable name
      start      1
      stop       180
      steps of   1
      commands
        Servo: Default   pin# 9
                         angle   angle
        set integer variable   variable   angle
                               value      angle + 1
        delay MILLIS milliseconds   100
    repeat between
      variable   variable name
      start      1
      stop       180
      steps of   1
      commands
        Servo: Servo: Default   pin# 9
                                angle   angle
        set integer variable   variable   angle
                               value      angle - 1
        delay MILLIS milliseconds   100
```

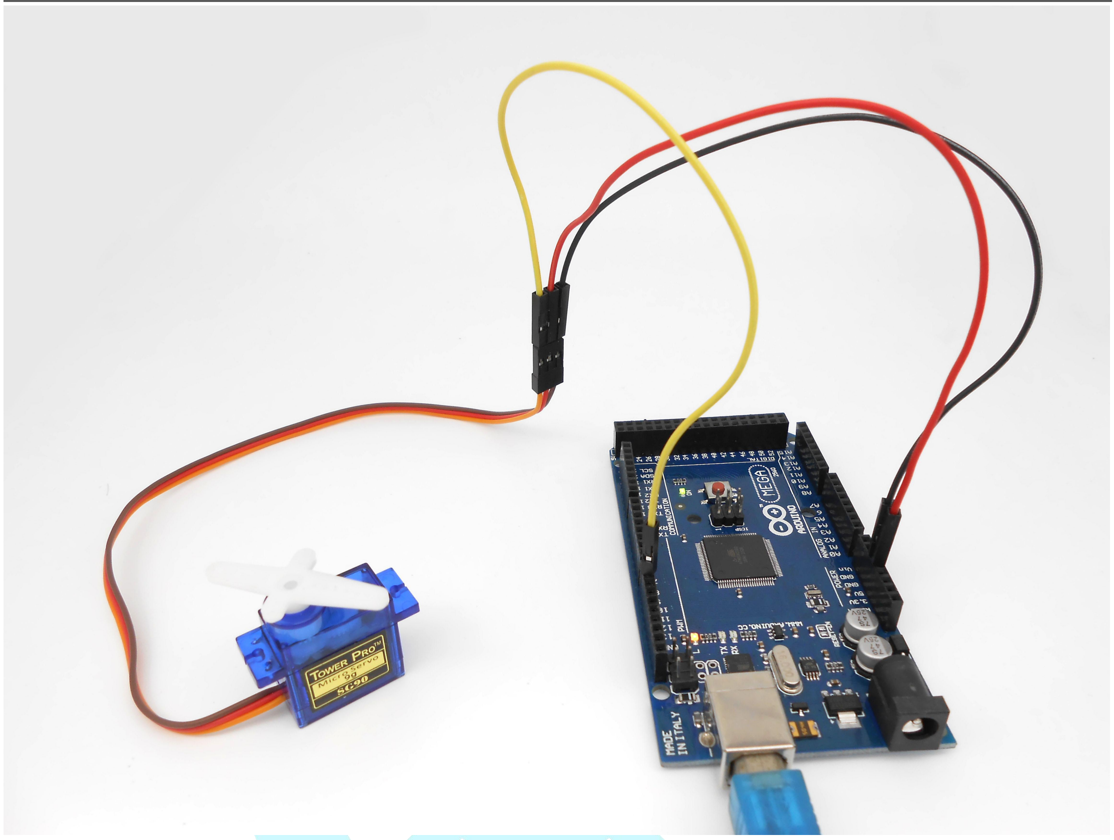
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_48_AdeeptArdublock7_segment | Arduino 1.6.11
File Edit Sketch Tools Help
_48_AdeeptArdublock7_segment

```
int _ABVAR_1_a = 0 ;
int _ABVAR_2_b = 0 ;
int _ABVAR_3_c = 0 ;
int _ABVAR_4_d = 0 ;
int _ABVAR_5_e = 0 ;
int _ABVAR_6_f = 0 ;
int _ABVAR_7_g = 0 ;
int _ABVAR_8_dp = 0 ;
void __ardublockDigitalWrite(int pinNumber, boolean status)
{
  pinMode(pinNumber, OUTPUT);
  digitalWrite(pinNumber, status);
}


void setup()
{
  _ABVAR_1_a = 7 ;

  _ABVAR_2_b = 6 ;

  _ABVAR_3_c = 4 ;

  _ABVAR_4_d = 11 ;
```

```
21                Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9
```

Now, you should see the servo rotate 180 degrees, and then rotate in opposite direction.



Summary

By learning this lesson, you should have known that the Arduino provided a servo library to control a servo. By using the servo library, you can easily control a servo. Just enjoy your imagination and make some interesting applications.

Adeept

Lesson 50 Adeept ardublock thermistor

Overview

In this lesson, we will learn how to use a thermistor to collect temperature by programming Arduino. The information which a thermistor collects temperature is displayed on the LCD1602.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* LCD1602
- 1* 10KΩ Potentiometer
- 1* 10KΩ Resistor
- 1* Thermistor
- 1* Breadboard
- Several Jumper Wires

Principle

A thermistor is a type of resistor whose resistance varies significantly with temperature, more so than in standard resistors. We are using MF52 NTC thermistor type. BTC thermistor is usually used as a temperature sensor.

MF52 thermistor key parameters:

B-parameter : 3470.

25℃ resistor : 10KΩ.

The relationship between the resistance of thermistor and temperature is as follows:

$$R_{thermistor} = R * e^{\left(B*\left(\frac{1}{T_1}-\frac{1}{T_2}\right)\right)}$$

$R_{thermistor}$: The resistance of thermistor at temperature T1

$\mathbf{R}$: The nominal resistance of thermistor at room temperature T2;

$\mathbf{e}$: 2.718281828459 ;

$\mathbf{B}$: It is one of the important parameters of thermistor;

T_1 : The Kelvin temperature that you want to measure.

T_2 : At the condition of room temperature 25 ℃ (298.15K), the standard resistance of MF52 thermistor is 10K;

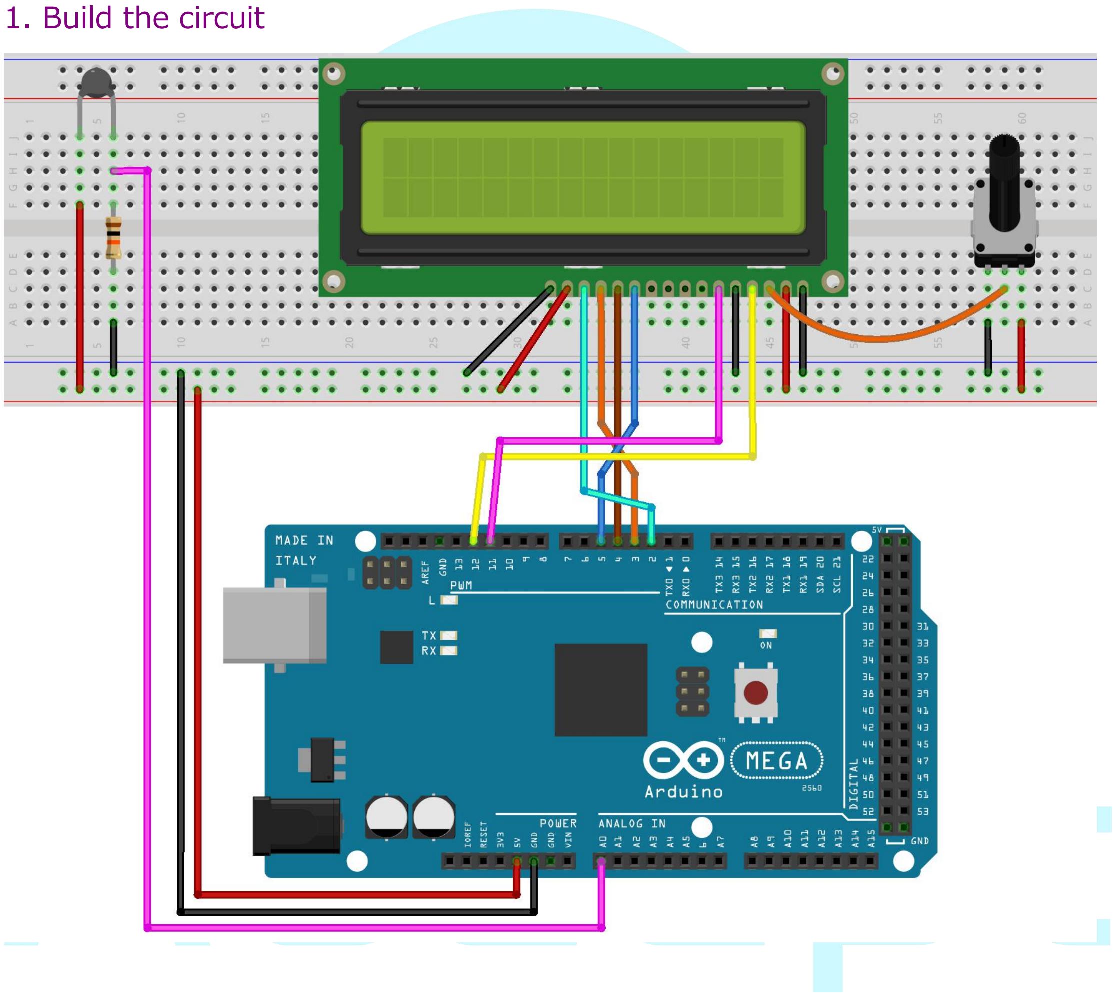
Kelvin temperature = 273.15 (absolute temperature) + degrees Celsius;

After transforming the above equation, we can get to the following formula:

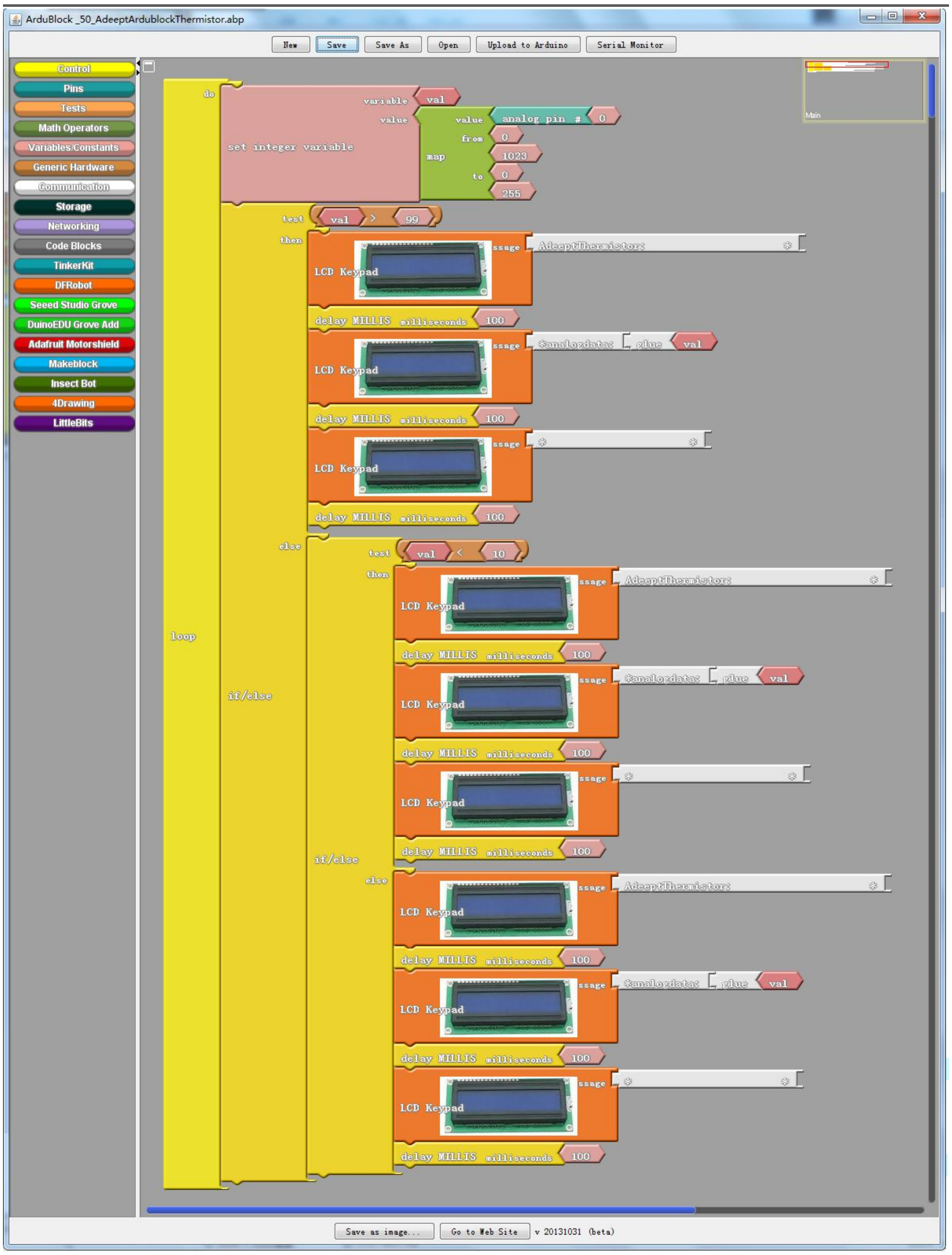
$$T_1 = \frac{B}{\left(ln^{\left(\frac{R_{thermistor}}{R}\right)} + \frac{B}{T_2}\right)}$$

Procedures

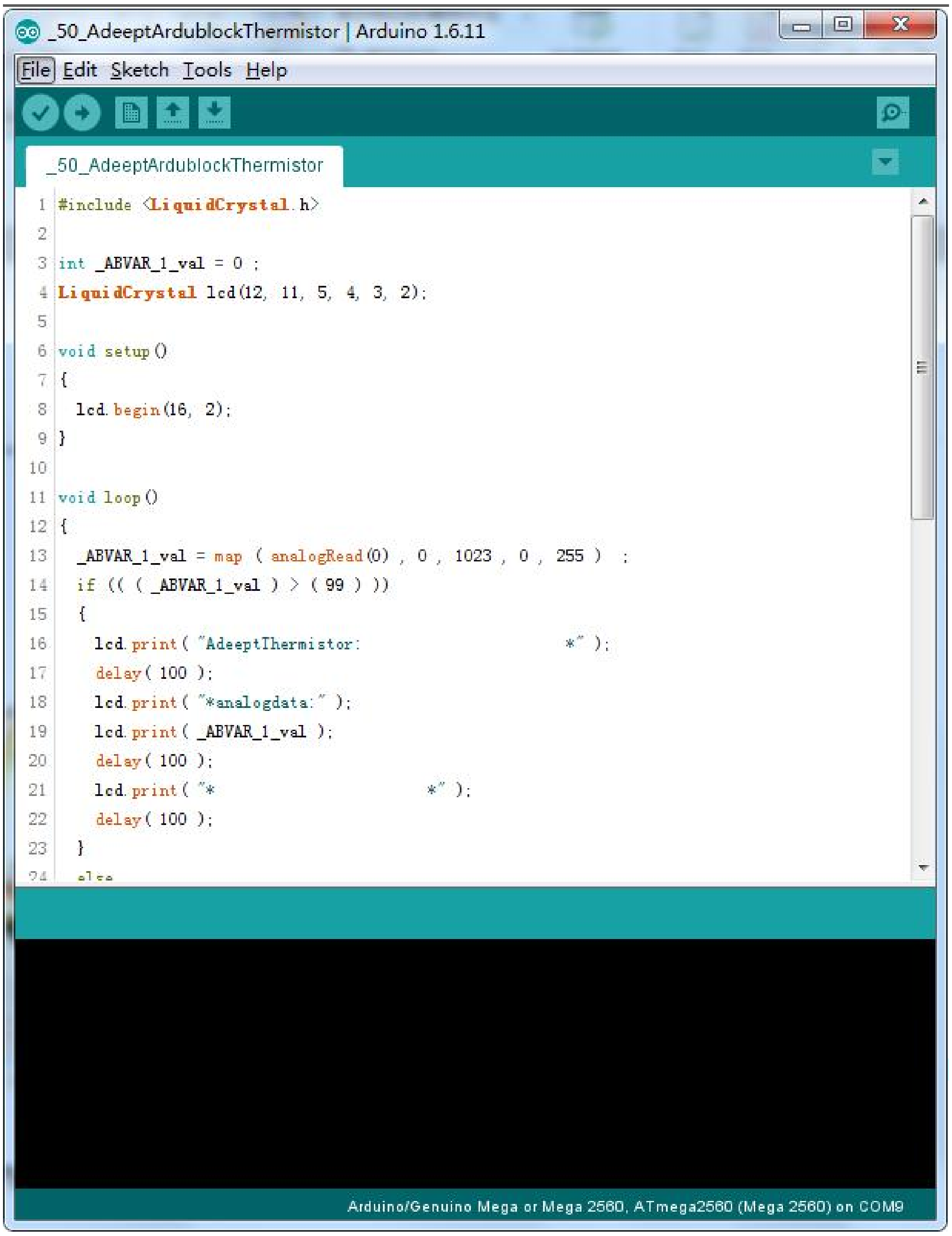
1. Build the circuit



2. Program

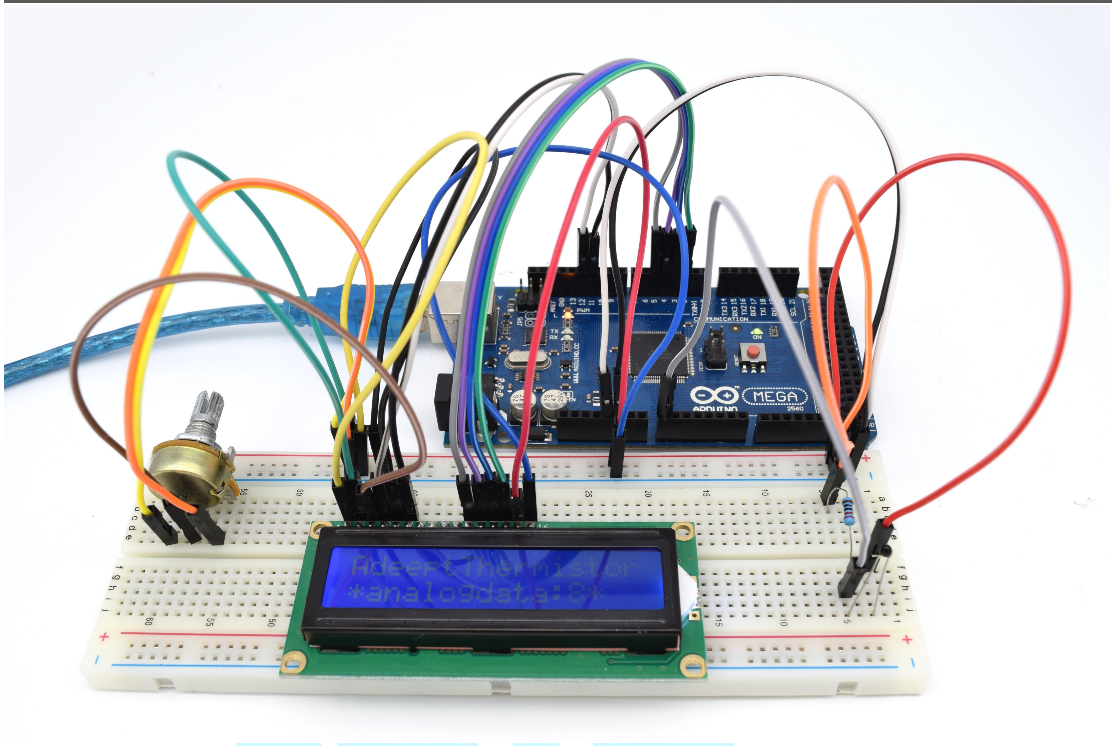


3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_50_AdeeptArdublockThermistor | Arduino 1.6.11
File Edit Sketch Tools Help
_50_AdeeptArdublockThermistor
Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

```
#include <LiquidCrystal.h>

int _ABVAR_1_val = 0 ;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  lcd.begin(16, 2);
}

void loop()
{
  _ABVAR_1_val = map ( analogRead(0) , 0 , 1023 , 0 , 255 )  ;
  if (( ( _ABVAR_1_val ) > ( 99 ) ))
  {
    lcd.print( "AdeeptThermistor:                *" );
    delay( 100 );
    lcd.print( "*analogdata:" );
    lcd.print( _ABVAR_1_val );
    delay( 100 );
    lcd.print( "*                       *" );
    delay( 100 );
  }
  else
```

Now, you can see the temperature which is collected by thermistor on the LCD1602.



Summary

By learning this lesson, I believe you have learned to use a thermistor to measure temperature. Next, you can use a thermistor to produce some interesting applications.

Adeept

Lesson 51 Adeept ardublock ultrasonic distance sensor

Overview

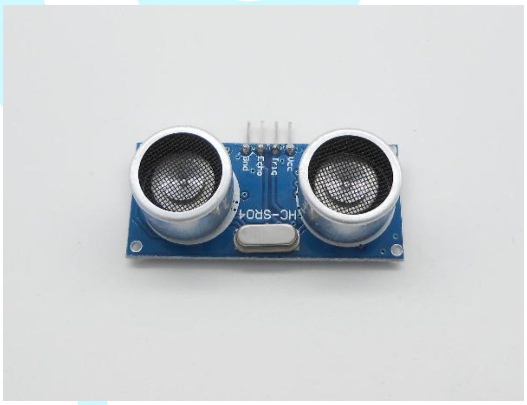
In this lesson, we will learn how to measure the distance by the ultrasonic distance sensor.

Requirement

- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* Ultrasonic Distance Sensor
- Several Jumper Wires

Principle

This recipe uses the popular Parallax PING ultrasonic distance sensor to measure the distance of an object ranging from 2 cm to around 3 m.

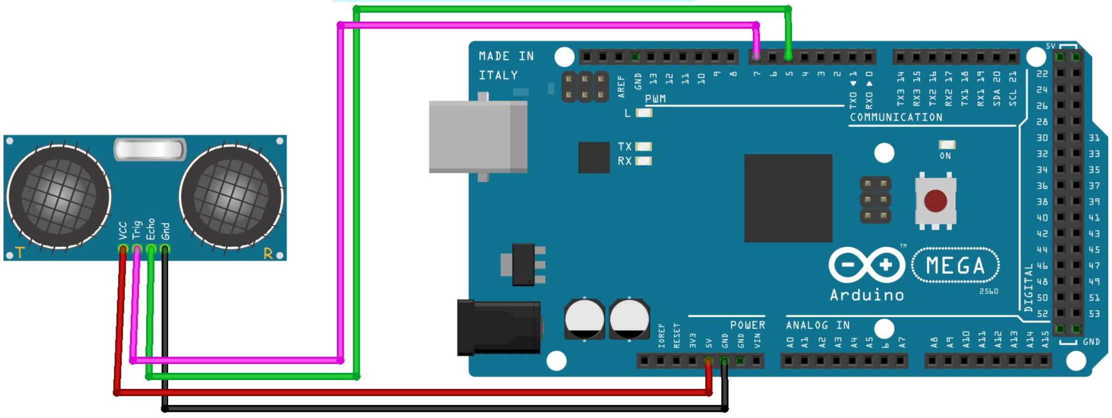


Ultrasonic sensors provide a measurement of the time it takes for sound to bounce off an object and return to the sensor. The “ping” sound pulse is generated when the pingPin level goes HIGH for two micro-seconds. The sensor will then generate a pulse that terminates when the sound returns. The width of the pulse is proportional to the distance the sound traveled and the sketch then uses the pulseIn function to measure that duration. The speed of sound is 340 meters per second, which is 29 microseconds per centimeter. The formula for the distance of the round trip is: RoundTrip = microseconds / 29.

So, the formula for the one-way distance in centimeters is: microseconds / 29 / 2

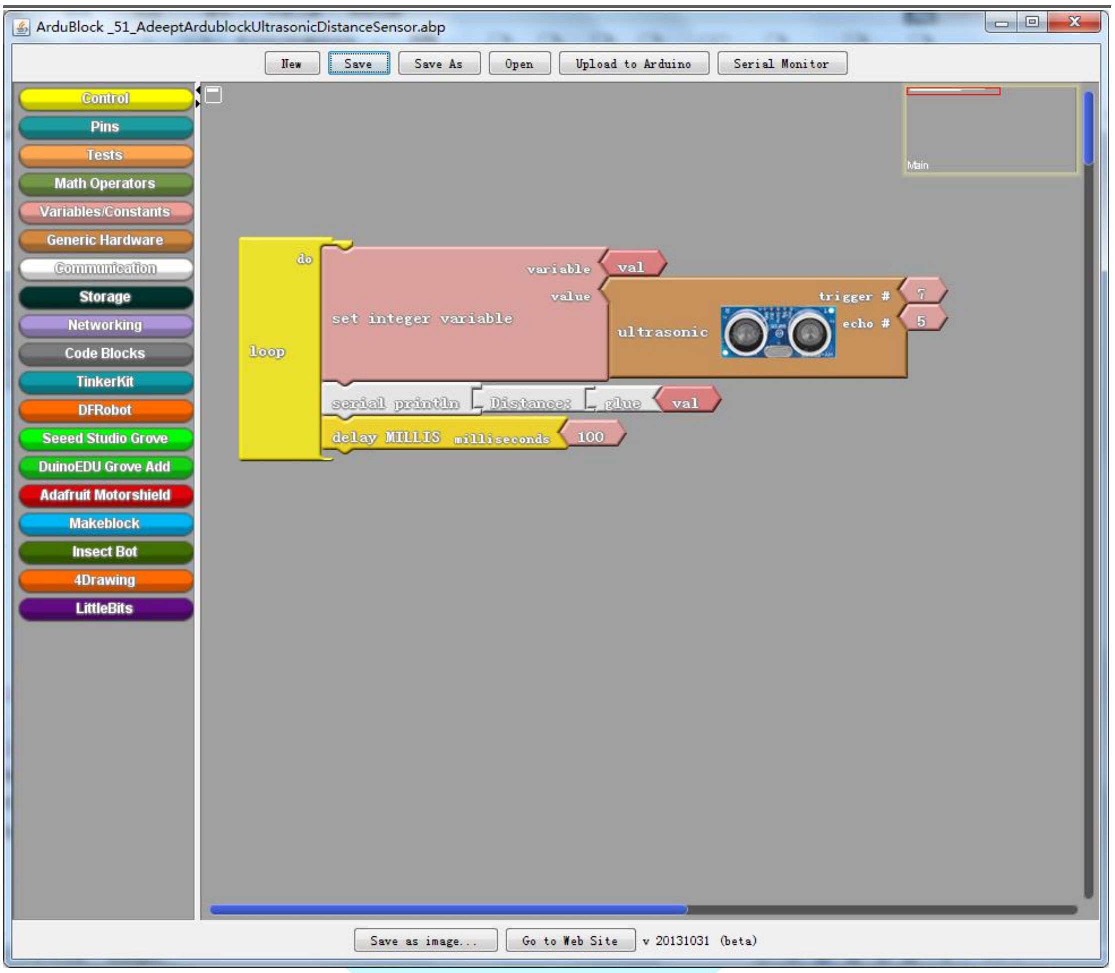
Procedures

1. Build the circuit

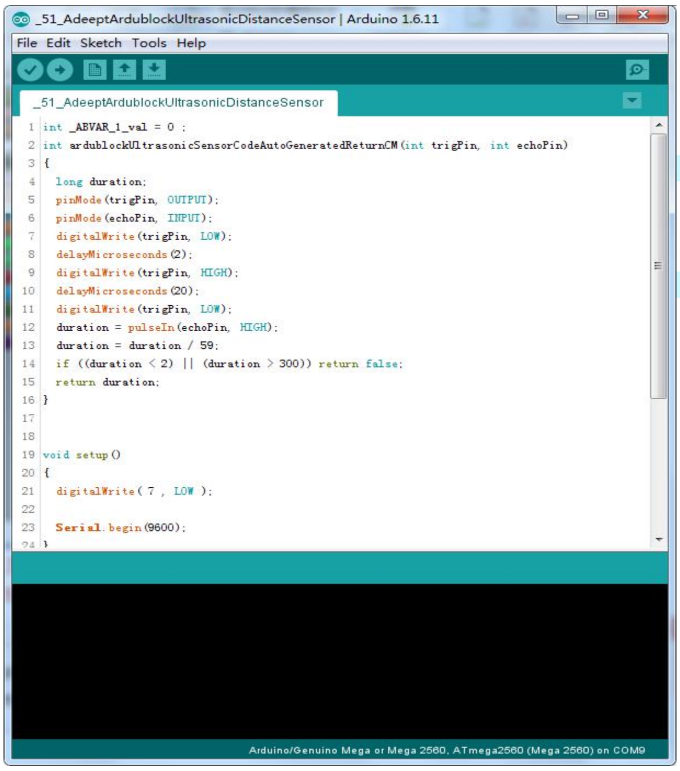


2. Program

Adeept



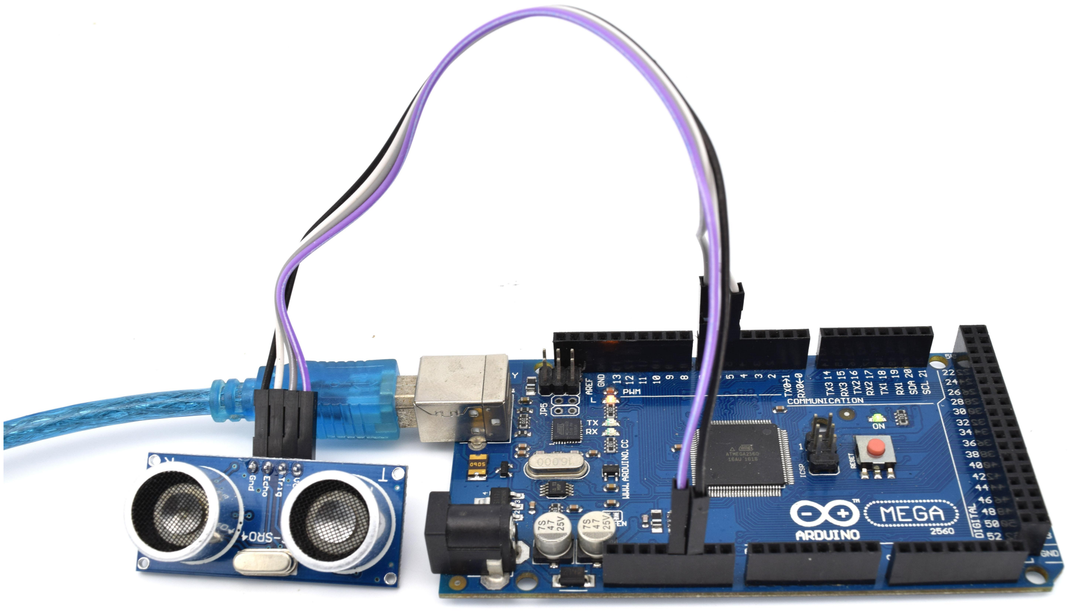
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_51_AdeeptArdublockUltrasonicDistanceSensor | Arduino 1.6.11
File Edit Sketch Tools Help
_51_AdeeptArdublockUltrasonicDistanceSensor

```
int _ABVAR_1_val = 0 ;
int ardublockUltrasonicSensorCodeAutoGeneratedReturnCM(int trigPin, int echoPin)
{
  long duration;
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(20);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  duration = duration / 59;
  if ((duration < 2) || (duration > 300)) return false;
  return duration;
}


void setup()
{
  digitalWrite( 7 , LOW );

  Serial.begin(9600);
}
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Now, when you try to change the distance between the ultrasonic module and the obstacles, you will find the distance value displayed on the LCD1602 will be changed.



Adeept

Lesson 52 Adeept ardublock joystick module

Introduction

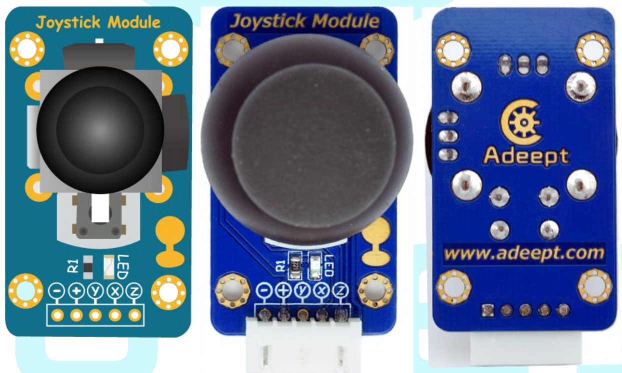
The PS2 Joystick Module is an input device. It consists of a station and the control knob onside. It functions by sending angle or direction signals to the device controlled. The button on the module can also be recognized by the microcontroller. The module supports two-channel analog output, namely, x- and y-axis offset, and one-channel digital output which indicates whether the user has pressed the button at z-axis or not. The Joystick Module can be used to easily control the object to move in a three-dimensional space. For example, it can be applied to control crane, truck, electronic games, robots, etc.

Components

- 1 * Arduino MEGA 2560
- 1 * Joystick Module
- 1 * USB Cable
- 1 * 5-Pin Wires

Experimental Principle

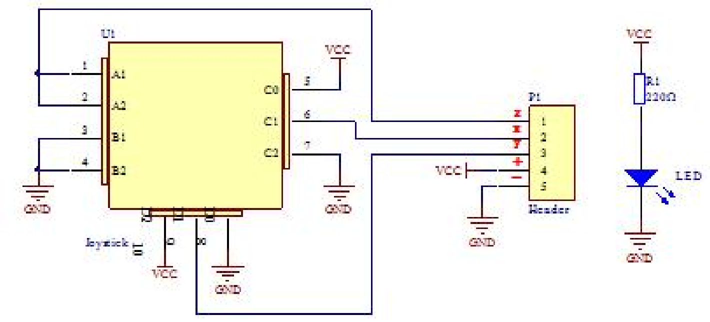
The Fritzing image:



Pin definition:

z	Digital key output
x	Analog output(X)
y	Analog output(Y)
+	VCC
-	GND

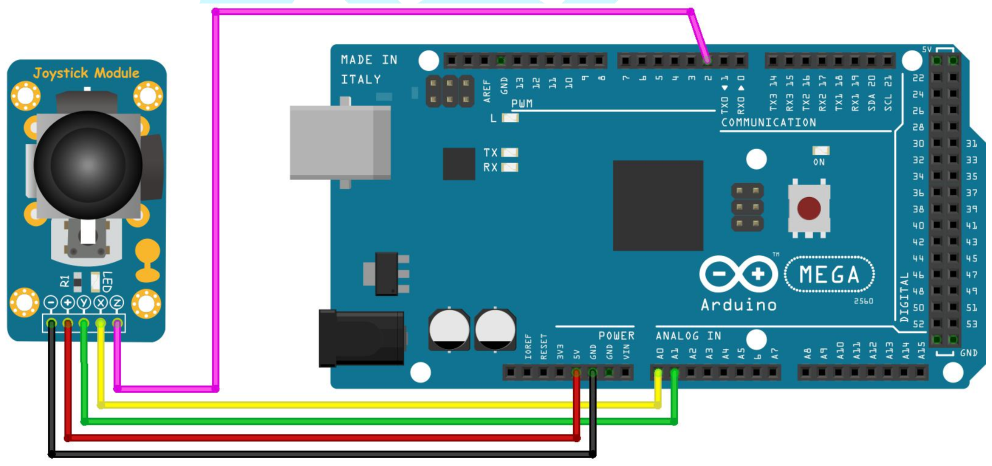
The schematic diagram:



The experiment reads the status of the PS2 Joystick Module, send the data to and display it on Serial Monitor.

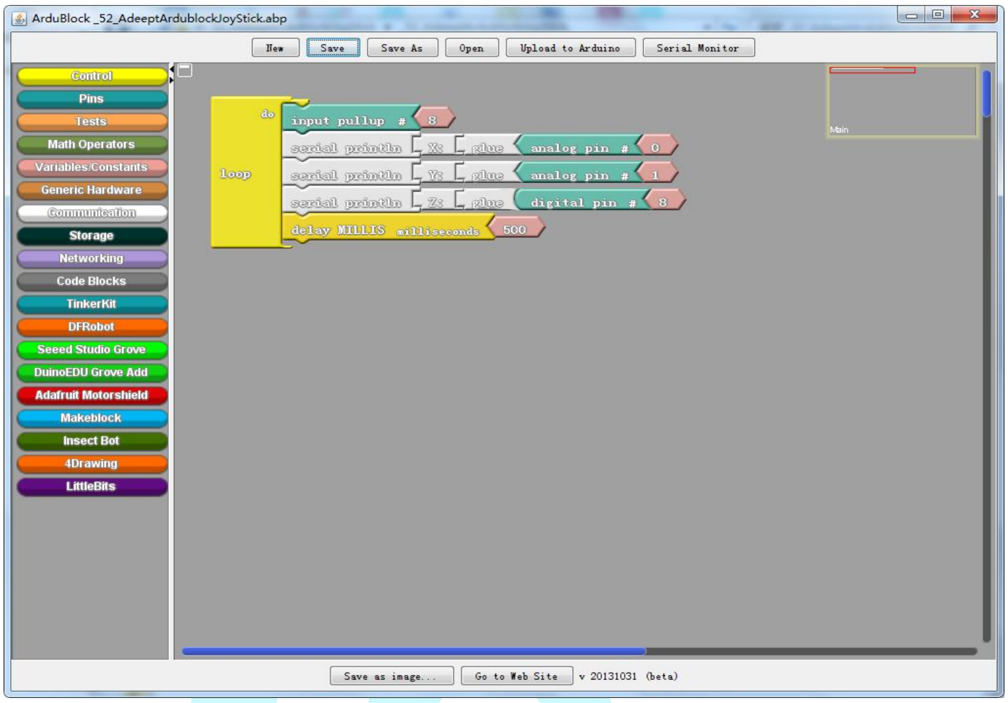
Experimental Procedures

Step 1: Build the circuit



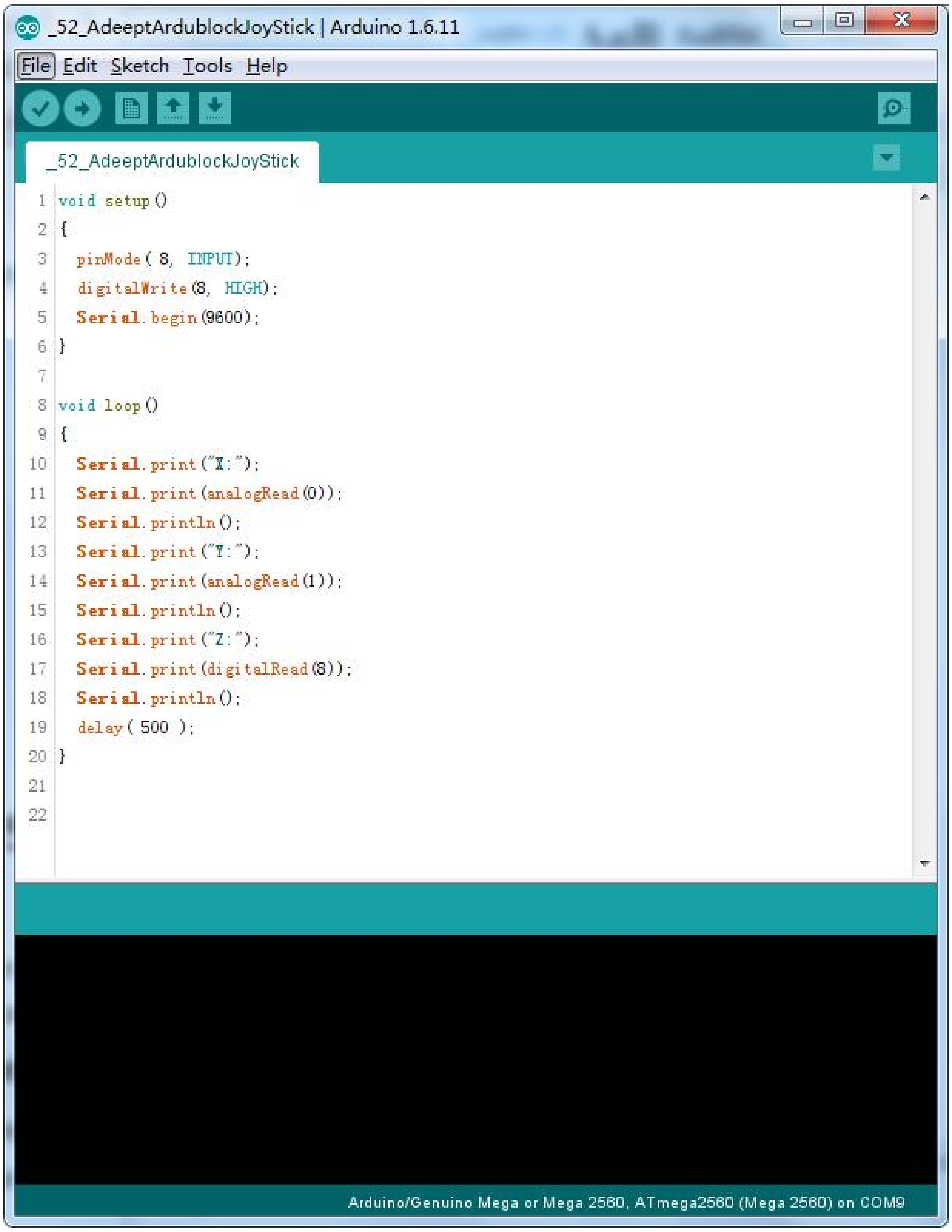
Step 2: Program

Adeept



```
loop
  do
    input pullup # 8
    serial println  X:  glue  analog pin # 0
    serial println  Y:  glue  analog pin # 1
    serial println  Z:  glue  digital pin # 8
    delay MILLIS milliseconds 500
```

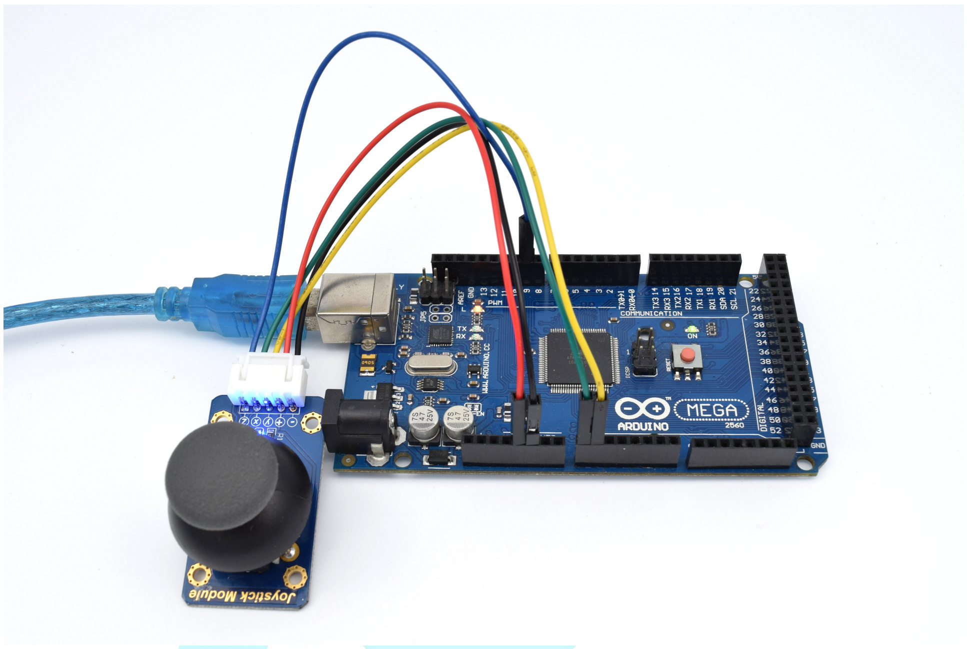
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_52_AdeeptArdublockJoyStick | Arduino 1.6.11
File Edit Sketch Tools Help
_52_AdeeptArdublockJoyStick

```
void setup()
{
  pinMode( 8, INPUT);
  digitalWrite(8, HIGH);
  Serial.begin(9600);
}

void loop()
{
  Serial.print("X:");
  Serial.print(analogRead(0));
  Serial.println();
  Serial.print("Y:");
  Serial.print(analogRead(1));
  Serial.println();
  Serial.print("Z:");
  Serial.print(digitalRead(8));
  Serial.println();
  delay( 500 );
}
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Open Serial Monitor of the Arduino IDE. Press or pull the knob and you will see the value of current status displayed on the window.



Adeept

Lesson 53 Adeept ardublock photoresistor

Overview

In this lesson, we will learn how to measure the light intensity by photoresistor and make the measurement result displayed on the LCD1602.

Requirement

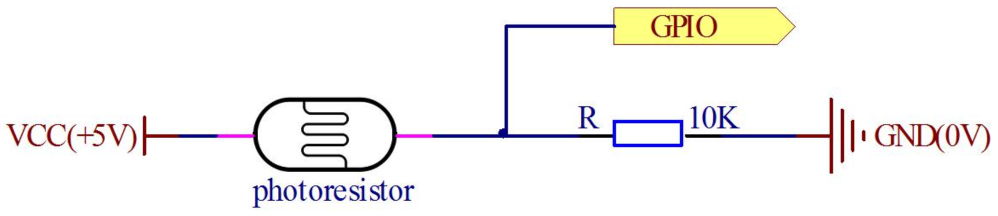
- 1* Arduino MEGA 2560
- 1* USB Cable
- 1* LCD1602
- 1* Photoresistor
- 1* 10KΩ Resistor
- 1* 10KΩ Potentiometer
- 1* Breadboard
- Several Jumper Wires

Principle

A photoresistor is a light-controlled variable resistor. The resistance of a photoresistor decreases with the increasing incident light intensity; in other words, it exhibits photoconductivity. A photoresistor can be applied in light-sensitive detector circuits.

A photoresistor is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as a few megohms (MΩ), while in the light, a photoresistor can have a resistance as low as a few hundred ohms. If incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons (and their hole partners) conduct electricity, thereby lowering resistance. The resistance range and sensitivity of a photoresistor can substantially differ among dissimilar devices. Moreover, unique photoresistors may react substantially differently to photons within certain wavelength bands.

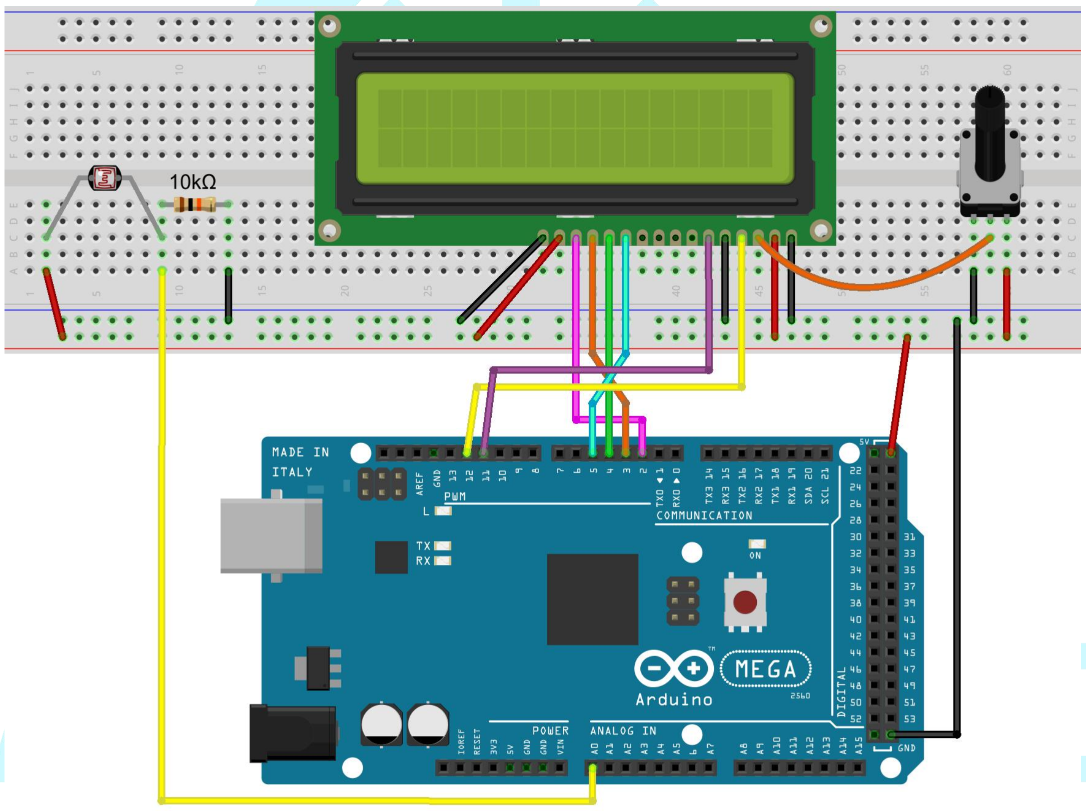
The schematic diagram of this experiment is shown below:



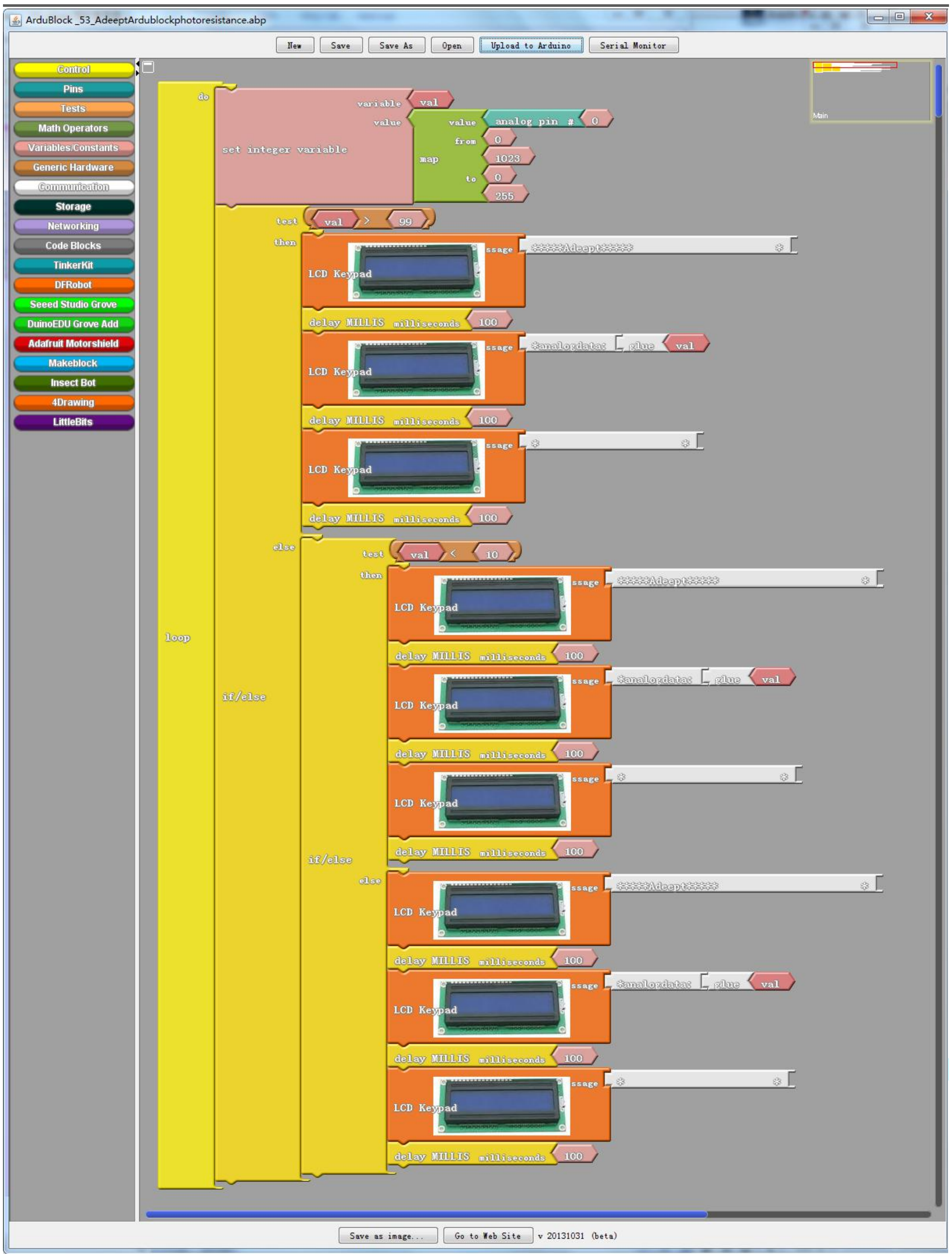
With the increase of the light intensity, the resistance of photoresistor will be decreased. The voltage of GPIO port in the above figure will become high.

Procedures

1. Build the circuit

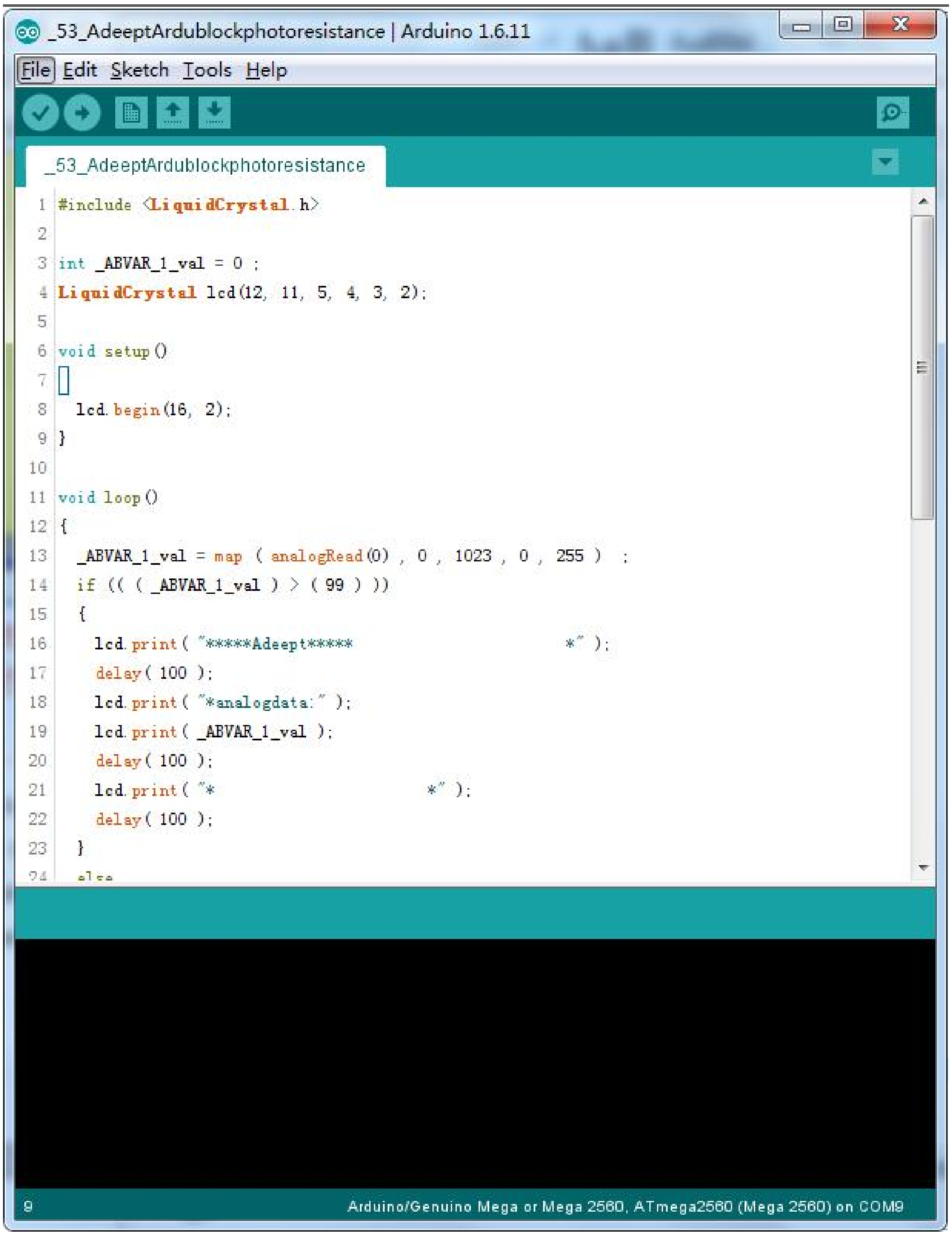


2. Program



```
loop
do
  set integer variable  variable val
                        value map  value analog pin # 0
                                   from 0
                                   1023
                                   to 0
                                   255
  if/else
    test val > 99
    then
      LCD Keypad  ssage  ****Adeept****
      delay MILLIS milliseconds 100
      LCD Keypad  ssage  analogdata: glue val
      delay MILLIS milliseconds 100
      LCD Keypad  ssage
      delay MILLIS milliseconds 100
    else
      if/else
        test val < 10
        then
          LCD Keypad  ssage  ****Adeept****
          delay MILLIS milliseconds 100
          LCD Keypad  ssage  analogdata: glue val
          delay MILLIS milliseconds 100
          LCD Keypad  ssage
          delay MILLIS milliseconds 100
        else
          LCD Keypad  ssage  ****Adeept****
          delay MILLIS milliseconds 100
          LCD Keypad  ssage  analogdata: glue val
          delay MILLIS milliseconds 100
          LCD Keypad  ssage
          delay MILLIS milliseconds 100
```

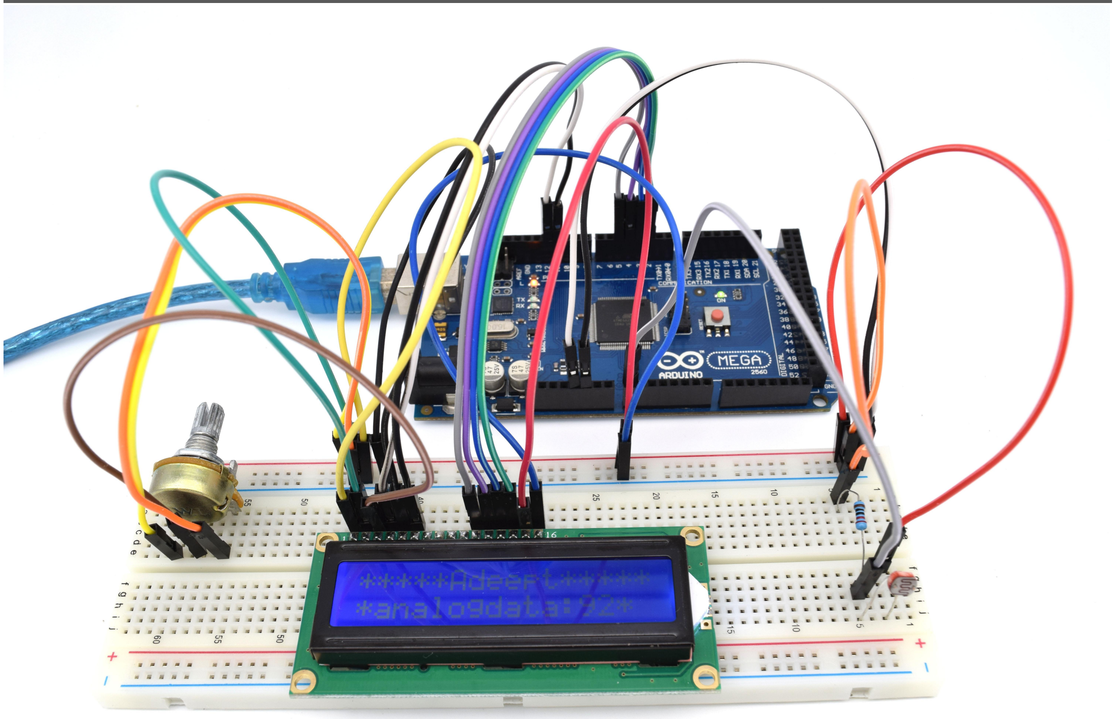
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:


_53_AdeeptArdublockphotoresistance | Arduino 1.6.11
File Edit Sketch Tools Help
_53_AdeeptArdublockphotoresistance

```
#include <LiquidCrystal.h>

int _ABVAR_1_val = 0 ;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  lcd.begin(16, 2);
}

void loop()
{
  _ABVAR_1_val = map ( analogRead(0) , 0 , 1023 , 0 , 255 )  ;
  if (( ( _ABVAR_1_val ) > ( 99 ) ))
  {
    lcd.print( "*****Adeept*****                         *" );
    delay( 100 );
    lcd.print( "*analogdata:" );
    lcd.print( _ABVAR_1_val );
    delay( 100 );
    lcd.print( "*                         *" );
    delay( 100 );
  }
  else
```

```
9                    Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9
```

Now, when you try to block the light towards the photoresistor, you will find that the value displayed on the LCD1602 will be reduced. Otherwise, when you use a powerful light to irradiate the photoresistor, the value displayed on the LCD1602 will be increased.



Summary

By learning this lesson, we have learned how to detect surrounding light intensity with the photoresistor. You can play your own wisdom, and make more originality based on this experiment and the former experiment.

Adeept

Lesson 54 Adeept ardublock soil moisture sensor module

Introduction

The Soil Moisture Sensor module is a simple sensor that measures the soil moisture. When the soil moisture is insufficient, the output value of the sensor will decrease; on the other hand, the value will increase when there's enough water. The surface of the sensor is gilded to prolong its life.
The CM Module consists of a comparator LM393 and extremely simple external circuits. When using the module, you can set a threshold via the blue potentiometer beforehand. When the input analog value reaches the threshold, the digital pin S will output a Low level.

Components

- 1 * Arduino MEGA 2560
- 1 * Soil Moisture Sensor Module
- 1 * CM Module
- 1 * USB Cable
- 1 * 4-Pin Wires
- 1 * 2-Pin Female to Female Wires

Experimental Principle

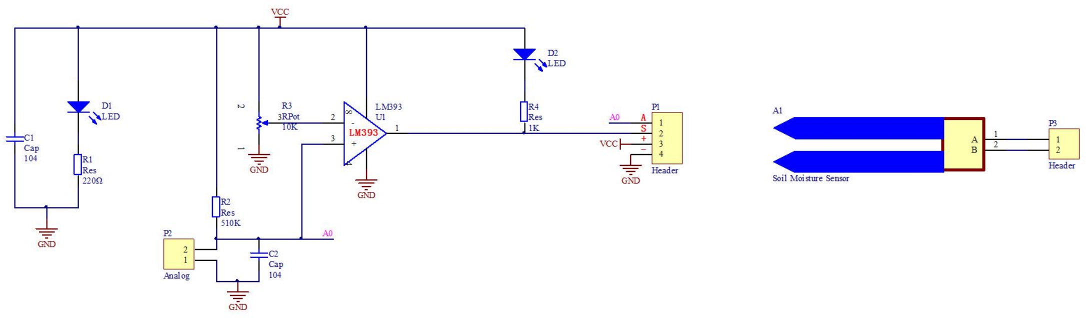
The Fritzing images:

Adeept

Pin definition:

Soil Moisture Sensor Module	
1	Analog output
2	Analog output
CM Module	
1	Analog output
2	Analog output
S	Digital output
A	Analog output
+	VCC
-	GND

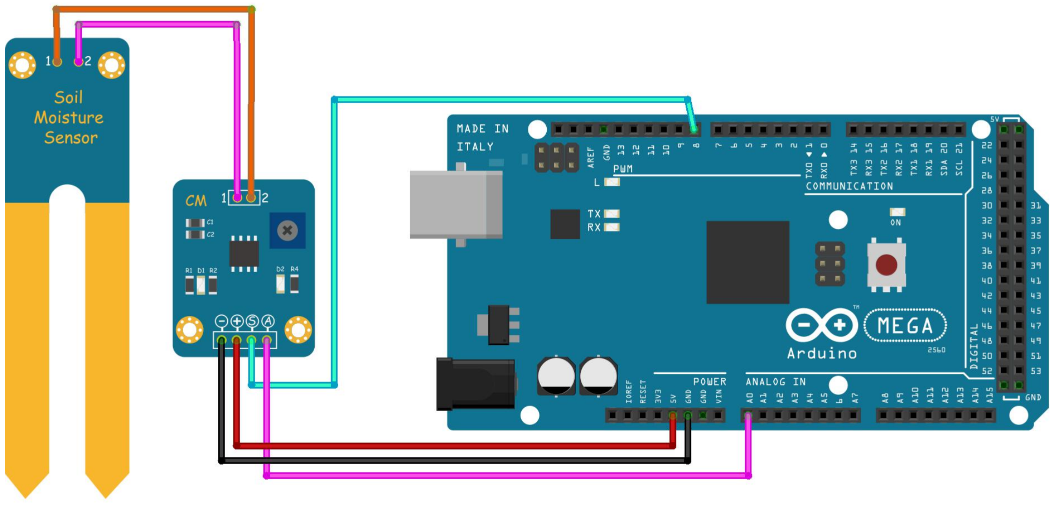
The schematic diagram:



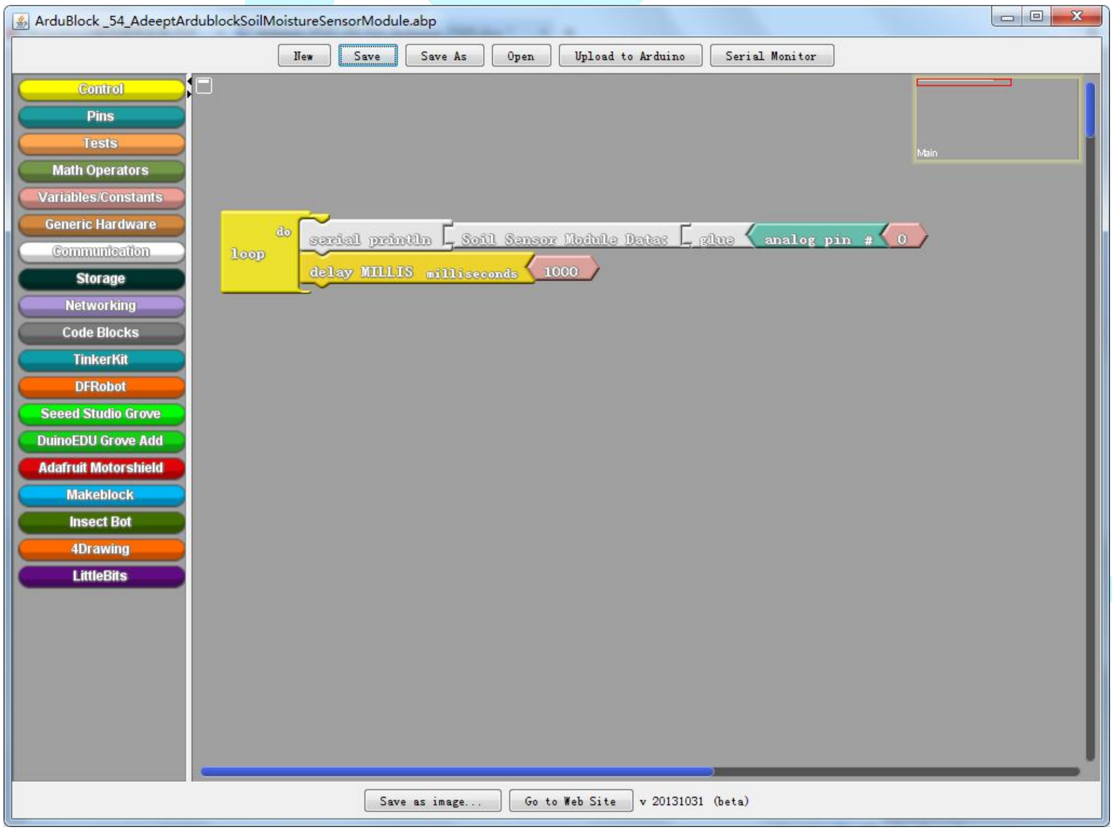
The experiment uses the Soil Moisture Sensor module to collect data of soil moisture and display it on Serial Monitor.

Experimental Procedures

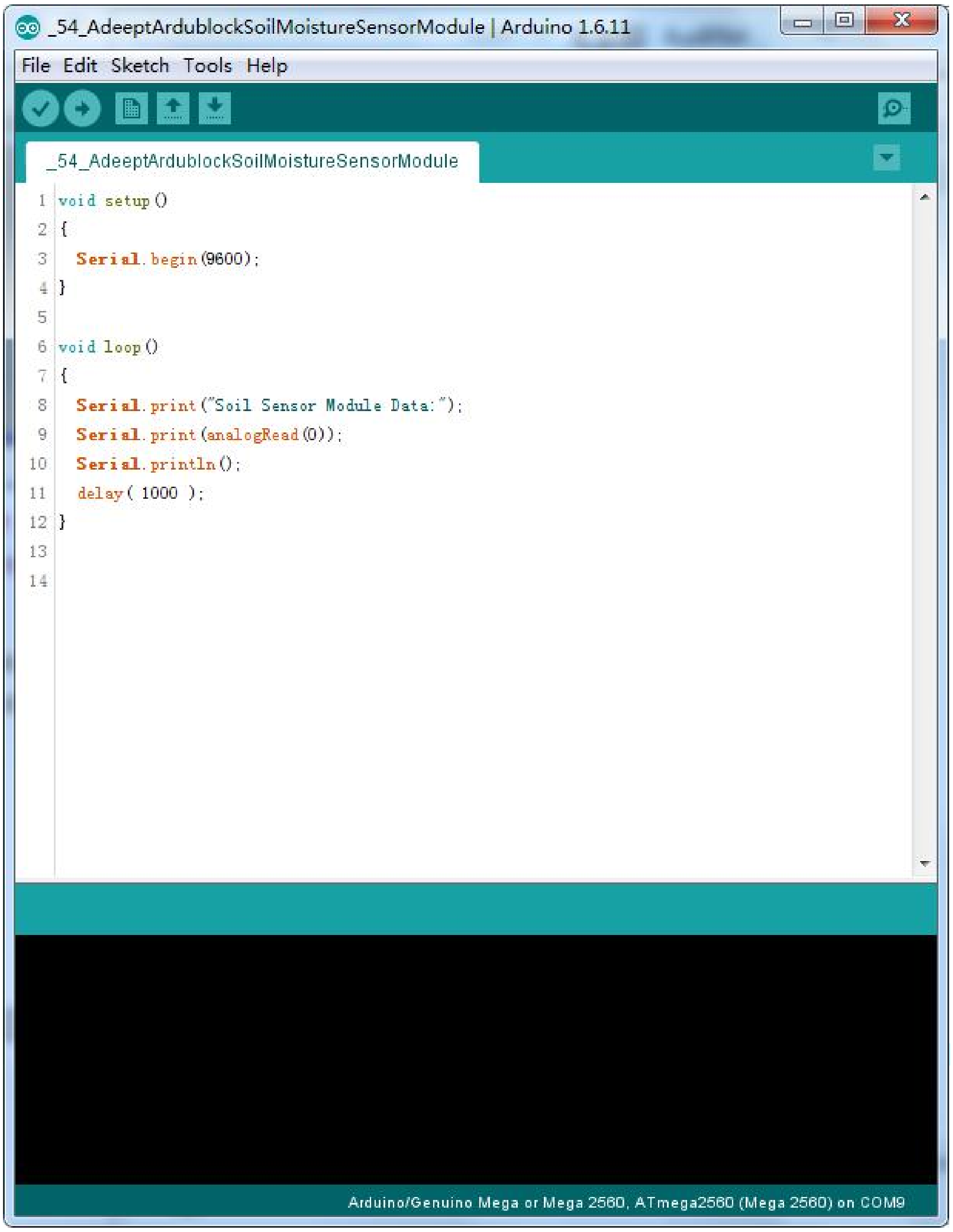
Step 1: Build the circuit



Step 2: Program



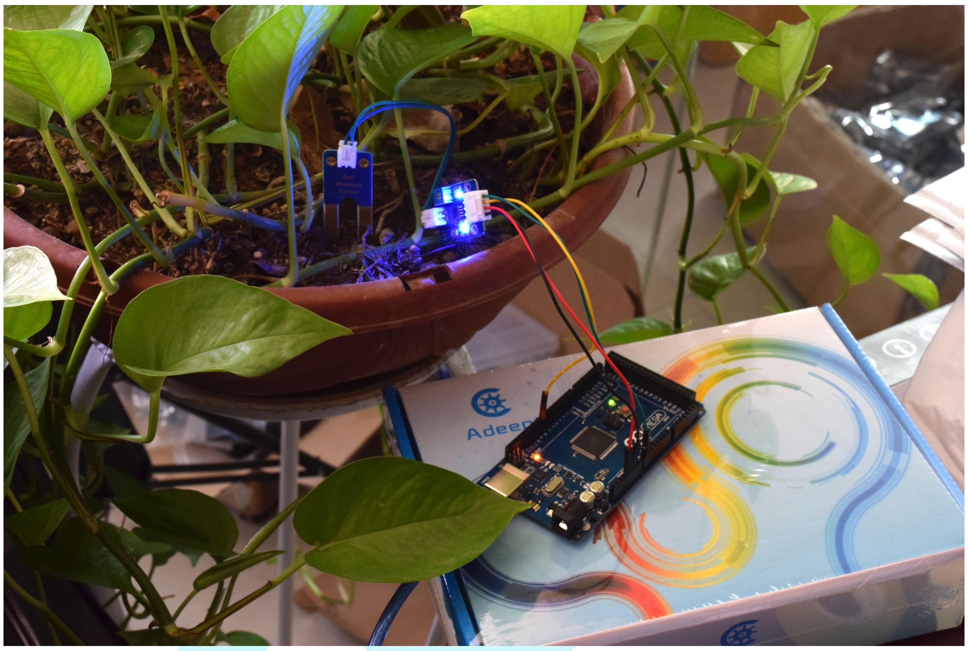
3. Compile the program and upload to Arduino MEGA 2560 board, code programming interface will appear:

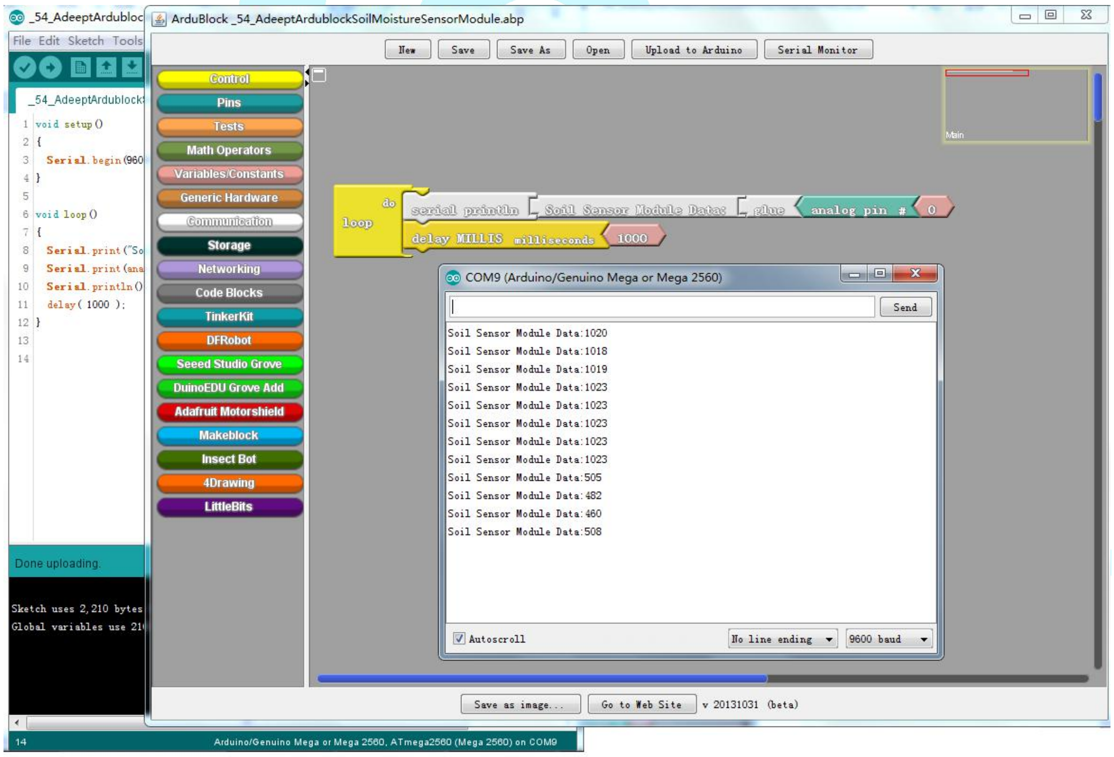

_54_AdeeptArdublockSoilMoistureSensorModule | Arduino 1.6.11
File Edit Sketch Tools Help
_54_AdeeptArdublockSoilMoistureSensorModule

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.print("Soil Sensor Module Data:");
  Serial.print(analogRead(0));
  Serial.println();
  delay( 1000 );
}
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM9

Open Serial Monitor of the Arduino IDE. You will see the value of soil moisture collected by the module displayed on the window.





```
void setup()
{
  Serial.begin(960
}

void loop()
{
  Serial.print("So
  Serial.print(ana
  Serial.println()
  delay( 1000 );
}
```

```
Done uploading.

Sketch uses 2,210 bytes
Global variables use 21
```

```
Soil Sensor Module Data:1020
Soil Sensor Module Data:1018
Soil Sensor Module Data:1019
Soil Sensor Module Data:1023
Soil Sensor Module Data:1023
Soil Sensor Module Data:1023
Soil Sensor Module Data:1023
Soil Sensor Module Data:1023
Soil Sensor Module Data:505
Soil Sensor Module Data:482
Soil Sensor Module Data:460
Soil Sensor Module Data:508
```

Adeept

Sharing Perfects Innovation

E-mail: support@adeept.com
website: www.adeept.com