



Ultimate Sensor kit

for UNO

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. We will do our best to help you solve the problem.



Adeept

Component List

Picture



1

Name • **Arduino compatible
UNO R3**

Qty • 1

Picture



2

Name • **LED Module**

Qty • 4

Picture



3

Name • **Button Module**

Qty • 4

Picture



4

Name • **RGB LED Module**

Qty • 1

Picture



5

Name • **Potentiometer
Module**

Qty • 1

Picture



6

Name • **Vibration Sensor
Module**

Qty • 1

Picture



7

Name • **Hall Sensor Module**

Qty • 1

Picture



8

Name • **Photoresistor
Module**

Qty • 1

Picture



9

Name • **Thermistor Module**

Qty • 1

Picture



10

Name • **DS18B20 Module**

Qty • 1

Picture



11

Name • **Active Buzzer Module**

Qty • 1

Picture



12

Name • **Passive Buzzer Module**

Qty • 1

Picture



13

Name • **PIR Sensor Module**

Qty • 1

Picture



14

Name • **LCD1602 Module**

Qty • 1

Picture



15

Name • **IIC Interface Module**

Qty • 1

Picture




16

Name • **Touch Button Module**

Qty • 1

Picture



17

Name • **Rotary Encoder Module**

Qty • 1

Picture



18

Name • **Limit Switch Module**

Qty • 1

Picture



19

Name • **Laser Transmitter Module**

Qty • 1

Picture




20

Name • **Laser Receiver Module**

Qty • 1

Picture



21

Name • **DHT-11 Sensor Module**

Qty • 1

Picture



22

Name • **Reed Module**

Qty • 1

Picture



23

Name • **Flame Sensor Module**

Qty • 1

Picture

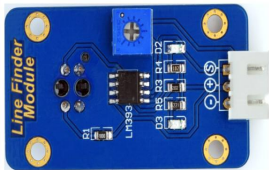


24

Name • **MQ-2 Gas Sensor Module**

Qty • 1

Picture



25

Name • **Line Finder Module**

Qty • 1

Picture



26

Name • **Slide Potentiometer Module**

Qty • 1

Picture



27

Name • **DC Motor Module**

Qty • 1

Picture



28

Name • **Joystick Module**

Qty • 1

Picture



29

Name • **MIC Module**

Qty • 1

Picture




30

Name • **Relay Module**

Qty • 1

Picture



31

Name • **Segment Display Module**

Qty • 1

Picture



32

Name • **8*8 LED Matrix Module**

Qty • 1

Picture



33

Name • **LED Bar Module**

Qty • 1

Picture



34

Name • **IR Receiver Module**

Qty • 1

Picture



35

Name • **Remote Controller**

Qty • 1

Picture

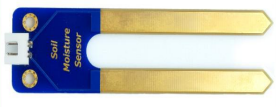


36

Name • **CM Module**

Qty • 1

Picture




37

Name • **Soil Moisture Sensor Module**

Qty • **1**

Picture




38

Name • **Water Level Sensor Module**

Qty • **1**

Picture



39

Name • **Ultrasonic Distance Sensor Module**

Qty • **1**

Picture




40

Name • **3-Pin Wires**

Qty • **8**

Picture



41

Name • **4-Pin Wires**

Qty • **5**

Picture



42

Name • **5-Pin Wires**

Qty • **3**

Picture




43

Name • **USB Cable**

Qty • **1**

Picture




44

Name • **Hookup Wire Set**

Qty • **1**

Picture



45

Name • **2-Pin Female to Female Wires**

Qty • **1**

Picture

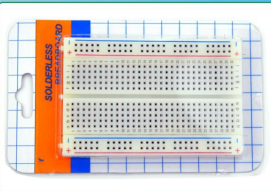


46

Name • **Male to Female Jumper Wires**

Qty • **20**

Picture

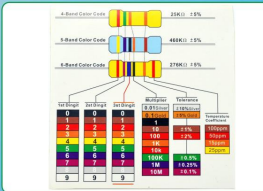


47

Name • **Breadboard**

Qty • **1**

Picture



48

Name • **Band Resistor Card**

Qty • **1**

Adeept

Contents

About Arduino.....	1
About Processing.....	2
Lesson 1 Blinking LED.....	3
Lesson 2 Controlling An LED by A Button.....	6
Lesson 3 Controlling An RGB LED by PWM.....	9
Lesson 4 How To Use Potentiometers.....	12
Lesson 5 Control An LED by Vibration.....	16
Lesson 6 How To Use The Hall Sensor.....	19
Lesson 7 How To Use The Photoresistor.....	23
Lesson 8 How To Use The Thermistor.....	27
Lesson 9 How To Use The DS18B20.....	31
Lesson 10 Alarm Prompt.....	37
Lesson 11 Playing Music.....	40
Lesson 12 Detection of Human Body Movement.....	43
Lesson 13 How To Use The LCD1602.....	47
Lesson 14 IIC Interface Application.....	52
Lesson 15 Application of Touch Button.....	58
Lesson 16 Pulse Count.....	62
Lesson 17 Impact checking.....	66
Lesson 18 Simple Laser Cannon.....	70
Lesson 19 Simple Laser Targeting.....	73
Lesson 20 Temperature And Humidity Detection.....	77
Lesson 21 How To Use The Reed.....	82
Lesson 22 Fire Detection.....	86
Lesson 23 Decetion of Flammable Gases.....	90
Lesson 24 Tracking Test.....	94
Lesson 25 How To Use Slide Potentiometer.....	98
Lesson 26 Small Fan Works.....	102
Lesson 27 How To Use The Joystick.....	105
Lesson 28 How To Use The MIC Module.....	109
Lesson 29 How To Use The Relay Module.....	113
Lesson 30 How To Use The Segment Module.....	117
Lesson 31 How To Use The 8*8 LED Matrix.....	121
Lesson 32 Indication of Signal.....	124
Lesson 33 Making A Simple Remote Control Device.....	129
Lesson 34 Detection of The Soil Moisture System.....	134
Lesson 35 Detection of The Water Height System.....	139
Lesson 36 Detection of The Distance System.....	144
Lesson 37 Control An LED by PC.....	148
Lesson 38 Upload The State of A Button to PC.....	151
Lesson 39 Simple Laser Pen.....	154

Lesson 40 Control Buzzer by Button.....	157
Lesson 41 A Simple Piano.....	160
Lesson 42 Change The Color of The RGB LED.....	163
Lesson 43 A Simple Light Control Lamp.....	166
Lesson 44 Control Segment Display by Rotary Encoder.....	169
Lesson 45 A Simple Temperature & Humidity Monitoring and Alarm System(1).....	172
Lesson 46 A Simple Temperature & Humidity Monitoring and Alarm System(2).....	175
Lesson 47 A Simple Flammable Gases Monitoring and Alarm System(1).....	178
Lesson 48 A Simple Flammable Gases Monitoring and Alarm System(2).....	181
Lesson 49 A Simple Clock.....	184
Lesson 50 Arduino Interacts with Processing(IED Module).....	186
Lesson 51 Arduino Interacts with Processing(Button Module).....	207
Lesson 52 Arduino Interacts with Processing(RGB Module).....	212
Lesson 53 Arduino Interacts with Processing(Potentiometer).....	217
Lesson 54 Arduino Interacts with Processing(Vibration Module).....	223
Lesson 55 Arduino Interacts with Processing(Photoresistor).....	229
Lesson 56 Arduino Interacts with Processing(DS18B20 Module).....	234
Lesson 57 Arduino Interacts with Processing(Buzzer Module).....	239
Lesson 58 Arduino Interacts with Processing(Rotary Encoder).....	243
Lesson 59 Arduino Interacts with Processing(Joystick Module).....	251
Lesson 60 Arduino Interacts with Processing(Ultrasonic Distance Module).....	256

Aadeept

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>

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

Lesson 1 Blinking LED

Introduction

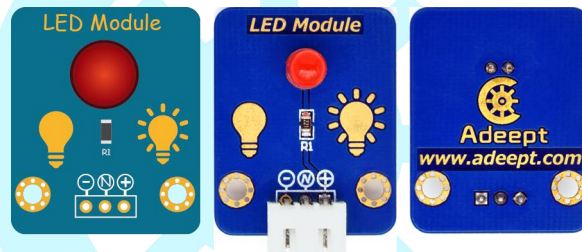
LED is usually used in office lighting, furniture, decoration, sign board, streetlight, etc.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * LED Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

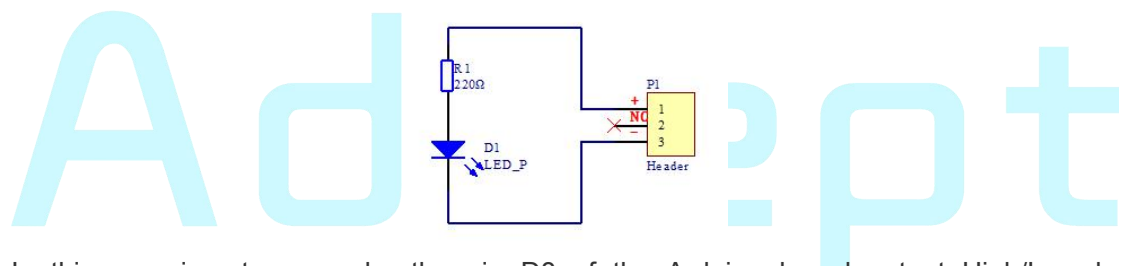
The Fritzing images:



Pin definition:

N	NULL
+	VCC
-	GND

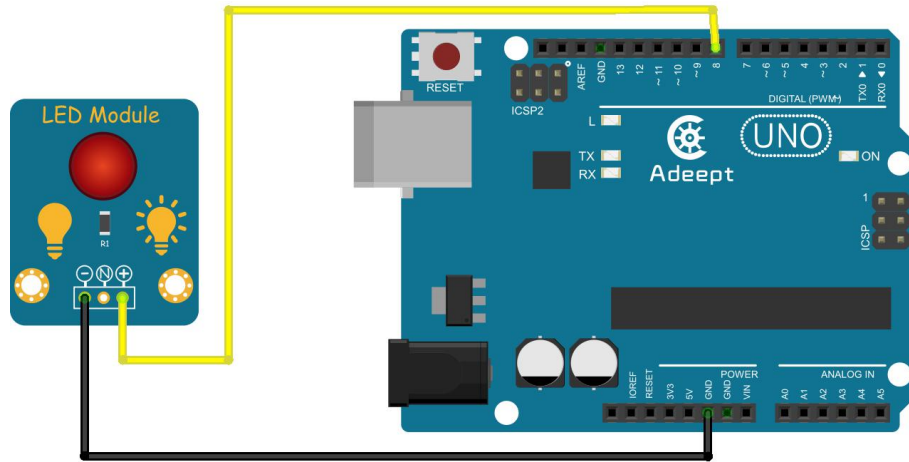
The schematic diagram:



In this experiment, we make the pin D8 of the Arduino board output High/Low by programming, to control the LED to blink in a certain frequency.

Experimental Procedures

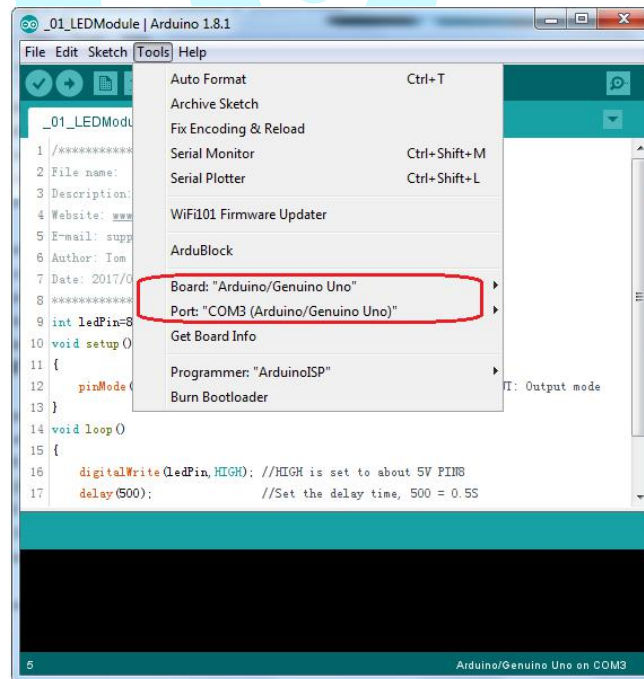
Step 1: Build the circuit

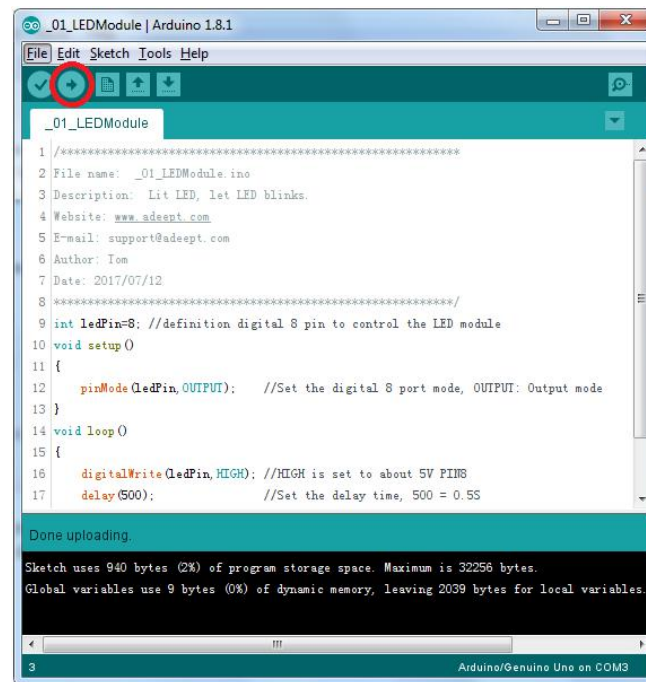


Adept UNO R3 Board	LED Module
D8	+
GND	-

Step 2: Program _01_LEDModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.



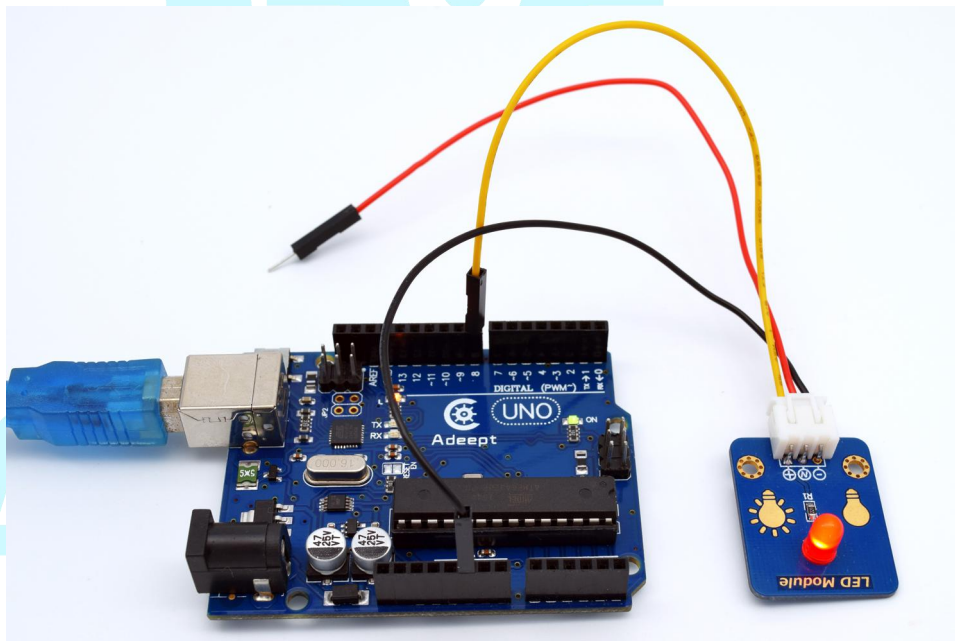


The screenshot shows the Arduino IDE interface with the file "_01_LEDModule" open. The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar has several icons, with the upload icon (a right-pointing arrow) circled in red. The code editor contains the following code:

```
1 /*****  
2 File name: _01_LEDModule.ino  
3 Description: Lit LED, let LED blinks.  
4 Website: www.adeept.com  
5 E-mail: support@adeept.com  
6 Author: Iom  
7 Date: 2017/07/12  
8 *****/  
9 int ledPin=8; //definition digital 8 pin to control the LED module  
10  
11 void setup()  
12 {  
13   pinMode(ledPin, OUTPUT); //Set the digital 8 port mode, OUTPUT: Output mode  
14 }  
15 void loop()  
16 {  
17   digitalWrite(ledPin, HIGH); //HIGH is set to about 5V PINS  
18   delay(500); //Set the delay time, 500 = 0.5S  
19 }
```

Below the code editor, a status bar indicates "Done uploading." and provides memory usage details: "Sketch uses 940 bytes (2%) of program storage space. Maximum is 32256 bytes. Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables." The bottom status bar shows "3" and "Arduino/Genuino Uno on COM3".

Now you can see the LED on the LED module blinks once per second.



Lesson 2 Controlling An LED by A Button

Introduction

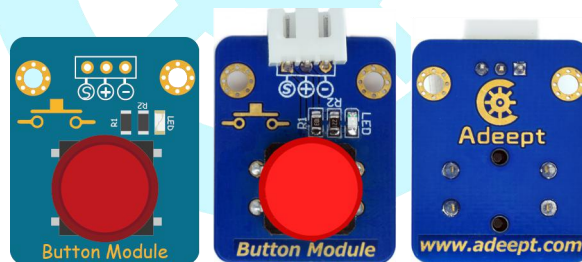
Buttons, or touch switches, are light-touch button switches. A button is an electronic switch and usually used for device control.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * Button Module
- 1 * LED Module
- 1 * USB Cable
- 2 * 3-Pin Wires
- 2 * Hookup Wire Set
- 1 * Breadboard

Experimental Principle

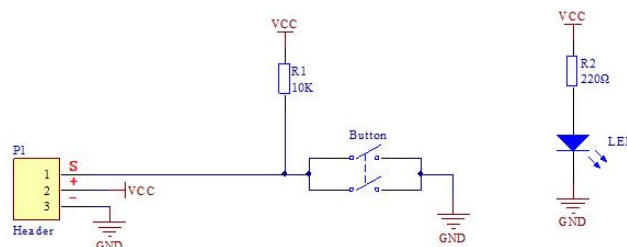
The Fritzing image:



Pin definition:

S	Digital keys output
+	VCC
-	GND

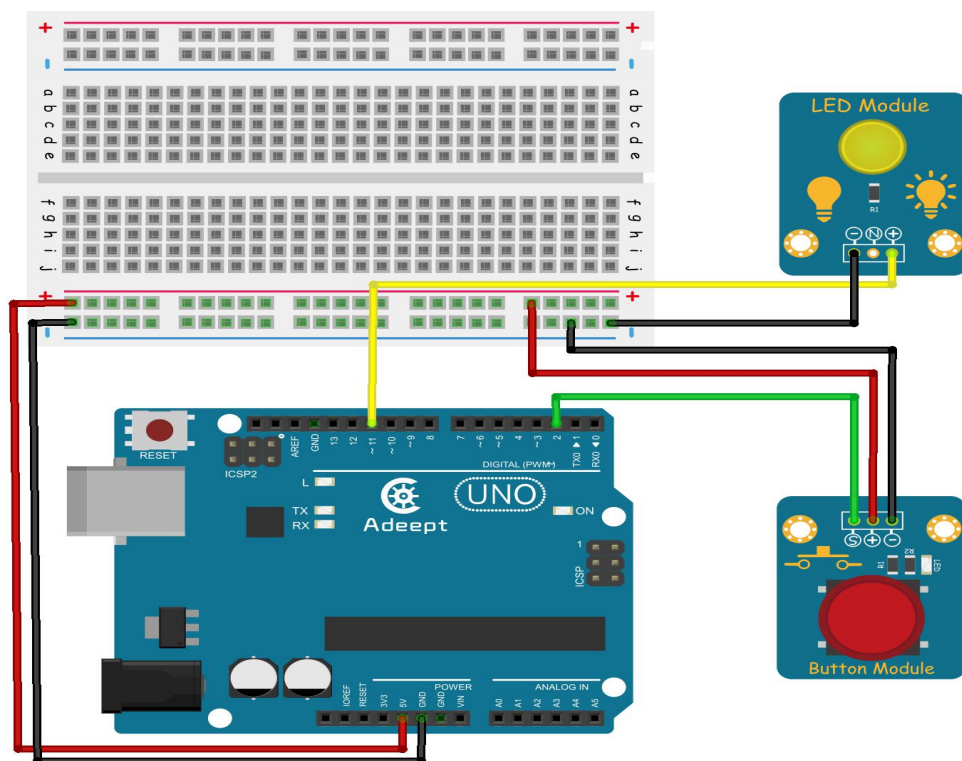
The schematic diagram:



In this experiment, we detect the High or Low level of pin D2 of the Arduino board and then control the LED connected to D8 accordingly.

Experimental Procedures

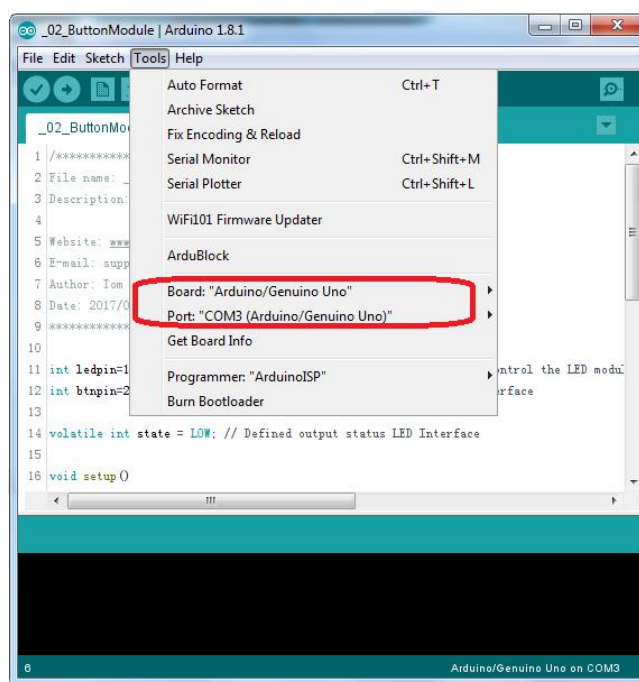
Step 1: Build the circuit

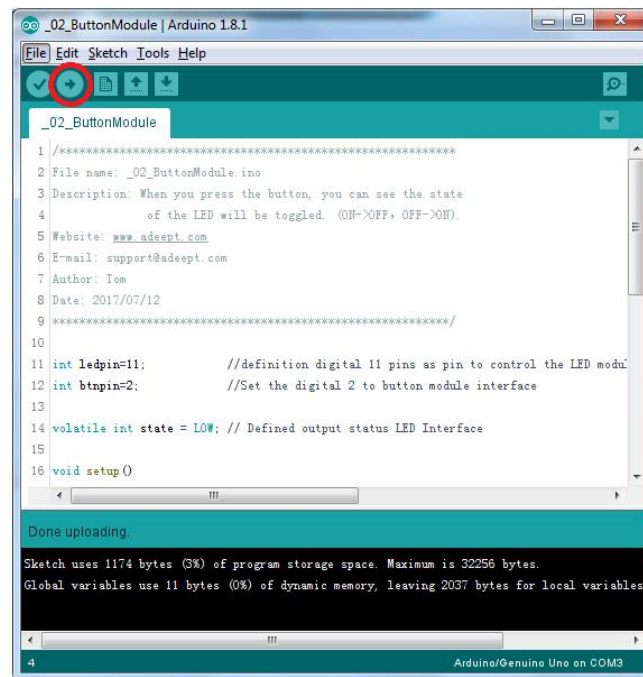


Adeept UNO R3 Board	Button Module	LED Module
GND	-	-
5V	+	
D2	S	
D11		+

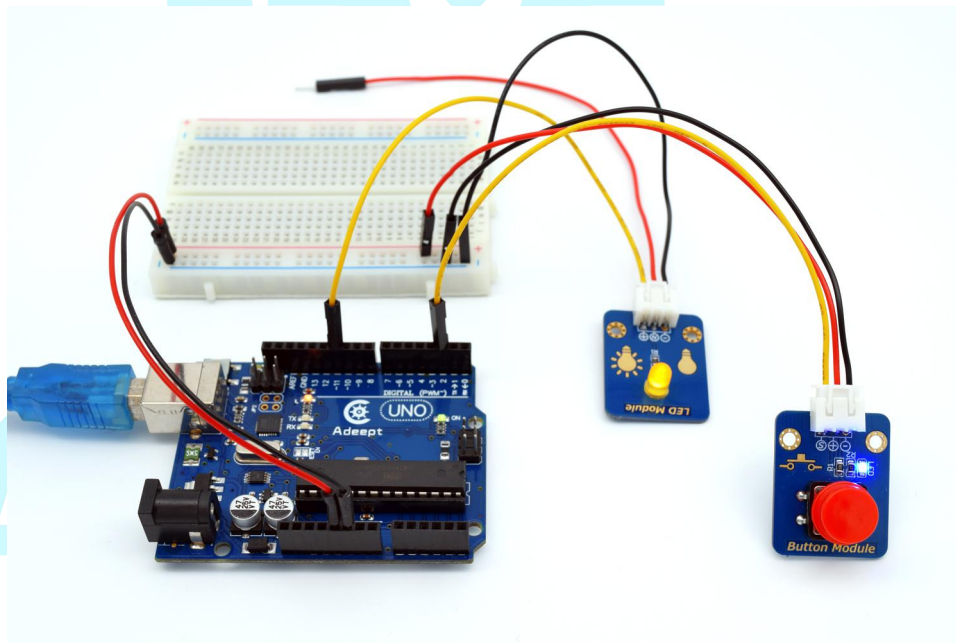
Step 2: Program _02_ButtonModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.





Press the button and you can see the LED toggle between on and off.



Lesson 3 Controlling An RGB LED by PWM

Introduction

RGB LED is designed based on the principle of three primary colors. In an RGB LED, three LEDs in red, green, and blue respectively are packaged together, thus by controlling the brightness of three LEDs, making the RGB LED flash multiple colors

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * RGB LED Module
- 1 * USB Cable
- 1 * 4-Pin Wires

Experimental Principle

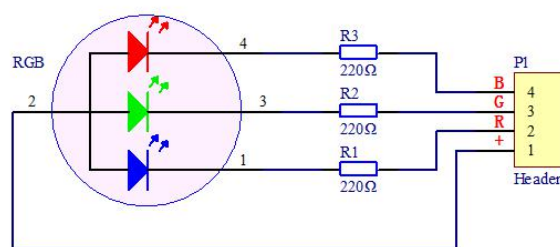
The Fritzing image:



Pin definition:

B	Blue Channel
G	Green Channel
R	Red Channel
+	VCC

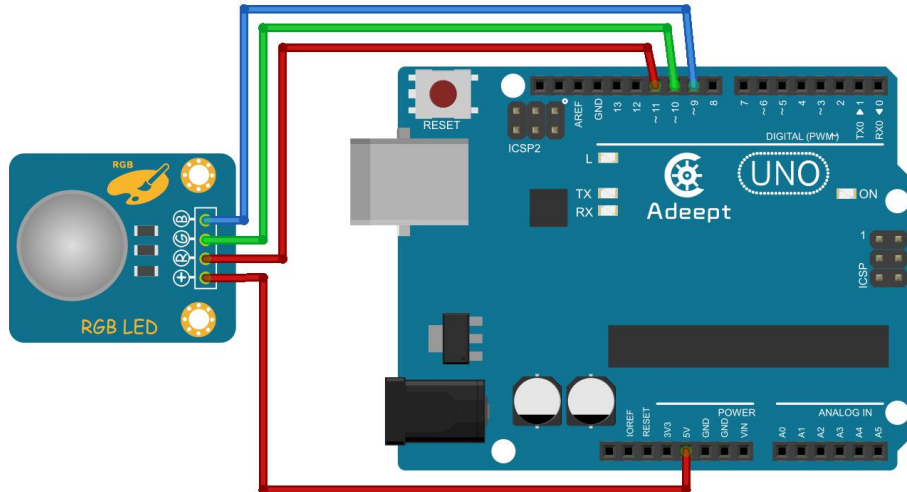
The schematic diagram:



In this experiment, we make the pin D9, D10, and D11 of the Arduino board output PWM (pulse-width modulation) signals by programming, to make the RGB LED flash different colors.

Experimental Procedures

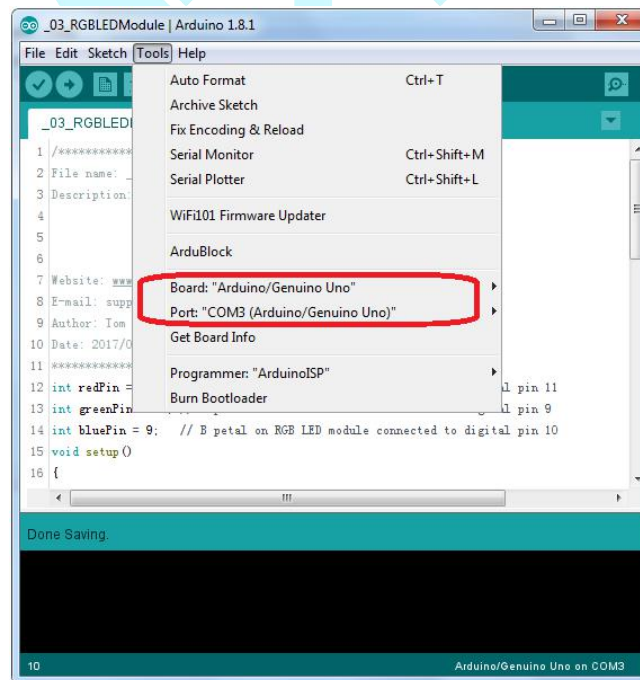
Step 1: Build the circuit

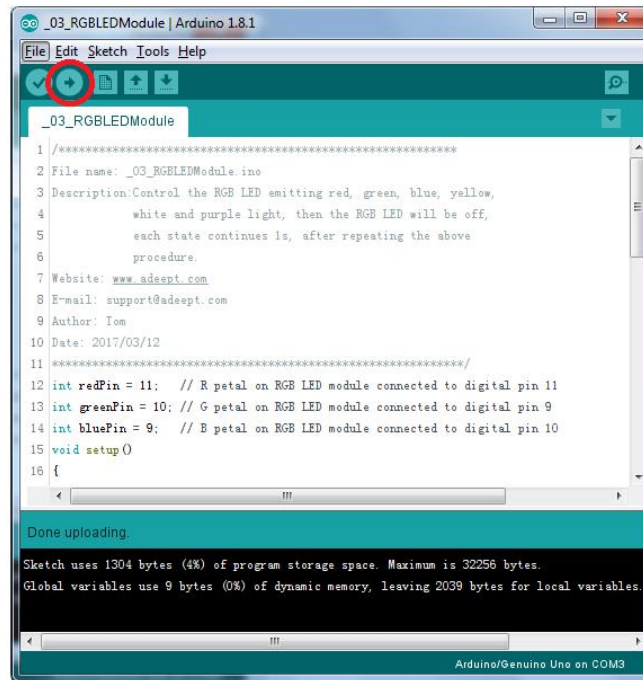


Adept UNO R3 Board	RGB LED Module
D9	B
D10	G
D11	R
5V	+

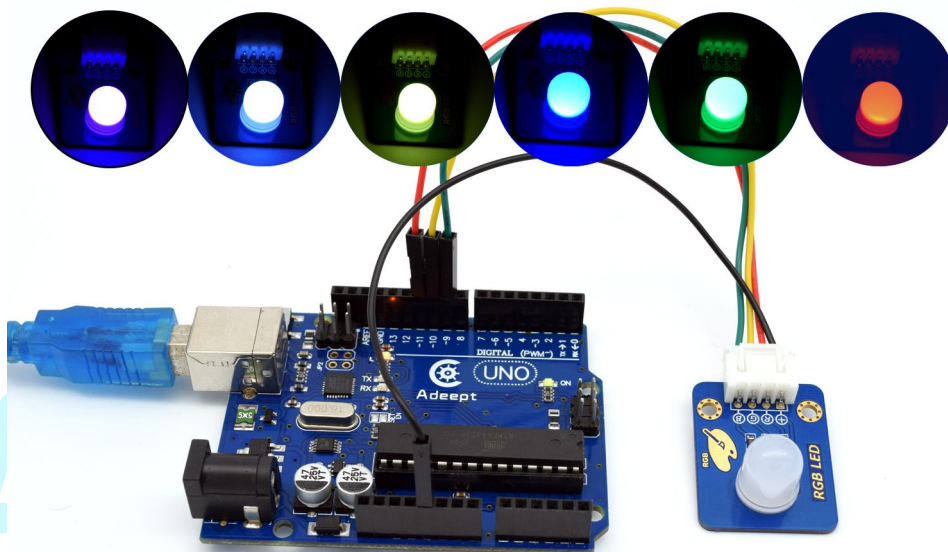
Step 2: Program _03_RGBLEDModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.





Now you can see the RGB LED flash different colors alternately.



Lesson 4 How To Use Potentiometers

Introduction

Potentiometer is a resistor whose resistance can be adjusted continuously. When its shaft is turned, the moving contact (or wiper) slides along the resistor.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Potentiometer Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

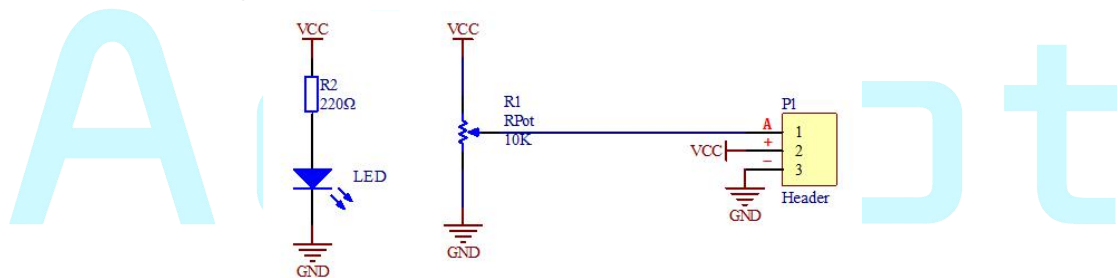
The Fritzing image:



Pin definition:

A	Analog output
+	VCC
-	GND

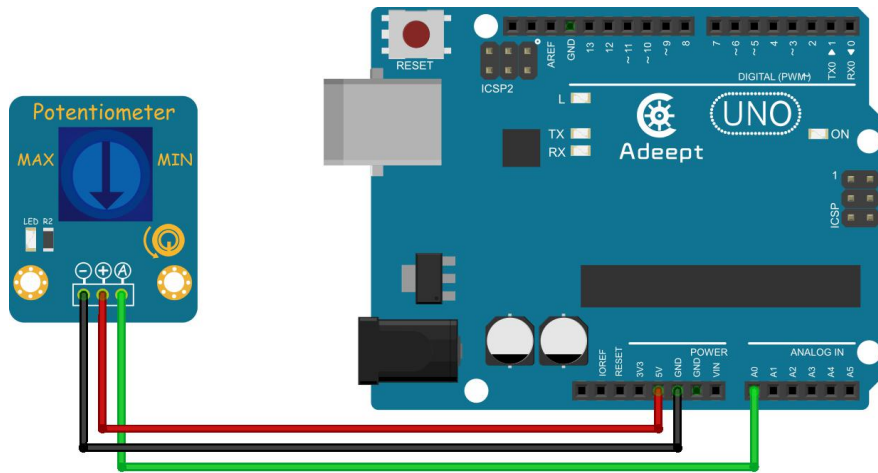
The schematic diagram:



In this experiment, by programming the Arduino, we collect the analog values output by the Potentiometer module through pin A0 of the Arduino board, convert it to digital values and display them on the computer via serial port.

Experimental Procedures

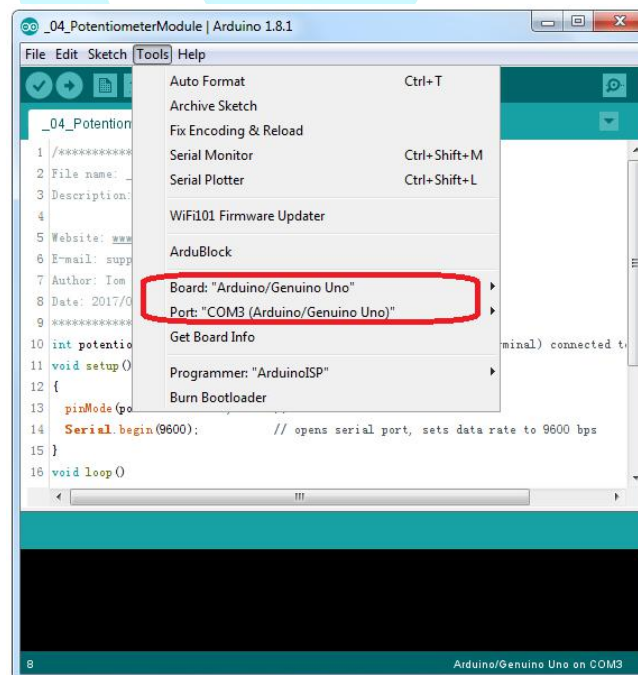
Step 1: Build the circuit

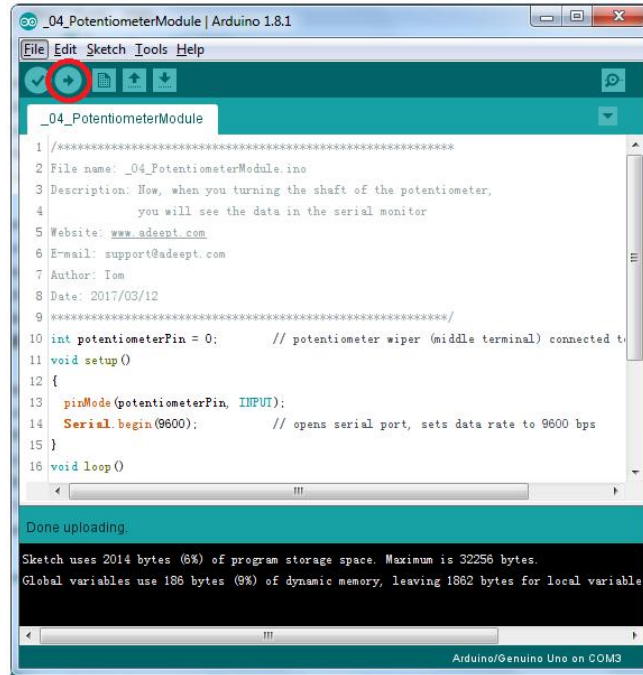


Adeept UNO R3 Board	Potentiometer Module
A0	A
5V	+
GND	-

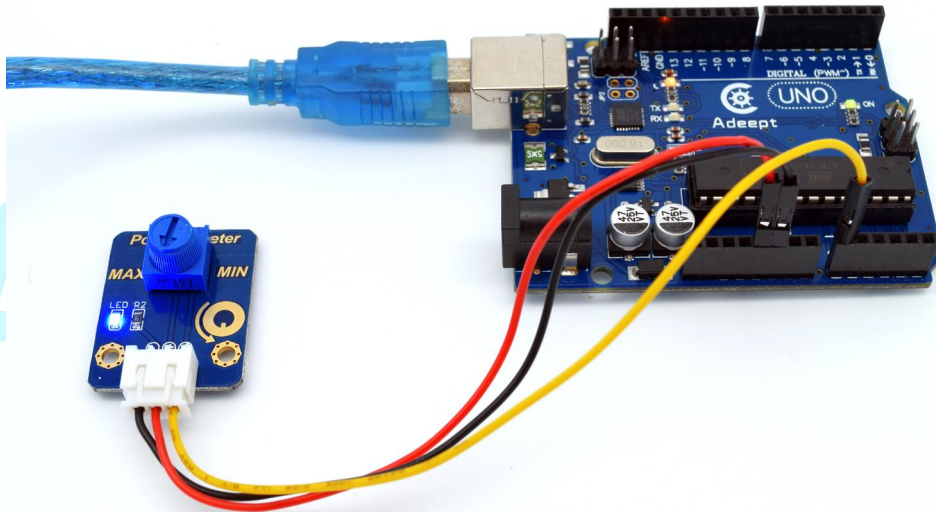
Step 2: Program _04_PotentiometerModule.ino

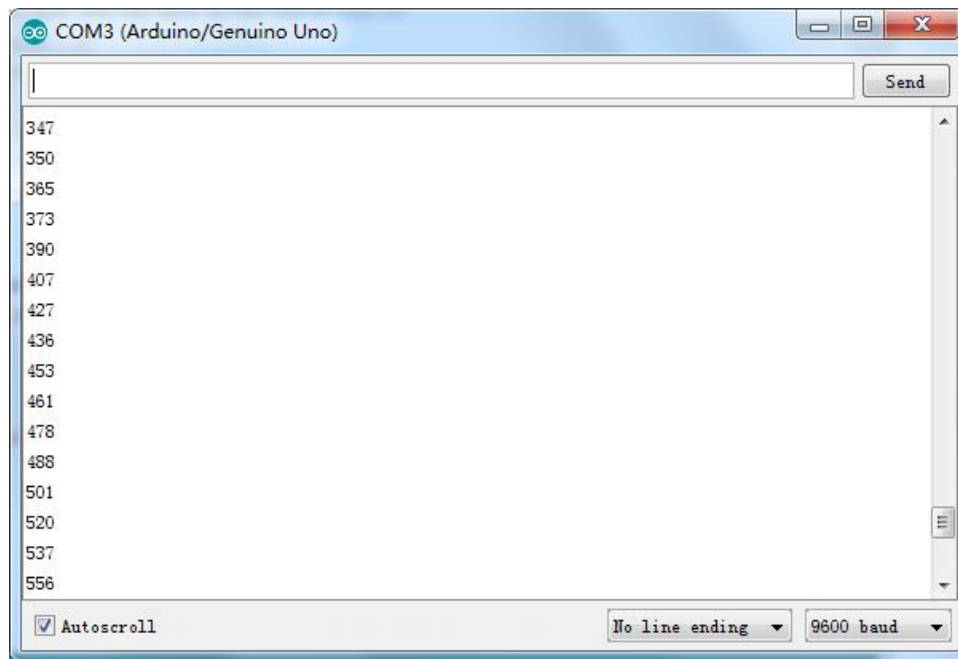
Step 3: Compile and download the sketch to the UNO R3 board.





Open the Serial Monitor in Arduino IDE and turn the knob on the potentiometer module. Then the UNO R3 board will upload the data collected to and display in the Serial Monitor. Turn the pot knob clockwise and the value on the window will decrease and turn counterclockwise, it will increase.





Adeept

Lesson 5 Control An LED by Vibration

Introduction

The vibration digital input module can sense weak vibration signals, thus it can be used for related interaction projects. The core sensor is SW-540, a spring component of no-directional vibration sensing which can be triggered at any angle. The module stays off at any angle when it's still. But when it's hit or knocked by external force, the distorted spring contacts and connects the electrode in the middle thus connecting the two pins and turning the sensor on. When the force disappears, the circuit is back to off.

This sensor module is suitable for small-current vibration detection circuits and has been widely used in products such as toys, light shoes, burglar alarms, electronic scales, flash dance shoes, hot wheels, flash balls, etc.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Vibration Sensor Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

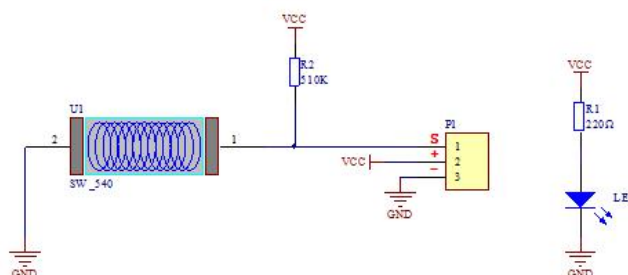
The Fritzing image:



Pin definition:

S	Digital output
+	VCC
-	GND

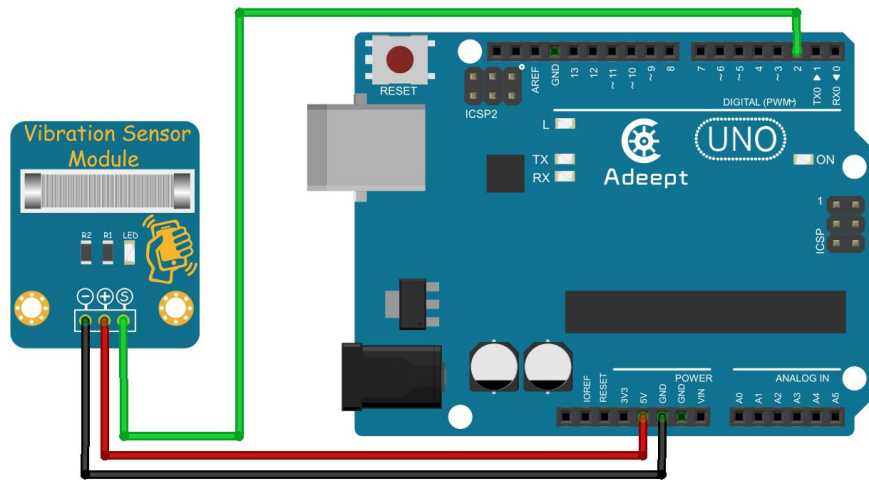
The schematic diagram:



This experiment is to make the LED on the Arduino Uno R3 board flicker with the actions of the Vibration Sensor module.

Experimental Procedures

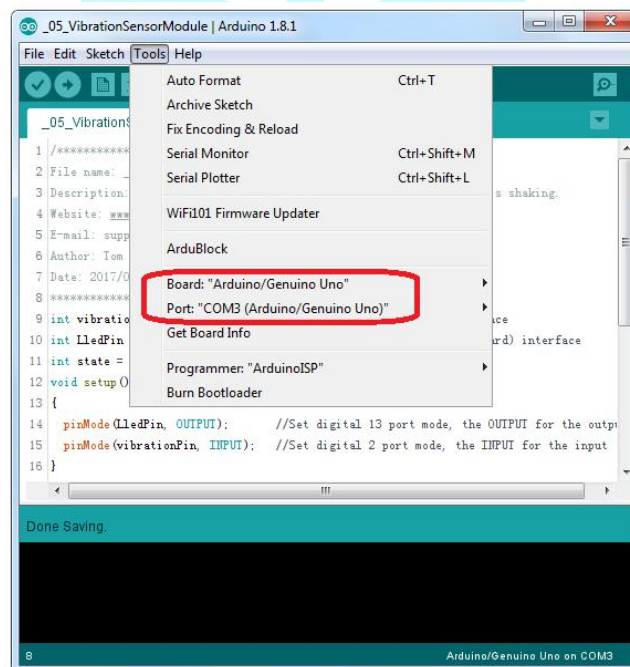
Step 1: Build the circuit

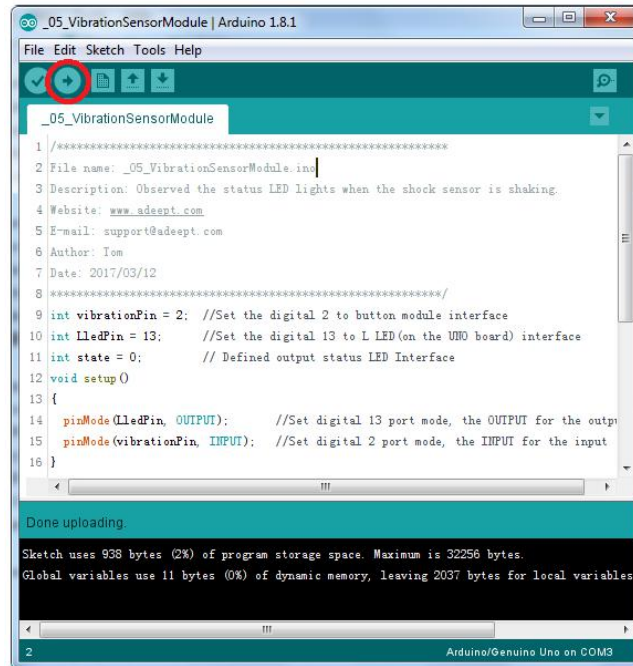


Adeept UNO R3 Board	Vibration Sensor Module
D2	S
5V	+
GND	-

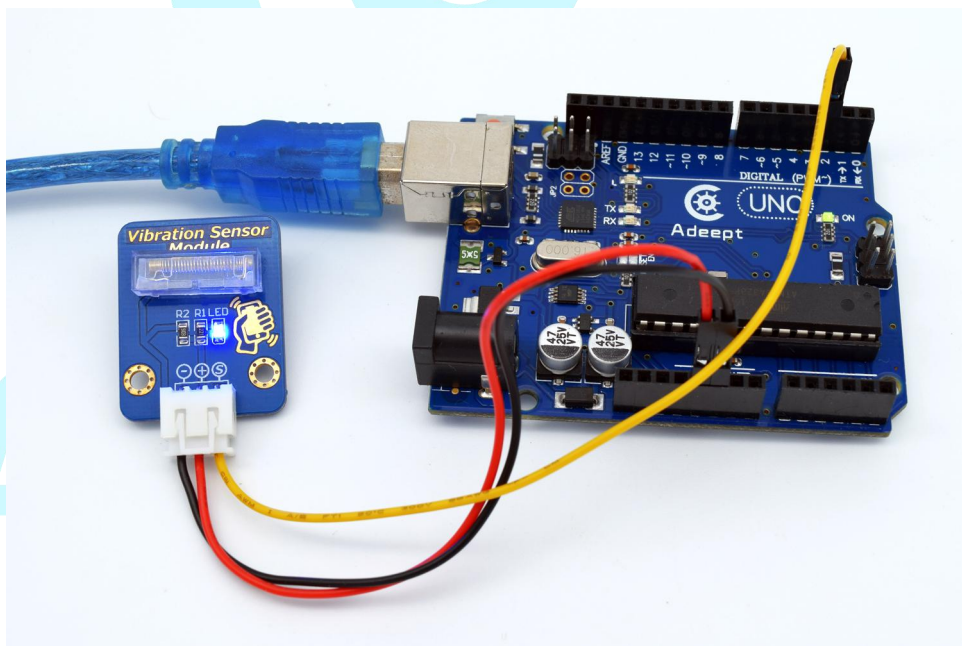
Step 2: Program _05_VibrationSensorModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.





Knock or tap the Vibration Sensor module and you'll see the LED on the Arduino UNO R3 board brighten.



Introduction

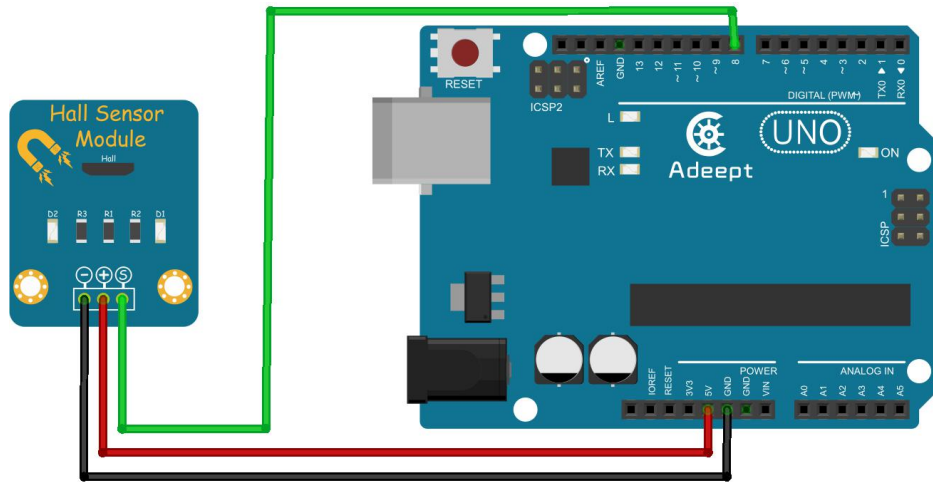
Components

- ## Experimental Principle

S	Digital output
+	VCC
-	GND

Experimental Procedures

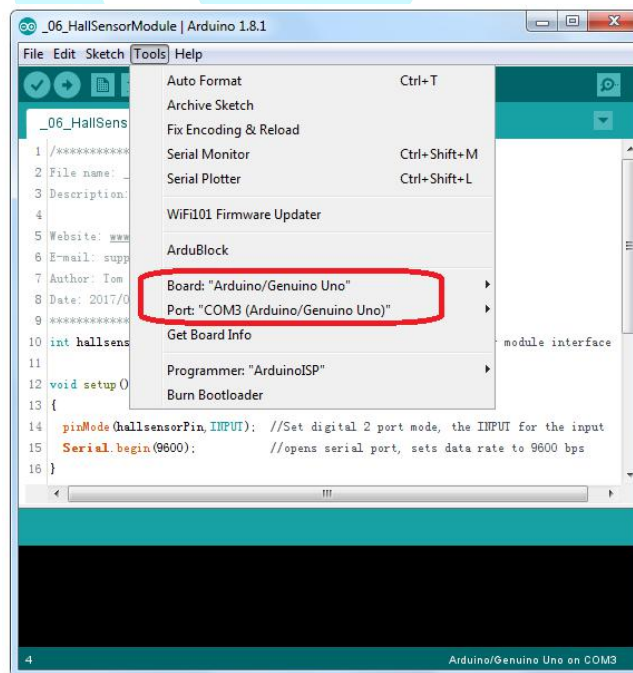
19

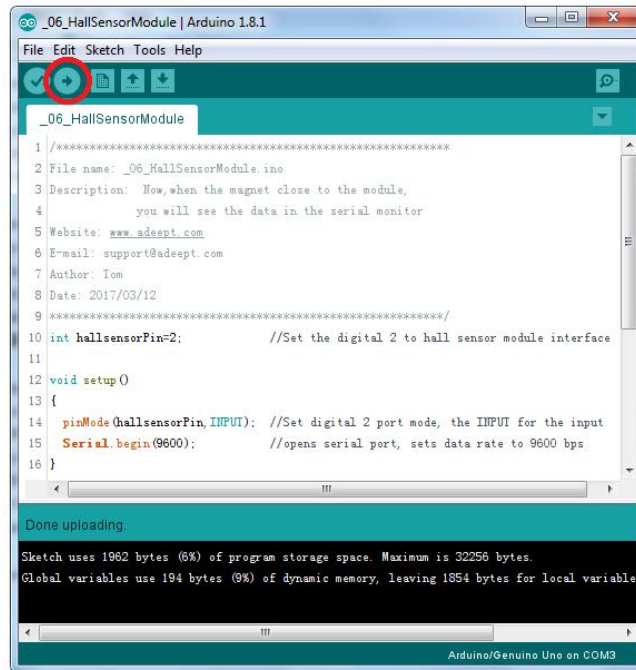


Adeept UNO R3 Board	Hall Sensor Module
D8	S
5V	+
GND	-

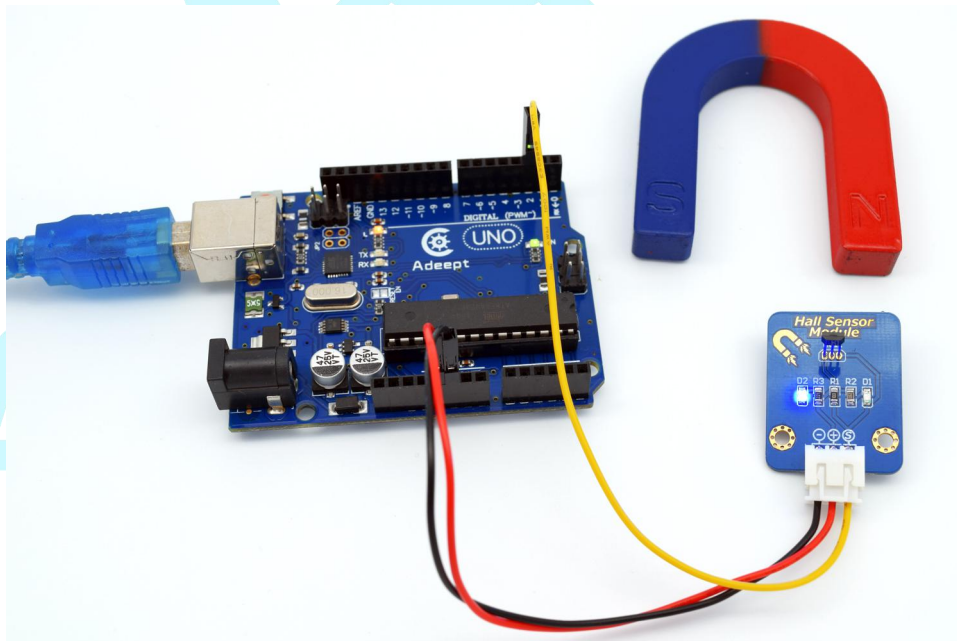
Step 2: Program _06_HallSensorModule.ino

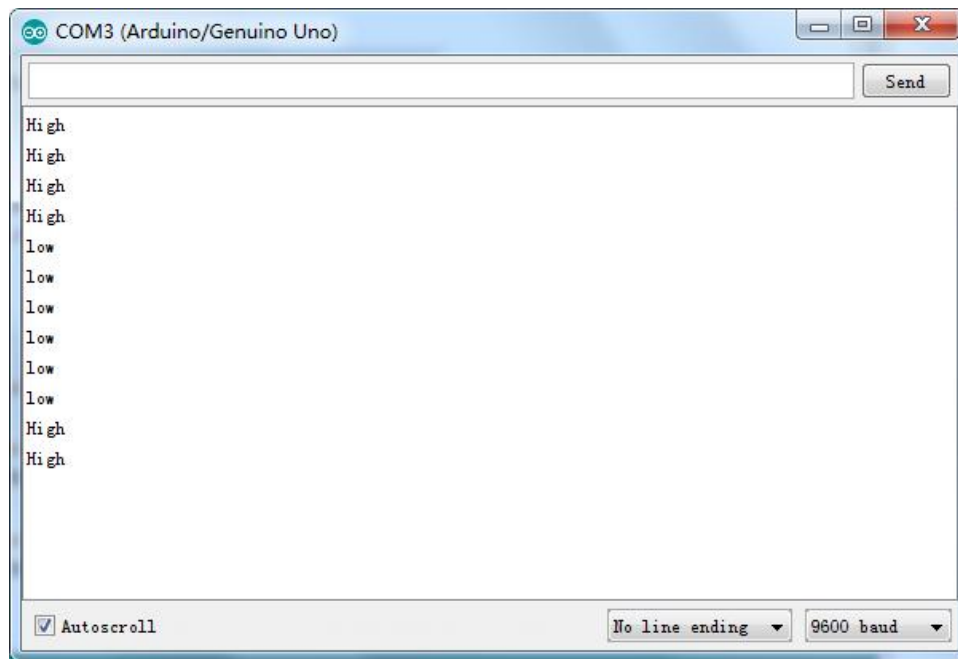
Step 3: Compile and download the sketch to the UNO R3 board.





Now open the Serial Monitor in Arduino IDE. When you place the N pole of a magnet close to or move it away from the Hall Sensor, corresponding status of pin D8 of the Arduino board, i.e. Low or High, will be displayed on the window.





Adeept

Lesson 7 How To Use The Photoresistor

Introduction

The photoresistor module is a resistor module designed based on the principle of photoconductive effect of semiconductors, of which the resistance varies with the intensity of incident light. The resistance of the photoresistor we use decreases with stronger incident light and increases with weaker light. In experiments and daily light, photoresistors are usually used for detecting light intensity.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * Photoresistor Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

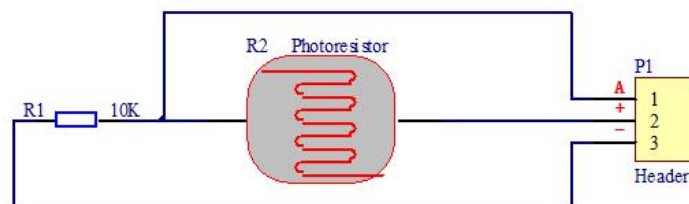
The Fritzing image:



Pin definition:

A	Analog output
+	VCC
-	GND

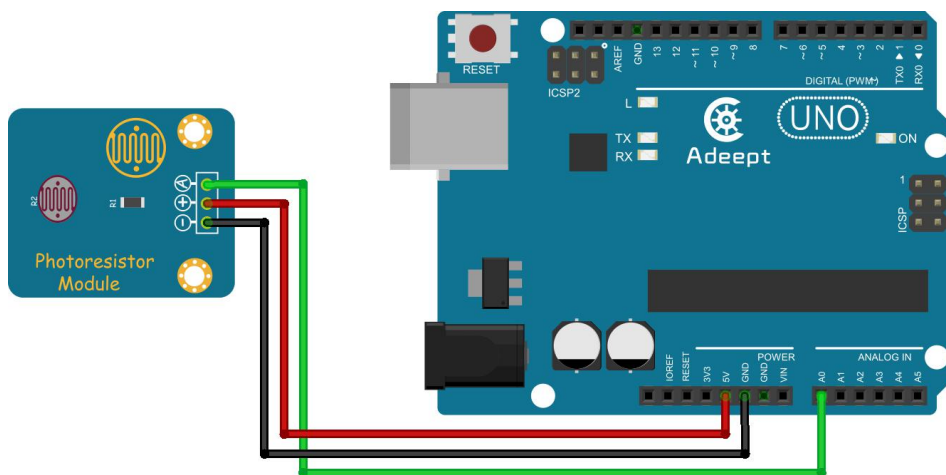
The schematic diagram:



This experiment is to collect the data of light intensity by the Photoresistor module and then display it on the computer.

Experimental Procedures

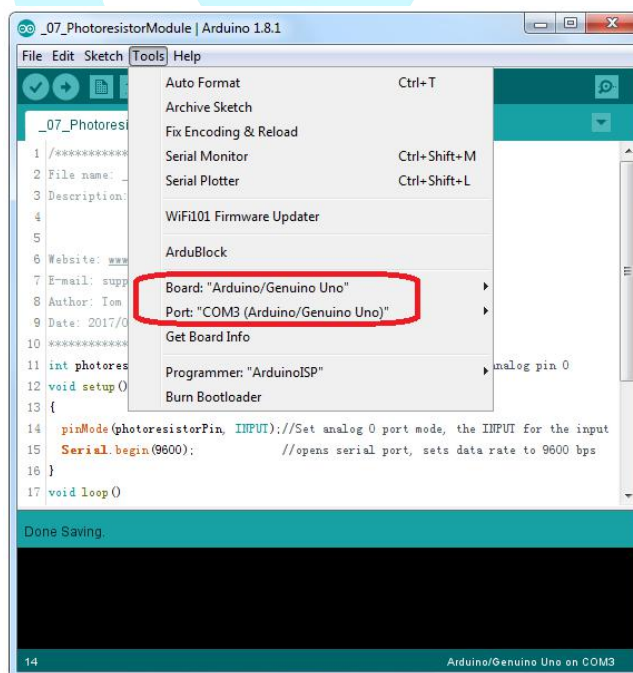
Step 1: Build the circuit

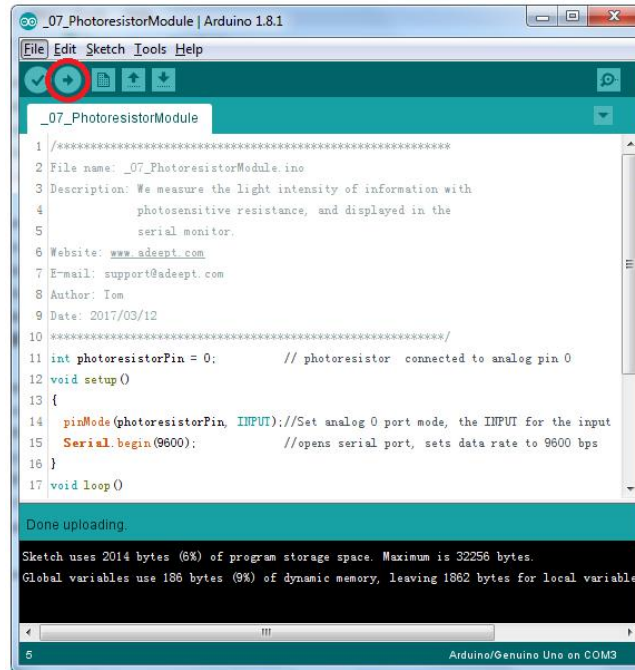


Adept UNO R3 Board	Photoresistor Module
A0	A
5V	+
GND	-

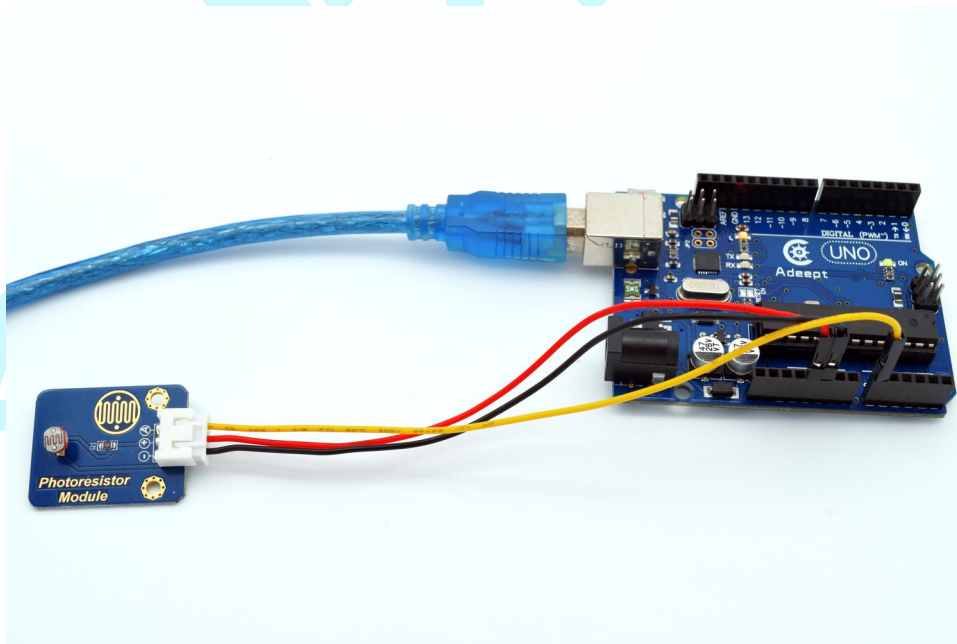
Step 2: Program _07_PhotoresistorModule.ino

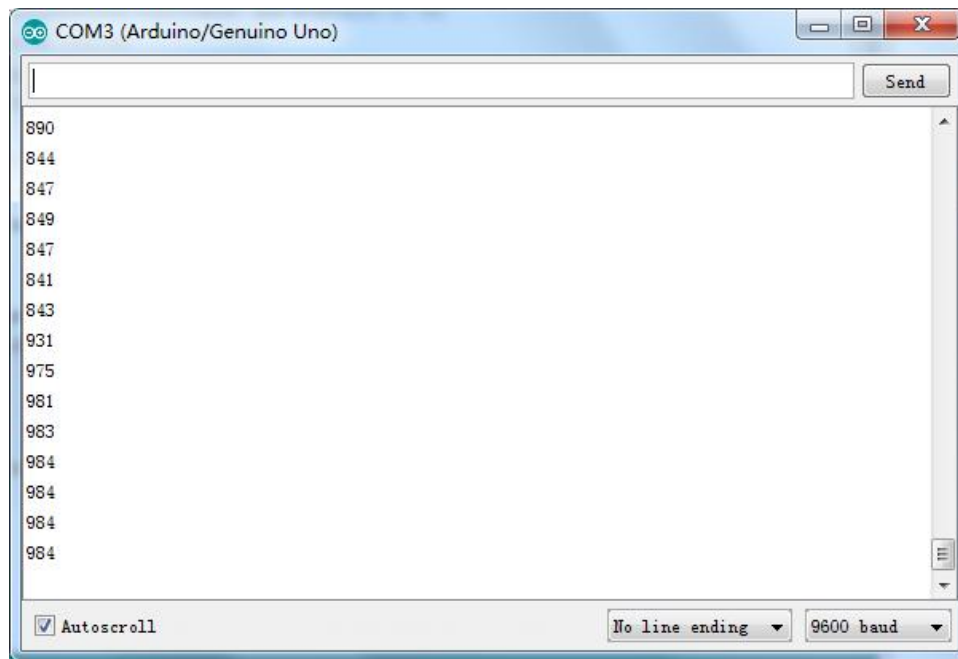
Step 3: Compile and download the sketch to the UNO R3 board.





Open the Serial Monitor in Arduino IDE. Change the intensity of light shone on the photoresistor module and you can see the value displayed on the window changes. Basically, when you shine the stronger light on the module, the value will become greater; when the light dims slowly on it, the value decreases.





Adeept

Lesson 8 How To Use The Thermistor

Introduction

Thermistors can be divided into two types by temperature coefficient: positive temperature coefficient (PTC) and negative temperature coefficient (NTC). The typical feature of a thermistor is sensitive to temperature – its resistance varies with ambient temperature changes. For PTC thermistor, when the temperature gets high, its resistance increases; for NTC thermistor, the case is the opposite. The thermistor we use is an NTC one.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * Thermistor Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

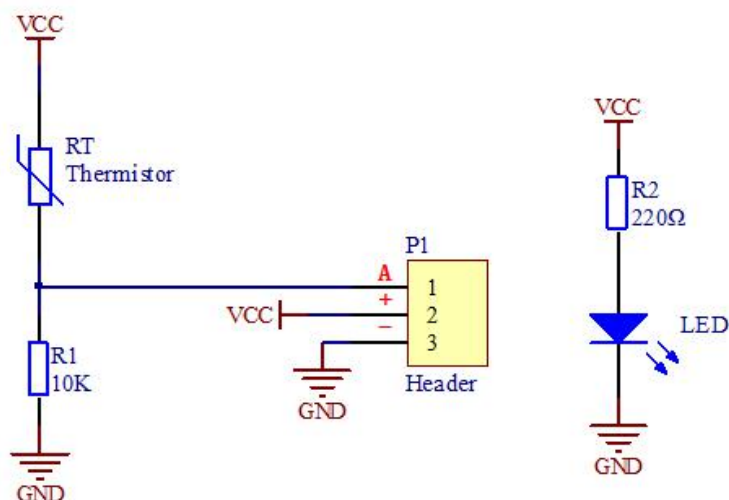
The Fritzing image:



Pin definition:

A	Analog output
+	VCC
-	GND

The schematic diagram:



The key parameters of an MF52 thermistor:

B-parameter: 3470.

25°C resistance: 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 the thermistor at temperature T1

R : the nominal resistance of the thermistor at room temperature T2

e : 2.718281828459

B : one of the important parameters of thermistor

T_1 : the Kelvin temperature that you want to measure

T_2 : Under the condition of a 25 °C (298.15K) room temperature, the standard resistance of MF52 thermistor is 10K;

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

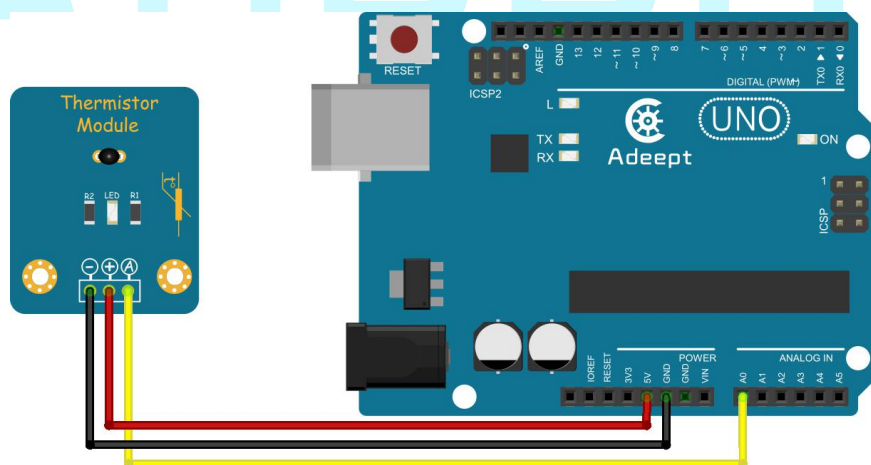
After transforming the above equation, we can get the following formula:

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

In this experiment, by programming the Arduino, we collect the analog values output by the Thermistor module through pin A0 of the Arduino board, change it to temperature values and display them on Serial Monitor.

Experimental Procedures

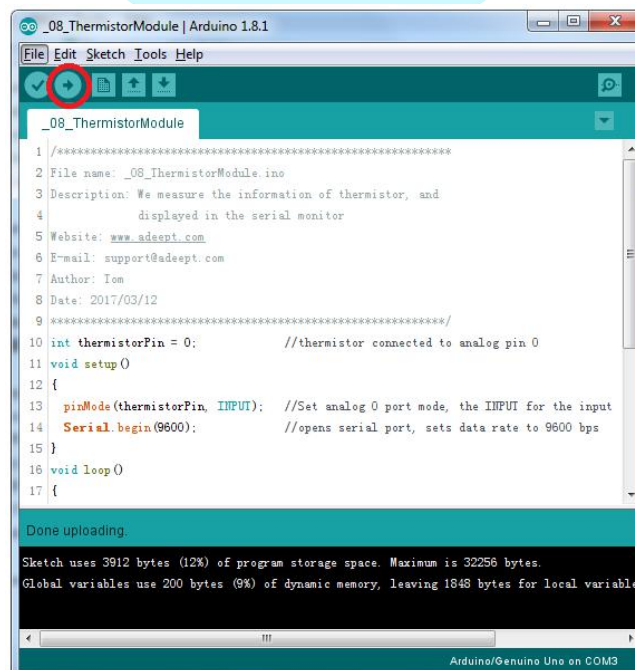
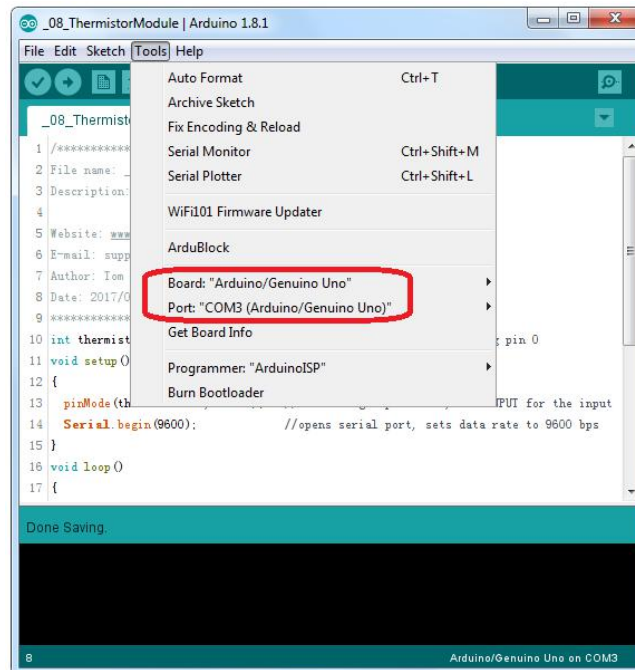
Step 1: Build the circuit



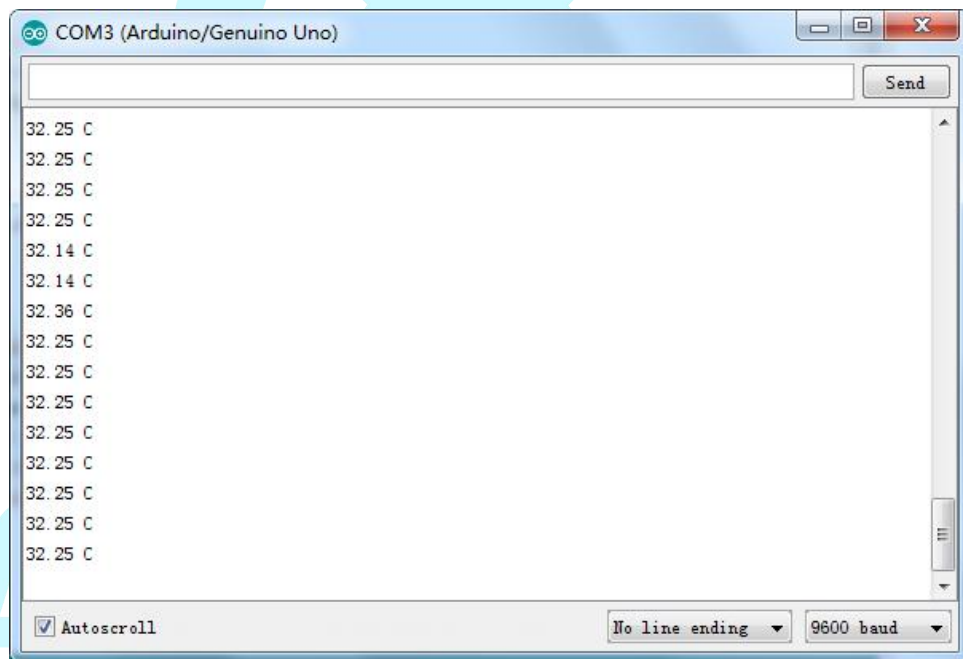
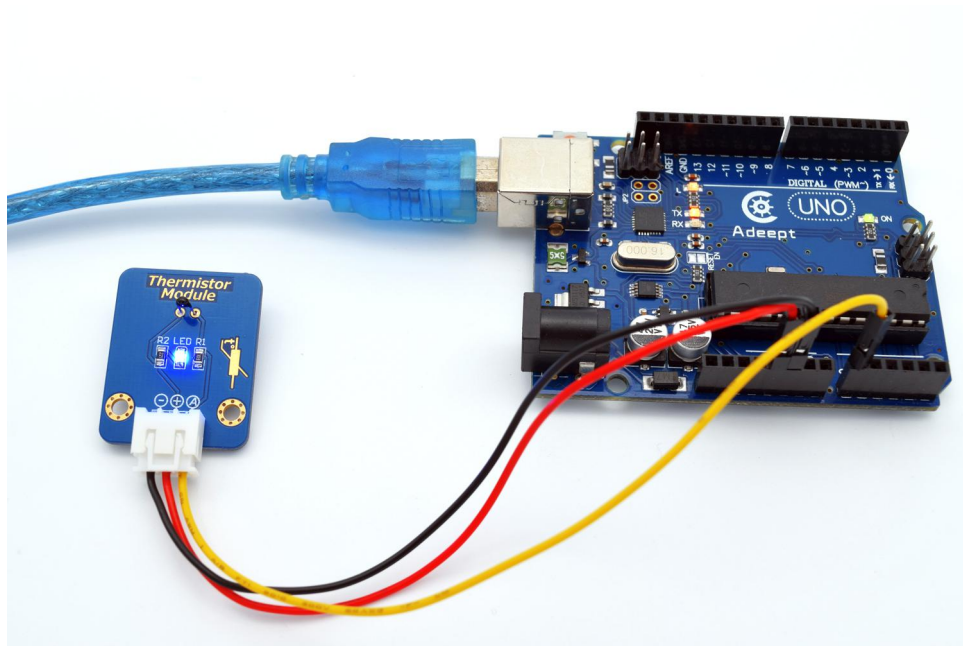
Aadept UNO R3 Board	Thermistor Module
A0	A
5V	+
GND	-

Step 2: Program _08_ThermistorModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.



Open the Serial Monitor in Arduino IDE and you can see the current temperature on the window.



Lesson 9 How To Use The DS18B20

Introduction

DS18B20 is a single-bus digital temperature sensor of high-precision. The measurement range is -55°C - $+125^{\circ}\text{C}$ and inherent temperature resolution is 0.5°C . The sensor support multi-point network and multi-point temperature measurement – the measured result is sent to the controller via serial port in the format of a 9-12-bit number. This digital sensor can be applied to various microcontrollers. It's simpler on Arduino boards – just call the related functions and the temperature can be measured then.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * DS18B20 Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

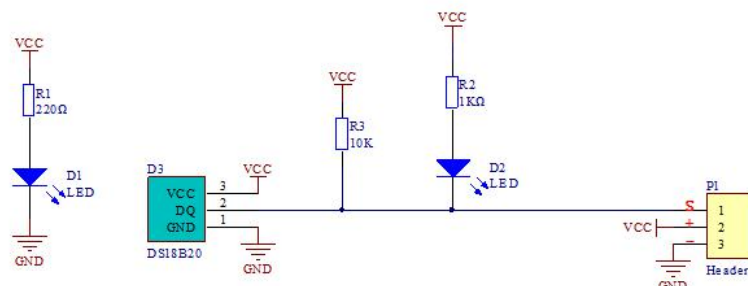
The Fritzing image:



Pin definition:

S	Digital pin
+	VCC
-	GND

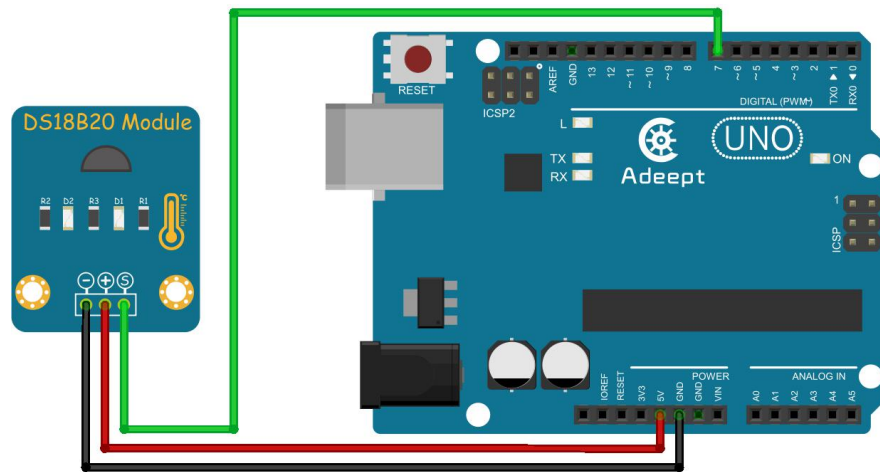
The schematic diagram:



In this experiment, by programming the Arduino, we read the temperature value collected by the DS18B20 module through pin D7 of the Arduino board, and display it on Serial Monitor.

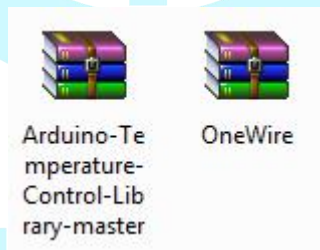
Experimental Procedures

Step 1: Build the circuit

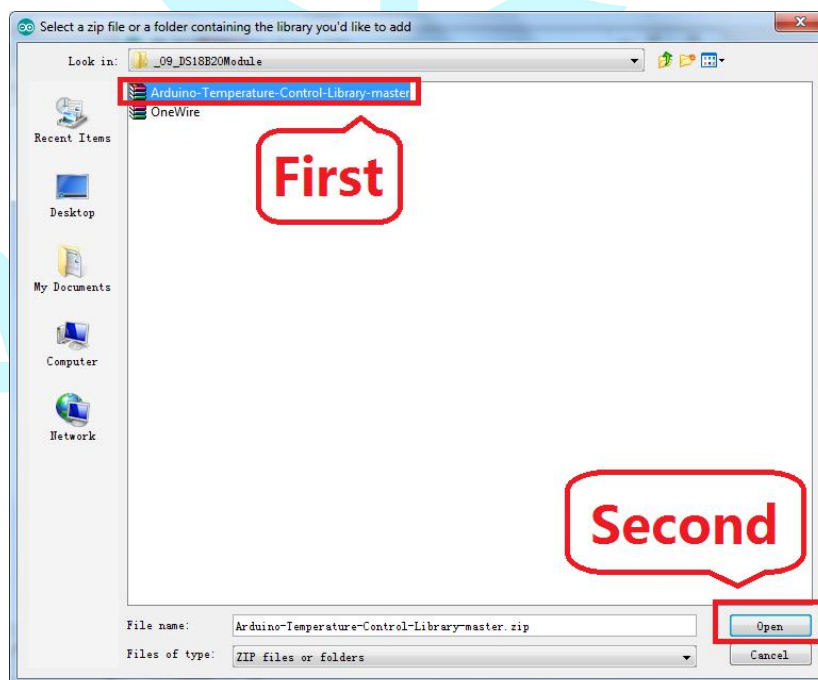
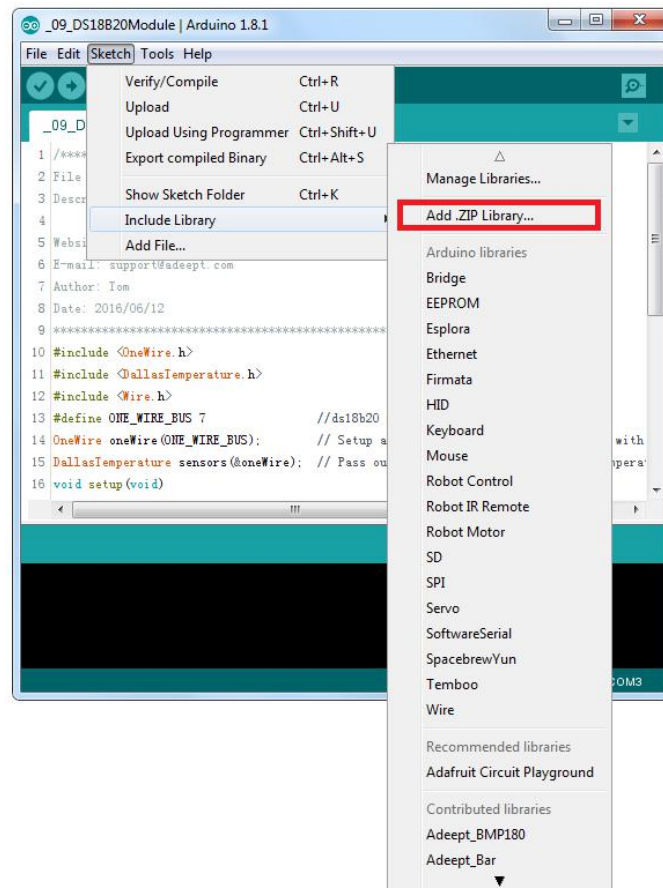


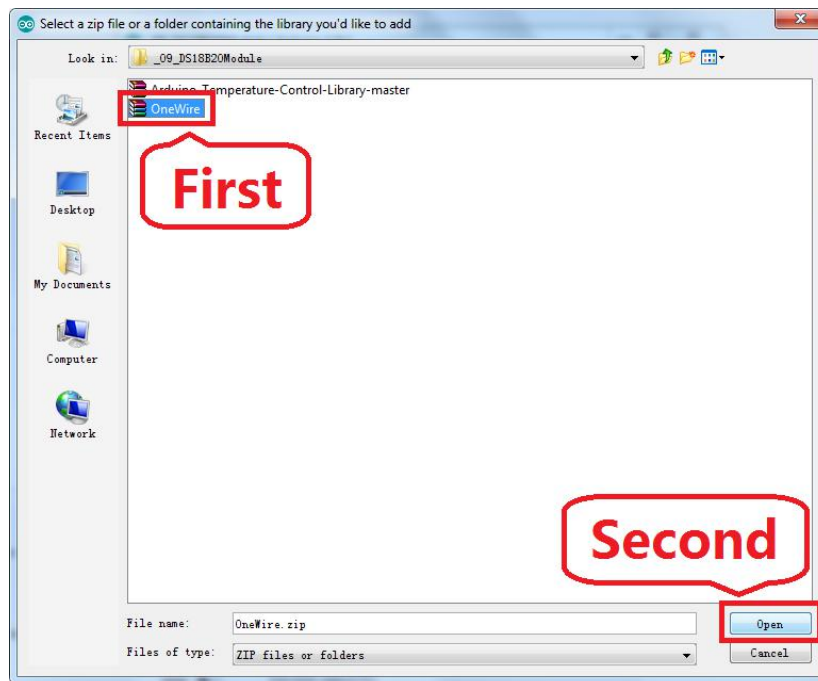
Adeept UNO R3 Board	DS18B20 Module
D7	S
5V	+
GND	-

Step 2: Install the function library (Arduino-Temperature-Control-Library-master.zip and OneWire.zip).



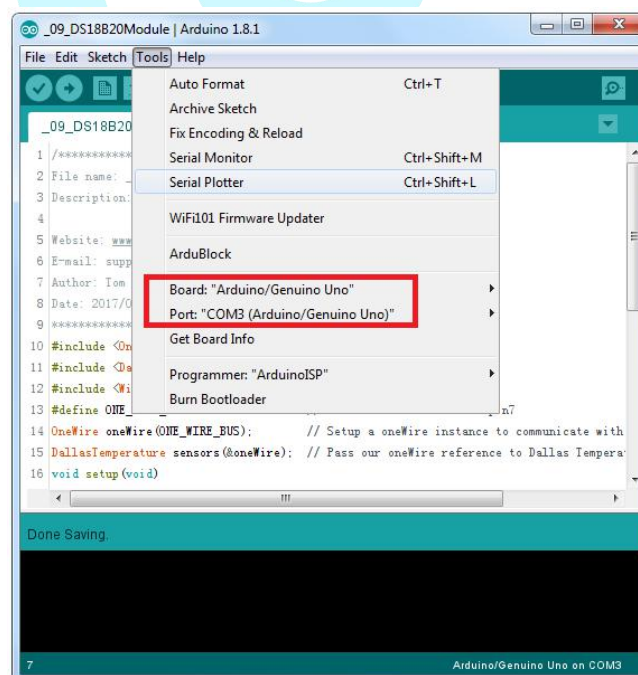
Adeept

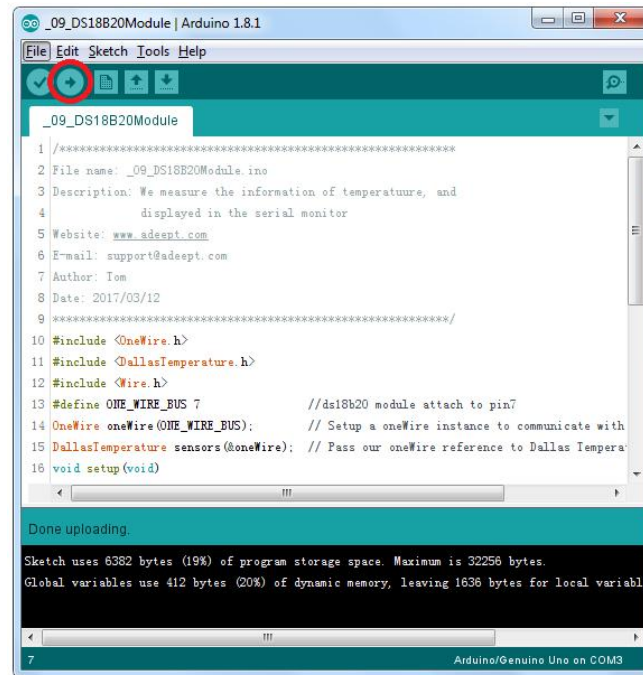




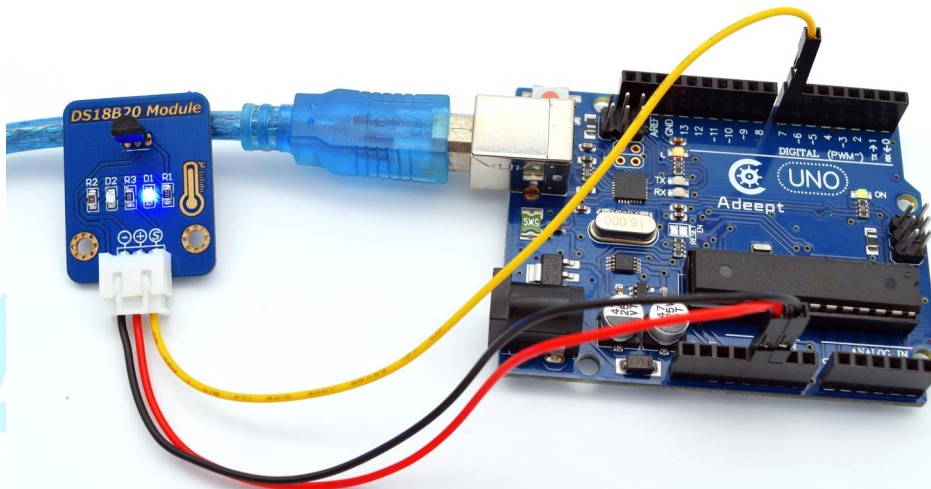
Step 3: Program `_09_DS18B20Module.ino`

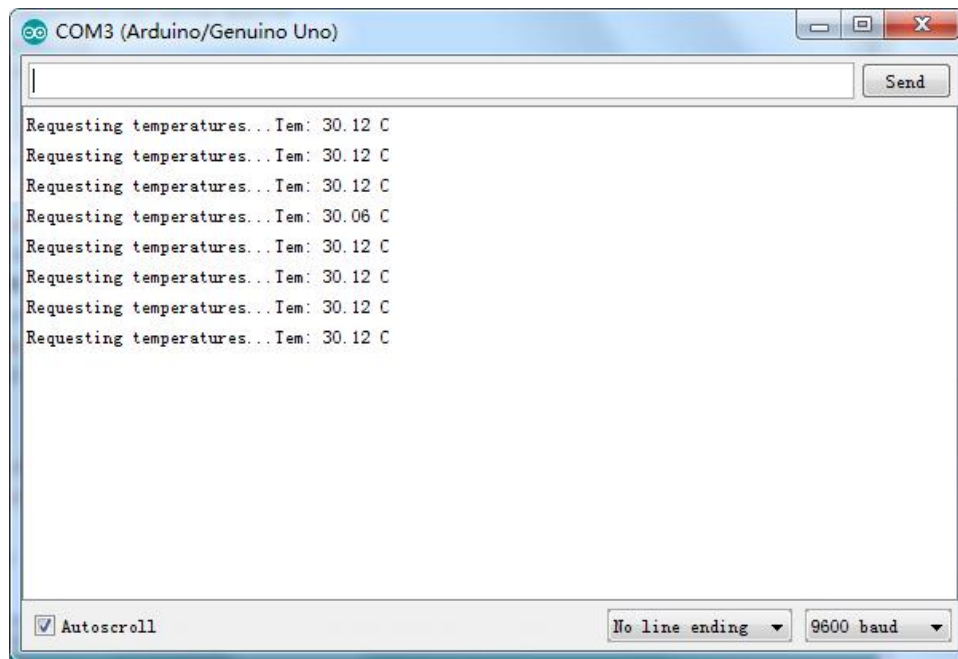
Step 4: Compile and download the sketch to the UNO R3 board.





Open the Serial Monitor in Arduino IDE and you can see the current temperature on the window.





Adeept

Lesson 10 Alarm Prompt

Introduction

Buzzers are a type of integrated electronic alarm devices and powered by DC supply. They are widely applied for sound producing in devices such as computer, printer, duplicator, alarm, electronic toy, vehicle electronic equipment, phone, and timer and so on. Active buzzers can make sounds constantly when connected with a 5V DC supply. This Active Buzzer module is connected to a digital pin of the Arduino Uno board. When the pin outputs High level, the buzzer will beep; when the pin gives Low, it stays dumb.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * Active Buzzer Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

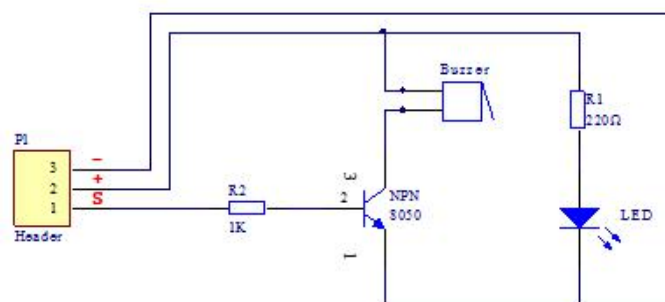
The Fritzing image:



Pin definition:

S	Digital input
+	VCC
-	GND

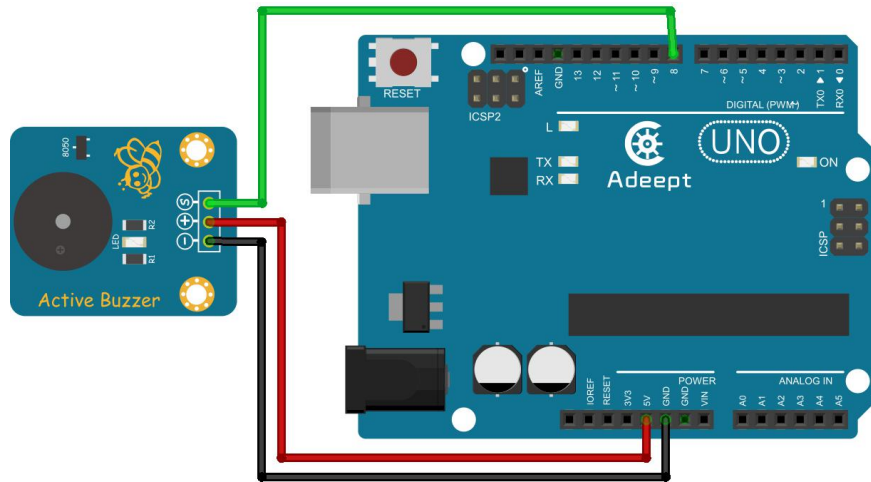
The schematic diagram:



In this experiment, by programming the Arduino, we make the pin D8 of the Arduino board output High and Low alternately, so the active buzzer makes sounds accordingly.

Experimental Procedures

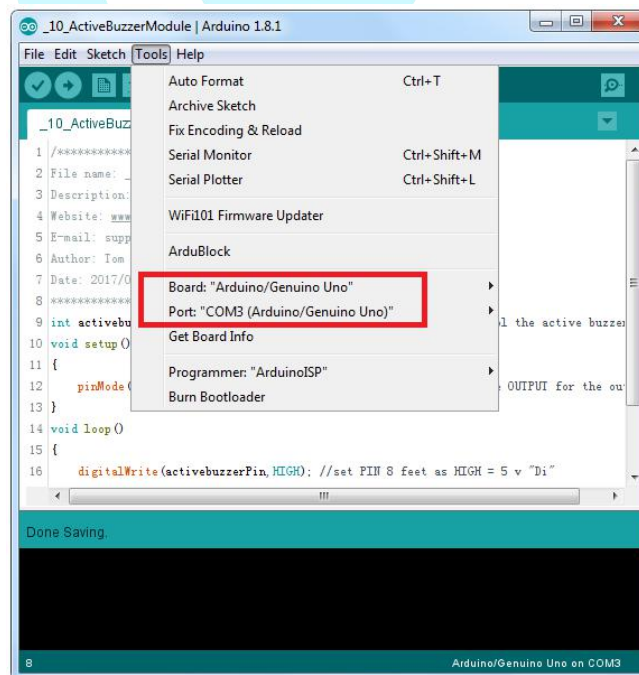
Step 1: Build the circuit

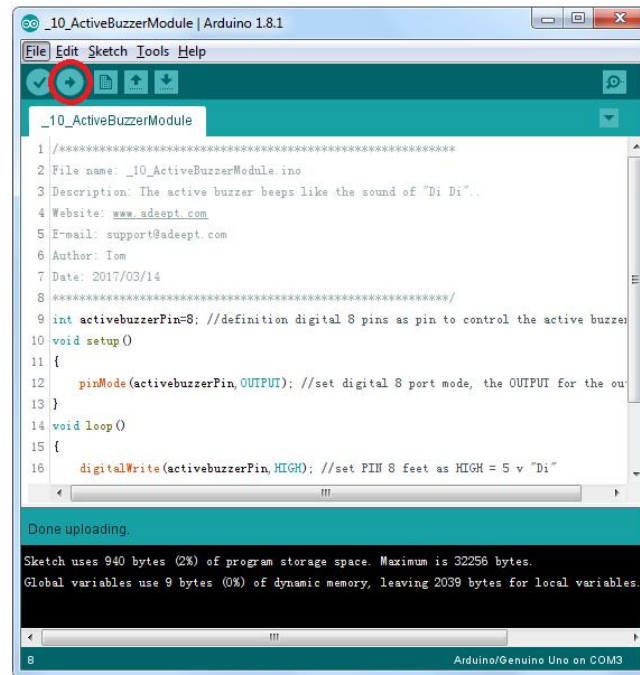


Adeept UNO R3 Board	Active Buzzer Module
D8	S
5V	+
GND	-

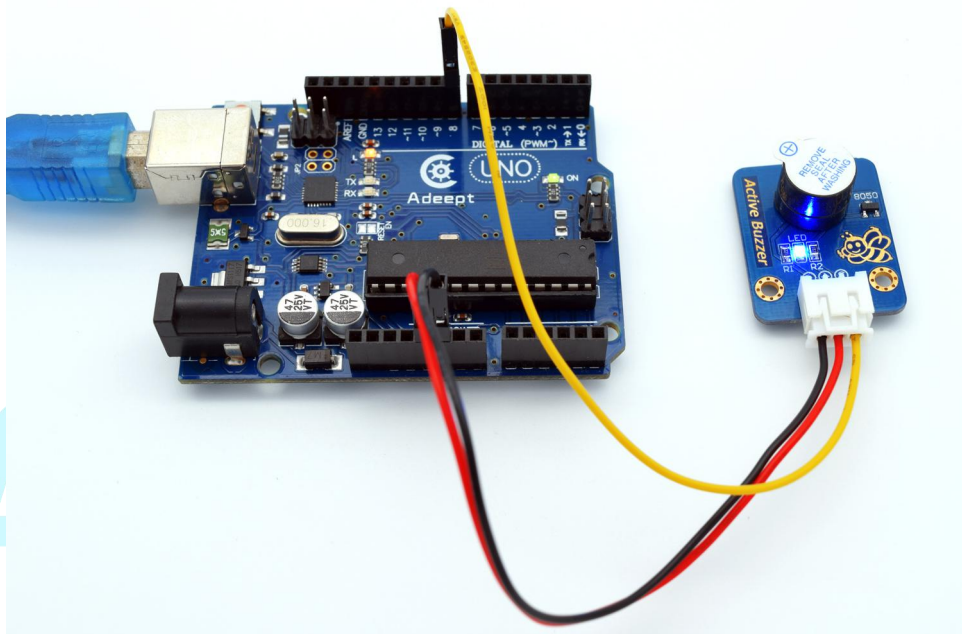
Step 2: Program _10_ActiveBuzzerModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.





Now you can hear the active buzzer beeps like the sound of "Di Di".



Lesson 11 Playing Music

Introduction

The difference between an active buzzer and a passive one radically lies in the requirement for input signals. The ideal signals for active buzzers are direct currents, usually marked with VCC or VDD. Inside them there are a simple oscillation circuit that can convert constant direct currents into pulse signal of a certain frequency, causing magnetic fields alternation and then Mo sheet vibrating and making sounds. On the other hand, there is no driving circuit in a passive buzzer. So the ideal signal for passive buzzer is square wave. If DC is given, it will not respond since the magnetic field is unchanged, the Mo sheet cannot vibrate and produce sounds.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * Passive Buzzer Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

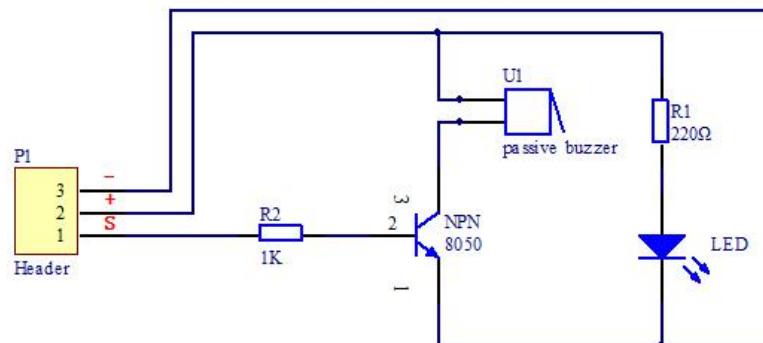
The Fritzing image:



Pin definition:

	S	Digital input
+		VCC
-		GND

The schematic diagram:

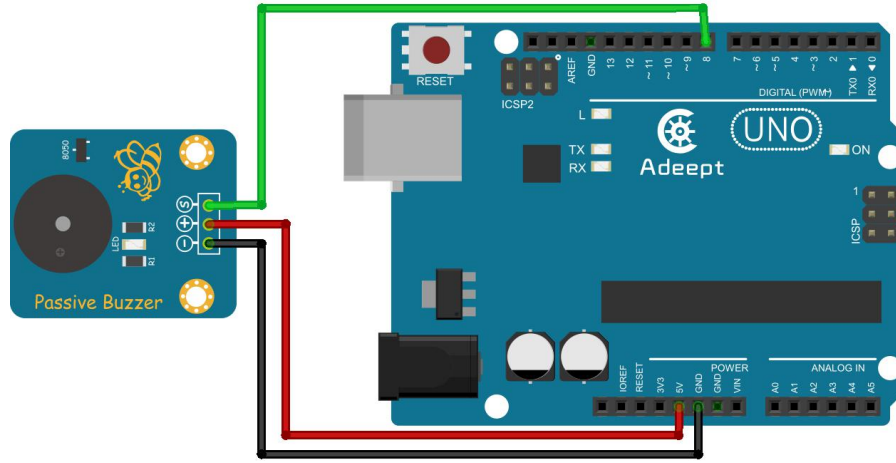


In this experiment, by programming the Arduino, we make the pin D8 of the Arduino board

output square waves of different frequencies alternately, thus driving the passive buzzer to play music.

Experimental Procedures

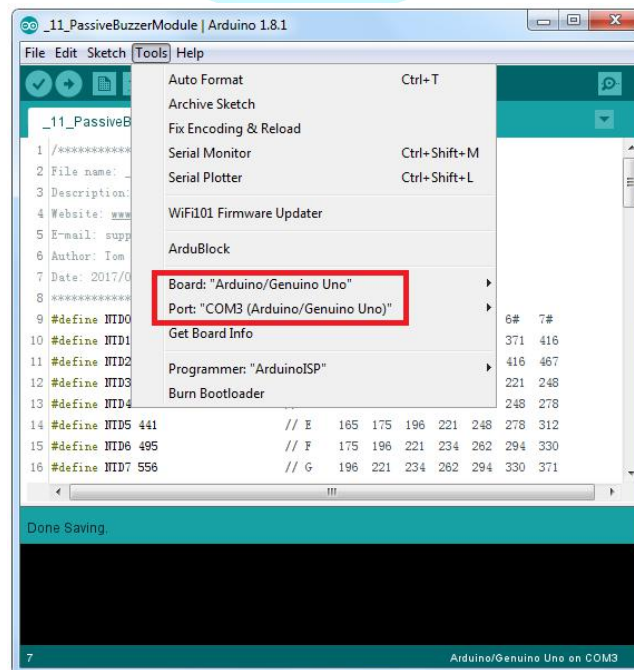
Step 1: Build the circuit

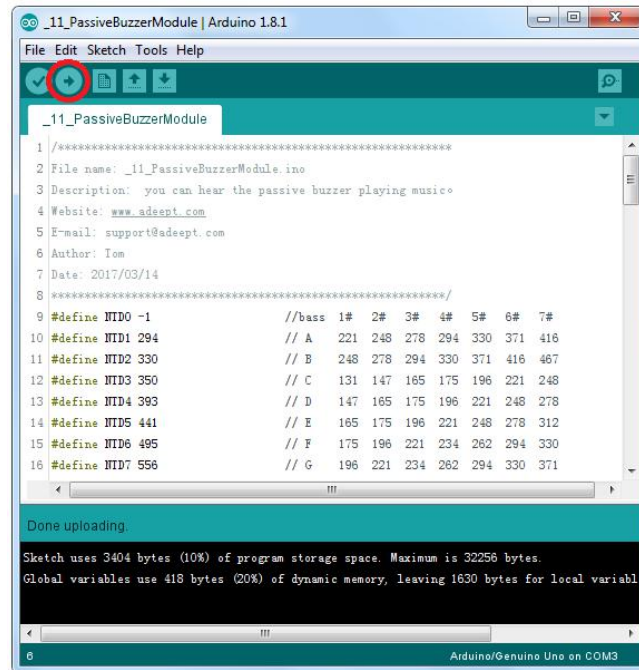


Adept UNO R3 Board	Passive Buzzer Module
D8	S
5V	+
GND	-

Step 2: Program _11_PassiveBuzzerModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.

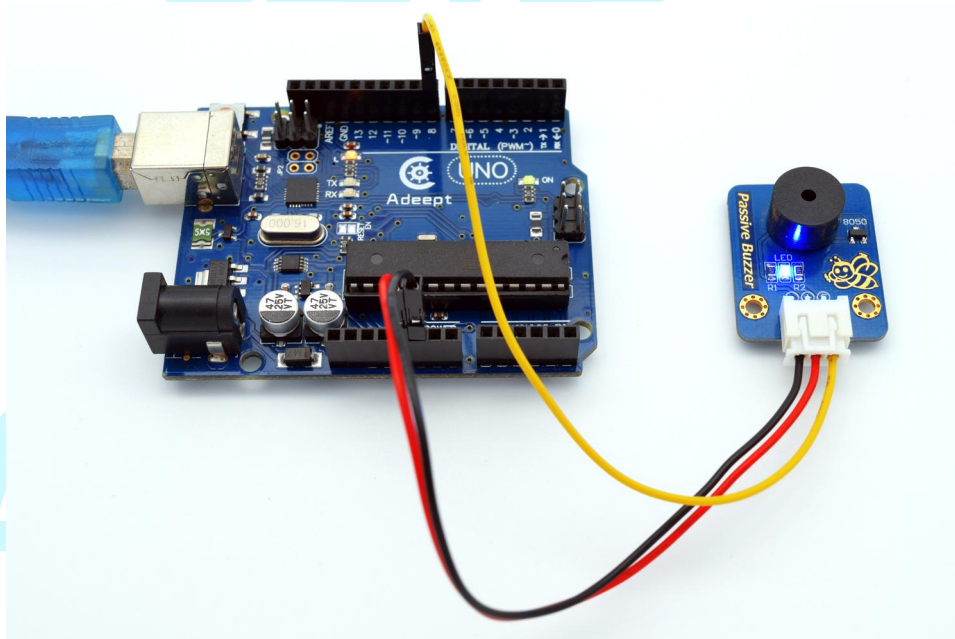




```
1 /*****  
2 File name: _11_PassiveBuzzerModule.ino  
3 Description: you can hear the passive buzzer playing music  
4 Website: www.adeept.com  
5 E-mail: support@adeept.com  
6 Author: Tom  
7 Date: 2017/03/14  
8 *****/  
9 #define INTD0 ~1 //bass 1# 2# 3# 4# 5# 6# 7#  
10 #define INTD1 294 // A 221 248 278 294 330 371 416  
11 #define INTD2 330 // B 248 278 294 330 371 416 467  
12 #define INTD3 350 // C 131 147 165 175 196 221 248  
13 #define INTD4 393 // D 147 165 175 196 221 248 278  
14 #define INTD5 441 // E 165 175 196 221 248 278 312  
15 #define INTD6 495 // F 175 196 221 234 262 294 330  
16 #define INTD7 556 // G 196 221 234 262 294 330 371
```

Done uploading.
Sketch uses 3404 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 418 bytes (20%) of dynamic memory, leaving 1630 bytes for local variables.

Now you can hear the passive buzzer play music.



Lesson 12 Detection of Human Body Movement

Introduction

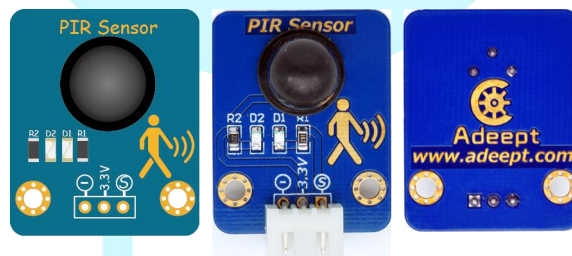
The PIR (passive Infrared) sensor can detect the Infrared rays emitted by human or animals and then output On/Off signals. Traditional pyroelectric PIR sensor needs pyroelectric Infrared probe, special chip and complex circuits to make the effect. In this case, the sensor is a large one with complicated circuits and comparatively less credible. But this Infrared pyroelectric motion sensor adopts the digital integration of the probe, so it boasts high credibility, low power consumption and simple outside circuit with a small size. It can be applied to any cases in detecting moving human or animals.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * PIR Sensor Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

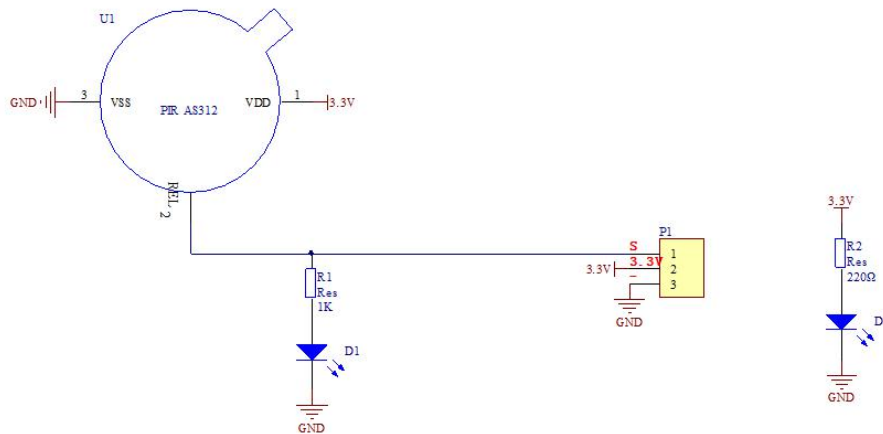
The Fritzing image:



Pin definition:

S	Digital output
3.3V	3.3V
-	GND

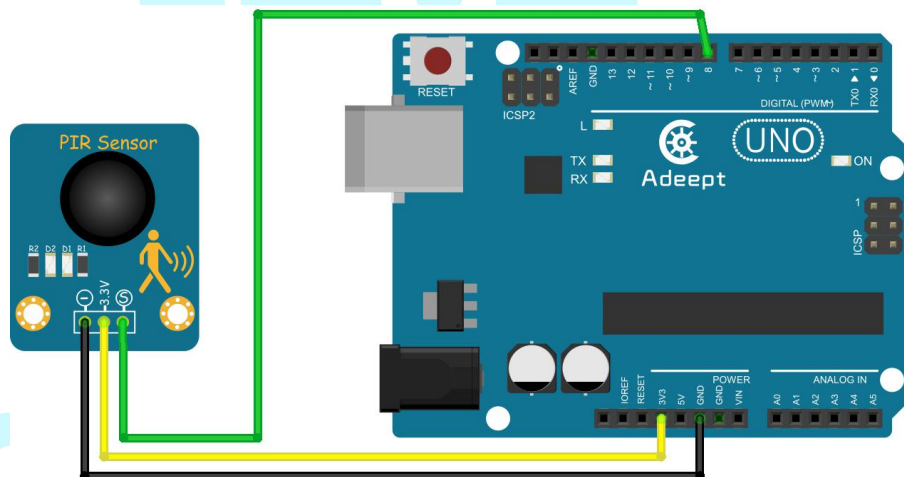
The schematic diagram:



This experiment is to use the PIR Sensor Module to detect whether there is any human activity and send the data to computer via serial port, and then display it on Serial Monitor.

Experimental Procedures

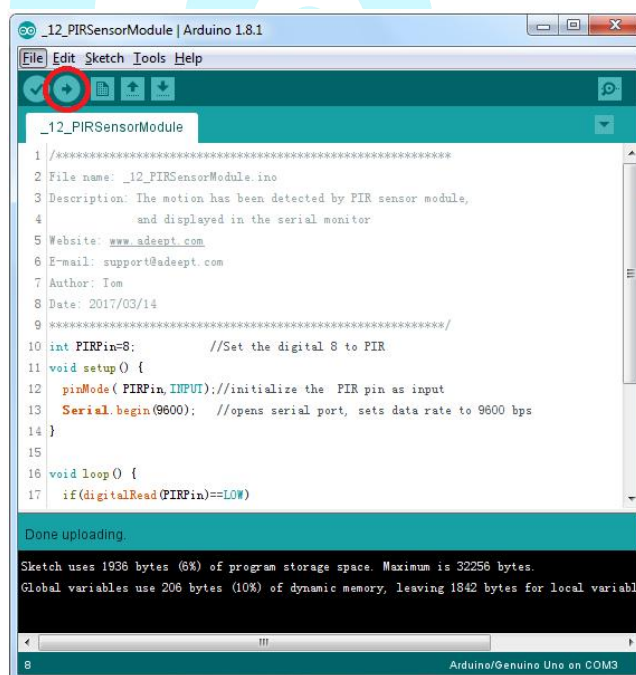
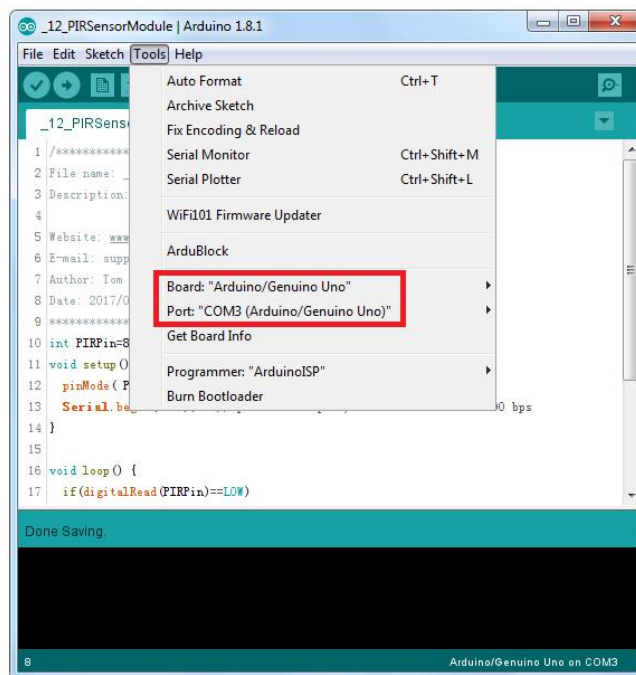
Step 1: Build the circuit



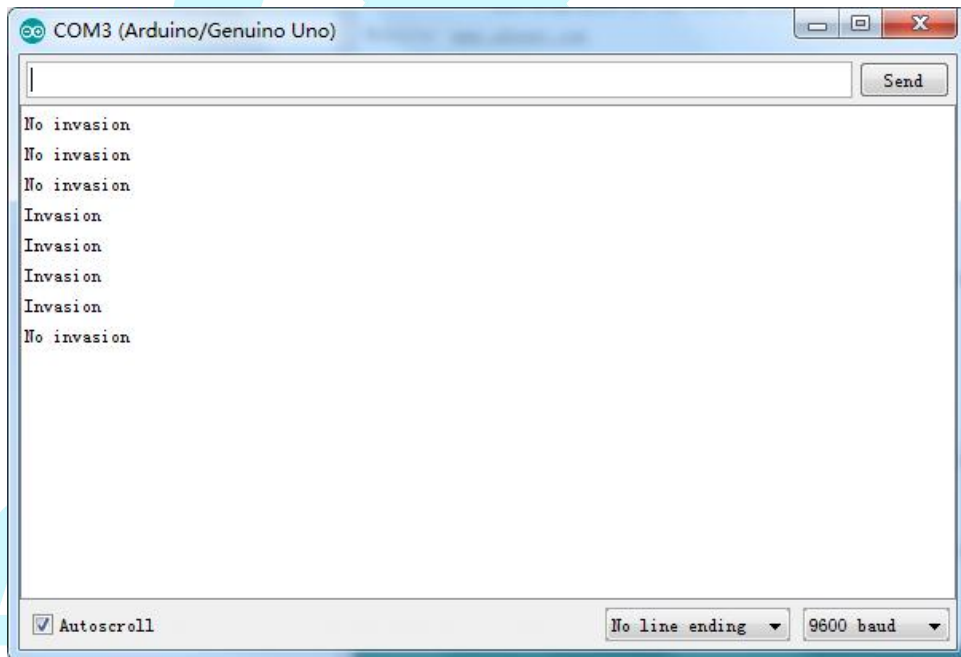
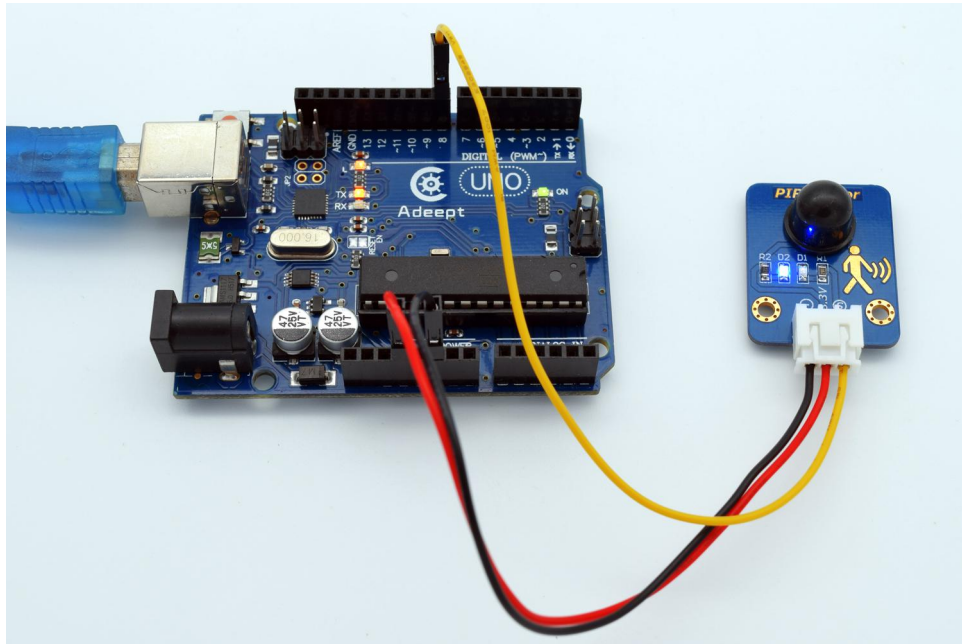
Adept UNO R3 Board	PIR Sensor Module
D8	S
3.3V	3.3V
GND	-

Step 2: Program _12_PIRSensorModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.



Open the Serial Monitor in Arduino IDE. When the PIR module detects human movement, "Invasion!" will be displayed on the window; otherwise, "No invasion".



Lesson 13 How To Use The LCD1602

Introduction

1602 crystal, or 1602 character crystal, is a dot-matrix crystal module used specially to display letters, numbers, symbol, etc. It's composed of several dot-matrix character bits, each of which can display one character. The character bits are separated by one dot pitch and there's a gap between each line. Therefore, characters are spaced within and between lines.

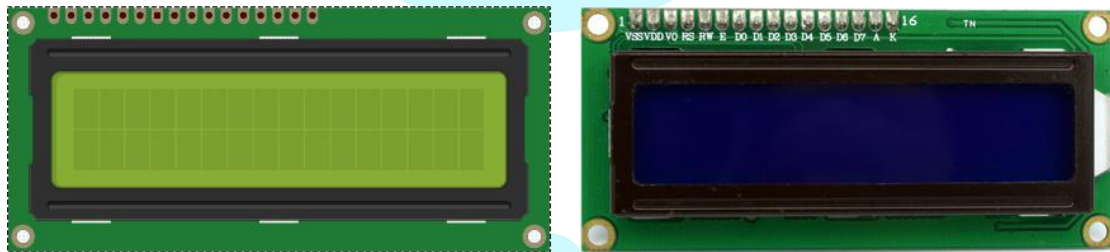
The name LCD 1602 indicates that the display is 16x2, that is, two lines with 16 characters in each.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * LCD1602 Display Module
- 1 * USB Cable
- 1 * 3-Pin Wires
- 2 * Hookup Wire Set
- 12 * Male to Female Jumper Wires
- 1 * Breadboard

Experimental Principle

The Fritzing image:



Pin definition:

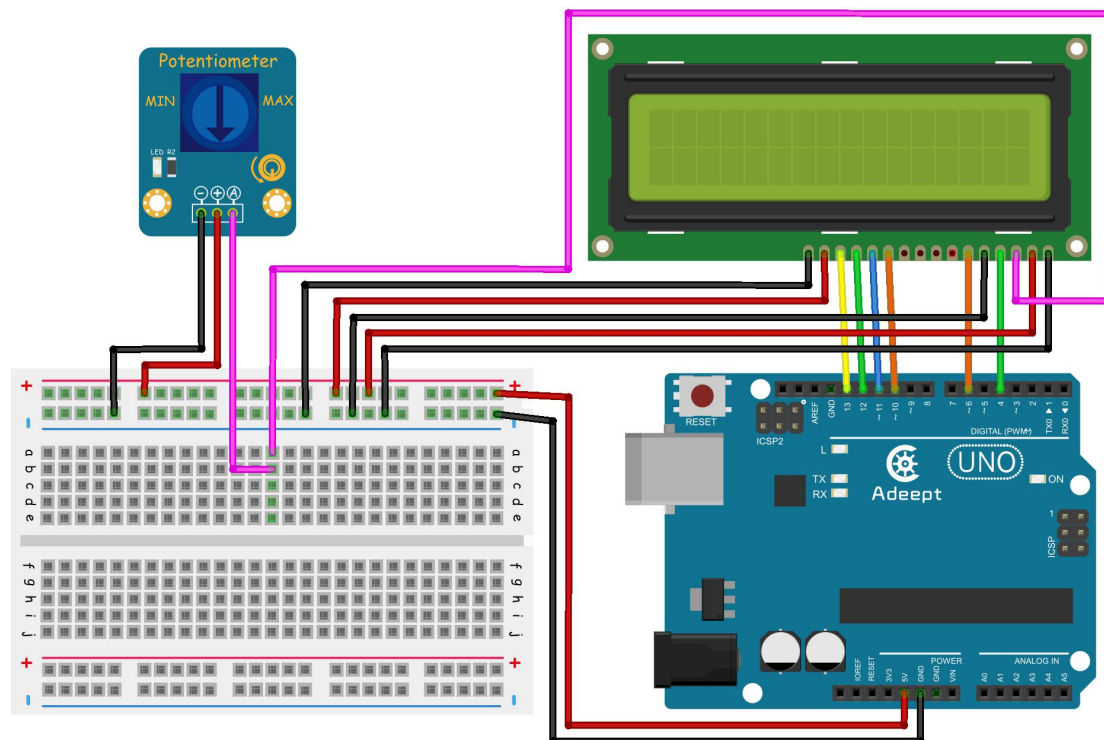
VSS	GND
VDD	VCC
VO	Analog input
RS	Digital input
RW	Digital input
E	Digital input
D0	Digital input
D1	Digital input
D2	Digital input
D3	Digital input
D4	Digital input
D5	Digital input
D6	Digital input

D7	Digital input
A	VCC
K	GND

In this experiment, by programming the Arduino, we make the LCD1602 display scrolling "Hello Geeks!" first and then "Adeept www.adeept.com" steadily.

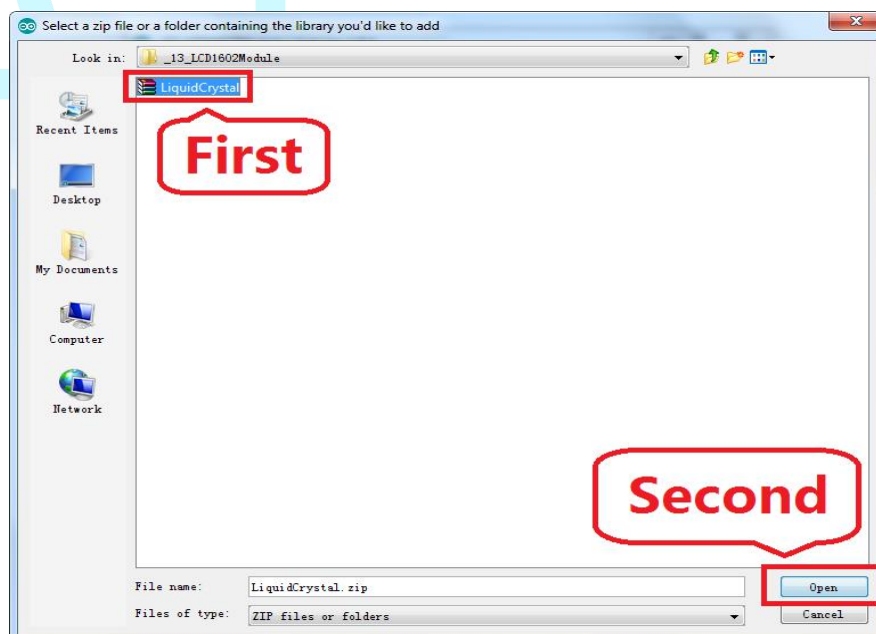
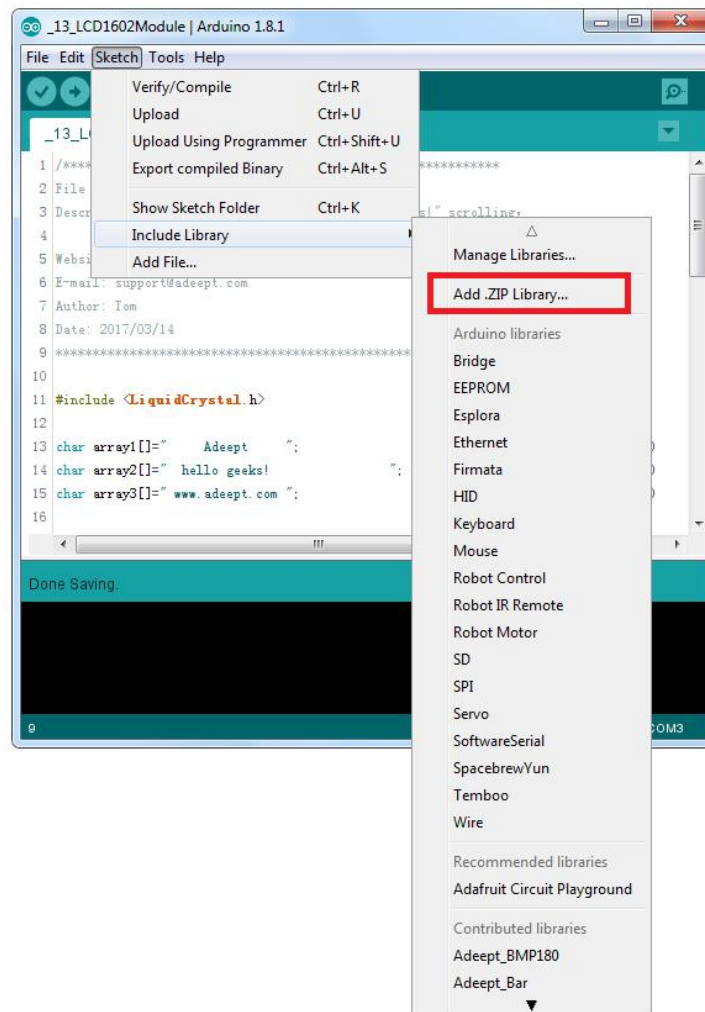
Experimental Procedures

Step 1: Build the circuit



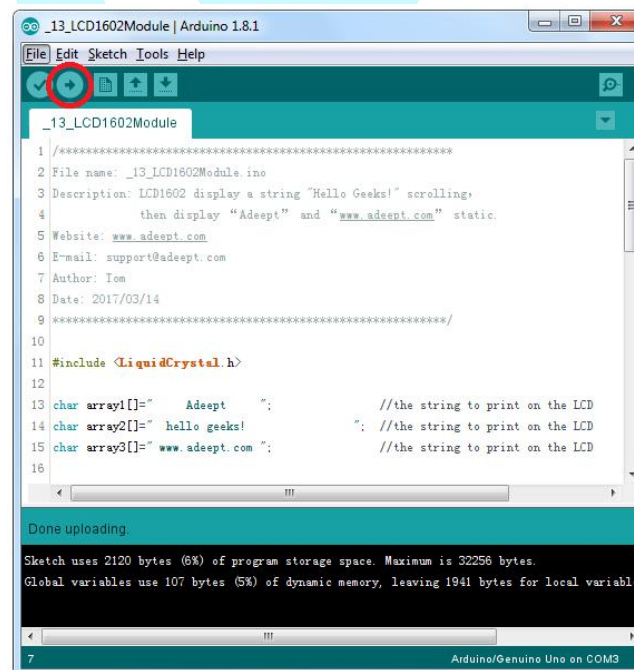
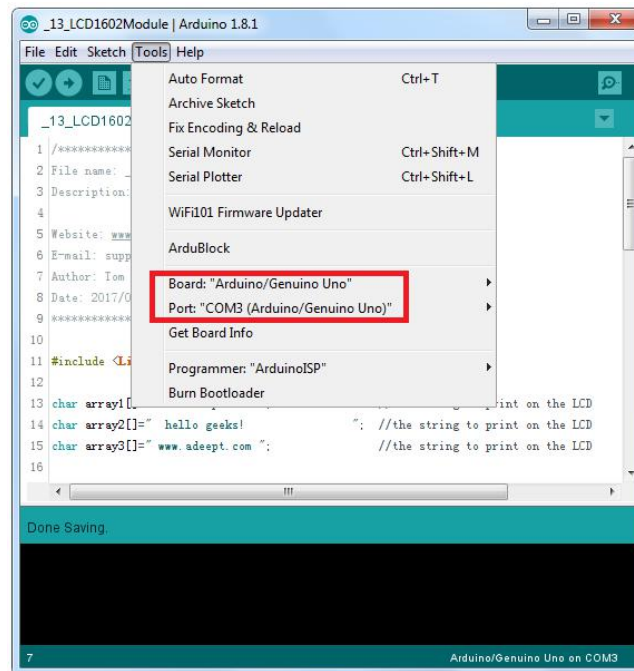
Adeept UNO R3 Board	LCD1602 Module	Potentiometer Module
GND	VSS	-
5V	VDD	+
	VO	A
D4	RS	
GND	RW	
D6	E	
D10	D4	
D11	D5	
D12	D6	
D13	D7	
5V	A	
GND	K	

Step 2: Install the function library (LiquidCrystal.zip).

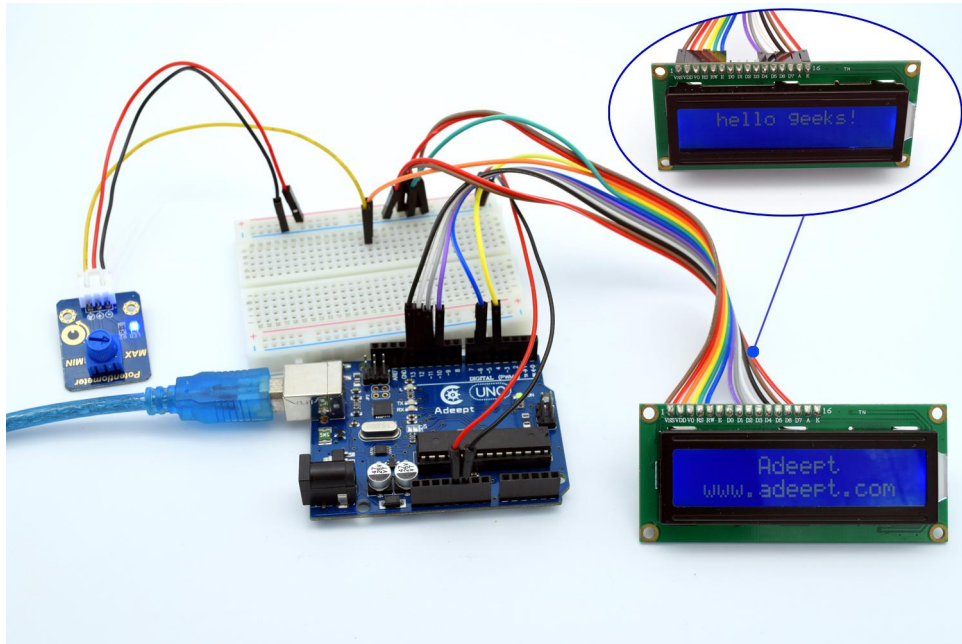


Step 3: Program _13_LCD1602Module.ino

Step 4: Compile and download the sketch to the UNO R3 board.



Now you can see on the LCD1602 display, "Hello Geeks!" is scrolling first, and then "Adeept www.adeept.com" appears and stays unchanged.



Adeept

Lesson 14 IIC Interface Application

Introduction

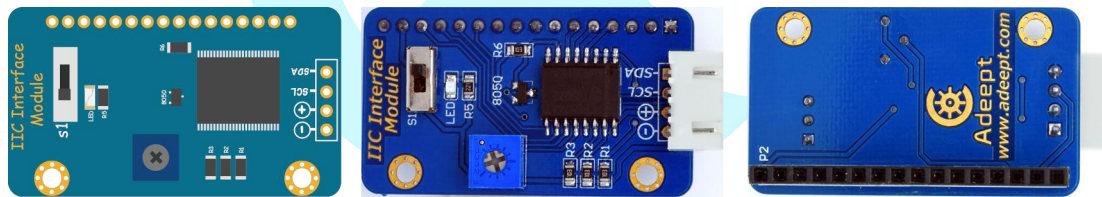
Since there are limited I/O ports on the Arduino UNO R3 board, if you use them to drive the LCD1602, it needs many of the ports and there may be insufficient to connect other devices. To solve this problem, an IIC (or I2C) Interface Module based on PCF8574 is designed to extend the I/O ports of the Arduino board. You only need two wires (SDA and SCL) to control the LCD1602 and save many ports for the board to connect more sensors.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * IIC Interface Module
- 1 * LCD1602 Display Module
- 1 * USB Cable
- 1 * 4-Pin Wires

Experimental Principle

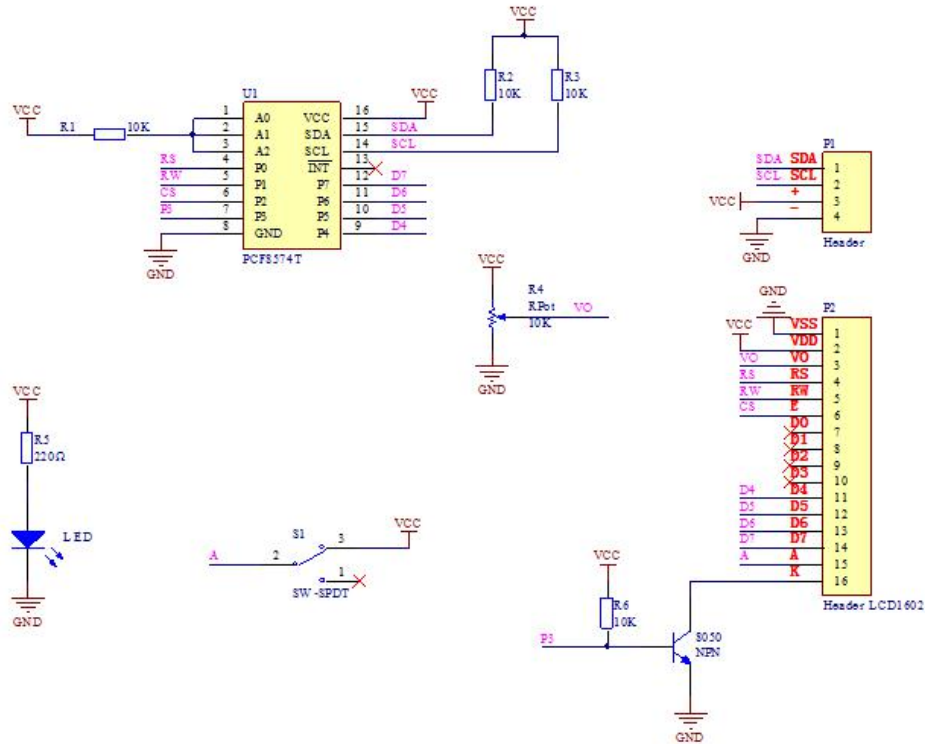
The Fritzing image:



Pin definition:

SDA	Data Pin
SCL	Clock Pin
+	VCC
-	GND

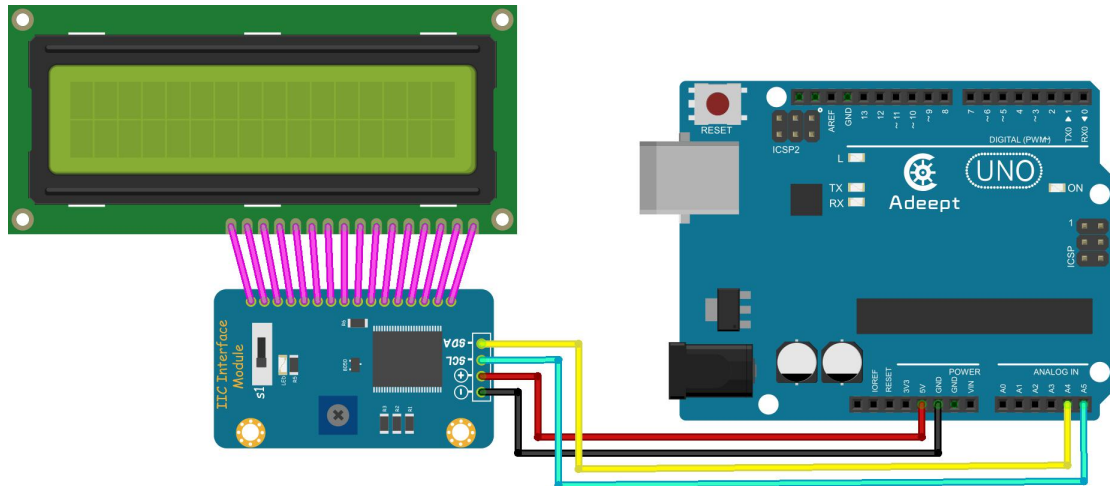
The schematic diagram:



In this experiment, by programming the Arduino, we write data to the IIC Interface Module via the I2C interface on the Arduino board so as to control the LCD1602 to display data as we want.

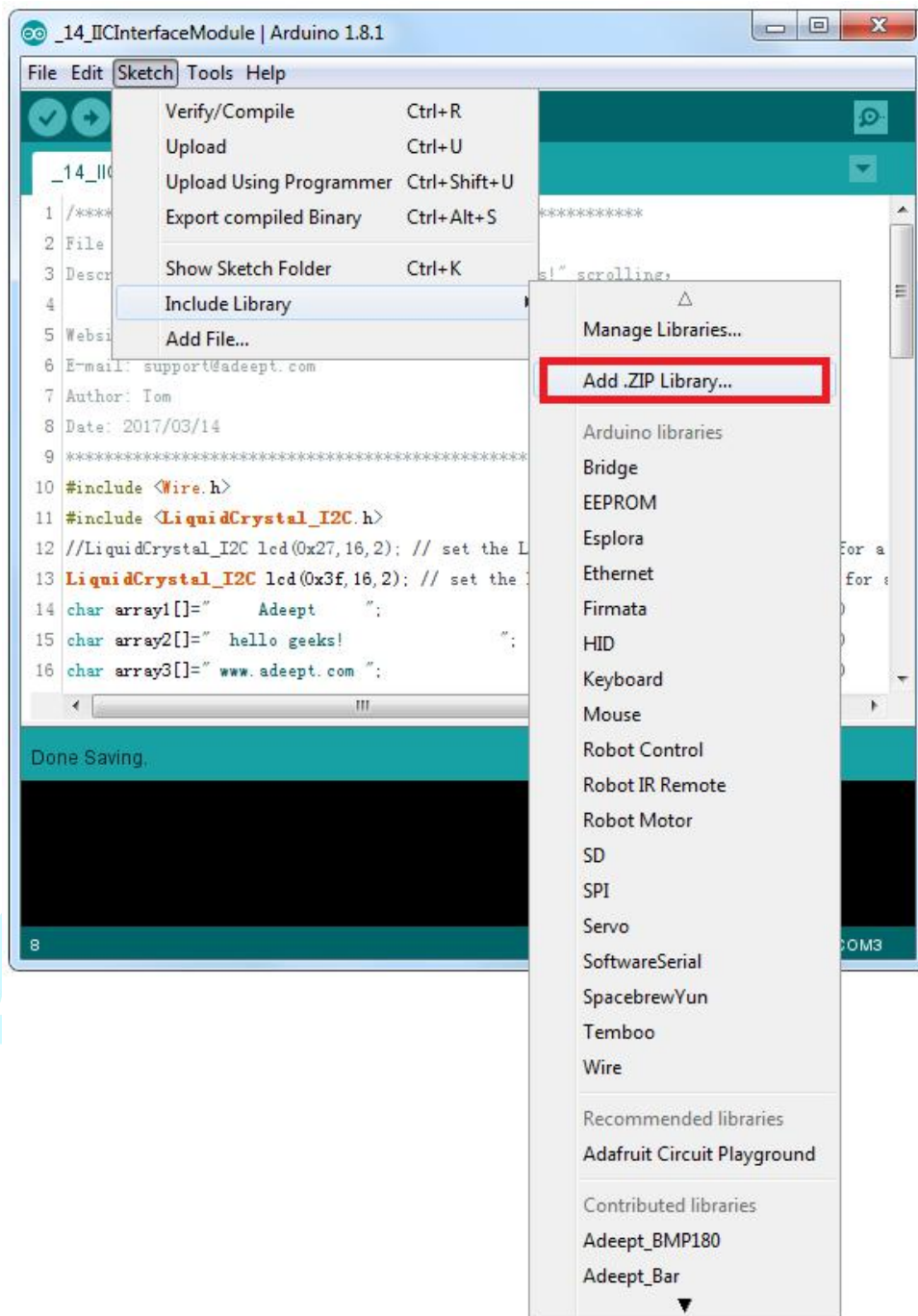
Experimental Procedures

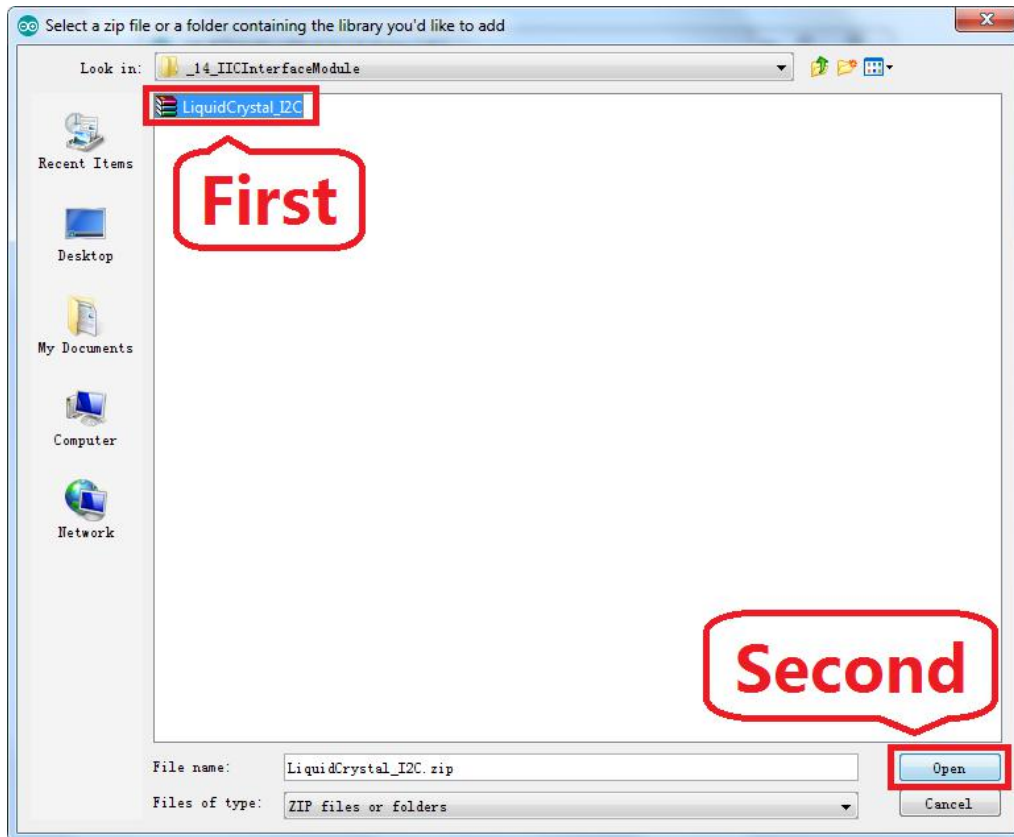
Step 1: Build the circuit



Adeept UNO R3 Board	IIC Interface Module
A4	SDA
A5	SCL
5V	+
GND	-

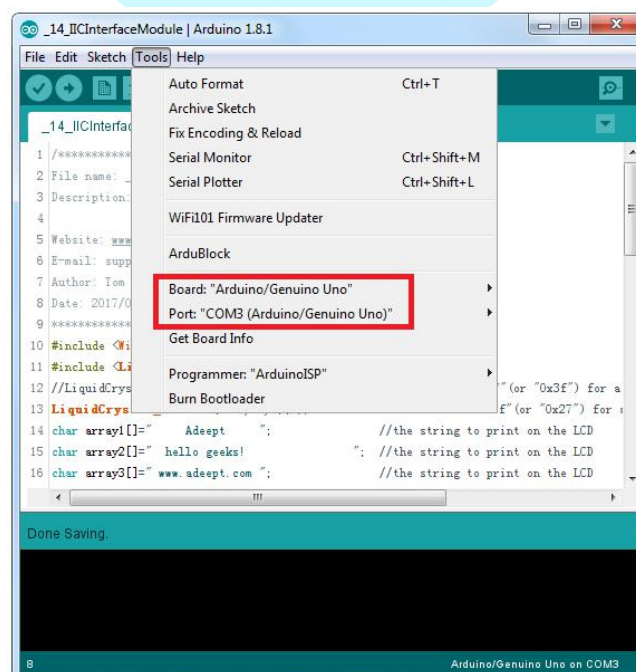
Step 2: Install the function library (LiquidCrystal.zip).

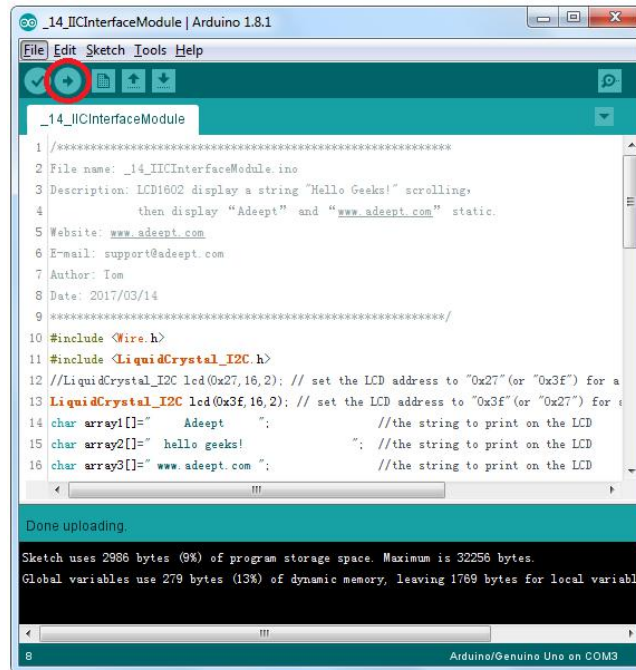




Step 3: Program `_14_IICInterfaceModule.ino`

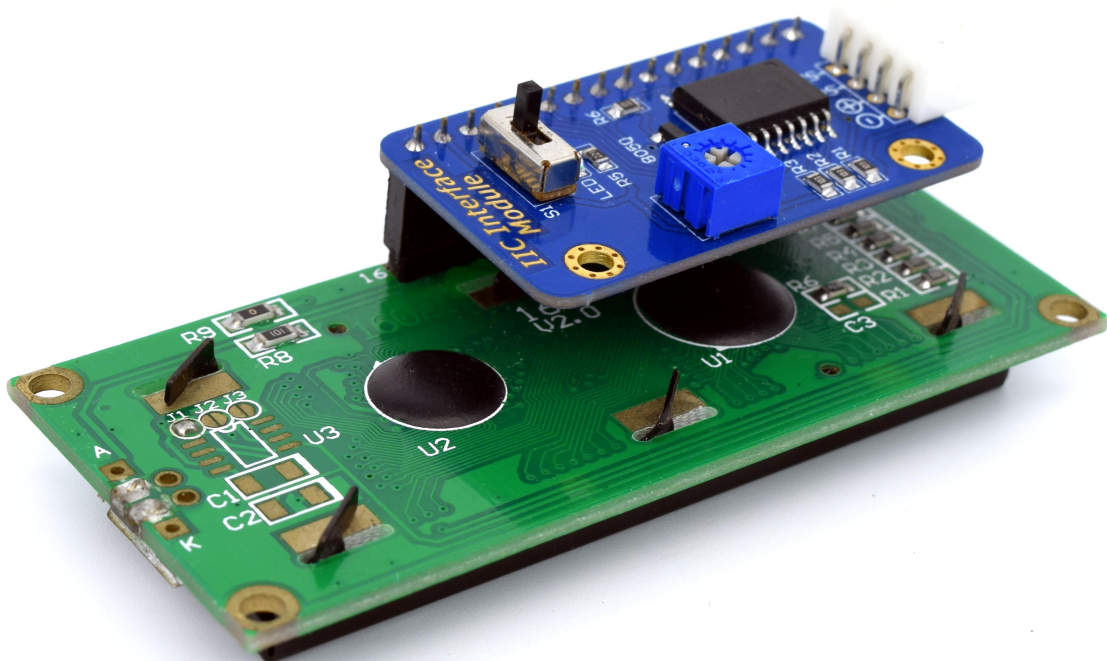
Step 4: Compile and download the sketch to the UNO R3 board.

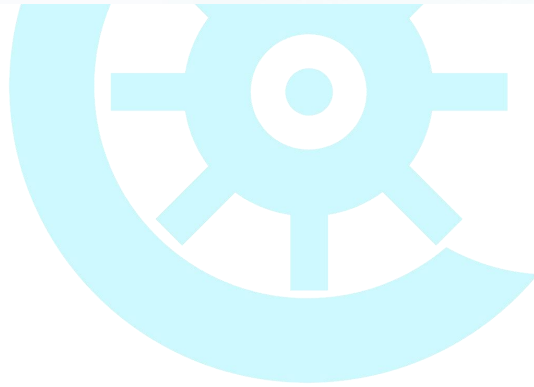
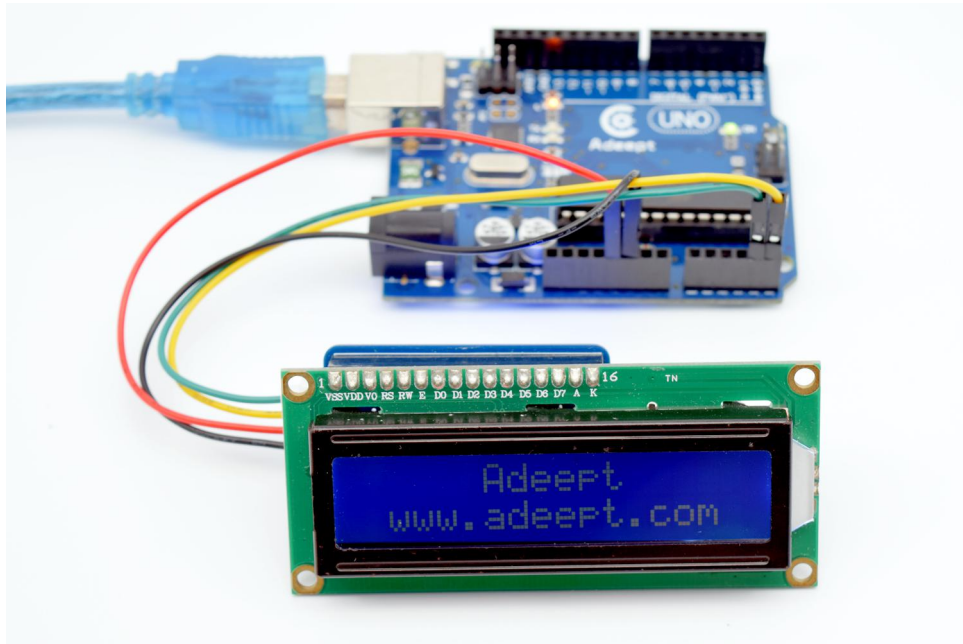




Now you can see on the LCD1602 display, "Hello Geeks!" is scrolling first, and then "Adeept www.adeept.com" appears and stays unchanged.

If characters are not displayed on the LCD, try to turn the potentiometer (a blue knob) for contrast adjustment on the I2C Interface Module. Or, you can toggle the switch on the module to see whether the backlight is on.





Adeept

Lesson 15 Application of Touch Button

Introduction

The Touch Button Module is a touch switch module developed based on the principle of capacitive sensing. Touch of human or metal onto the gilded touch surface can be sensed. Besides, it can also detect other such touch with certain materials like plastic and glass between. The sensitivity in these cases depends on the touched area and thickness of the material between the touch pad and human or metal. The module can be conveniently used to replace physical buttons.

Note: During the application, remember to leave some space in height between the module and fixed surface in case of signal errors.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Touch Button Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

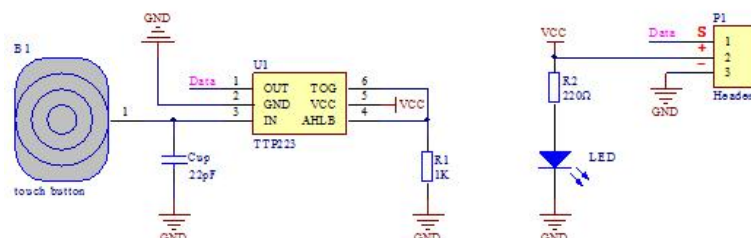
The Fritzing image:



Pin definition:

S	Digital output
+	VCC
-	GND

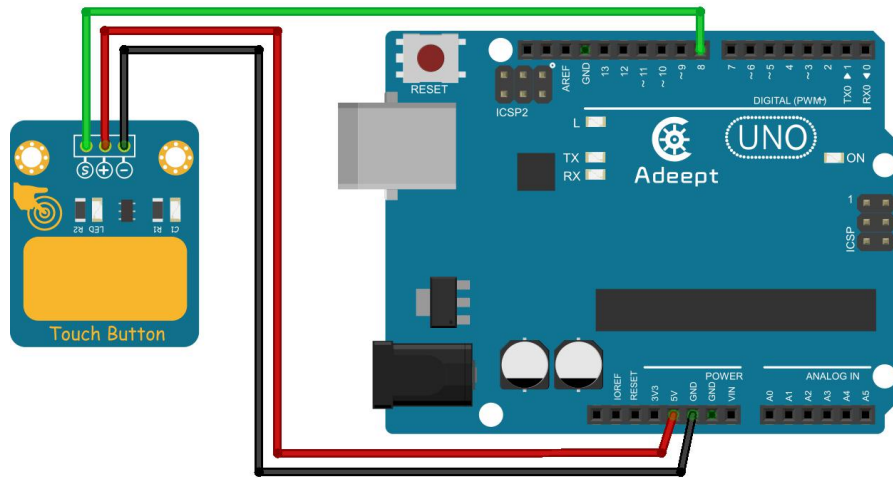
The schematic diagram:



In this experiment, by programming the Arduino, we detect the High or Low of the output terminal of the Touch Button Module by pin D8 of the Arduino board, so as to tell whether fingers touched the touch button or not.

Experimental Procedures

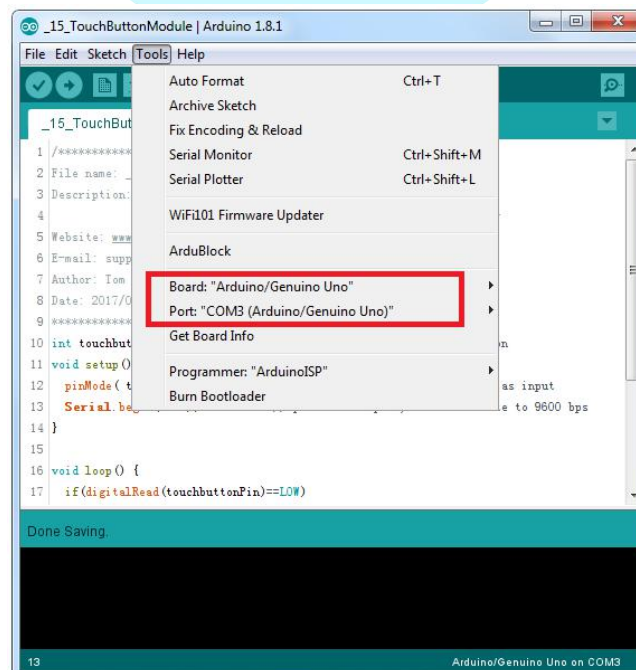
Step 1: Build the circuit

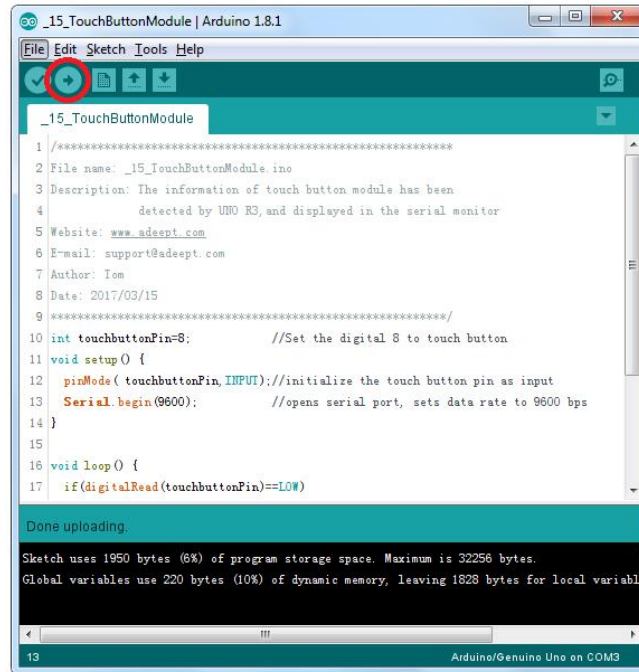


Adept UNO R3 Board	Touch Button Module
D8	S
5V	+
GND	-

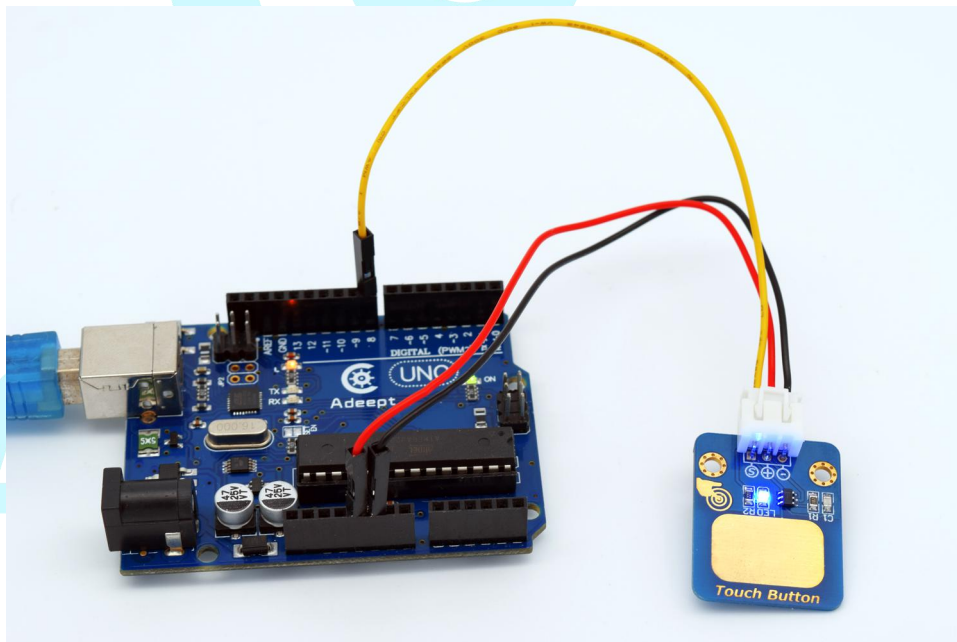
Step 2: Program _15_TouchButtonModule.ino

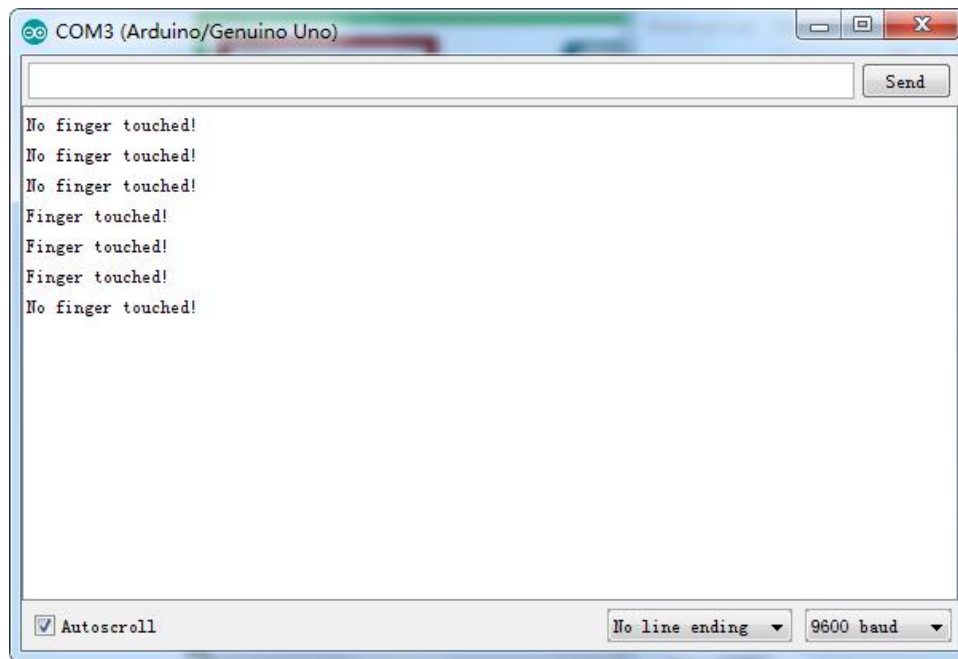
Step 3: Compile and download the sketch to the UNO R3 board.





Now open the Serial Monitor in Arduino IDE. Touch the Touch Button Module and you'll see "Finger touched!" on the window; move away and "No finger touched!" will appear.





Adeept

Lesson 16 Pulse Count

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 * Adept Arduino UNO R3 Board
- 1 * Rotary Encoder Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

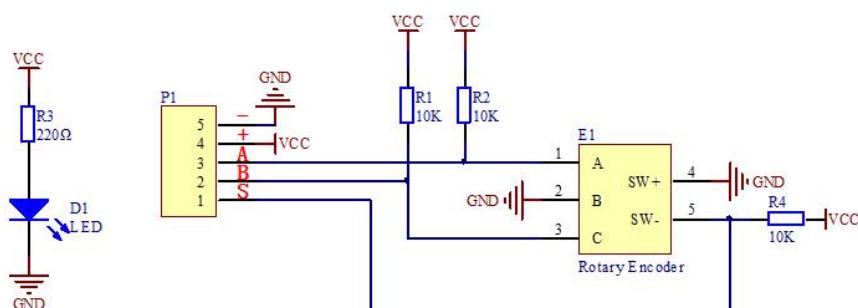
The Fritzing image:



Pin definition:

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

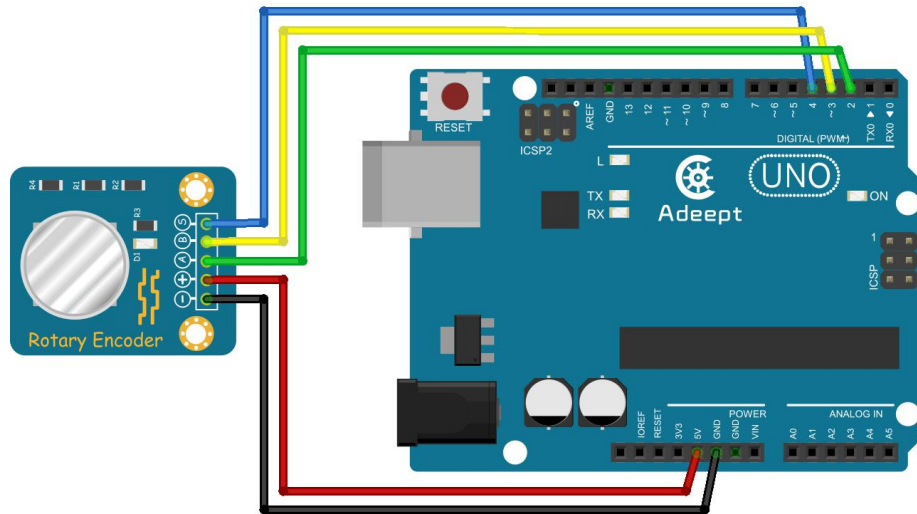
The schematic diagram:



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

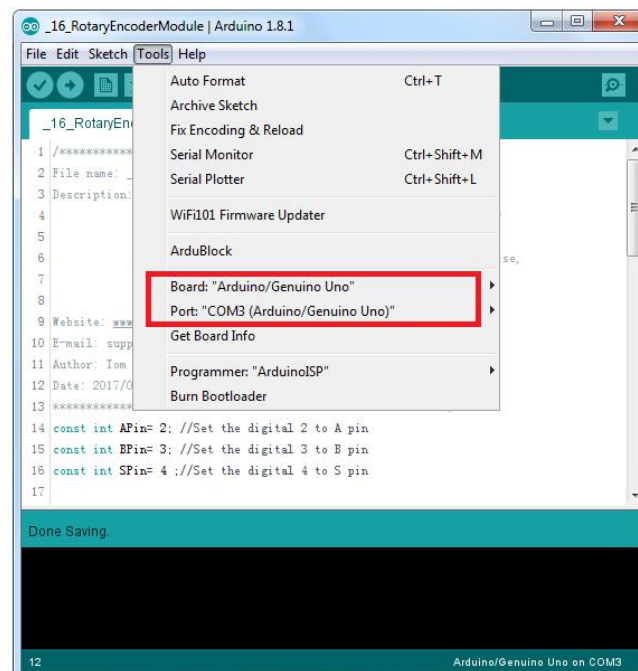
Step 1: Build the circuit

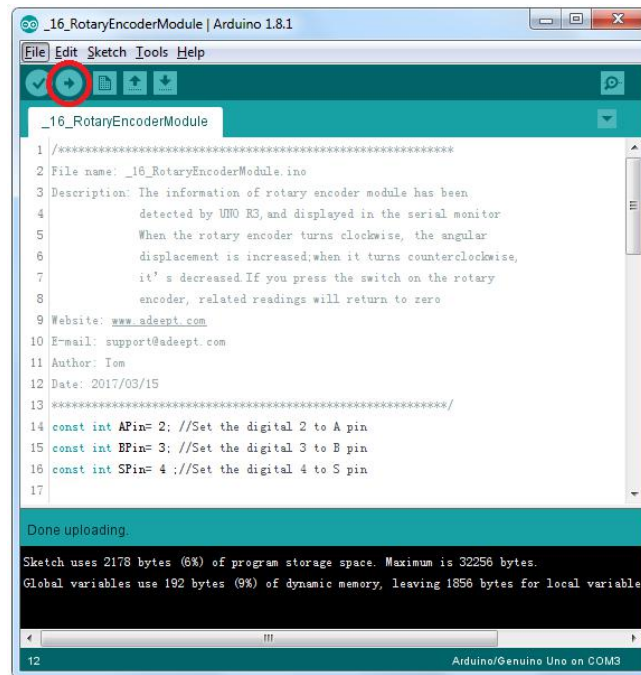


Adept UNO R3 Board	Rotary Encoder Module
D4	S
D3	B
D2	A
5V	+
GND	-

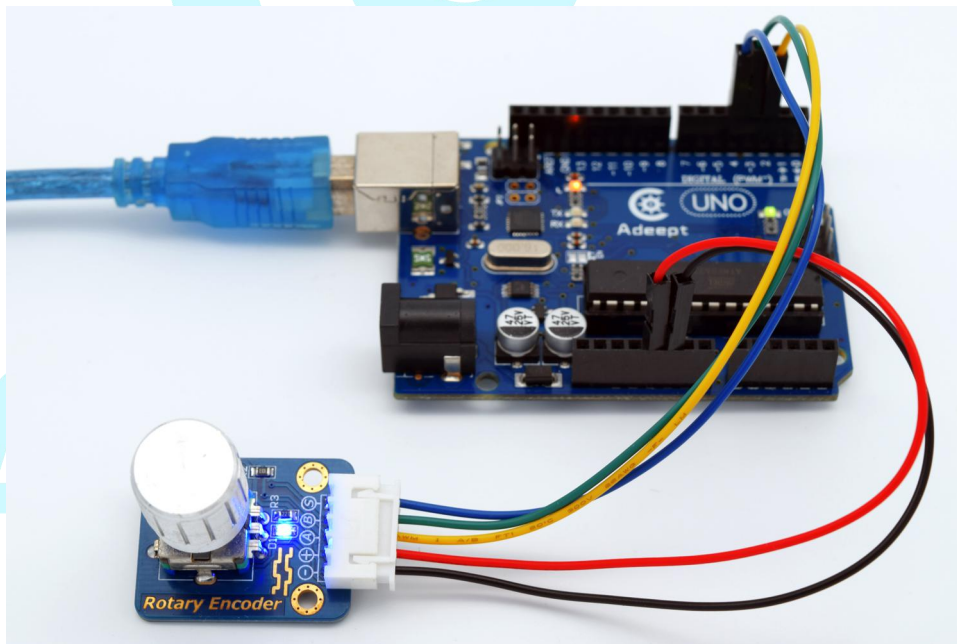
Step 2: Program _16_RotaryEncoderModule.ino

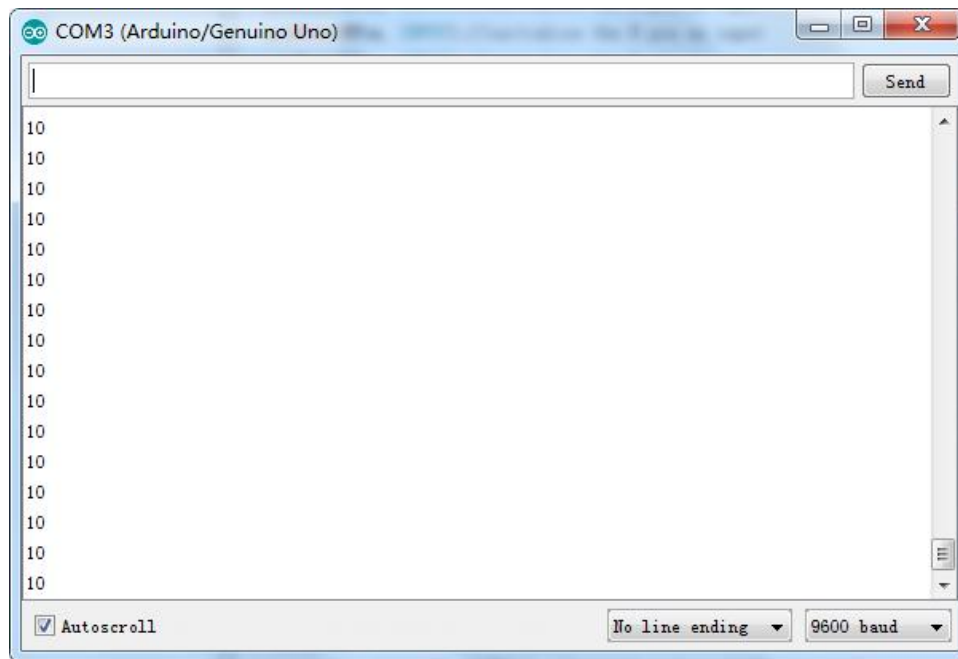
Step 3: Compile and download the sketch to the UNO R3 board.





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.





Adeept

Lesson 17 Impact checking

Introduction

Limit Switch, or travel switch, can be installed on relatively stationary objects such as mounting bracket and door frame, or moving objects like car and door. When the moving object approaches the stationary one, the switch is closed; when the moving one moves away from the static one, the switch is open.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * Limit Switch Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

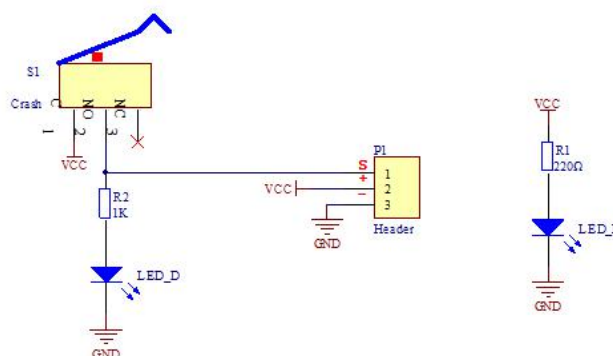
The Fritzing image:



Pin definition:

S	Digital output
+	VCC
-	GND

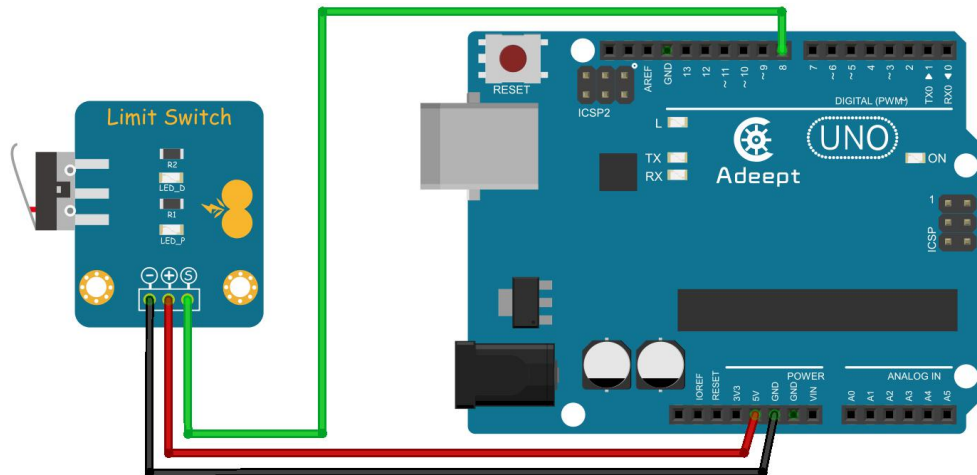
The schematic diagram:



In this experiment, by programming the Arduino, we detect the status of the Limit Switch module through pin D8 of the Arduino board and display it on Serial Monitor via the serial port.

Experimental Procedures

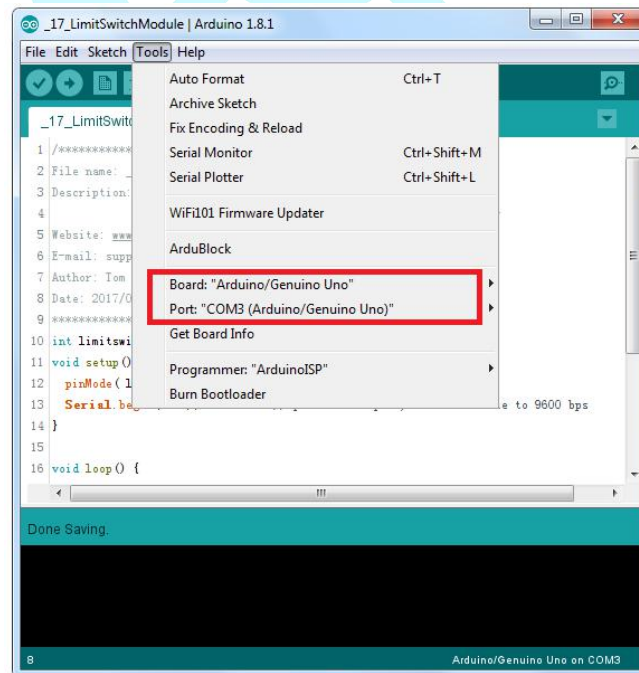
Step 1: Build the circuit

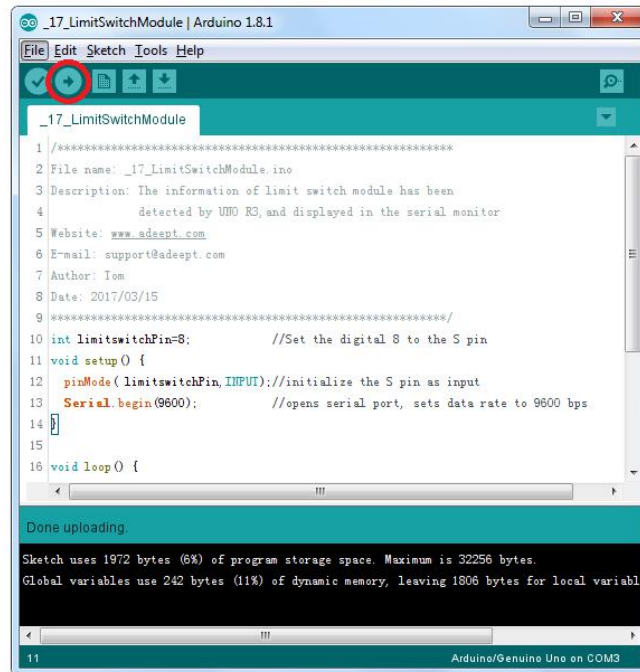


Adept UNO R3 Board	Limit Switch Module
D8	S
5V	+
GND	-

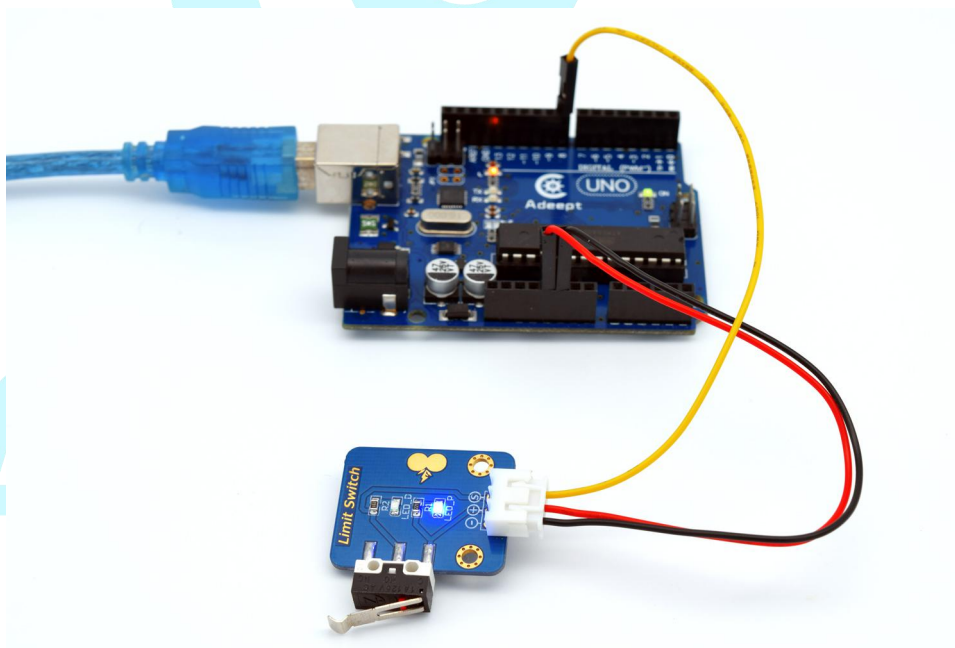
Step 2: Program _17_LimitSwitchModule.ino

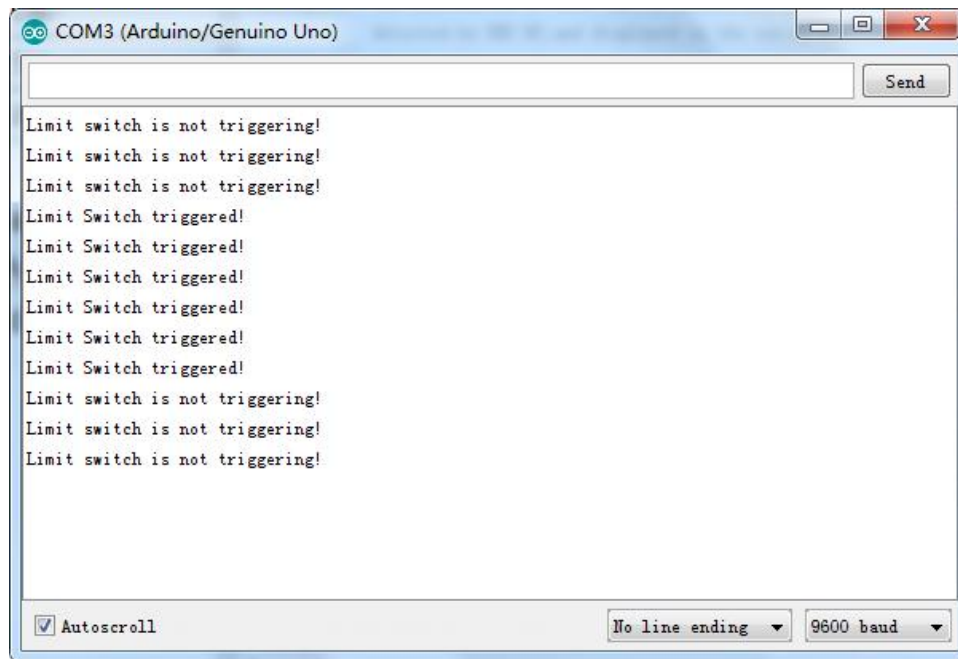
Step 3: Compile and download the sketch to the UNO R3 board.





Open the Serial Monitor in Arduino IDE. Press the Limit Switch and you can see "Limit Switch triggered!" on the window.





Adeept

Lesson 18 Simple Laser Cannon

Introduction

Semiconductor laser modules are widely used in laser communication, ranging, lidar, ignition and blasting, and testing instruments.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * Laser Transmitter Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

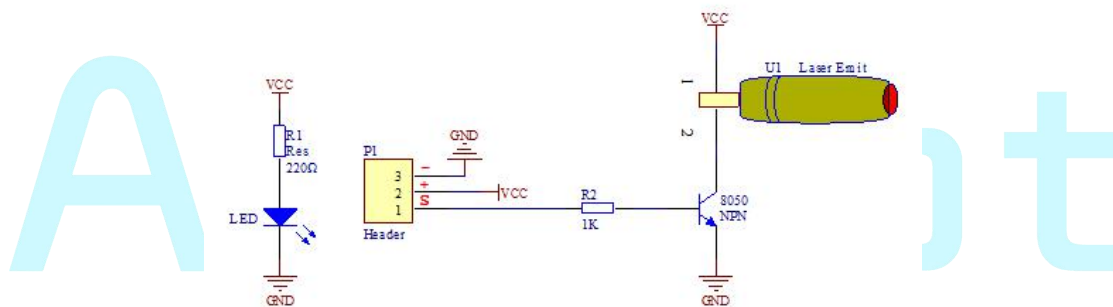
The Fritzing image:



Pin definition:

S	Digital Input
+	VCC
-	GND

The schematic diagram:

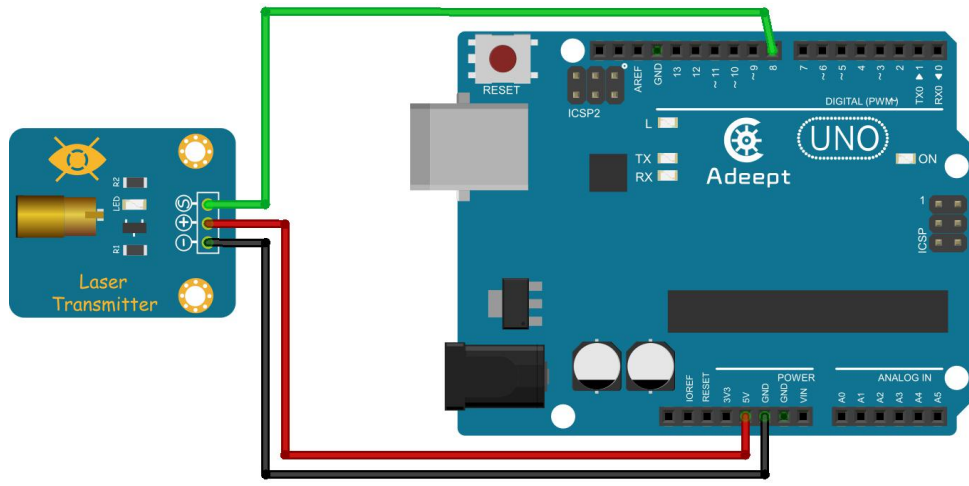


In this experiment, by programming the Arduino, we control the Laser Transmitter Module to emit laser by pin A0 of the Arduino board.

Note: DO NOT look directly into the laser!

Experimental Procedures

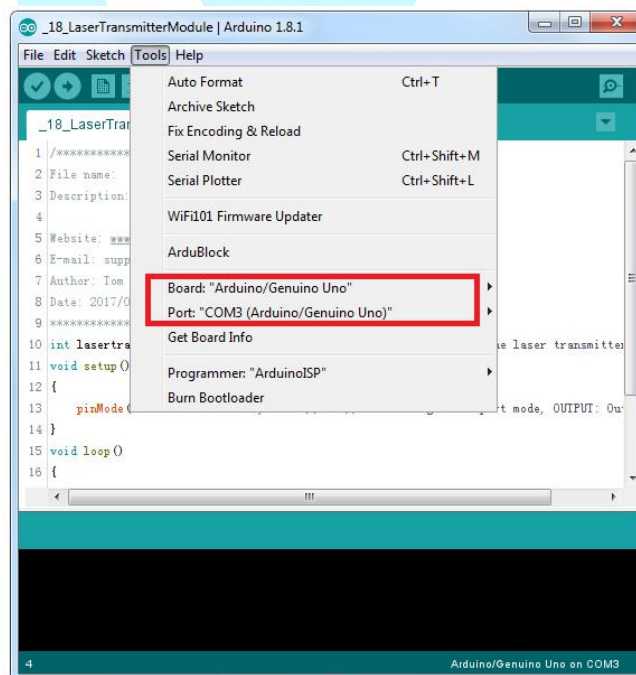
Step 1: Build the circuit

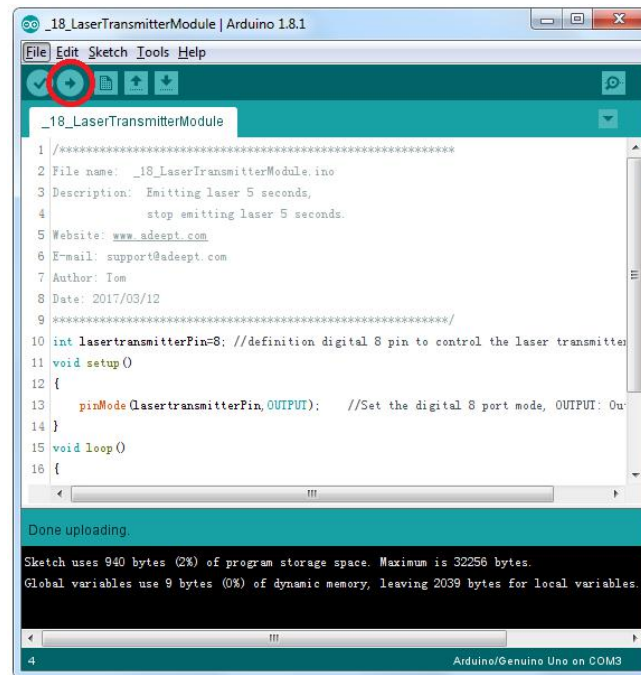


Adeept UNO R3 Board	Laser Transmitter Module
D8	S
5V	+
GND	-

Step 2: Program _18_LaserTransmitterModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.

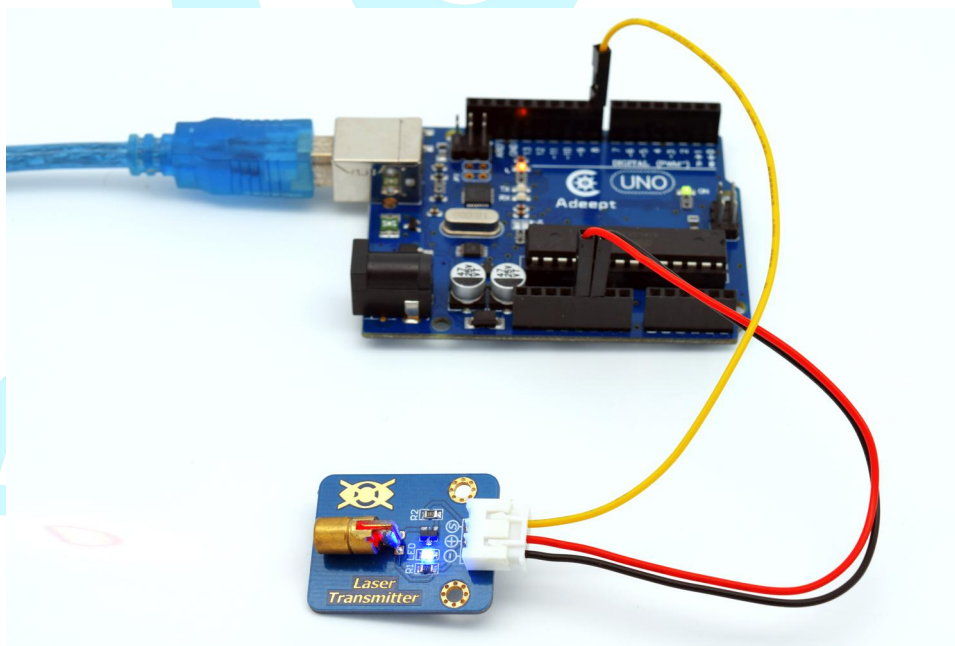




The screenshot shows the Arduino IDE interface with the sketch '_18_LaserTransmitterModule' loaded. The 'Upload' button is highlighted with a red circle. Below the code editor, a status bar indicates 'Done uploading.' and provides memory usage statistics: 'Sketch uses 940 bytes (2%) of program storage space. Maximum is 32256 bytes. Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables.'

```
1 /*****  
2 File name: _18_LaserTransmitterModule.ino  
3 Description: Emitting laser 5 seconds,  
4             stop emitting laser 5 seconds.  
5 Website: www.adeept.com  
6 E-mail: support@adeept.com  
7 Author: Tom  
8 Date: 2017/03/12  
9 *****/  
10 int lasertransmitterPin=8; //definition digital 8 pin to control the laser transmitter  
11 void setup()  
12 {  
13     pinMode(lasertransmitterPin, OUTPUT); //Set the digital 8 port mode, OUTPUT: Out  
14 }  
15 void loop()  
16 {
```

Now you can see the Laser Transmitter Module emit laser and the emission lasts for 5 seconds, and then it stops. After 5s, the cycle repeats.



Lesson 19 Simple Laser Targeting

Introduction

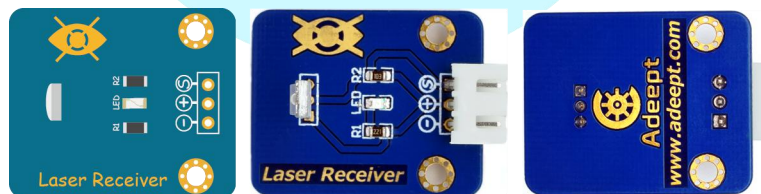
The principle for many laser receiving devices is the same. The laser ray goes through the optical lens and then is received by the photosensitive device, i.e. the photodiode. After receiving the rays, the photodiode will generate currents accordingly (based on the light intensity) which then output electrical signal after running through the amplifier. Then use an I/O port of the Arduino Uno board to detect the output terminal of the Laser Receiver module, and thus we can tell whether there are rays shone on the module.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * Laser Receiver Module
- 1 * Laser Transmitter Module
- 1 * USB Cable
- 2 * 3-Pin Wires
- 2 * Hookup Wire Set
- 1 * Breadboard

Experimental Principle

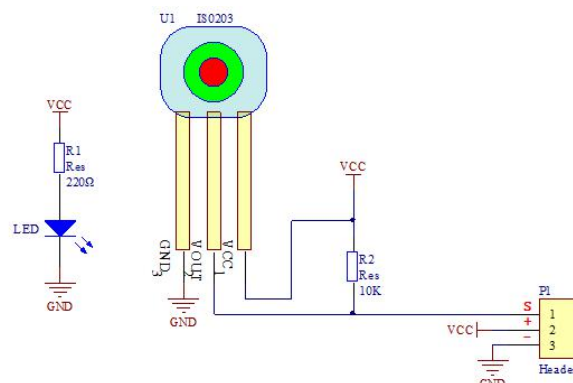
The Fritzing image:



Pin definition:

S	Digital output
+	VCC
-	GND

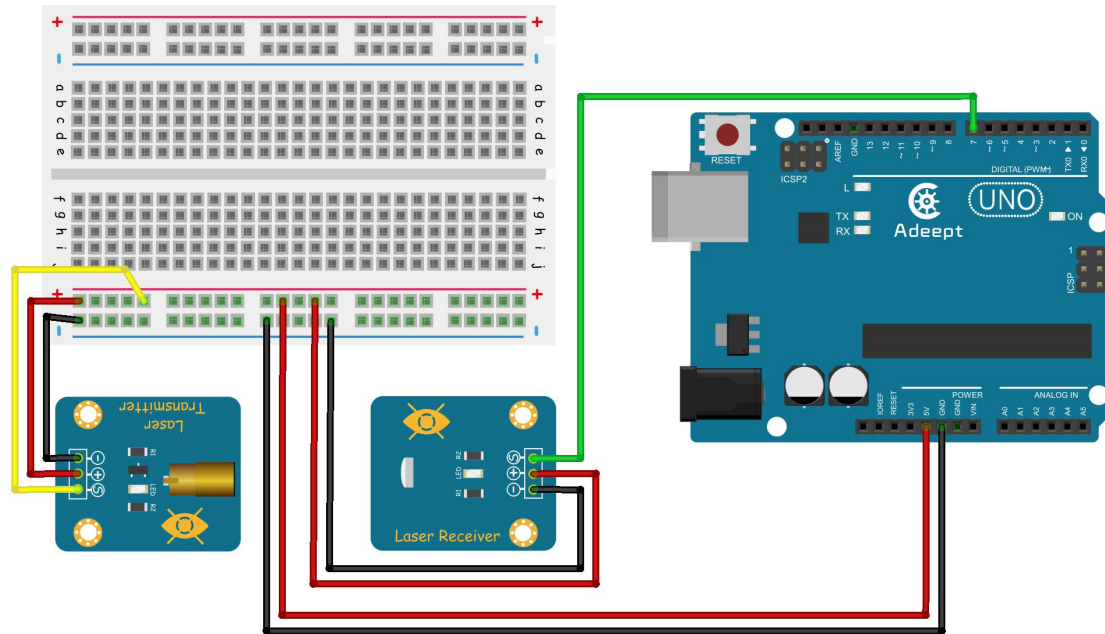
The schematic diagram:



In this experiment, we use the Laser Receiver module to detect whether there is laser ray shining on the module. If yes, the output pin (S) of the module will output Low.

Experimental Procedures

Step 1: Build the circuit



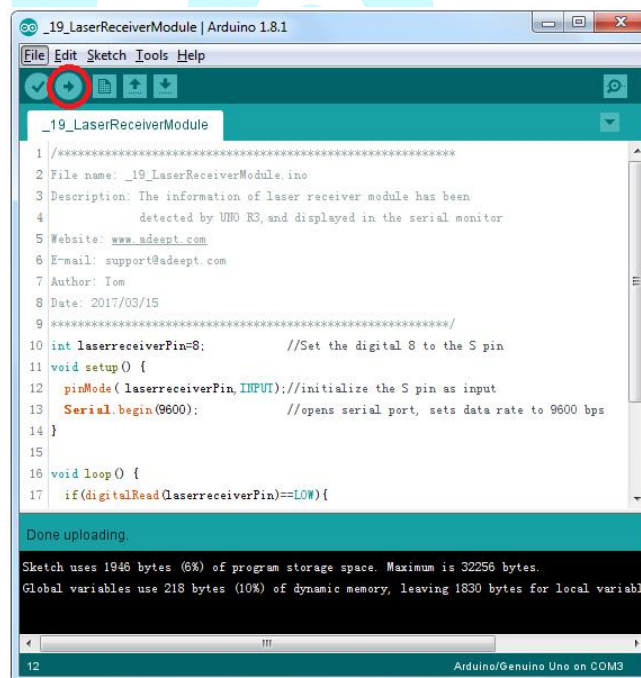
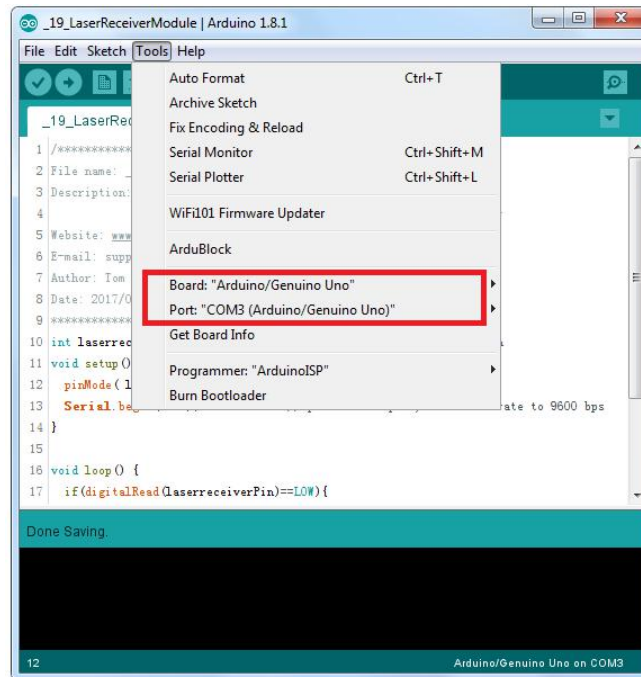
Adept UNO R3 Board	Laser receiver Module
D7	S
5V	+
GND	-

Note: DO NOT look directly into the laser!

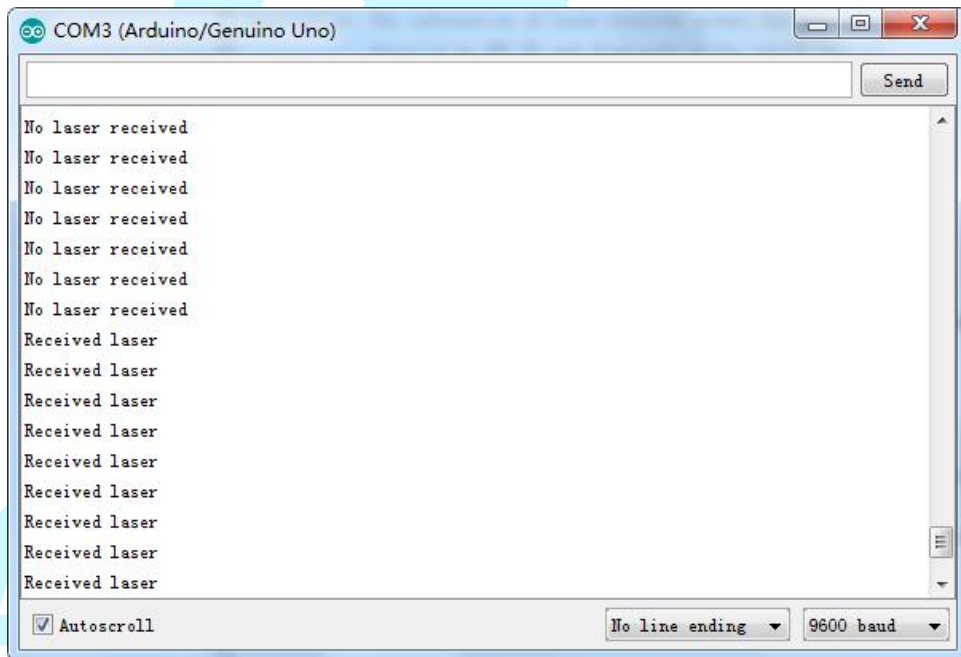
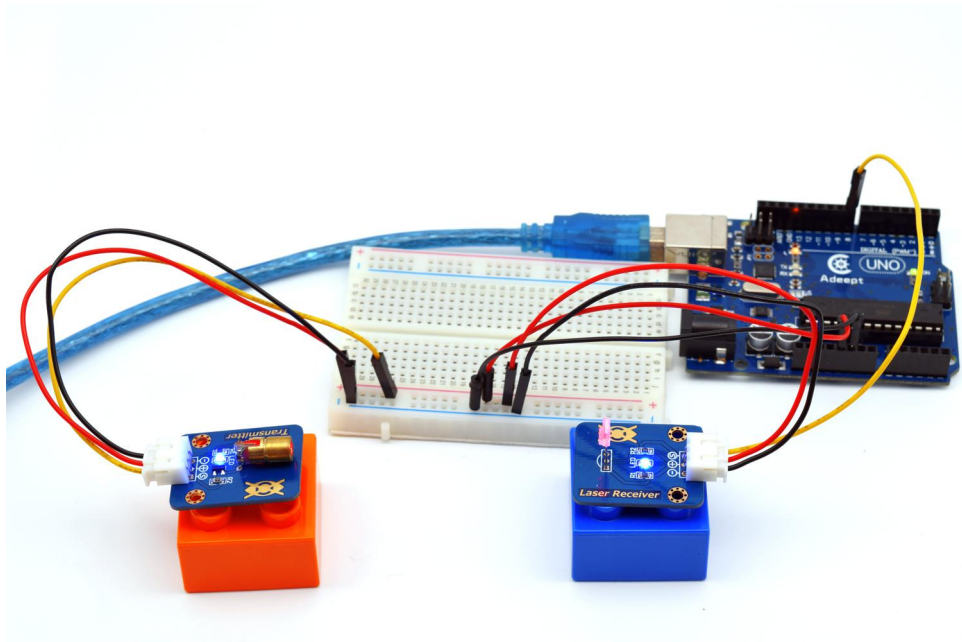
Step 2: Program _19_LaserReceiverModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.

Adept



Open the Serial Monitor in Arduino IDE. Make the Laser Transmitter module to shoot laser ray onto the Laser Receiver module. Then "Received laser" will be displayed on the window. Remove the transmitter module and "No laser received" will be shown.



Lesson 20 Temperature And Humidity Detection

Introduction

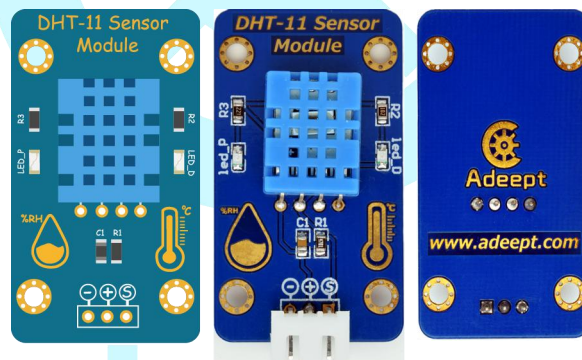
DHT11 is a composite digital thermal sensor that integrates temperature and humidity detection. It can convert the temperature and humidity analog values into digital values via corresponding sensitive components and built-in circuits, which can be directly read by computer or other data collecting devices.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * DHT-11 Sensor Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

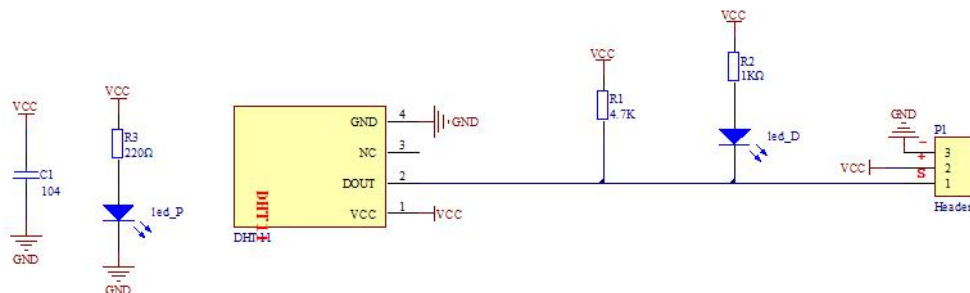
The Fritzing image:



Pin definition:

S	Digital output
+	VCC
-	GND

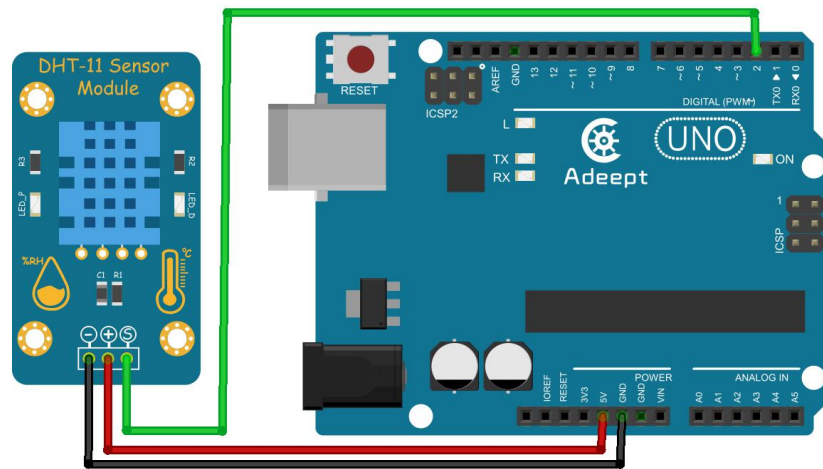
The schematic diagram:



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.

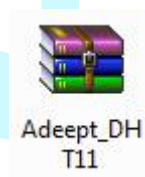
Experimental Procedures

Step 1: Build the circuit

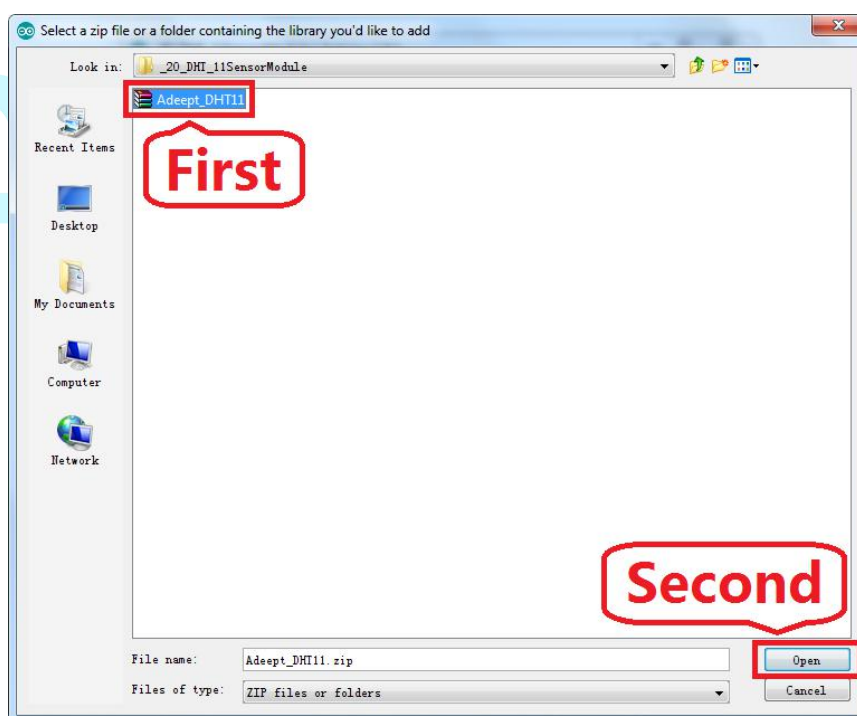
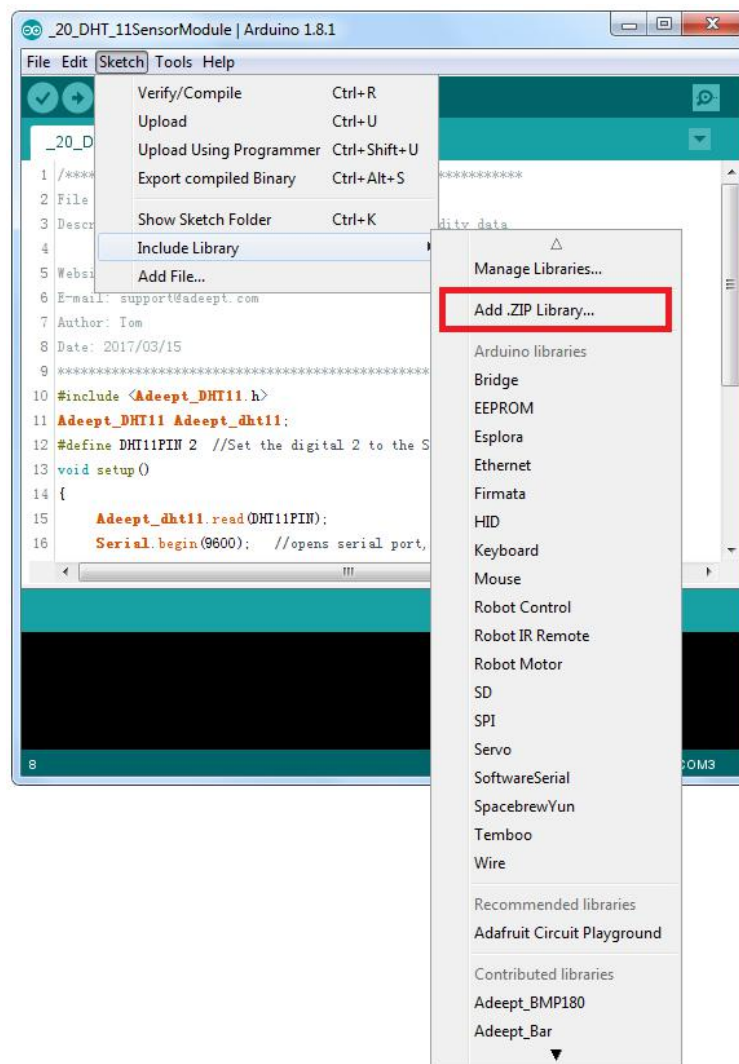


Adeept UNO R3 Board	DHT-11 Sensor Module
D2	S
5V	+
GND	-

Step 2: Install the function library (Adeept_DHT11.zip).

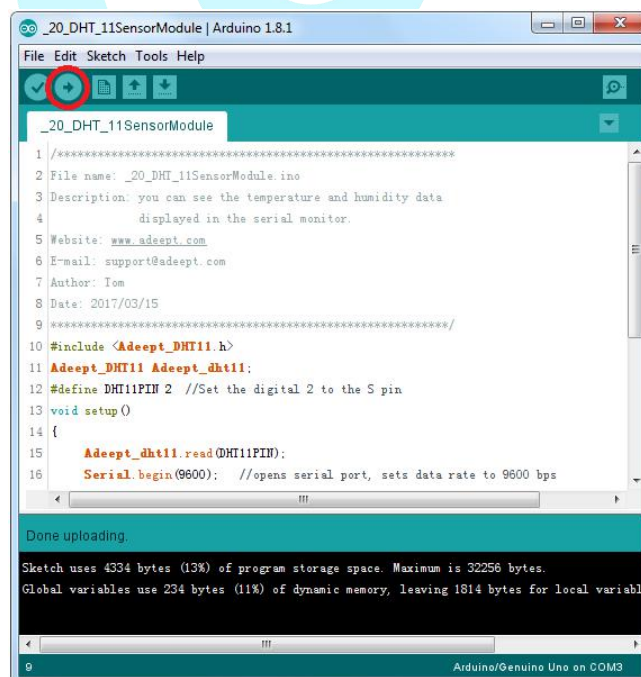
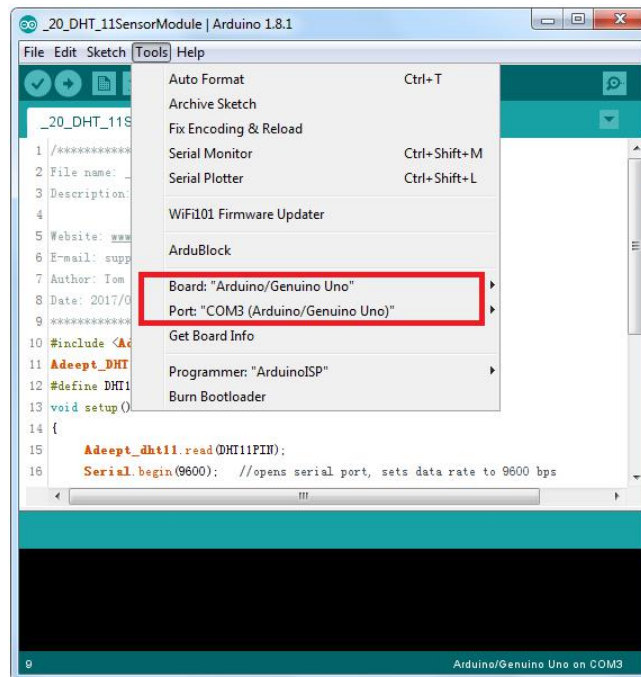


Adeept

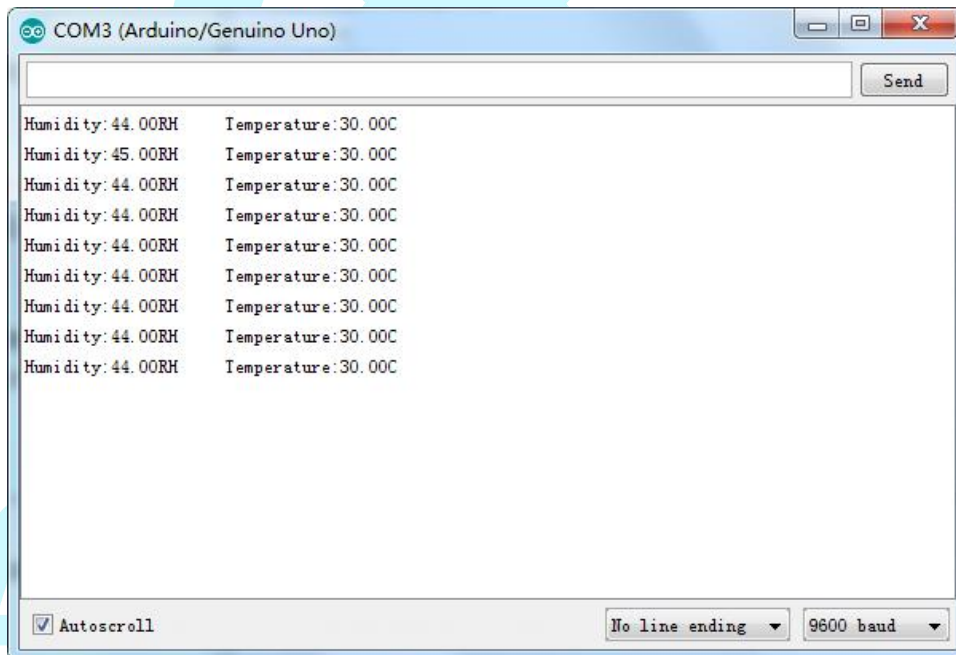
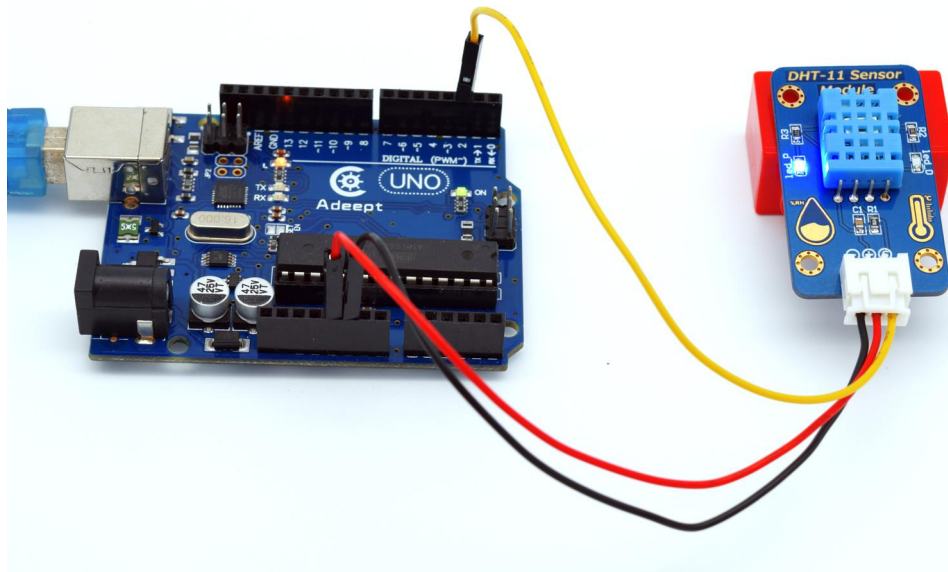


Step 3: Program _20_DHT_11SensorModule.ino

Step 4: Compile and download the sketch to the UNO R3 board.



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



Lesson 21 How To Use The Reed

Introduction

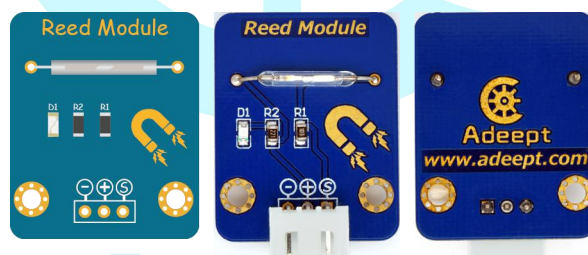
Reed switch is a special magnet-sensitive switch. Inside the glass tube, the reed sheets placed in parallel with a gap between compose the normally-open contact. When a magnet approaches the reed switch, or after the coil wrapped on the reed is electrified and a magnet field comes into existence thus magnetizing the reed, the contact of the reed will sense it and become the opposite pole. Due to the principle that different poles attract each other, when the magnetic force is larger than resistance of the reed, the open contacts (sheets) are closed; when the magnetic force decreases to a certain degree, the resistance takes charge again and the reed will back to the original state.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Reed Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

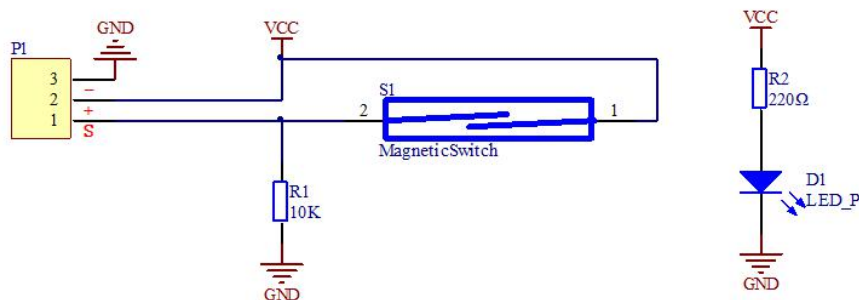
The Fritzing image:



Pin definition:

S	Digital output
+	VCC
-	GND

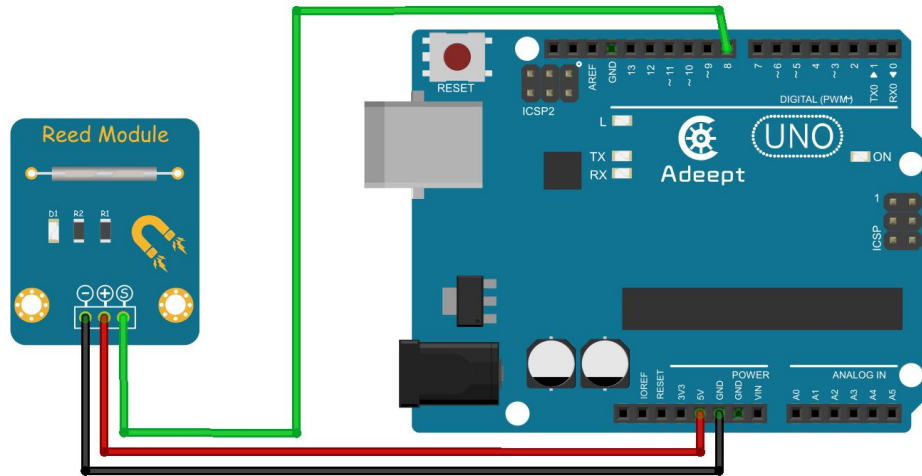
The schematic diagram:



In this experiment, by programming the Arduino, we detect pin D8 of the Arduino board and display the High or Low at D8 on the Serial Monitor.

Experimental Procedures

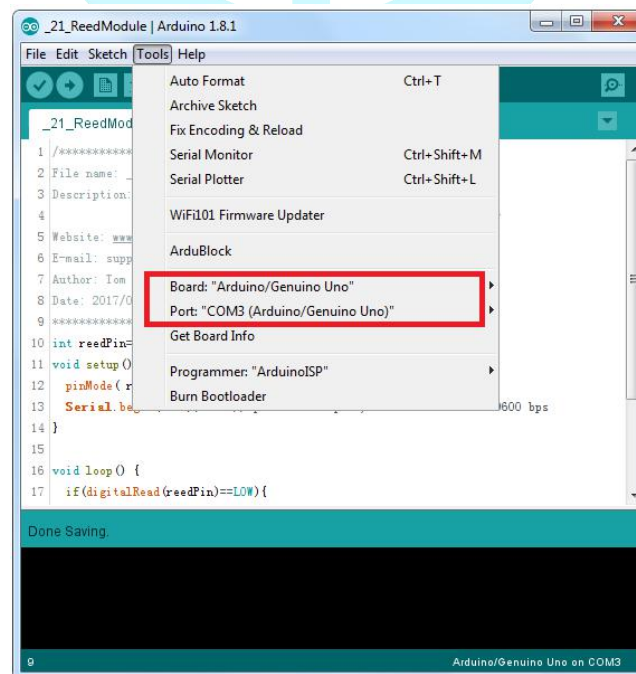
Step 1: Build the circuit

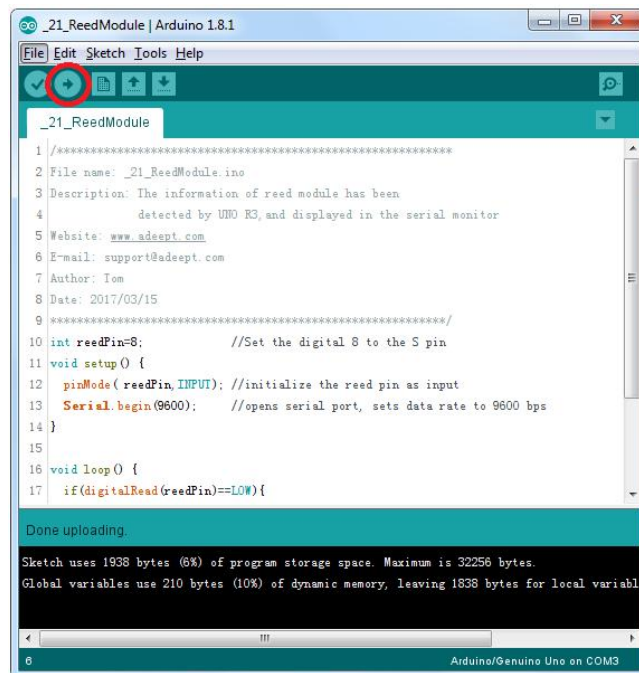


Adept UNO R3 Board	Reed Module
D8	S
5V	+
GND	-

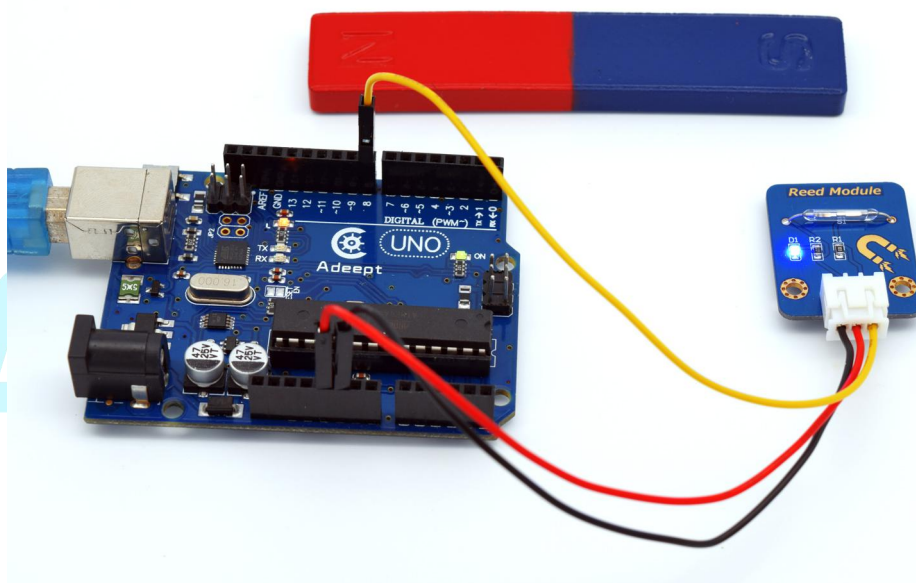
Step 2: Program _21_ReedModule.ino

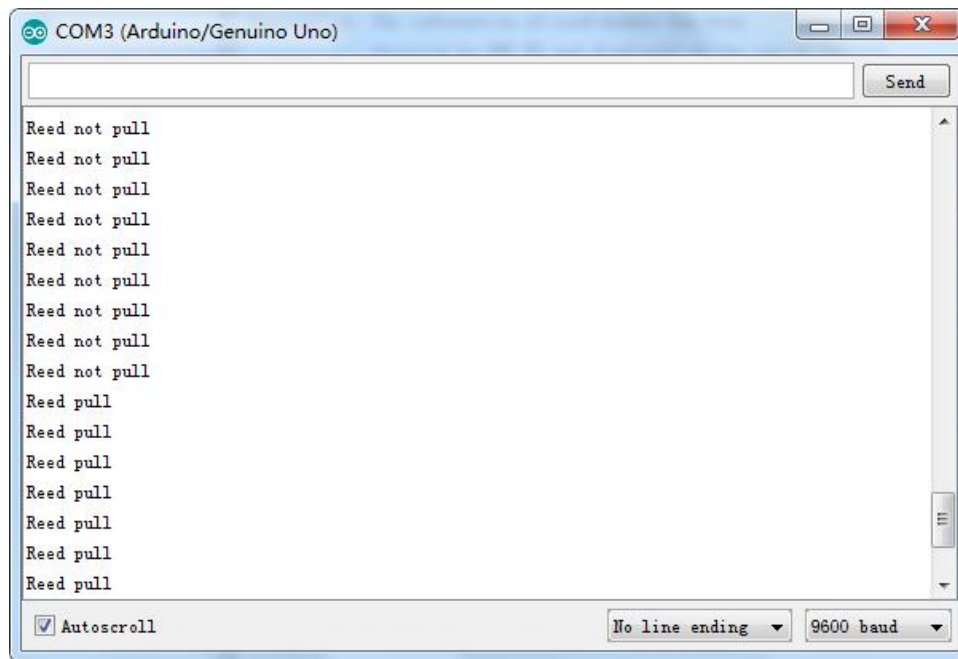
Step 3: Compile and download the sketch to the UNO R3 board.





Open the Serial Monitor in Arduino IDE. Place the magnet near or away from the Reed Module and you will see the High ("Reed pull") and Low ("Reed not pull") change at D8 of the Arduino board on Serial Monitor.





Adeept

Lesson 22 Fire Detection

Introduction

The Flame Sensor detects flames by the special infrared receiver to capture the infrared rays of a specific wavelength in the flames. It supports a detection angle of as high as 60 degrees and works within $-25 - 85^{\circ}\text{C}$. When in use, you need to pay attention and do not place the probe of the sensor too close to the flames in case of damages. Besides, the sensor can be used to detect light intensity. It can detect a light source with a wavelength of 760 - 1100nm. Pin A outputs the analog data collected by the sensor. You can adjust the potentiometer on the module to set the alarm threshold of the sensor. So when the value reaches the threshold, pin S will switch between High and Low.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * Flame Sensor Module
- 1 * USB Cable
- 1 * 4-Pin Wires

Experimental Principle

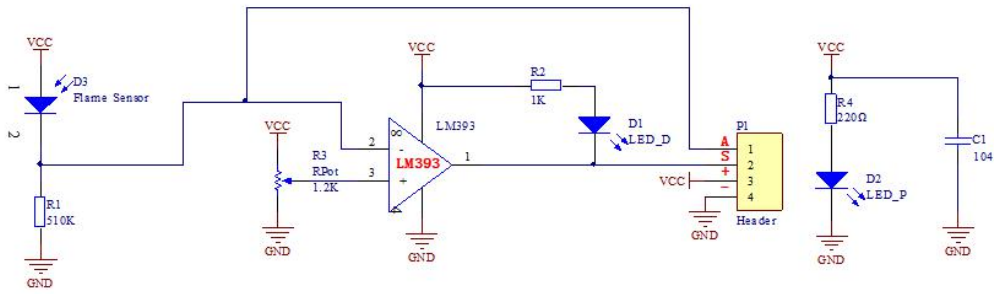
The Fritzing image:



Pin definition:

S	Digital output
A	Analog output
+	VCC
-	GND

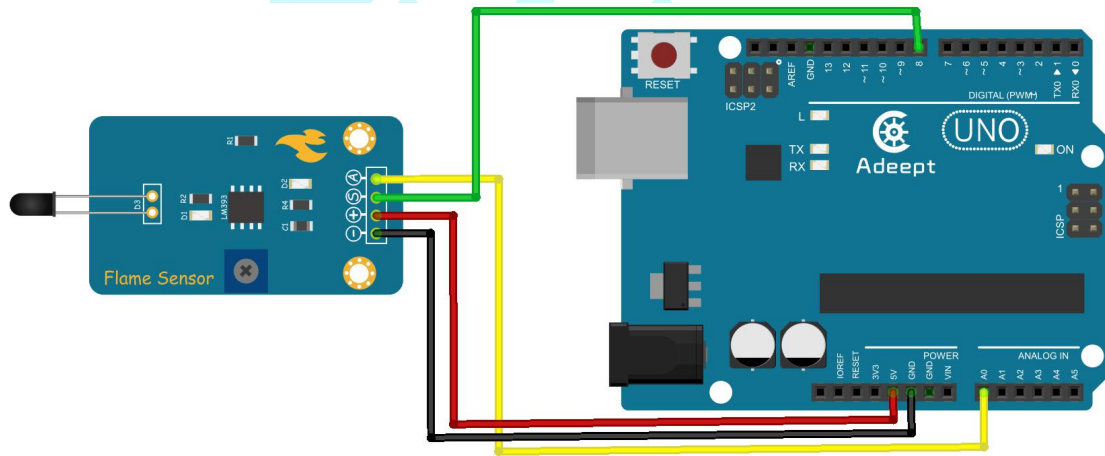
The schematic diagram:



In this experiment, by programming the Arduino, we collect the analog values and switch output from the Flame Sensor module through pin A0 and D8 of the Arduino board, and display them on the Serial Monitor via serial port.

Experimental Procedures

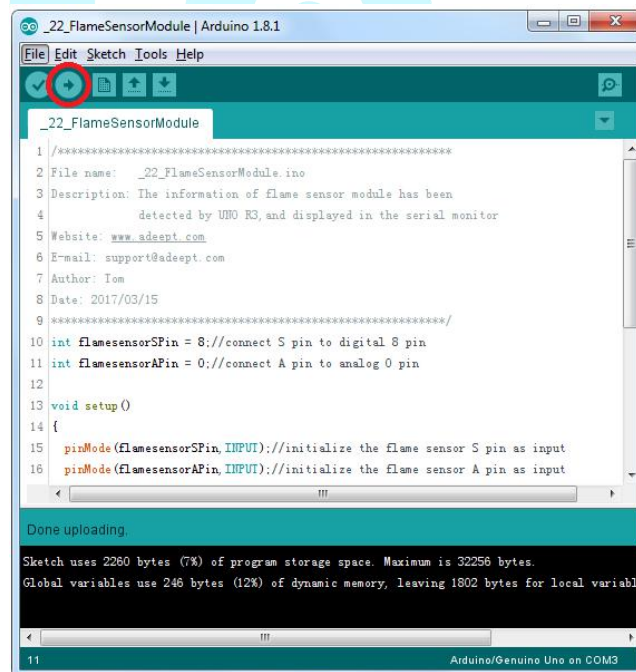
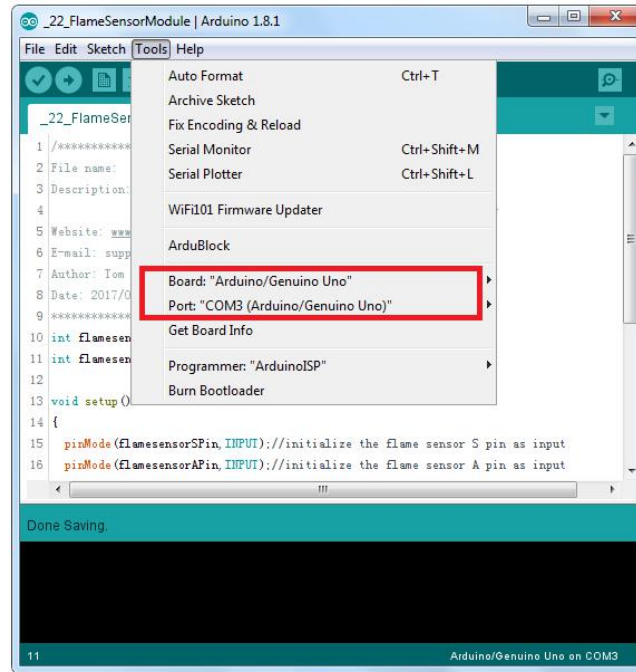
Step 1: Build the circuit



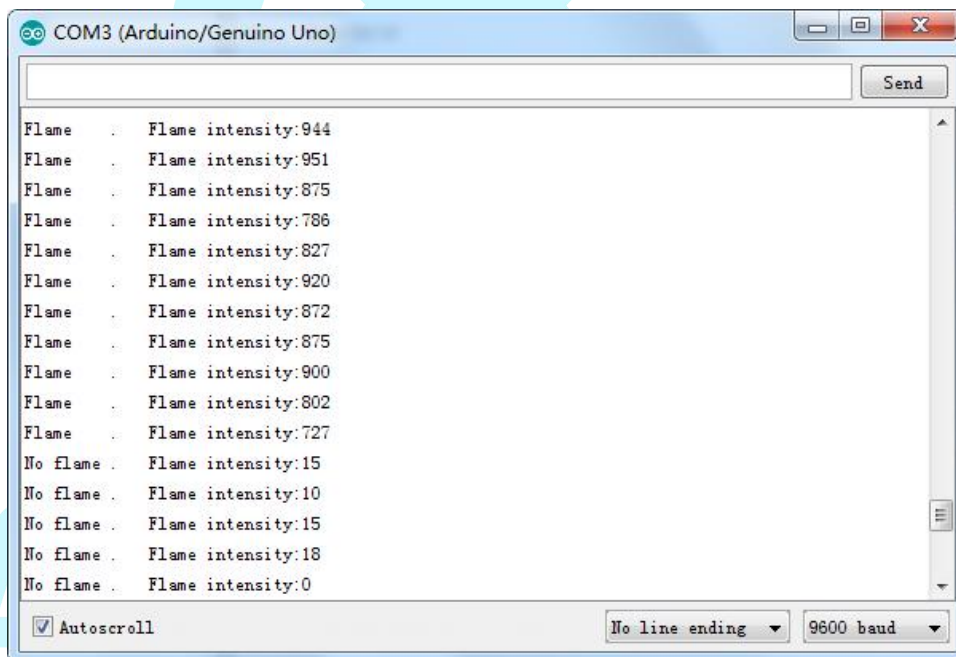
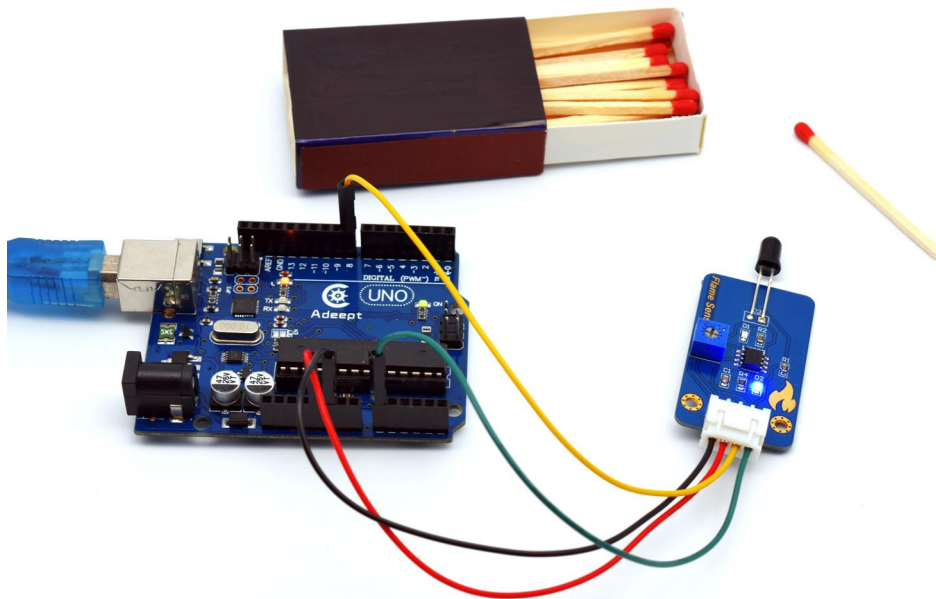
Adept UNO R3 Board	Flame Sensor Module
D8	S
A0	A
5V	+
GND	-

Step 2: Program _22_FlameSensorModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.



Open the Serial Monitor in Arduino IDE. Then you'll see the data detected by the Flame Sensor module on the window. The data includes two parts: "Flame"/"No flame" and "Flame intensity".



Lesson 23 Decetion of Flammable Gases

Introduction

MQ-2 is a sensor that can detect flammable gases such as methane, hydrogen, and propane and so on. It adopts the low conductivity stannic oxide for the basic material. When there are flammable gases in the ambient environment, the conductivity of the sensor will increase as the gases become denser. This type can detect a wide range of gases thus making it a low cost multifunctional sensor. The sensor can be used for methane leak alarm and automatic smoke exhaust fan. With the features, it boasts a perfect sensor for indoor air regulation that meets the environmental standards.

Notes:

1. This sensor is equipped with an adjustment potentiometer for the alarm threshold. Spin the knob of the pot clockwise, and the threshold will be increased; spin it counterclockwise, the threshold will be reduced.
2. The sensor may not output a steady and accurate data immediately; it needs to be preheated for about 1 minute to collect data steadily.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * MQ-2 Gas Sensor Module
- 1 * USB Cable
- 1 * 4-Pin Wires

Experimental Principle

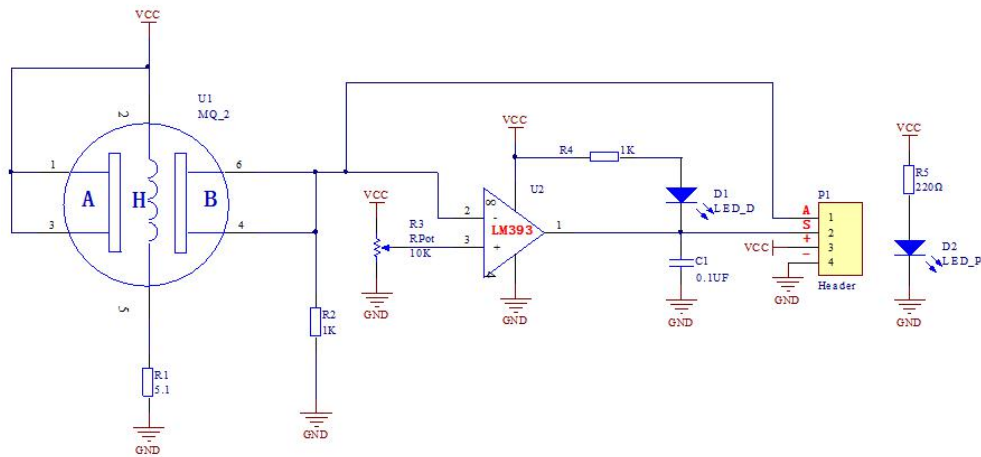
The Fritzing image:



Pin definition:

S	Digital output
A	Analog output
+	VCC
-	GND

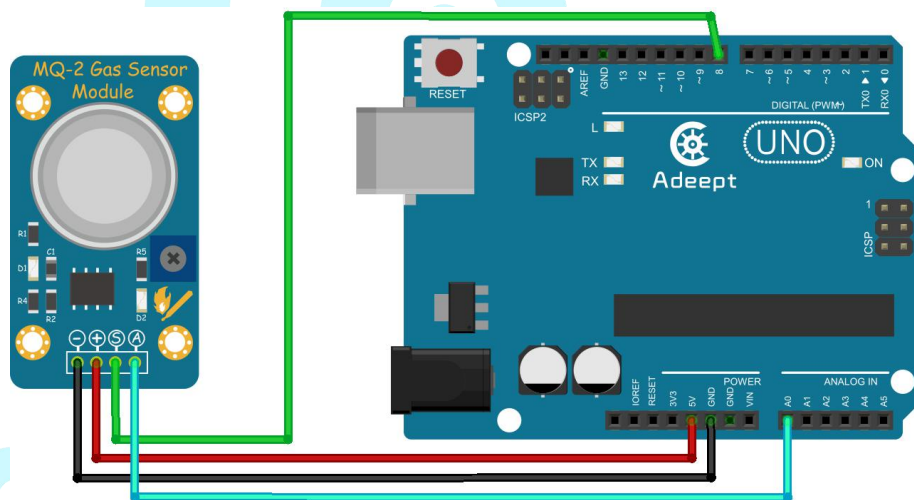
The schematic diagram:



In this experiment, by programming the Arduino, we read the analog and switch values collected by the MQ-2 Gas Sensor and display them on the Serial Monitor via serial port.

Experimental Procedures

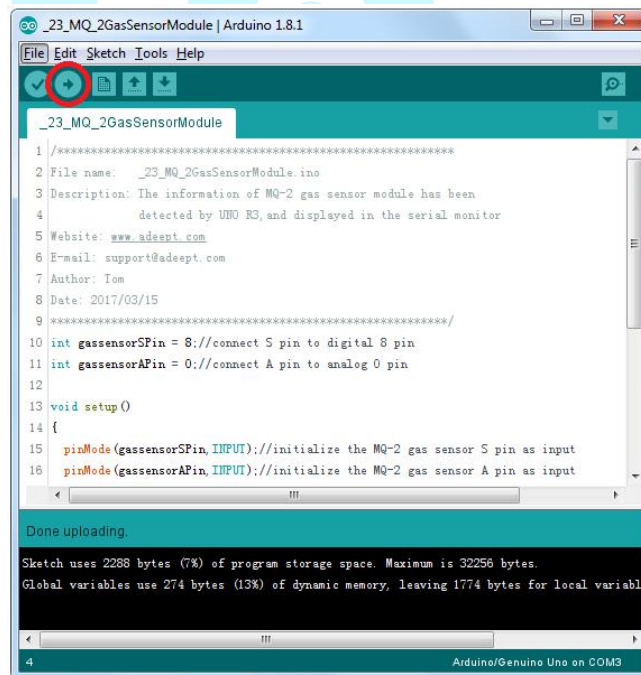
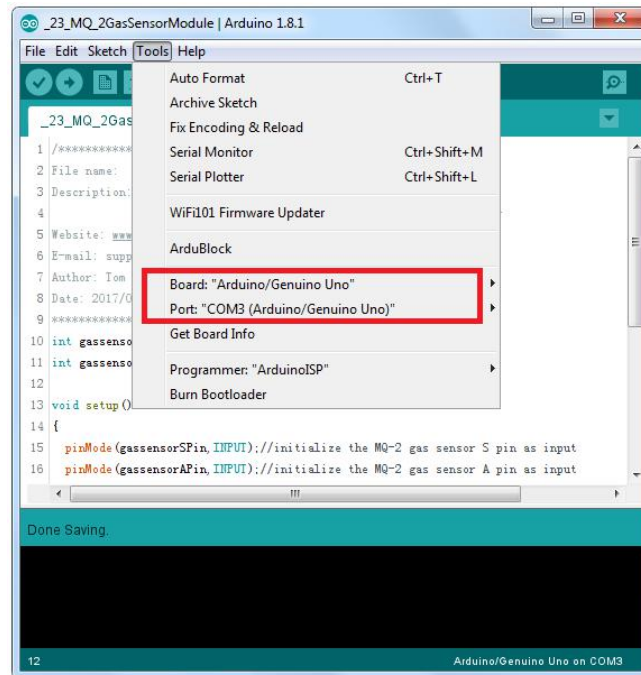
Step 1: Build the circuit



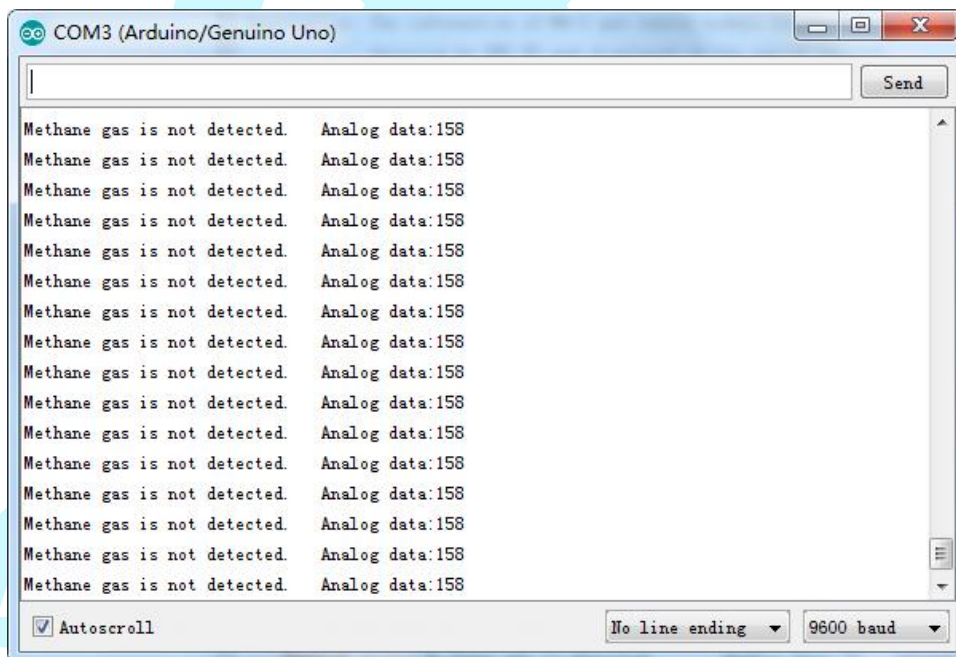
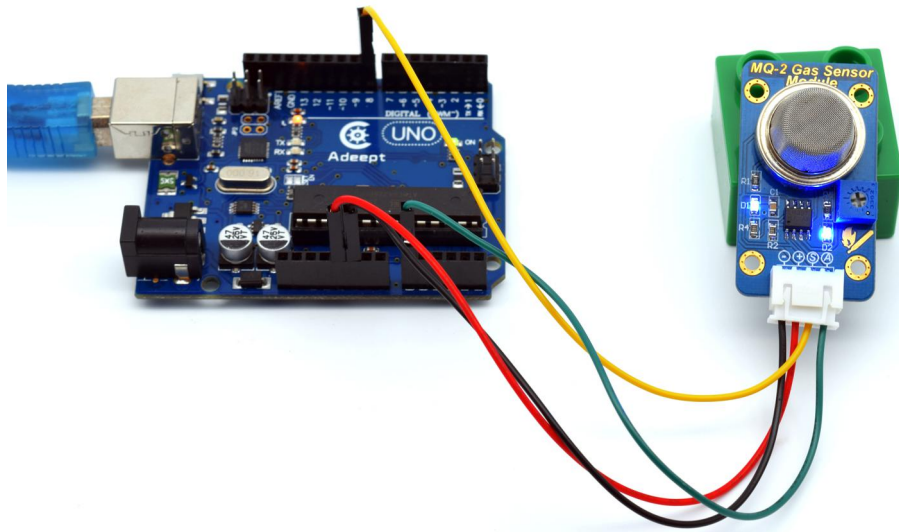
Adept UNO R3 Board	MQ-2 Gas Sensor Module
A0	A
D8	S
5V	+
GND	-

Step 2: Program _23_MQ_2GasSensorModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.



Open the Serial Monitor in Arduino IDE. Release some methane near the module. And you will see the corresponding message on the window indicating flammable gases. Also the value output by the analog pin of the module will be printed.



Lesson 24 Tracking Test

Introduction

The Line Finder Module applies the principle that infrared rays reflect differently on surfaces of different colors. After electrified, the infrared diode on the module sends out infrared rays constantly. When they encounter a white surface, the diffused reflection happens and the reflected rays are received by the receiver on the module. On the other hand, when they come across a black one, the receiver cannot get any infrared.

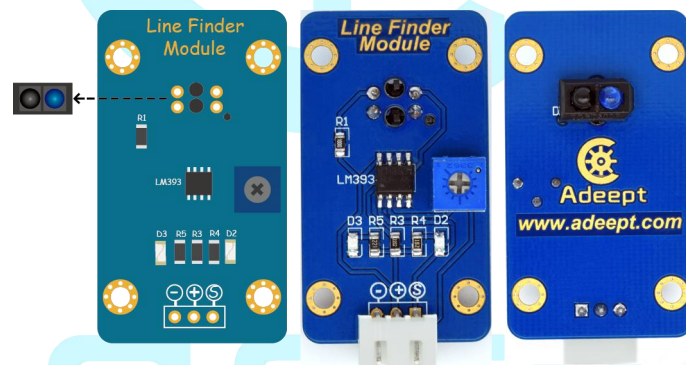
Thus, the processor can tell whether it is a white or black detected surface by receiving the reflected infrared rays or not. Based on this, the module is usually used in line finding on a smart car.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Line Finder Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

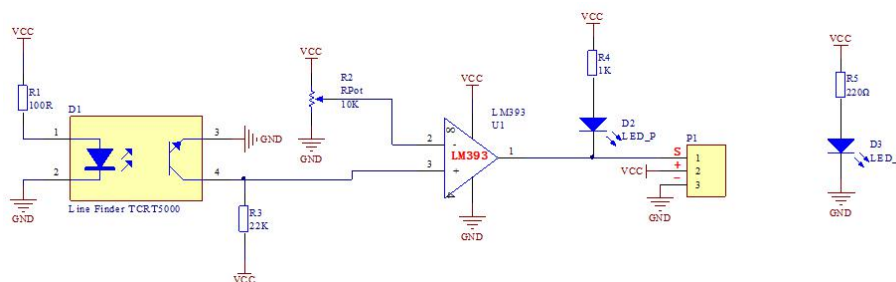
The Fritzing image:



Pin definition:

S	Digital output
+	VCC
-	GND

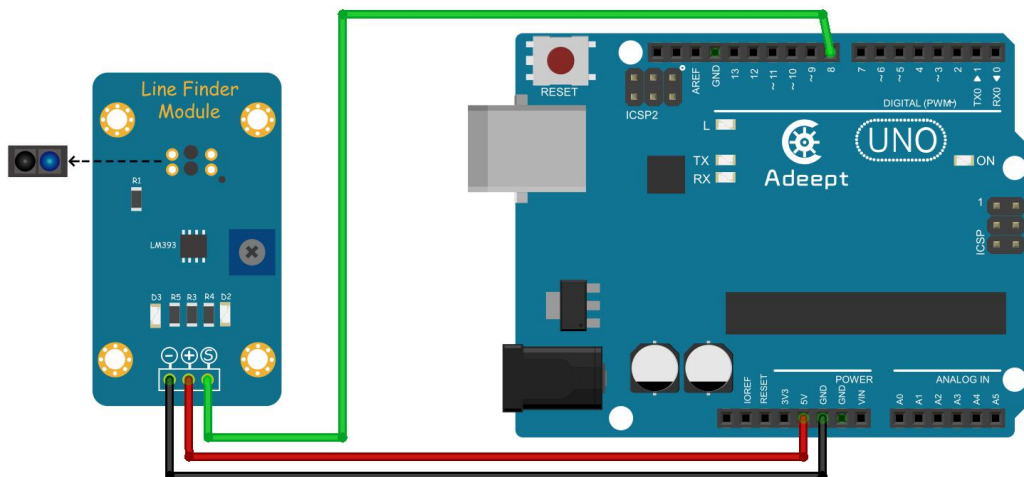
The schematic diagram:



In this experiment, we detect a piece of white and another of black paper via the Line Finder Module.

Experimental Procedures

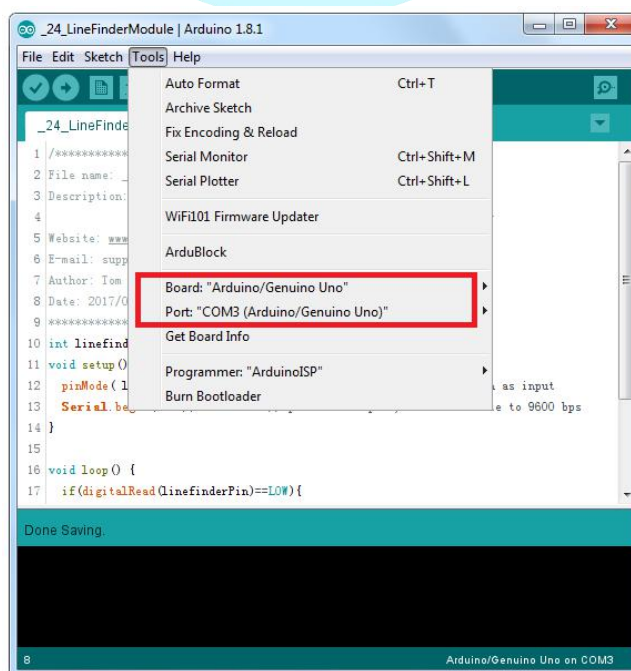
Step 1: Build the circuit

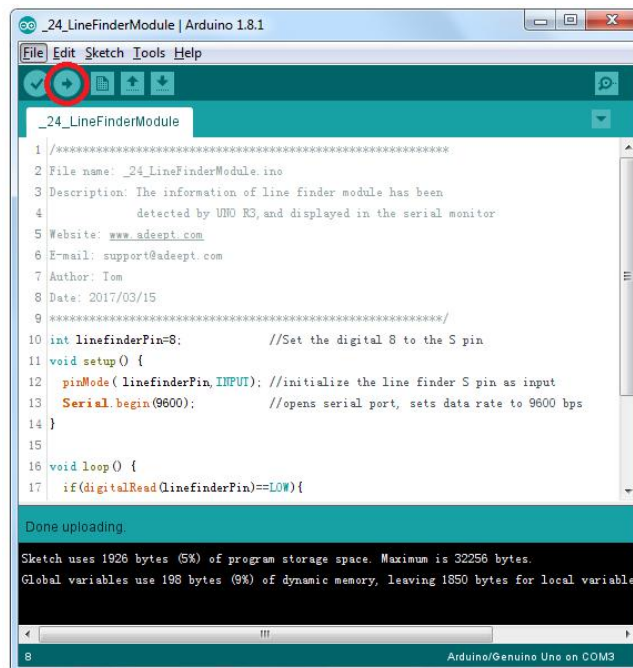


Adept UNO R3 Board	Line Finder Module
D8	S
5V	+
GND	-

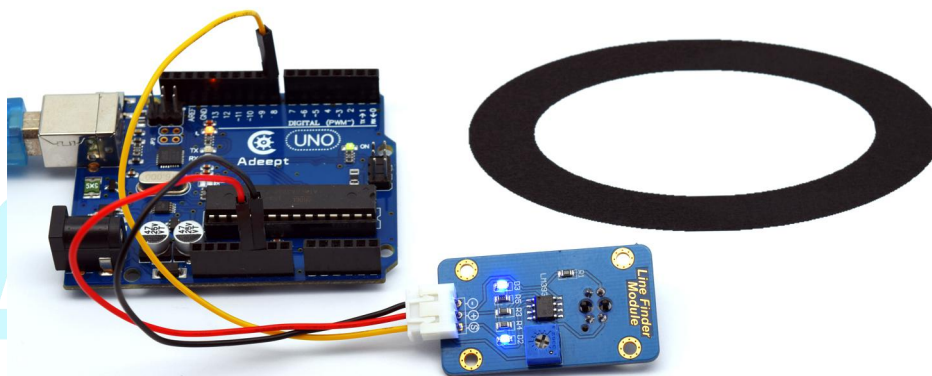
Step 2: Program _24_LineFinderModule.ino

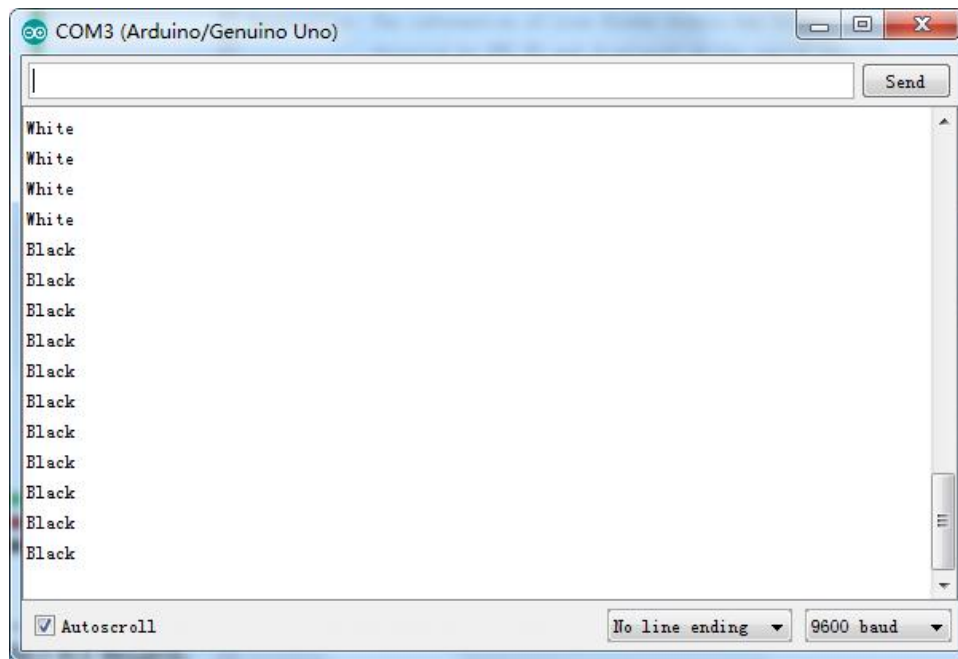
Step 3: Compile and download the sketch to the UNO R3 board.





Open the Serial Monitor in Arduino IDE. Place the sensor module over a piece of white paper and another of black and you will see the data detected on the window. You can adjust the blue potentiometer on the module to change the sensitivity.





Adeept

Lesson 25 How To Use Slide Potentiometer

Introduction

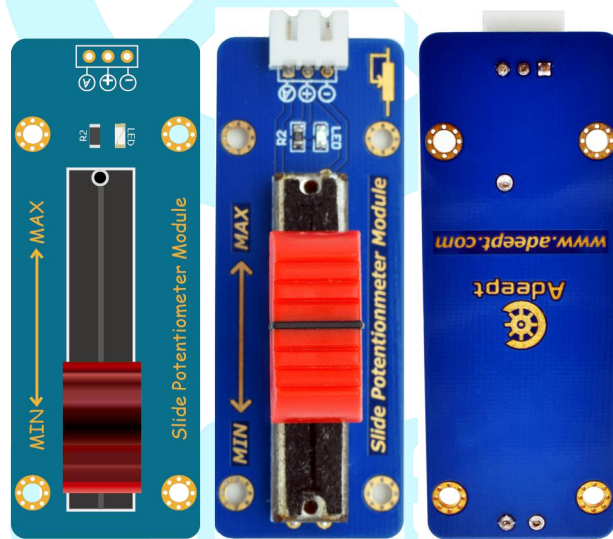
The similarity of the Slide Potentiometer Module and Potentiometer Module lies in: holding three terminals; changing the resistance between the changeable terminal and one end by changing the position of the slider. When the difference is: the Slide Potentiometer usually has a larger power (and size) 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 * Adept Arduino UNO R3 Board
- 1 * Slide Potentiometer Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Experimental Principle

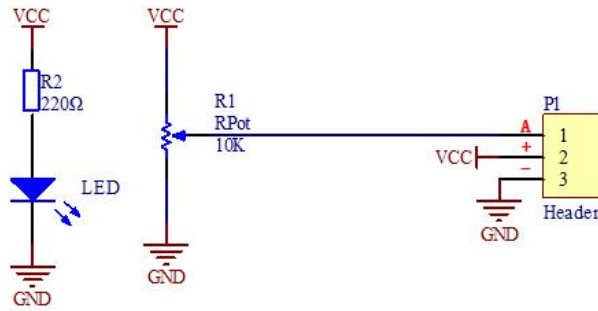
The Fritzing image:



Pin definition:

A	Analog output
+	VCC
-	GND

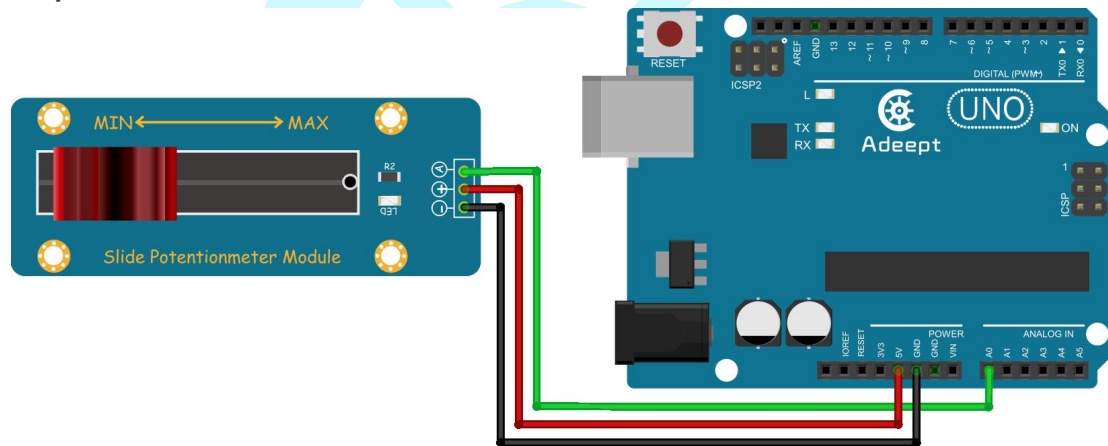
The schematic diagrams:



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

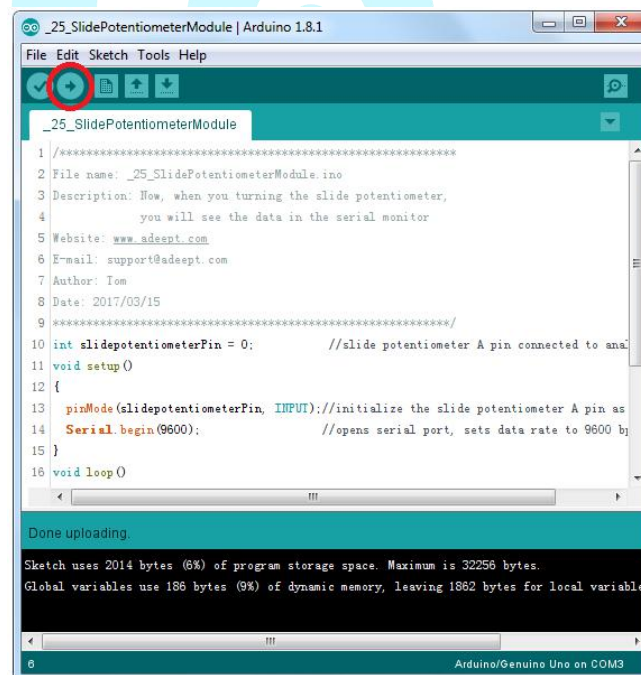
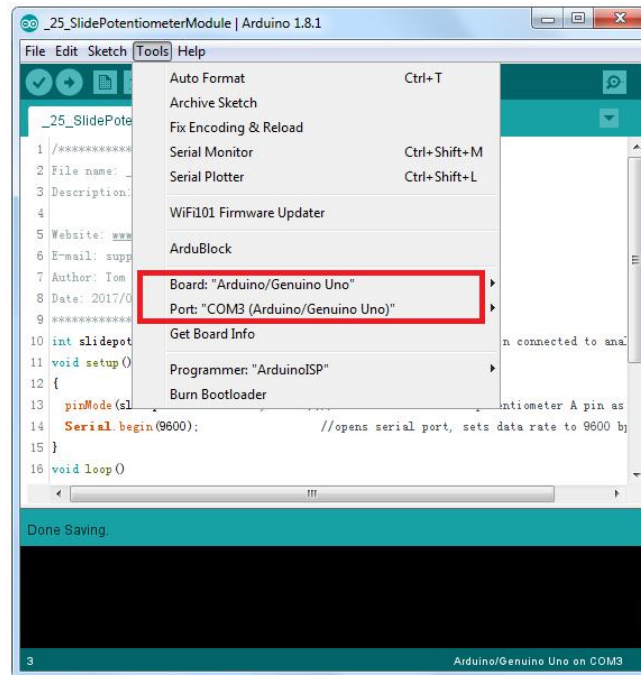
Step 1: Build the circuit



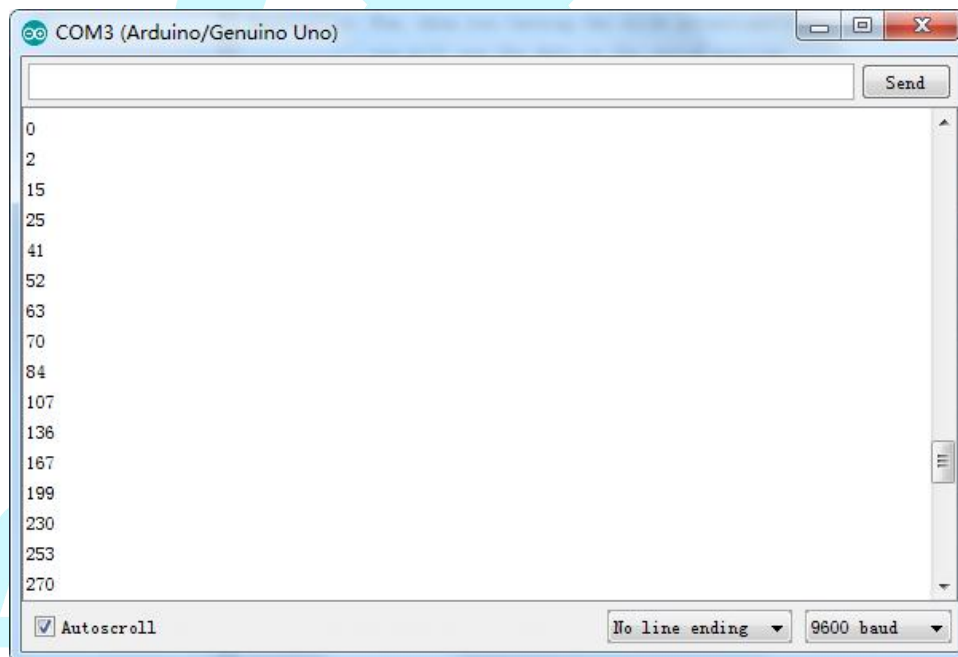
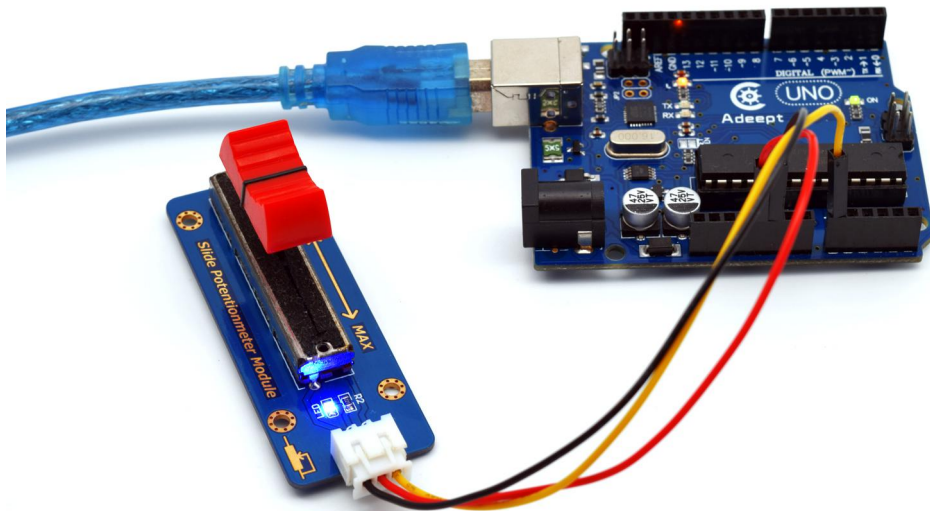
Adept UNO R3 Board	Slide Potentiometer Module
A0	A
5V	+
GND	-

Step 2: Program _25_SlidePotentiometerModule.ino

Step 3: Compile and download the sketch to the UNO R3 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.



Lesson 26 Small Fan Works

Introduction

DC motor is a device that converts electrical energy into mechanical energy. Due to the ease of control, it is usually used in fan, electronic toy, shaver, etc.

Components

- 1 * Aadept Arduino UNO R3 Board
- 1 * DC Motor Module
- 1 * USB Cable
- 1 * 4-Pin Wires

Experimental Principle

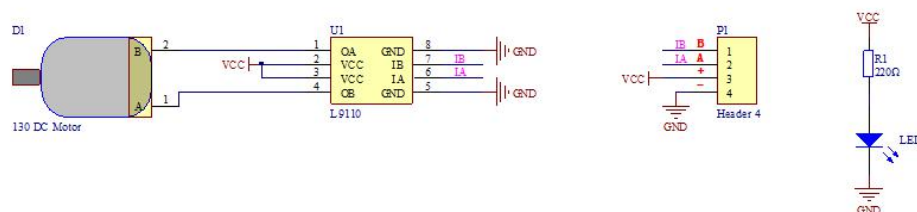
The Fritzing image:



Pin definition:

B	Digital output
A	Digital output
+	VCC
-	GND

The schematic diagram:



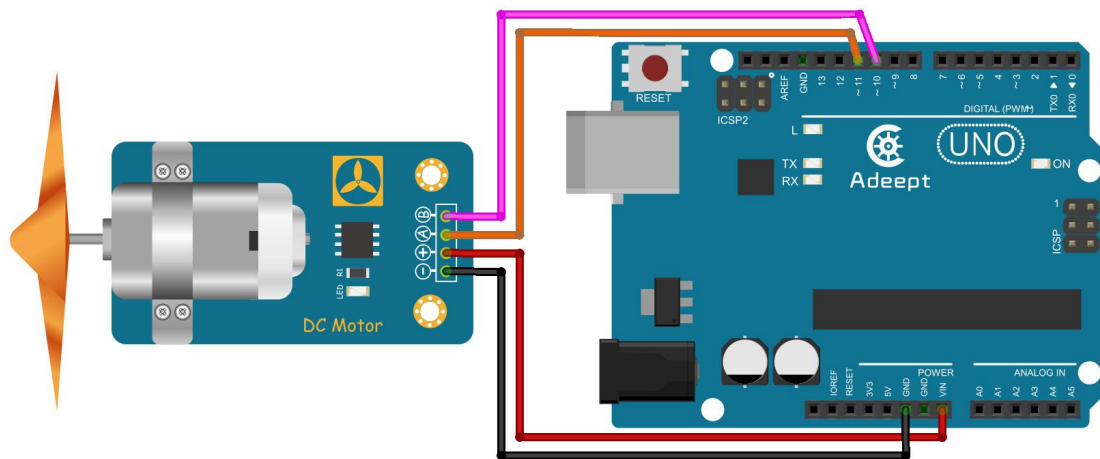
This experiment is to control the status of the DC motor via the Arduino board. The statuses include running, stop, forward, reversing, acceleration, and deceleration.

note:

We need to connect the power first. After powering on the module, we can connect the module signal pins. Otherwise the motor module will be hot and may not work.

Experimental Procedures

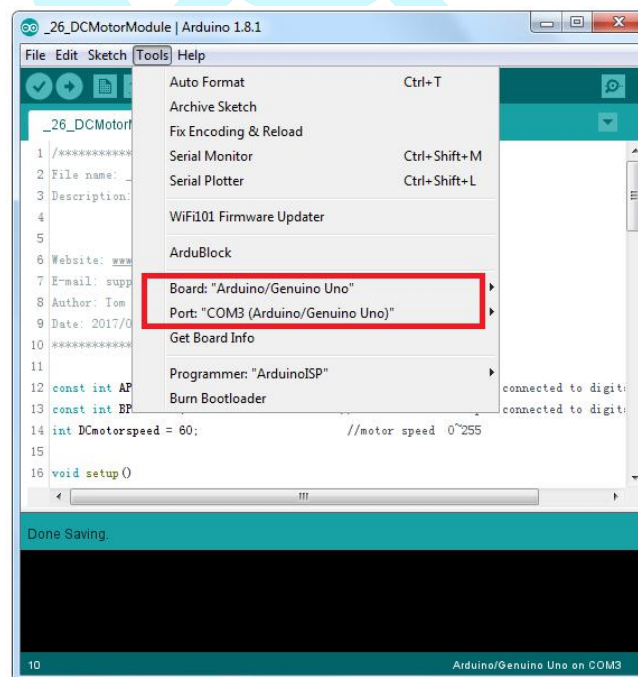
Step 1: Build the circuit

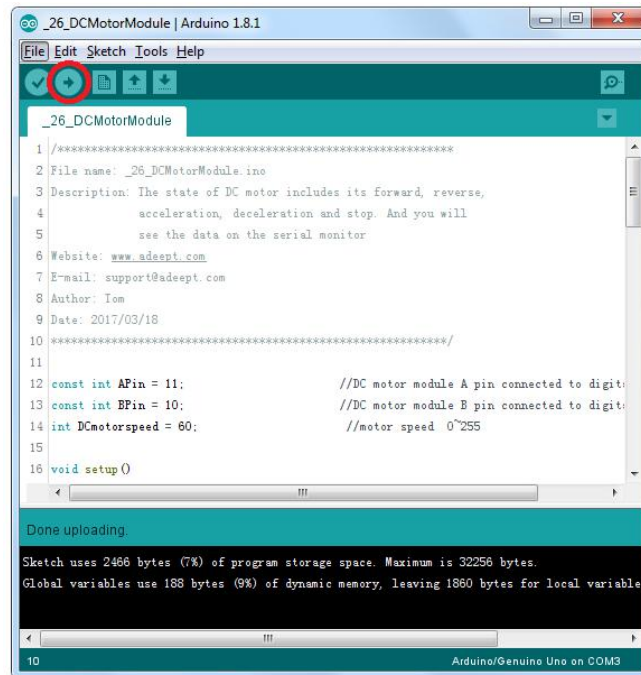


Adept UNO R3 Board	DC Motor Module
D10	B
D11	A
Vin(+5V)	+
GND	-

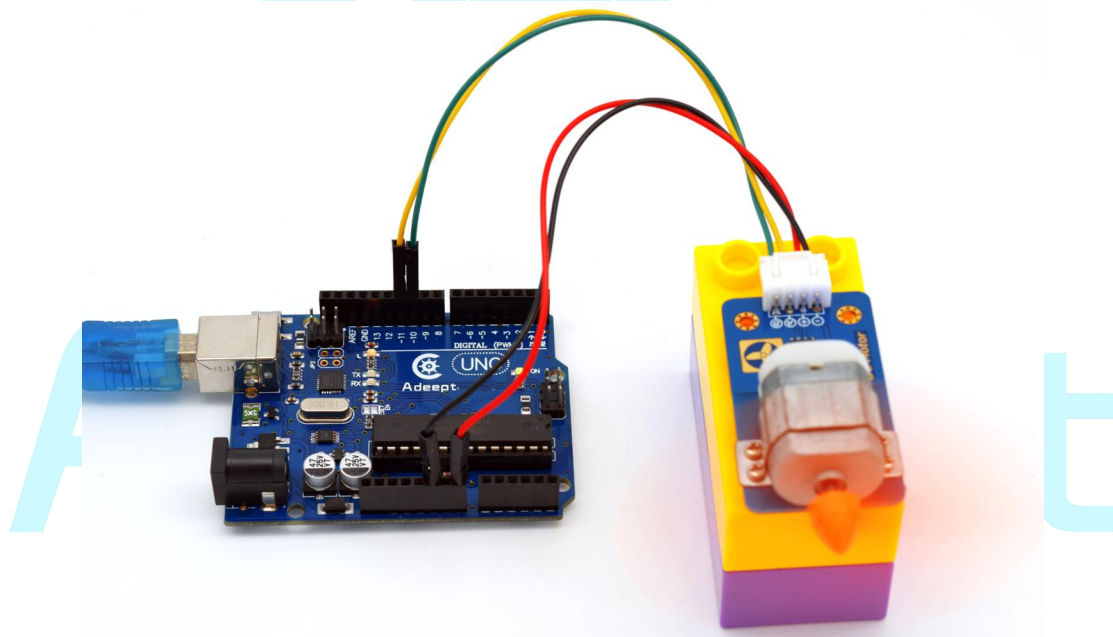
Step 2: Program _26_DCMotorModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.





Now you can see the fan rotates forward, accelerate, speed down, stop, start and accelerate, speed down again, and stop again. At the same time the value of the speed set will be sent to and displayed on Serial Monitor.



Lesson 27 How To Use The Joystick

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 * Adept Arduino UNO R3 Board
- 1 * Joystick Module
- 1 * USB Cable
- 1 * 5-Pin Wires

Experimental Principle

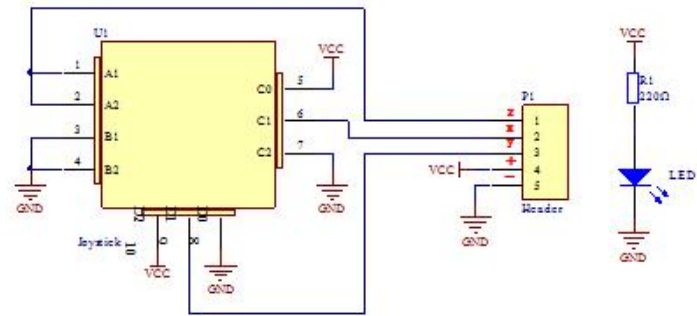
The Fritzing image:



Pin definition:

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

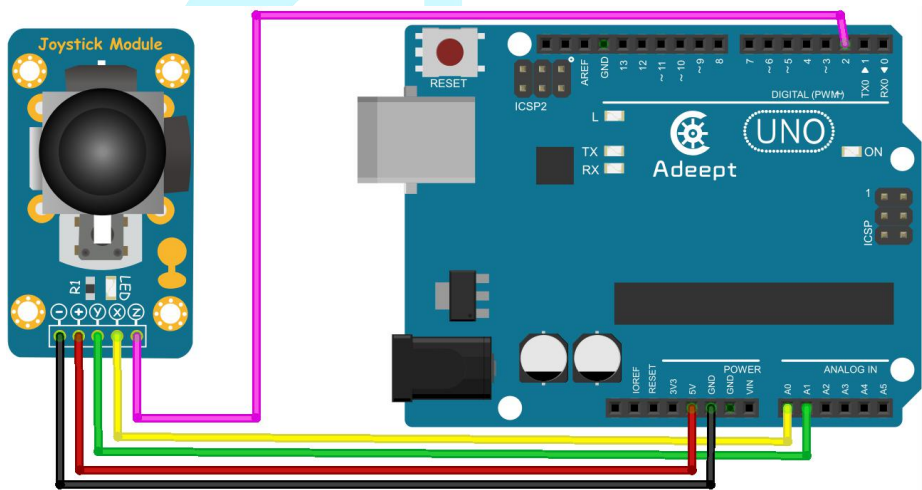
The schematic diagram:



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

Experimental Procedures

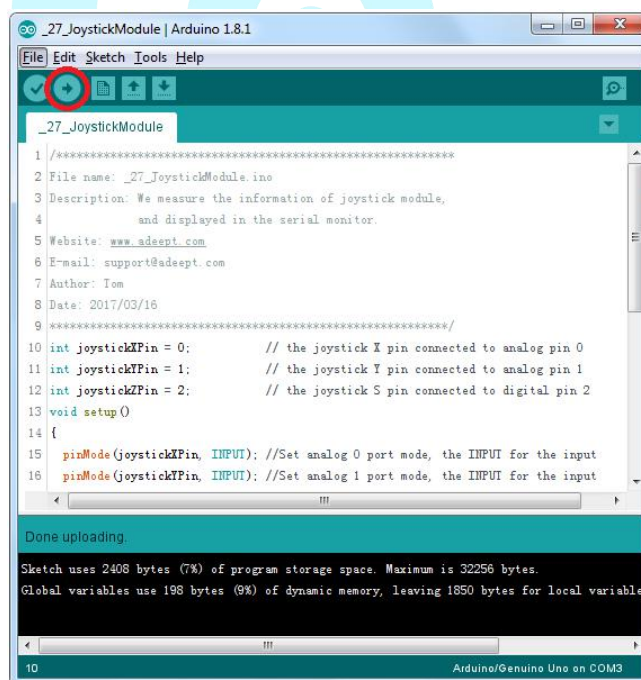
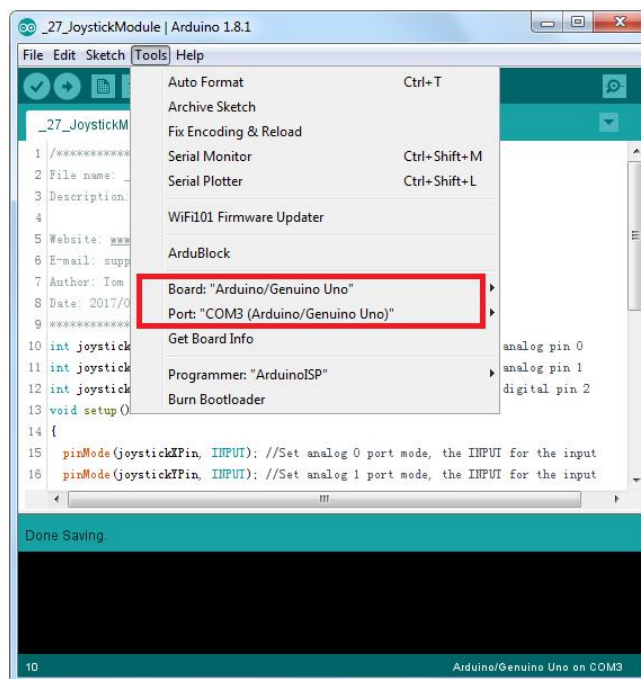
Step 1: Build the circuit



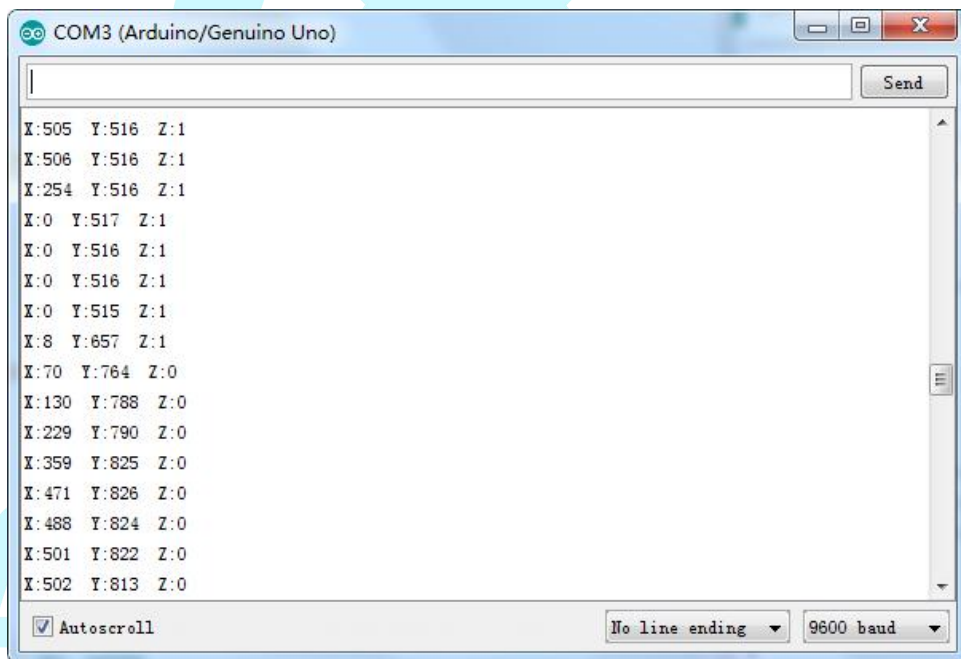
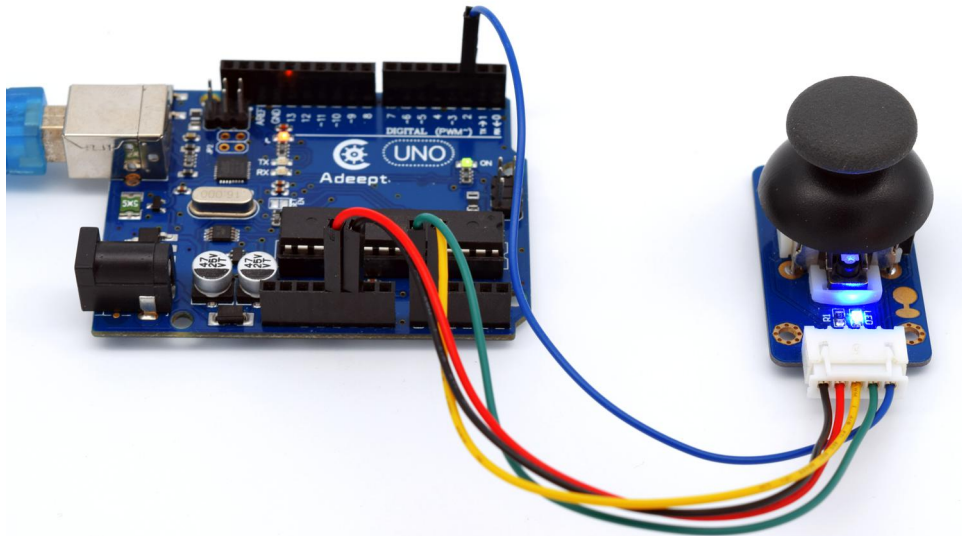
Adept UNO R3 Board	Joystick Module
D2	z
A0	x
A1	y
5V	+
GND	-

Step 2: Program `_27_JoystickModule.ino`

Step 3: Compile and download the sketch to the UNO R3 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.



Lesson 28 How To Use The MIC Module

Introduction

The MIC Module is composed of a small microphone and an LM393 voltage comparator. It can capture minor sound signals and convert them into electric ones. The threshold of the comparator can be adjusted by the blue potentiometer on the module. This module can be applied in sound alarm system.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * MIC Module
- 1 * USB Cable
- 1 * 4-Pin Wires

Experimental Principle

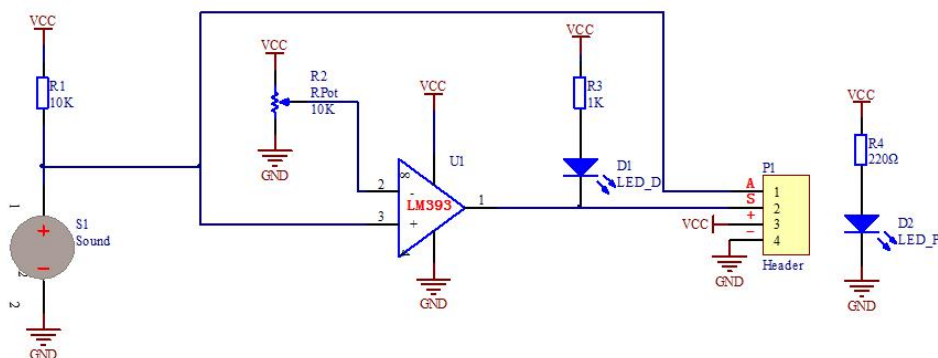
The Fritzing image:



Pin definition:

S	Digital output
A	Analog output
+	VCC
-	GND

The schematic diagram:

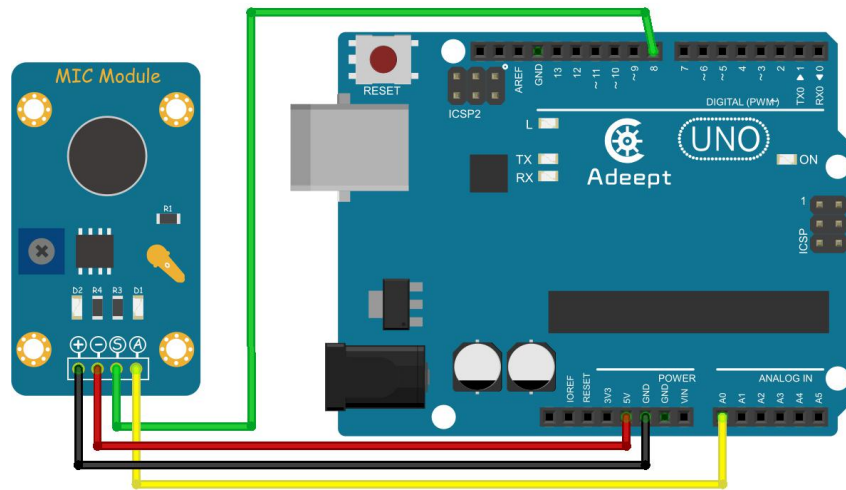


This experiment uses the MIC Module to detect the sound and display the data on Serial

Monitor via the serial port.

Experimental Procedures

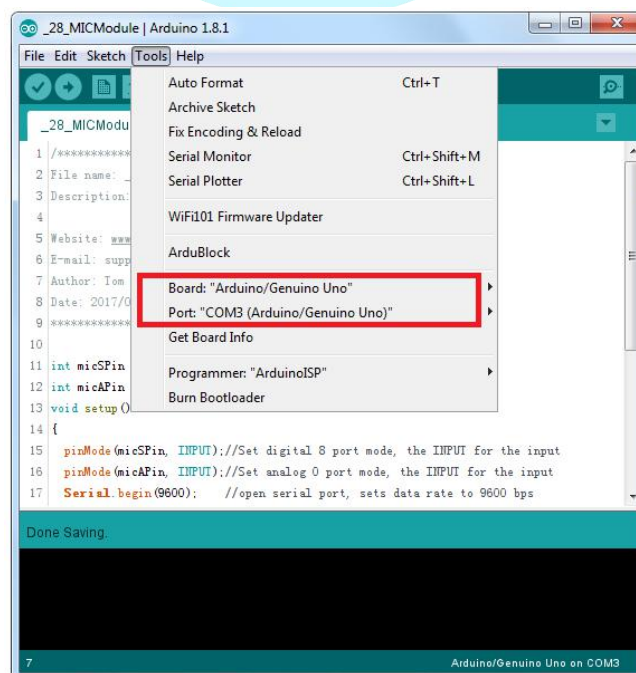
Step 1: Build the circuit

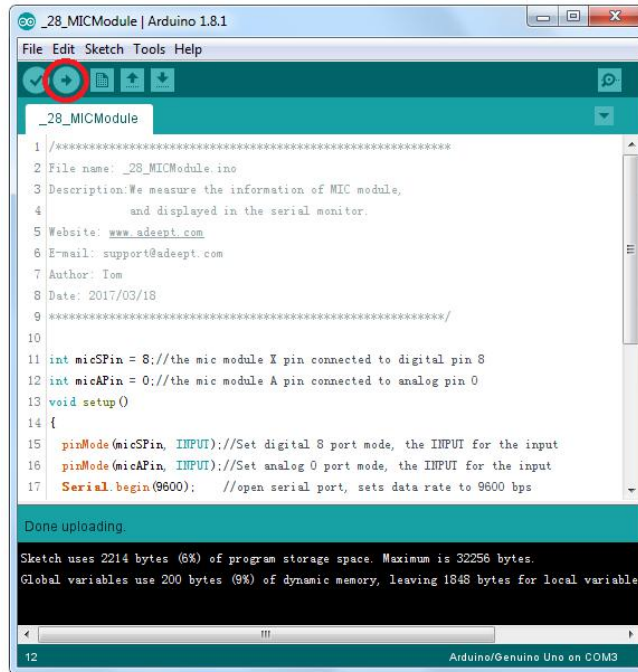


Adept UNO R3 Board	MIC Module
D8	S
A0	A
5V	+
GND	-

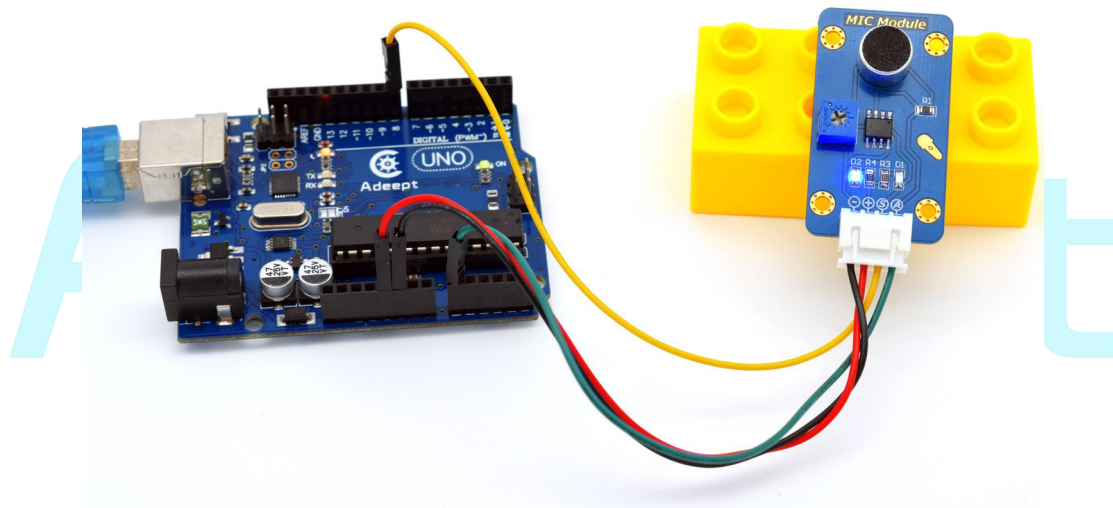
Step 2: Program _28_MICModule.ino

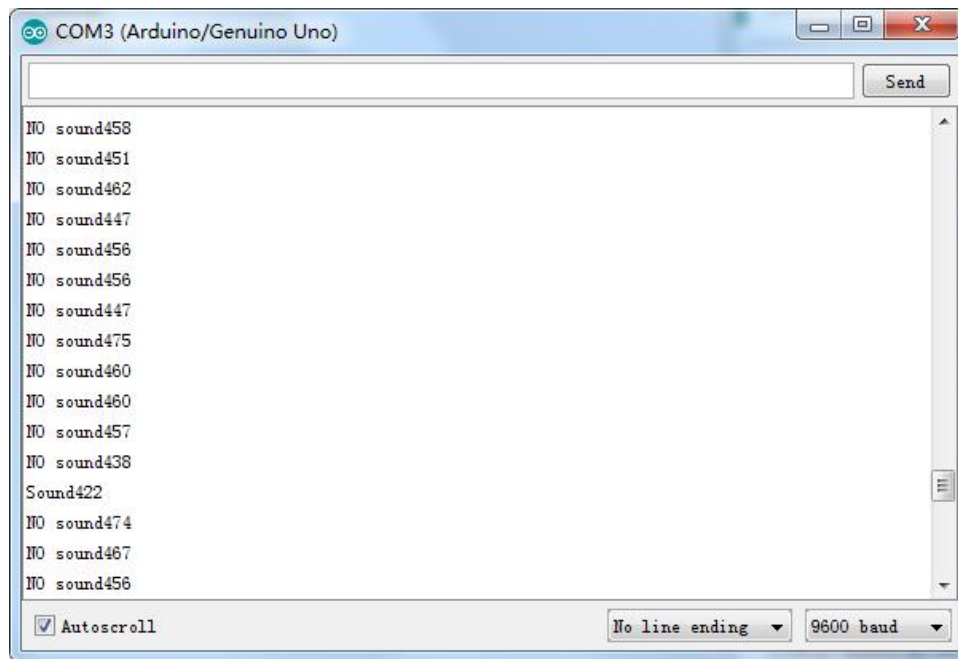
Step 3: Compile and download the sketch to the UNO R3 board.





Open Serial Monitor of the Arduino IDE. Blow at the MIC Module or make some other sounds near it, and you can see the value on the window indicating the sound intensity. The higher the sound volume, the larger value on the window; the lower volume, the smaller value.





Adeept

Lesson 29 How To Use The Relay Module

Introduction

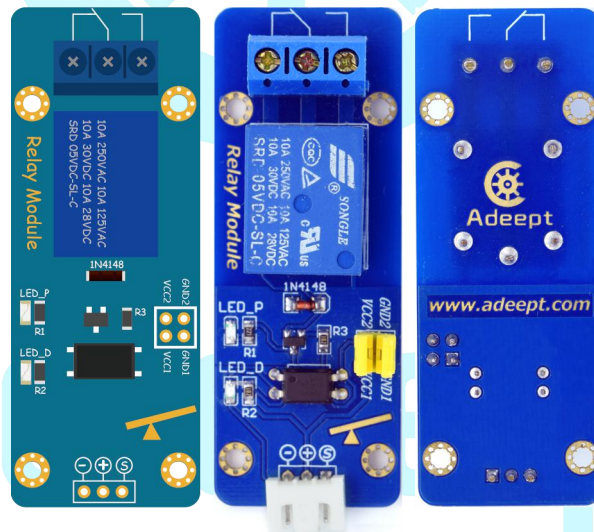
The relay is an electronic and electrical component that controls large currents by small currents. In the course of building an Arduino project, generally many large current or high volume devices like solenoid valve, lamp and motor cannot be connected directly to digital I/Os of the Arduino board.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

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Relay Module
- 1 * LED Module
- 1 * USB Cable
- 2 * 3-Pin Wires
- 3 * Hookup Wire Set
- 1 * Breadboard

Experimental Principle

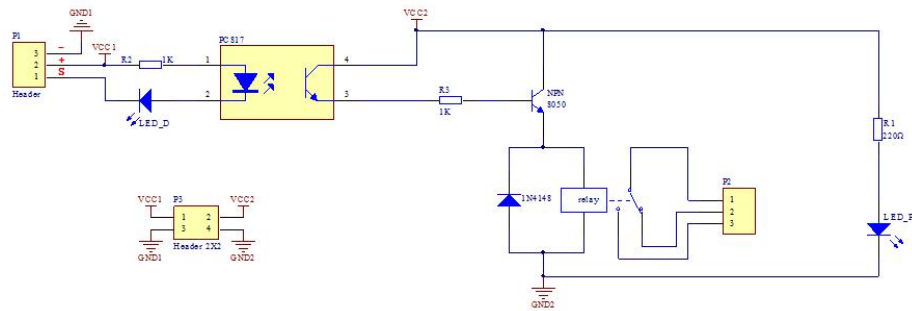
The Fritzing image:



Pin definition:

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

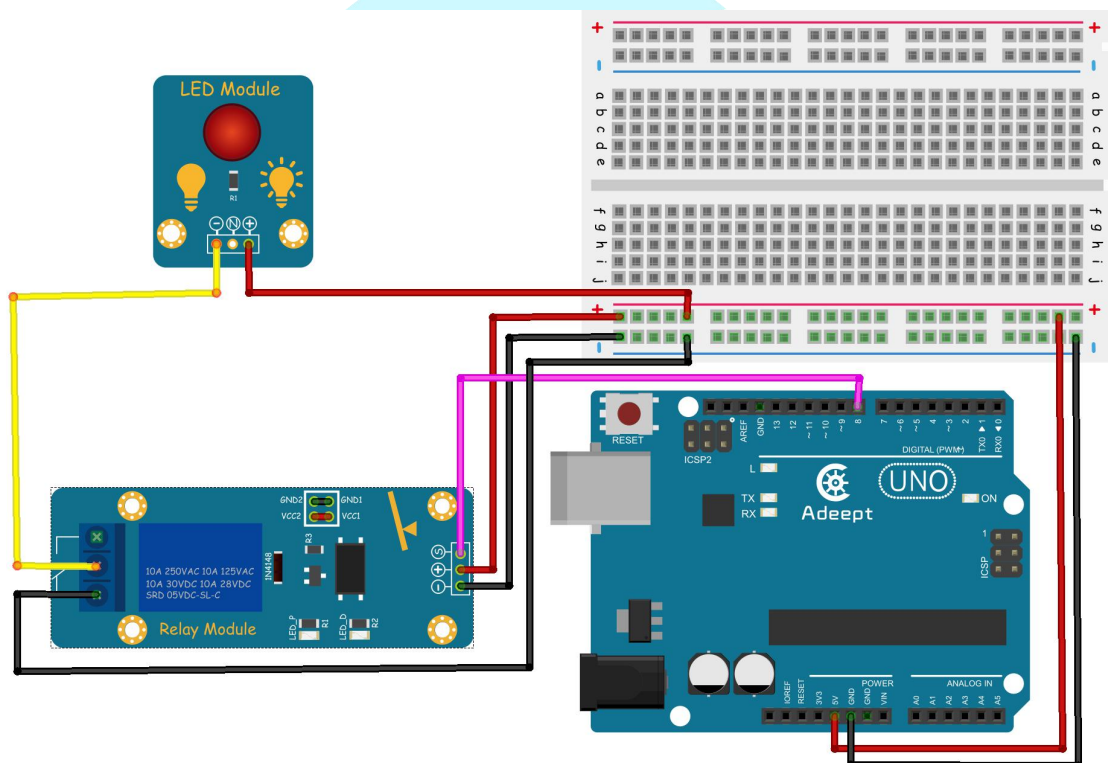
The schematic diagram:



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

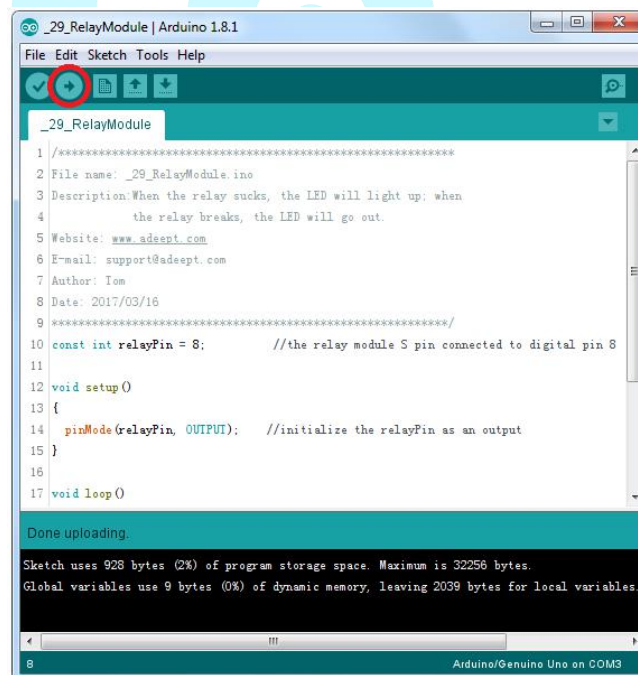
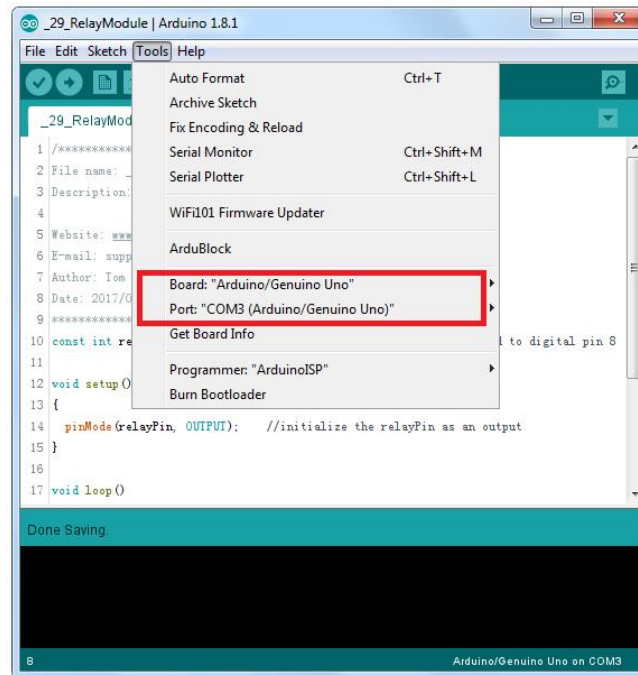
Experimental Procedures

Step 1: Build the circuit

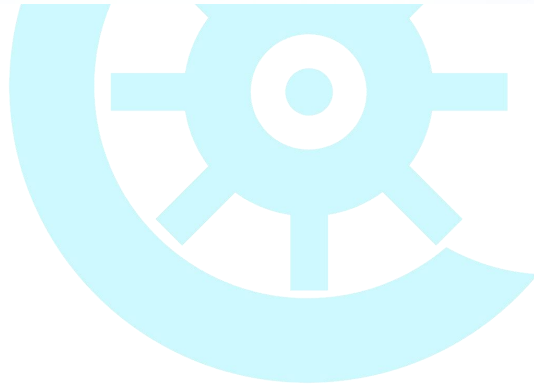
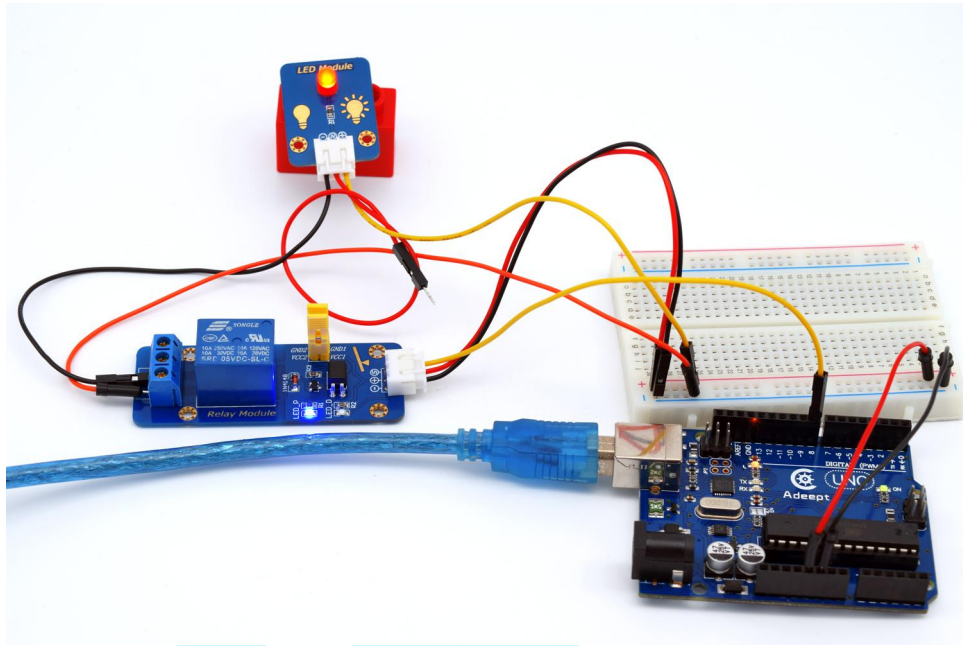


Step 2: Program _29_RelayModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.



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



Adept

Lesson 30 How To Use The Segment Module

Introduction

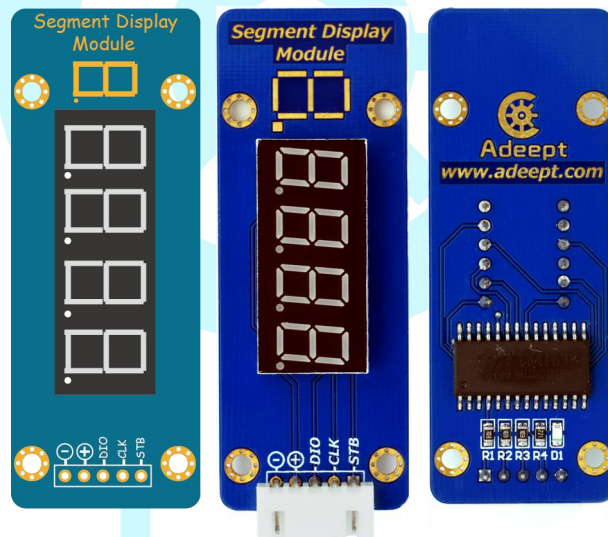
The module consists of a 4-bit 7-segment common-cathode (CC) diode and a control chip TM1638. It communicates with the Arduino board via three wires and can show numbers and simple characters.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * Segment Display Module
- 1 * USB Cable
- 1 * 5-Pin Wires

Experimental Principle

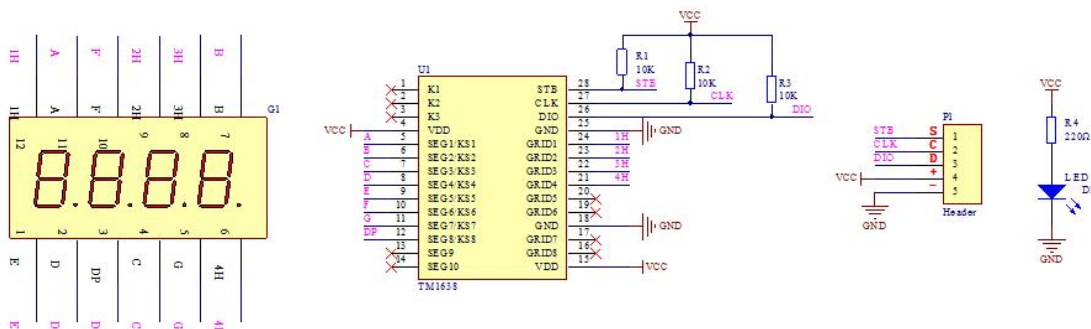
The Fritzing image:



Pin definition:

STB	Digital input
CLK	Digital input
DIO	Digital input
+	VCC
-	GND

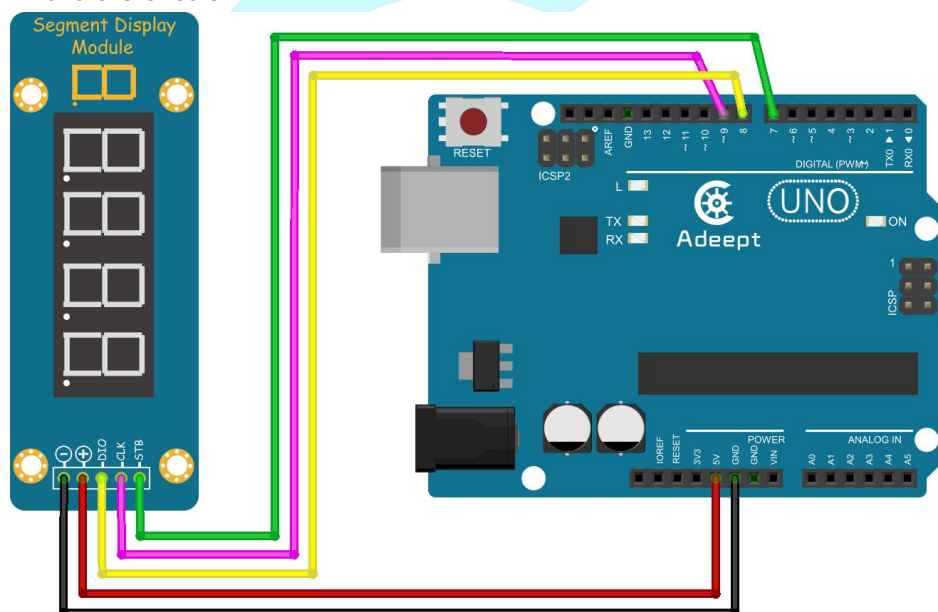
The schematic diagram:



Through programming the Arduino board, make the module display 0000~9999.

Experimental Procedures

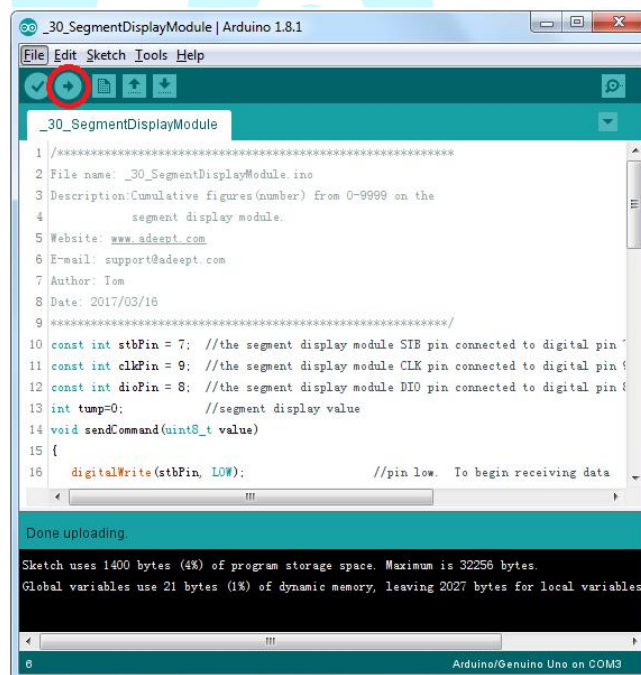
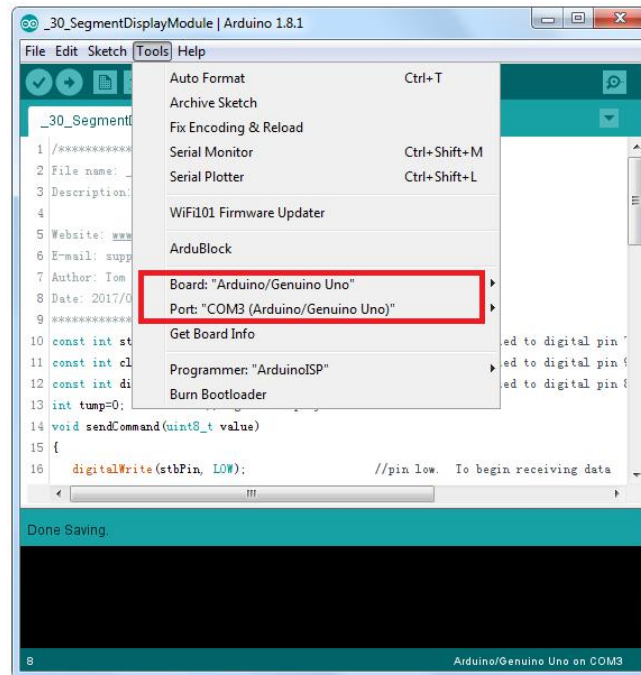
Step 1: Build the circuit



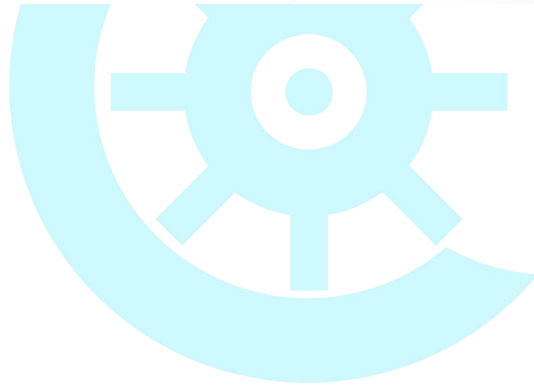
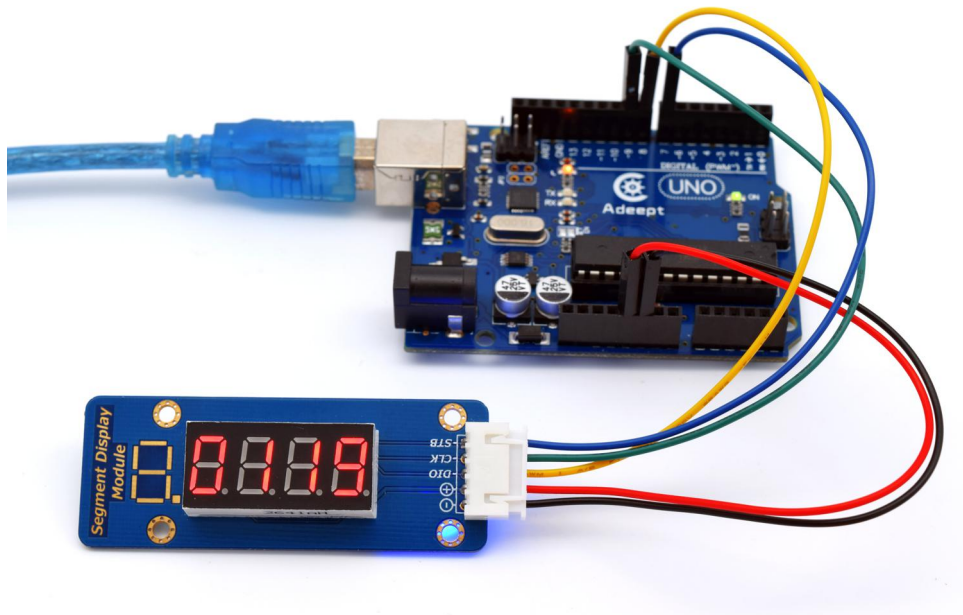
Adeept UNO R3 Board	Segment Display Module
D7	STB
D9	CLK
D8	DIO
5V	+
GND	-

Step 2: Program _30_SegmentDisplayModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.



Now you can see the number 0~9999 shown repeatedly on the digital display.



Adeept

Lesson 31 How To Use The 8*8 LED Matrix

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 * Adept Arduino UNO R3 Board
- 1 * 8*8 LED Matrix Module
- 1 * USB Cable
- 1 * 5-Pin Wires

Experimental Principle

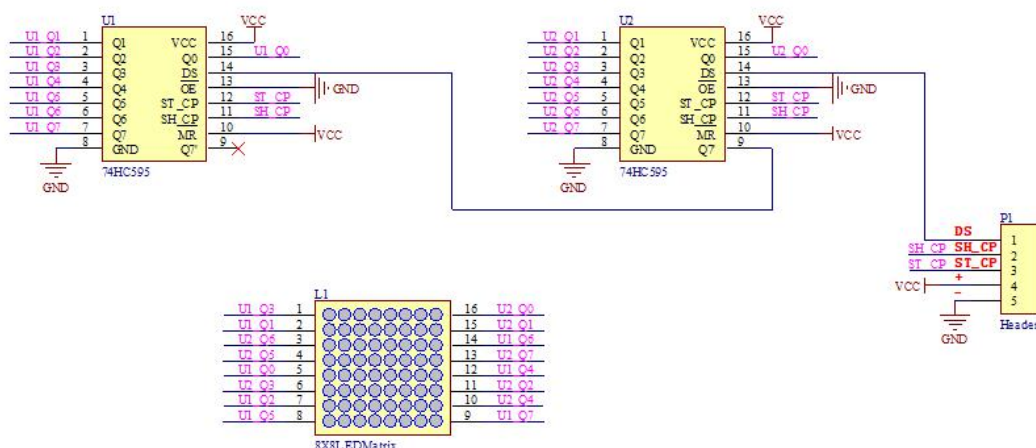
The Fritzing image:



Pin definition:

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

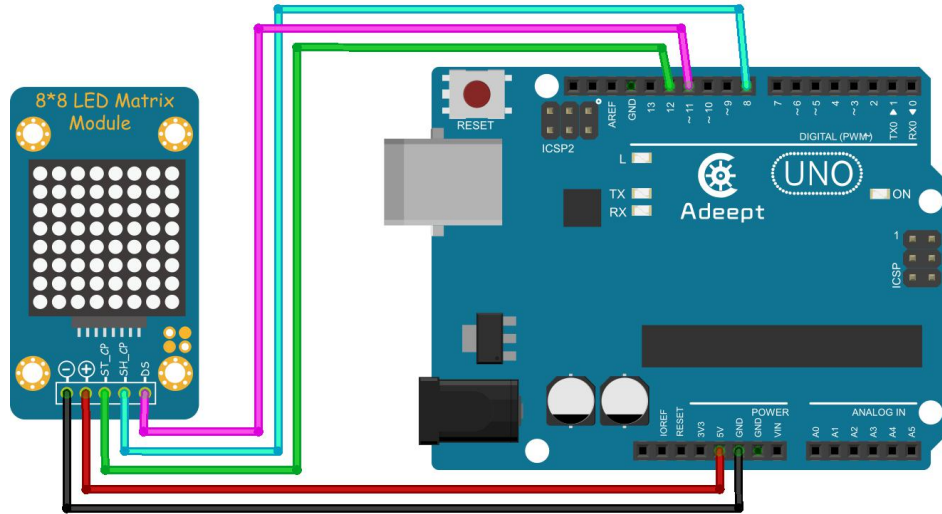
The schematic diagram:



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 “Adept”.

Experimental Procedures

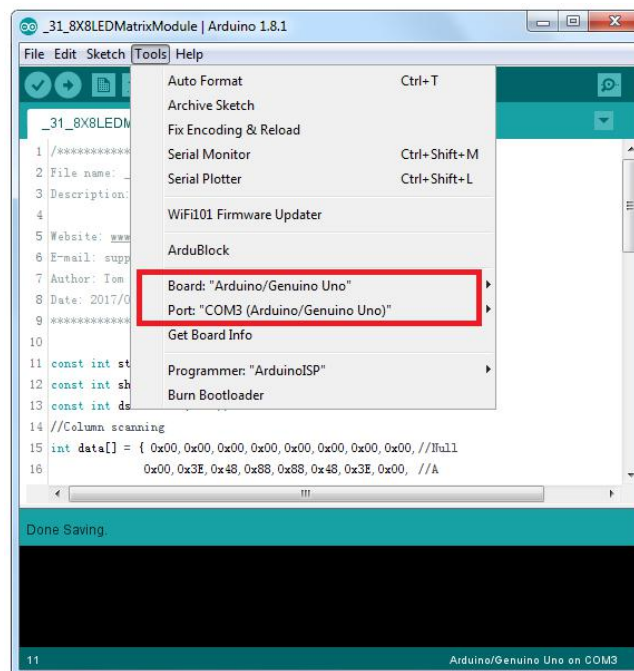
Step 1: Build the circuit

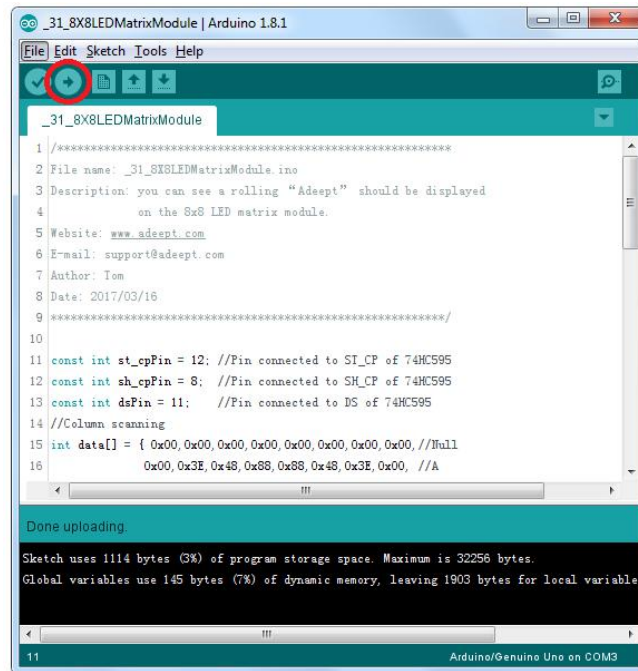


Adept UNO R3 Board	8*8 LED Matrix Module
D11	DS
D8	SH_CP
D12	ST_CP
5V	+
GND	-

Step 2: Program _31_8X8LEDMatrixModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.





The screenshot shows the Arduino IDE interface with the sketch "_31_8X8LEDMatrixModule" loaded. The code includes file metadata and pin definitions for an 8x8 LED matrix module. The status bar at the bottom indicates "Done uploading." and "Sketch uses 1114 bytes (3%) of program storage space. Maximum is 32256 bytes. Global variables use 145 bytes (7%) of dynamic memory, leaving 1903 bytes for local variables."

```
1 /*****
2 File name: _31_8X8LEDMatrixModule.ino
3 Description: you can see a rolling "Adeept" should be displayed
4             on the 8x8 LED matrix module.
5 Website: www.adeept.com
6 E-mail: support@adeept.com
7 Author: Tom
8 Date: 2017/03/16
9 *****/
10
11 const int st_cpPin = 12; //Pin connected to ST_CP of 74HC595
12 const int sh_cpPin = 8;  //Pin connected to SH_CP of 74HC595
13 const int dsPin = 11;    //Pin connected to DS of 74HC595
14 //Column scanning
15 int data[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //full
16               0x00, 0x3E, 0x48, 0x88, 0x88, 0x48, 0x3E, 0x00, //A
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2
```

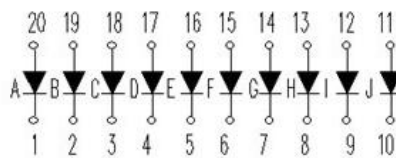
Lesson 32 Indication of Signal

Introduction

The LED bar is an analog indicating component usually used for volume indication.



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

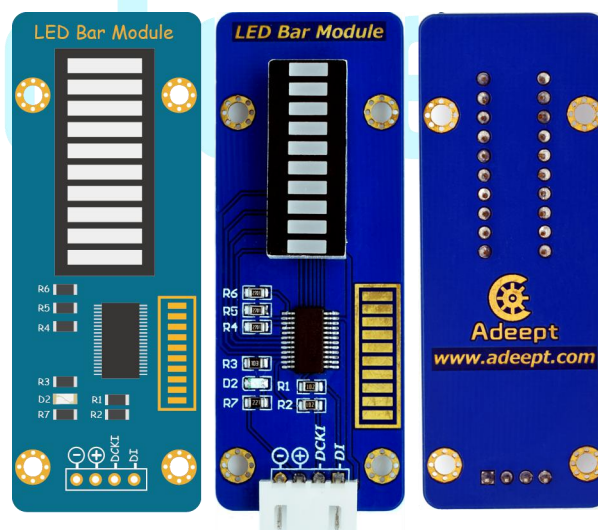


Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * LED Bar Module
- 1 * USB Cable
- 1 * 4-Pin Wires

Experimental Principle

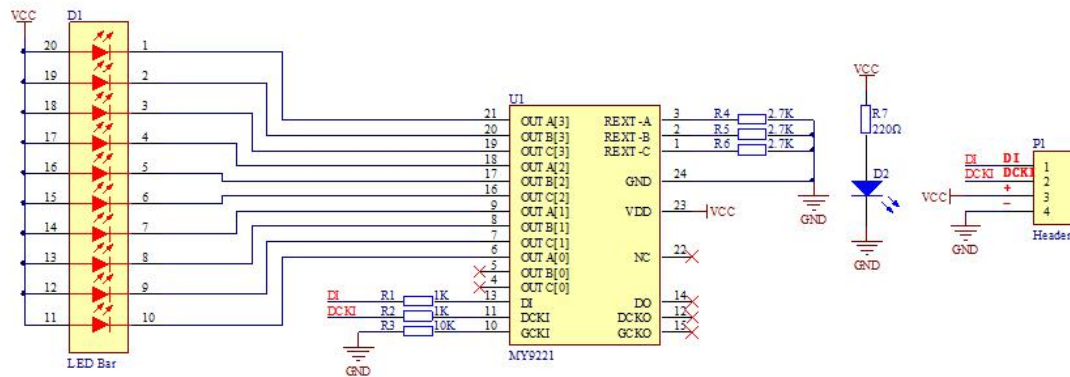
The Fritzing image:



Pin definition:

DI	Digital input
DCLK	Digital input
+	VCC
-	GND

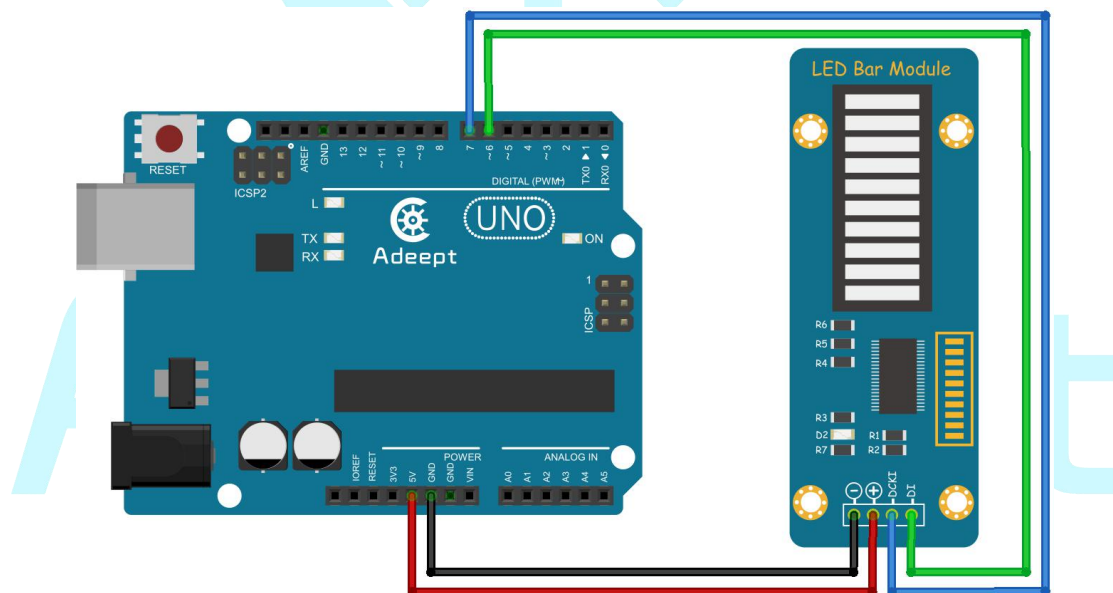
The schematic diagram:



The experiment is to control the number of LEDs brightened on the LED bar graph by programming the Arduino.

Experimental Procedures

Step 1: Build the circuit

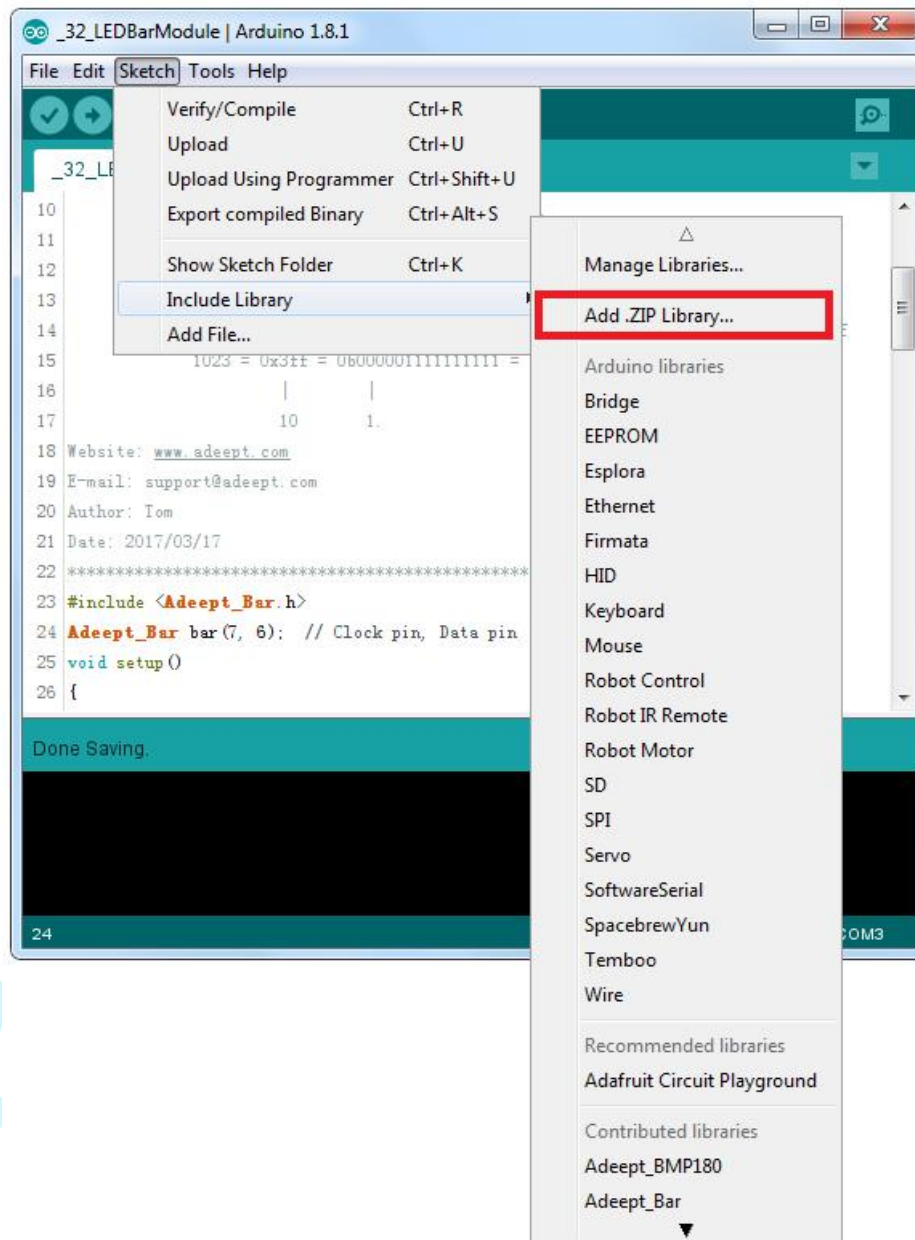


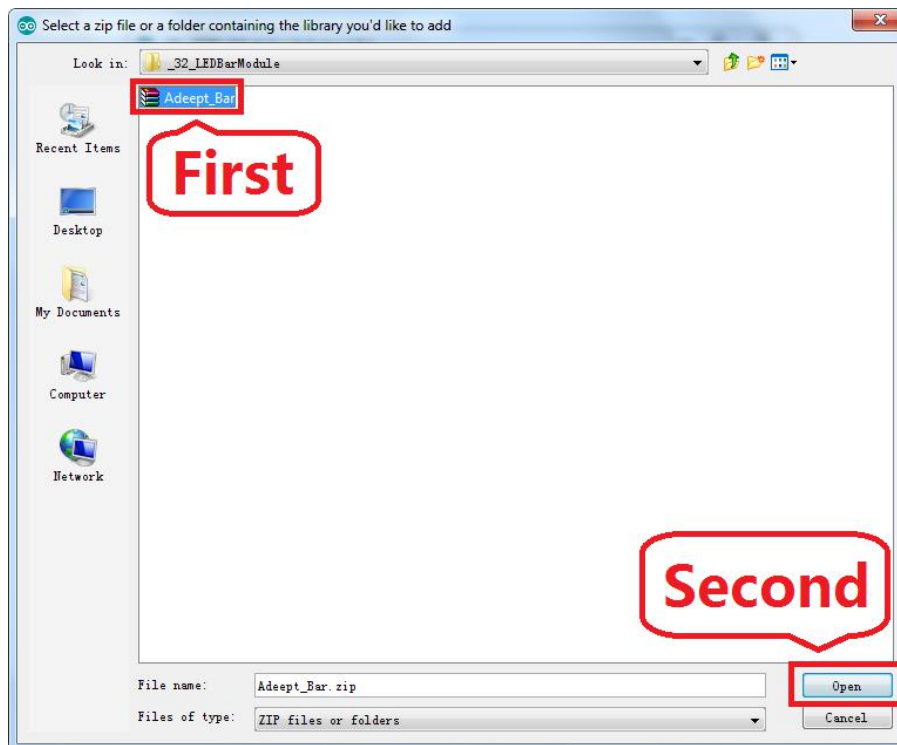
Adept UNO R3 Board	LED Bar Module
D7	DCLK
D6	DI
5V	+
GND	-

Step 2: Install the function library (Adept_Bar.zip).



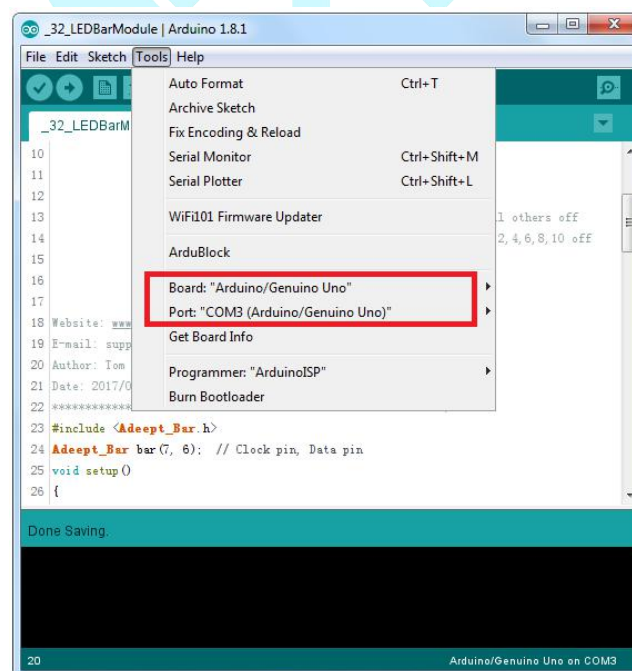
Adept_Bar

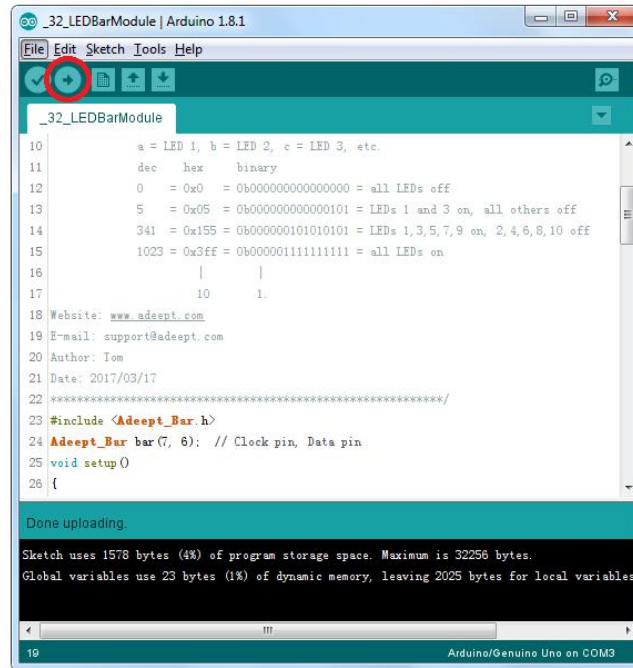




Step 3: Program _32_LEDBarModule.ino

Step 4: Compile and download the sketch to the UNO R3 board.





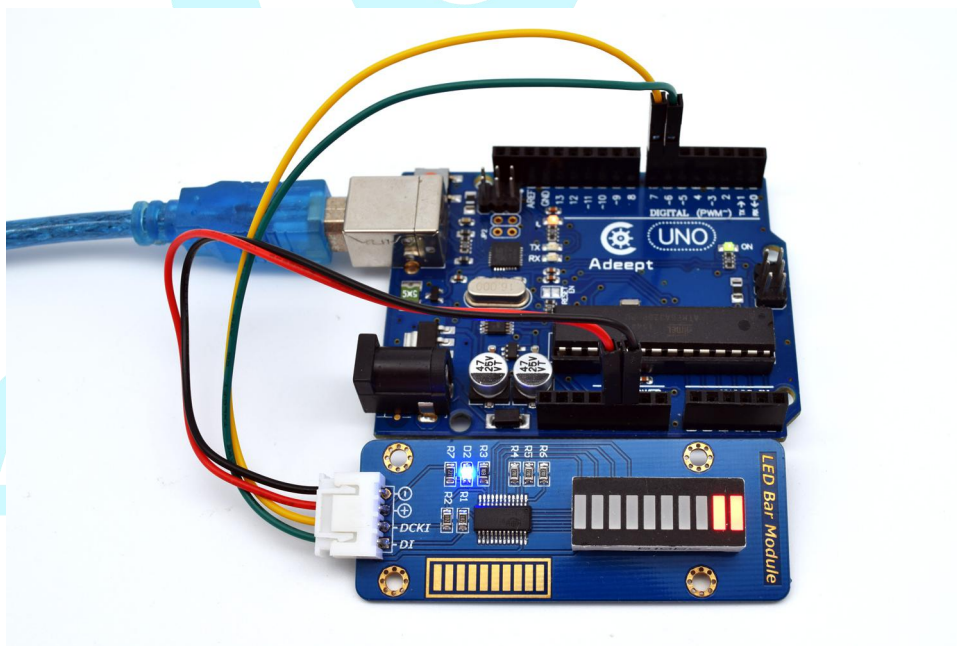
```
10  a = LED 1, b = LED 2, c = LED 3, etc.
11  dec    hex    binary
12  0      = 0x0   = 0b0000000000000000 = all LEDs off
13  5      = 0x05  = 0b0000000000000101 = LEDs 1 and 3 on, all others off
14  341    = 0x155 = 0b0000000101010101 = LEDs 1,3,5,7,9 on, 2,4,6,8,10 off
15  1023   = 0x3ff = 0b0000001111111111 = all LEDs on
16
17      |      |
18      10      1.
19
20 Website: www.adeept.com
21 E-mail: support@adeept.com
22 Author: Tom
23 Date: 2017/03/17
24 *****/
25 #include <Adeept_Bar.h>
26 Adeept_Bar bar(7, 6); // Clock pin, Data pin
27 void setup()
28 {
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Done uploading.

Sketch uses 1578 bytes (4%) of program storage space. Maximum is 32256 bytes.
Global variables use 23 bytes (1%) of dynamic memory, leaving 2025 bytes for local variables.

Arduino/Genuino Uno on COM3

Now you can see the LEDs on the LED Bar Graph module light up and dim one by one repeatedly.



Lesson 33 Making A Simple Remote Control Device

Introduction

The 1838B IR Receiver is a 38KHz IR receiver that can receive signals modulated by a standard 38KHz remote control. By programming the Arduino UNO R3 board, we can decode the signals.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * IR Receiver Module
- 1 * Remote Controller Module
- 1 * USB Cable
- 1 * 4-Pin Wires

Experimental Principle

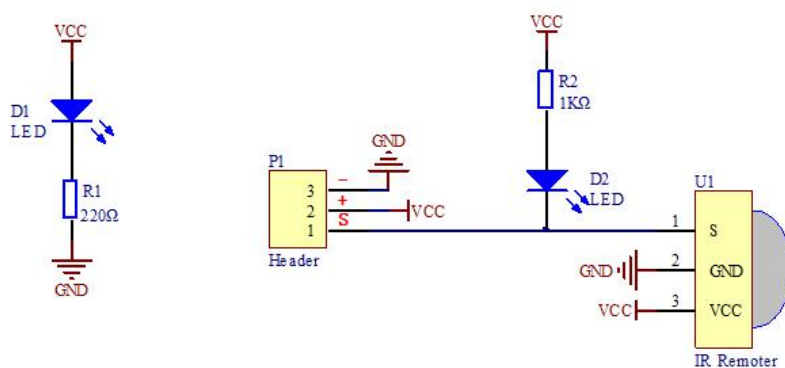
The Fritzing image:



Pin definition:

S	Digital output
+	VCC
-	GND

The schematic diagram:



In this experiment, by programming the Arduino board, we encode the data received by the IR Receiver that is sent by the remote control, and display the deciphered data on

Serial Monitor via the serial port. In the program, we use the Arduino-IRremote-master library (provided).

Note:

Before using this library, you have to delete the [RobotIRremote](#) directory in your Arduino IDE directory (check in IDE by File->Preference, and see the path in the Browse dialog box), and delete the [RobotIRremote](#) directory in the system Documents folder. For example, if your computer is running on Windows 7, you need to delete the [RobotIRremote](#)

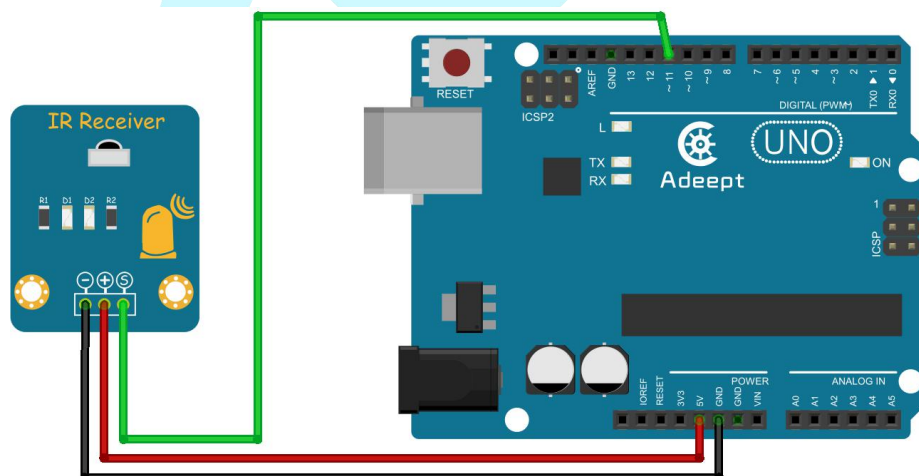
Directory in

[C:\ProgramFiles\(x86\)\Arduino\libraries](#) and [C:\Users\SJG\Documents\Arduino\libraries](#).

Otherwise, when you compile the program, errors will be prompted.

Experimental Procedures

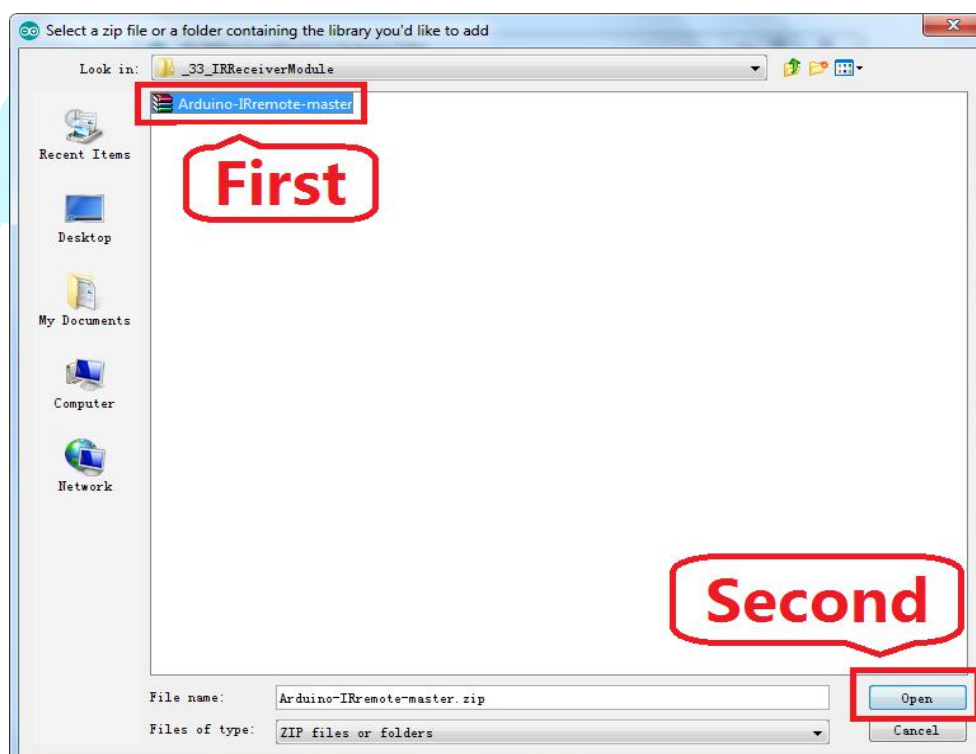
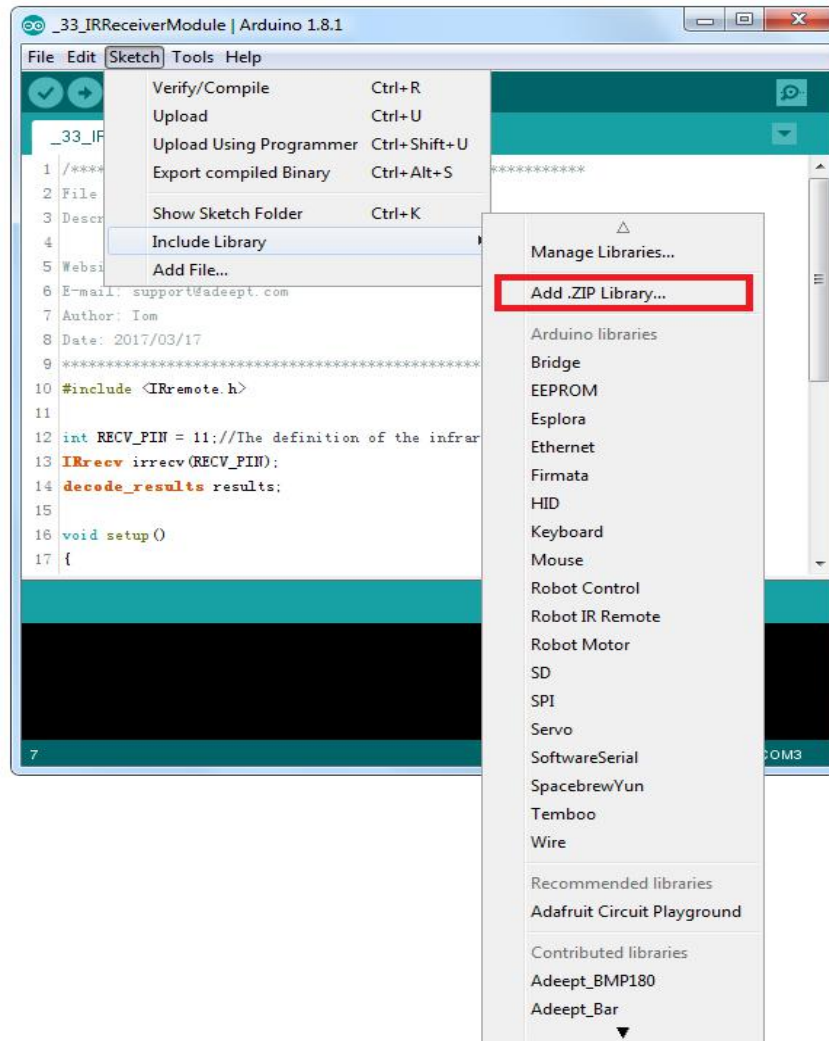
Step 1: Build the circuit



Adept UNO R3 Board	IR Receiver Module
D11	S
5V	+
GND	-

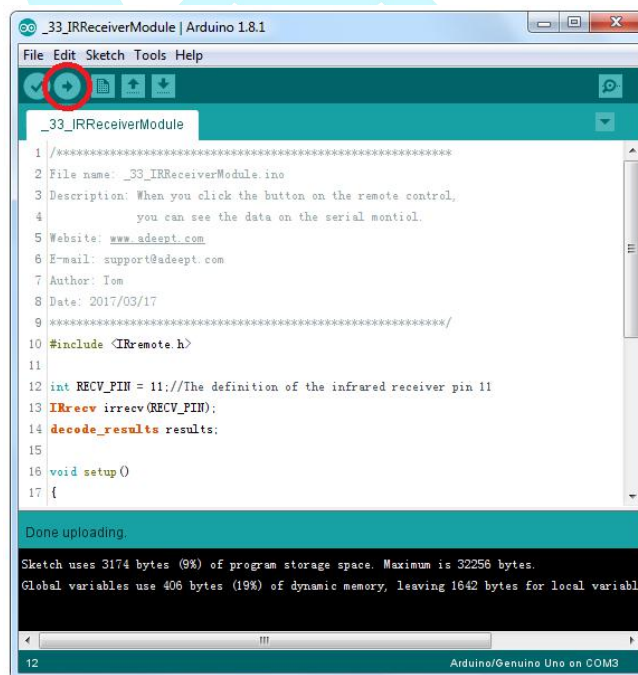
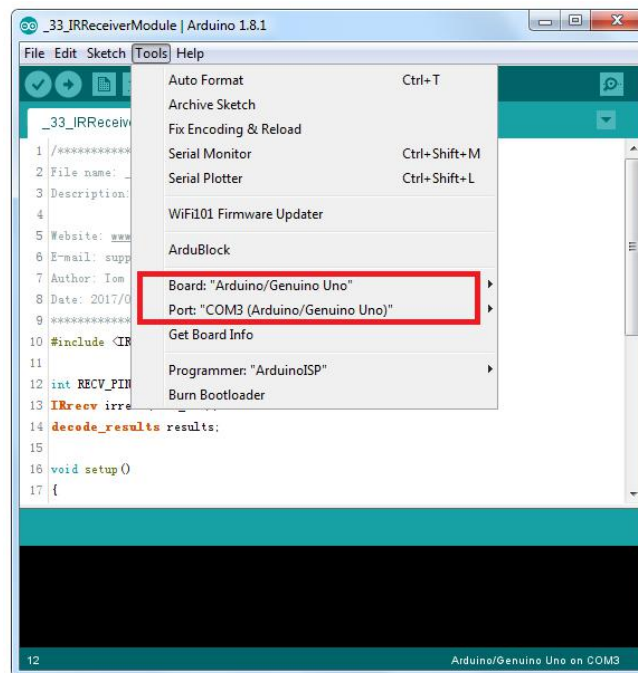
Step 2: Install the function library (Arduino-IRremote-master.zip).



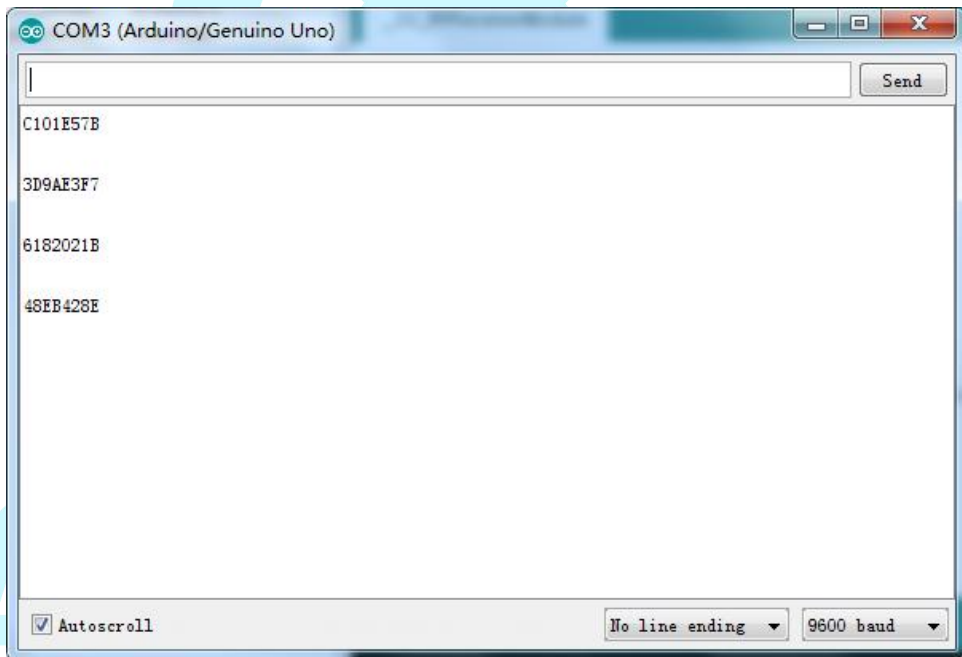
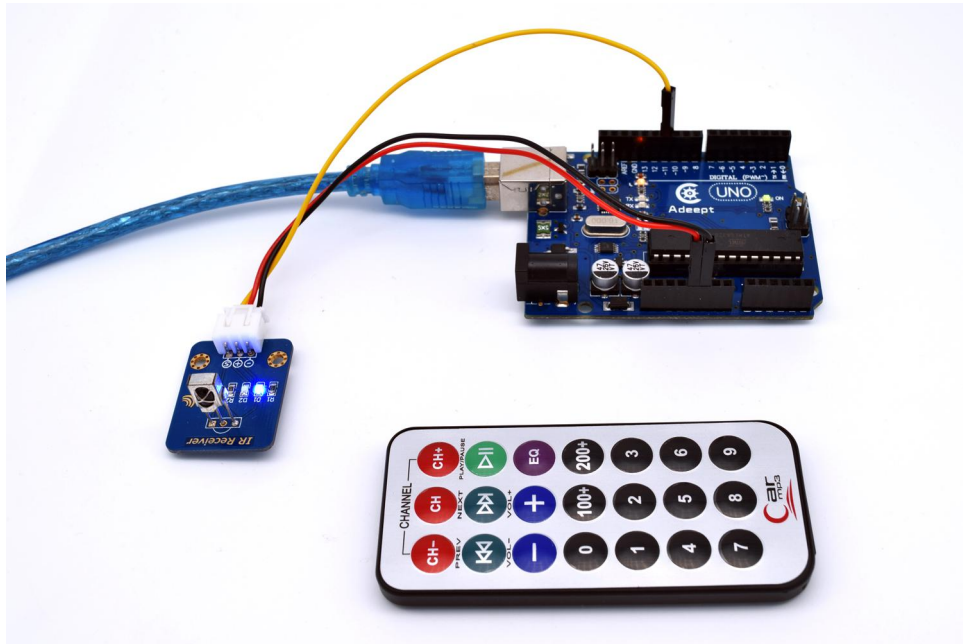


Step 3: Program _33_IRReceiverModule.ino

Step 4: Compile and download the sketch to the UNO R3 board.



Now press any key on the remote control, and you will see the corresponding code displayed on Serial Monitor.



Lesson 34 Detection of The Soil Moisture System

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.

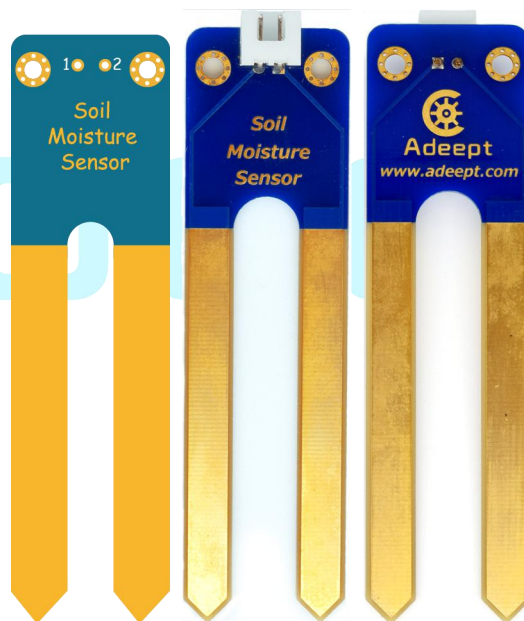
The CM Module consists of a comparator LM393 and extremely simple external circuits. When using the module, you can set a threshold via the blue potentiometer beforehand. When the input analog value reaches the threshold, the digital pin S will output a Low level.

Components

- 1 * Adeept Arduino UNO R3 Board
- 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:

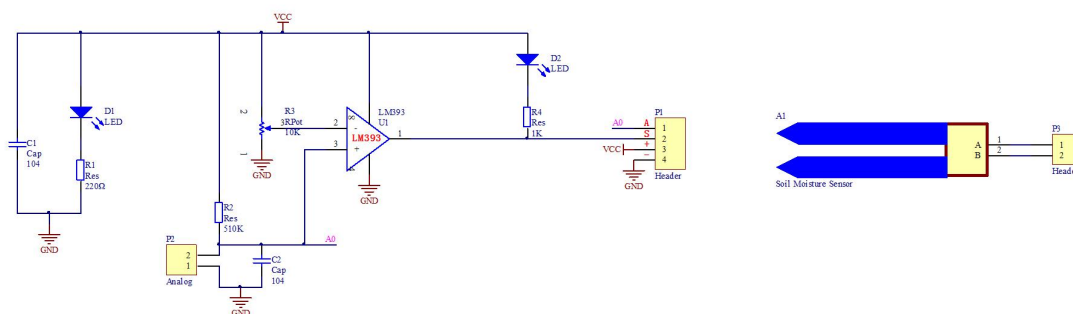




Pin definition:

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

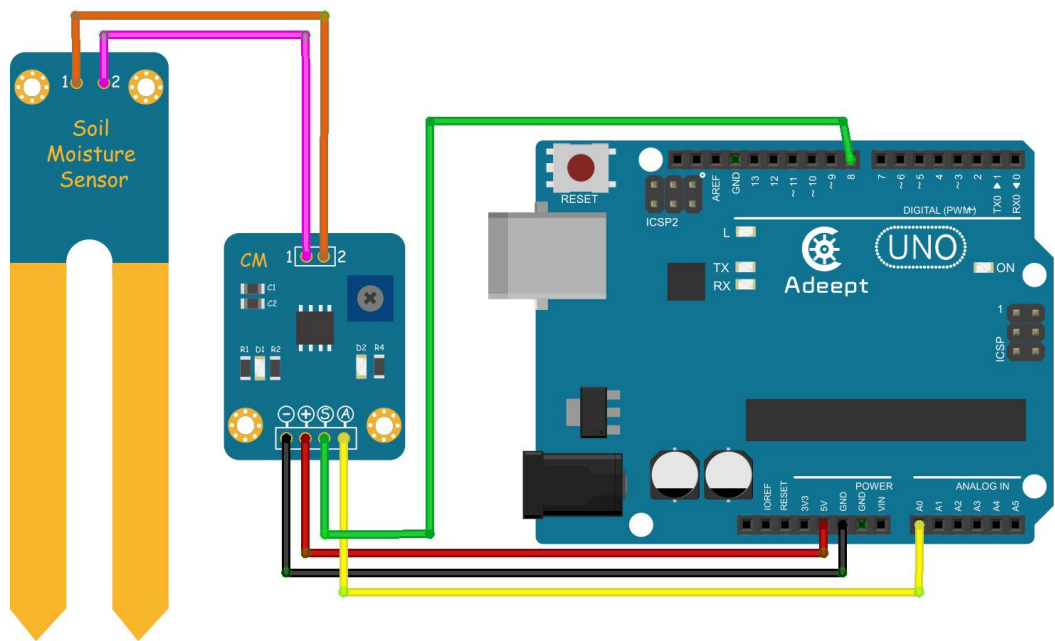
The schematic diagram:



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

Experimental Procedures

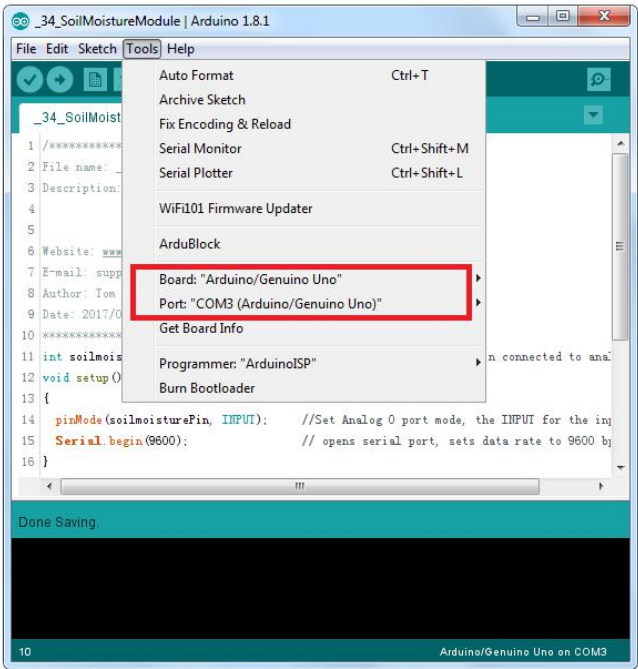
Step 1: Build the circuit

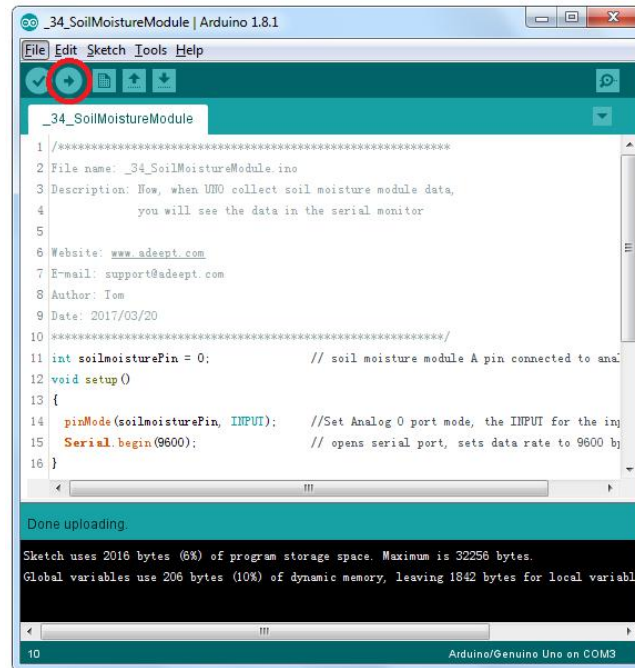


Adeept UNO R3 Board	CM Module	Soil Moisture Sensor Module
D8	S	
A0	A	
5V	+	
GND	-	
	1	2
	2	1

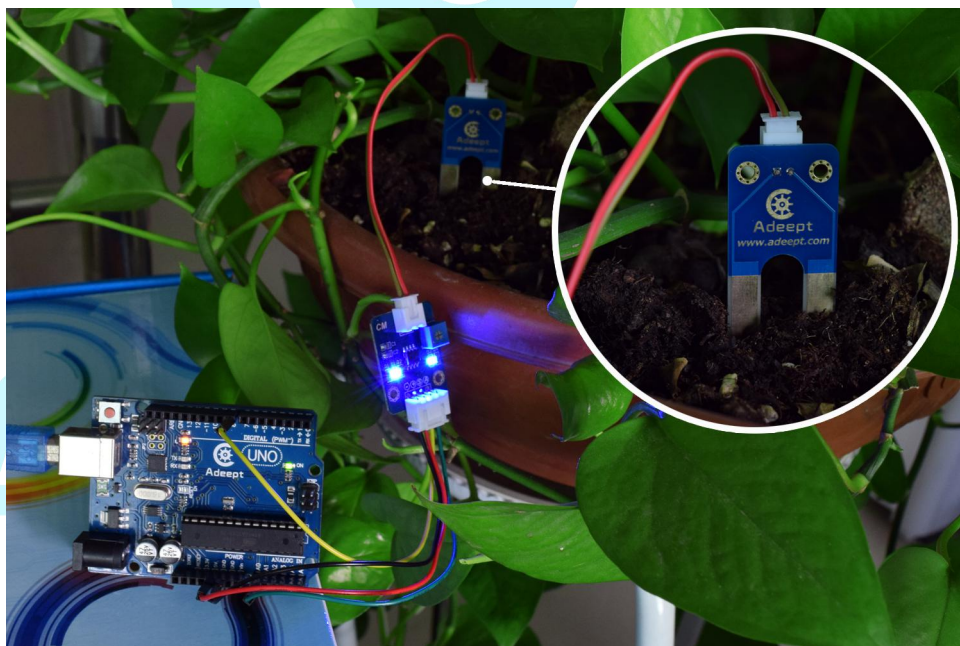
Step 2: Program _34_SoilMoistureModule.ino

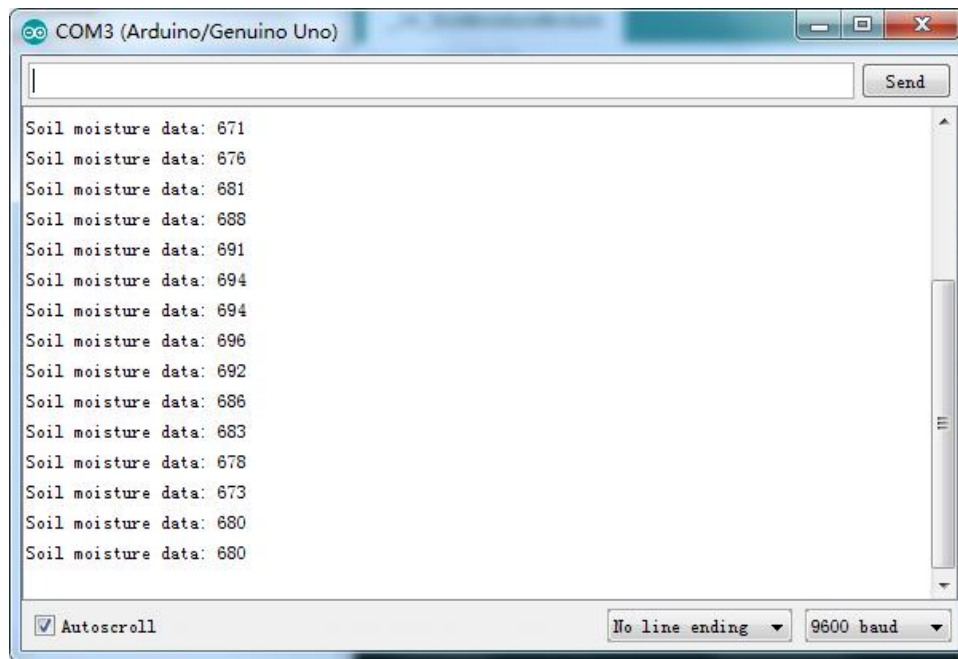
Step 3: Compile and download the sketch to the UNO R3 board.





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





Adeept

Lesson 35 Detection of The Water Height System

Introduction

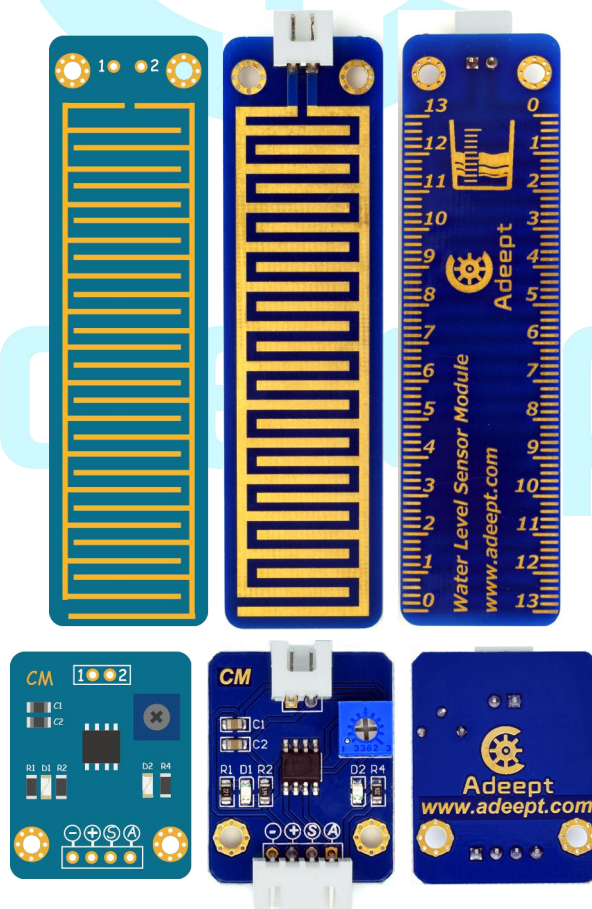
The module is a simple water level sensor. It measures the water volume by the printed wires exposed to the air on the module. The more water on the surface, more wires connected. Thus, the area of electrified wires gets larger, so the output voltage will increase. The surface of the sensor is gilded to prolong its life.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * Water Level Sensor Module
- 1 * CM Module
- 1 * USB Cable
- 1 * 4-Pin Wires
- 1 * 2-Pin Female to Female Wires

Experimental Principle

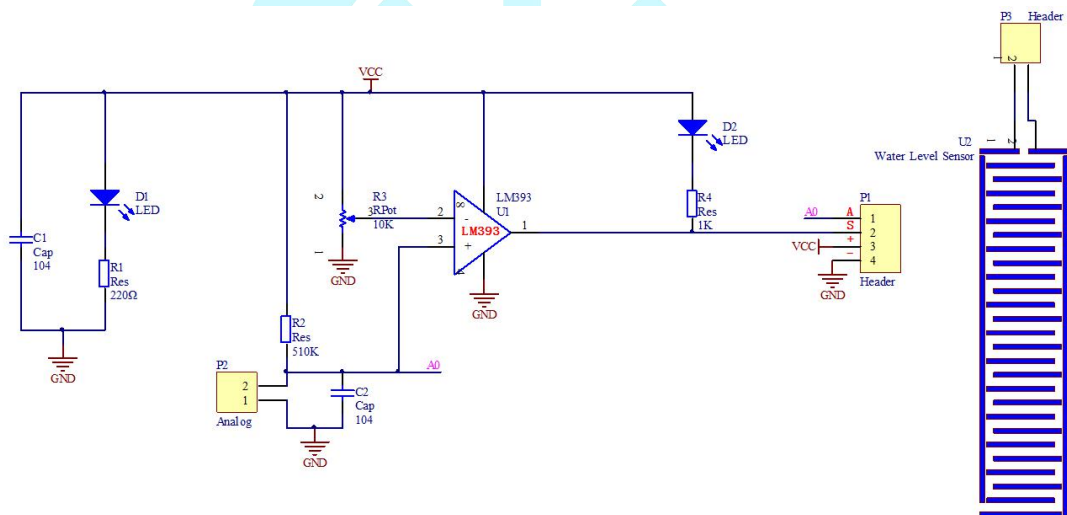
The Fritzing images:



Pin definition:

Water Level Sensor Module	
1	Analog output
2	Analog output
CM Module	
1	Analog output
2	Analog output
S	Digital output
A	Analog output
+	VCC
-	GND

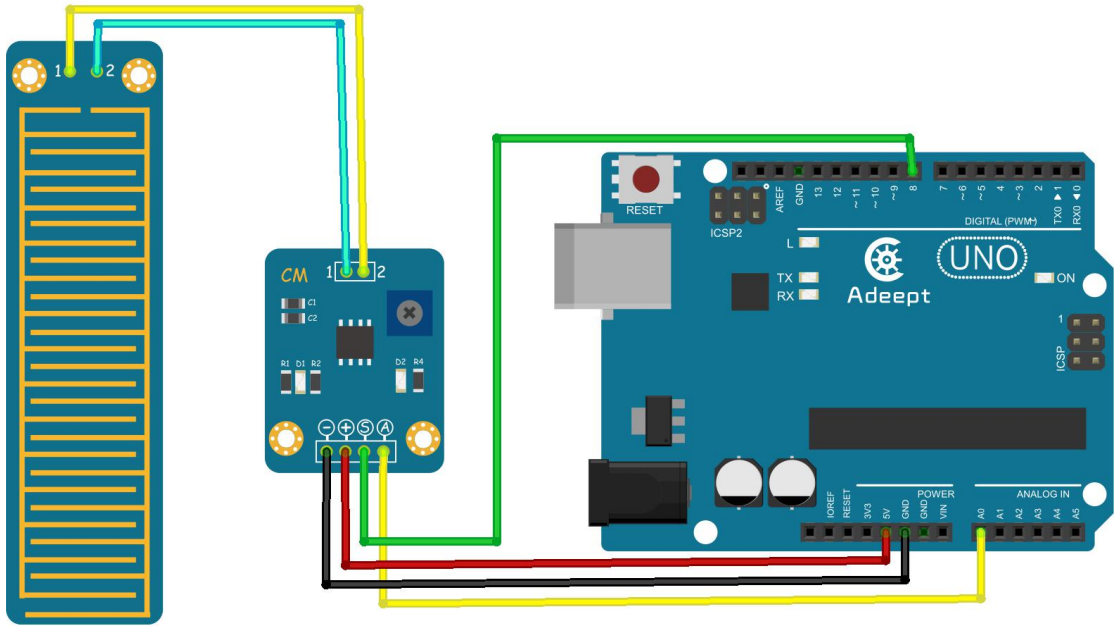
The schematic diagram:



This experiment collects the data of water level by the Water Level Sensor module and displays it on Serial Monitor.

Experimental Procedures

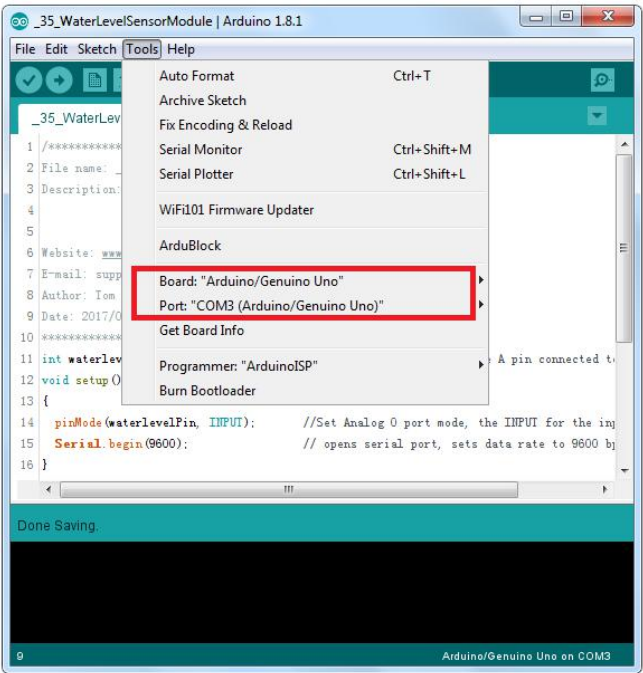
Step 1: Build the circuit

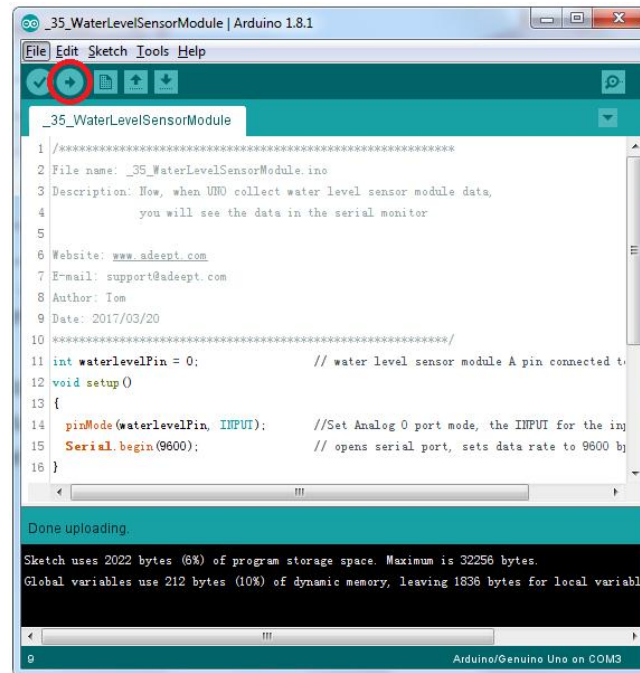


Adeept UNO R3 Board	CM Module	Water Level Sensor Module
D8	S	
A0	A	
5V	+	
GND	-	
	1	2
	2	1

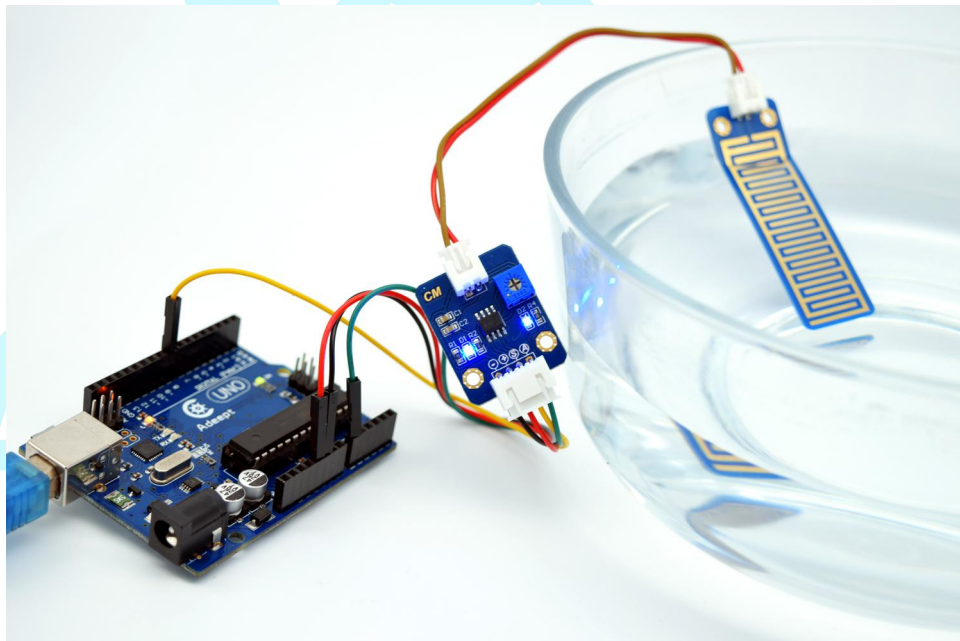
Step 2: Program _35_WaterLevelSensorModule.ino

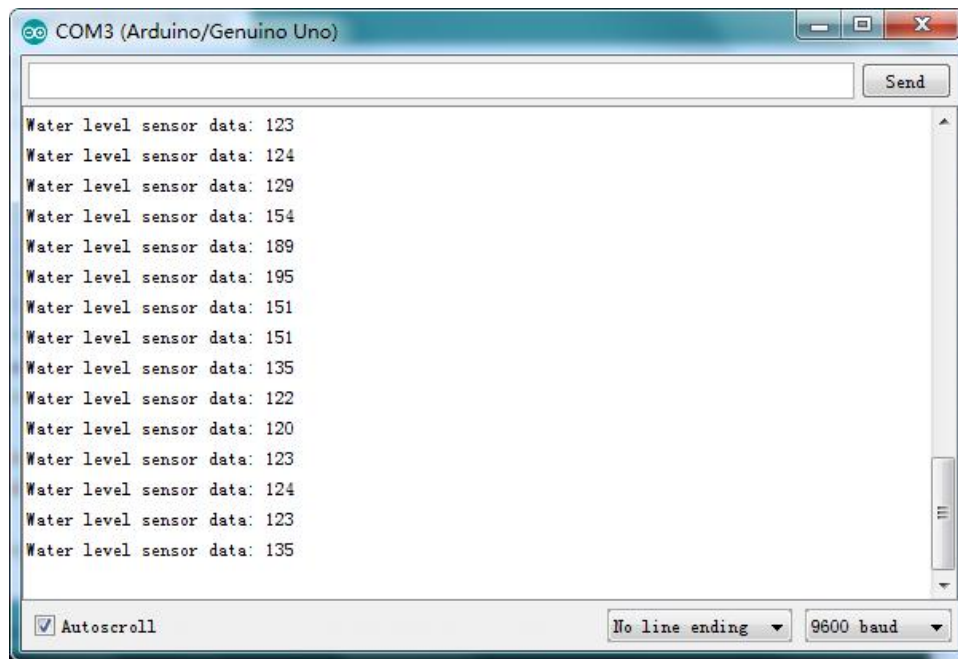
Step 3: Compile and download the sketch to the UNO R3 board.





Open Serial Monitor of the Arduino IDE. Take a bottle with some water. Place the Water Level Sensor module in the water at end and add water into the bottle. You will see the value of water level displayed on the window and changes as you add water.





Lesson 36 Detection of The Distance System

Introduction

Ultrasonic Distance Sensor module supports a contactless detection within a distance of 2cm-400cm. It contains an ultrasonic emitter, receiver and control circuits.

Notes:

1. The module is not suggested to connect wires when power is on. If you have to do so, please first connect the GND and then other pins; otherwise, the module may not work.
2. During the ranging, the area of the targeted object should be no less than 0.5mm and the surface facing the module should be as flat as possible; otherwise the result may be inaccurate.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Ultrasonic Distance Sensor Module
- 1 * USB Cable
- 4 * Male to Female Wires

Experimental Principle

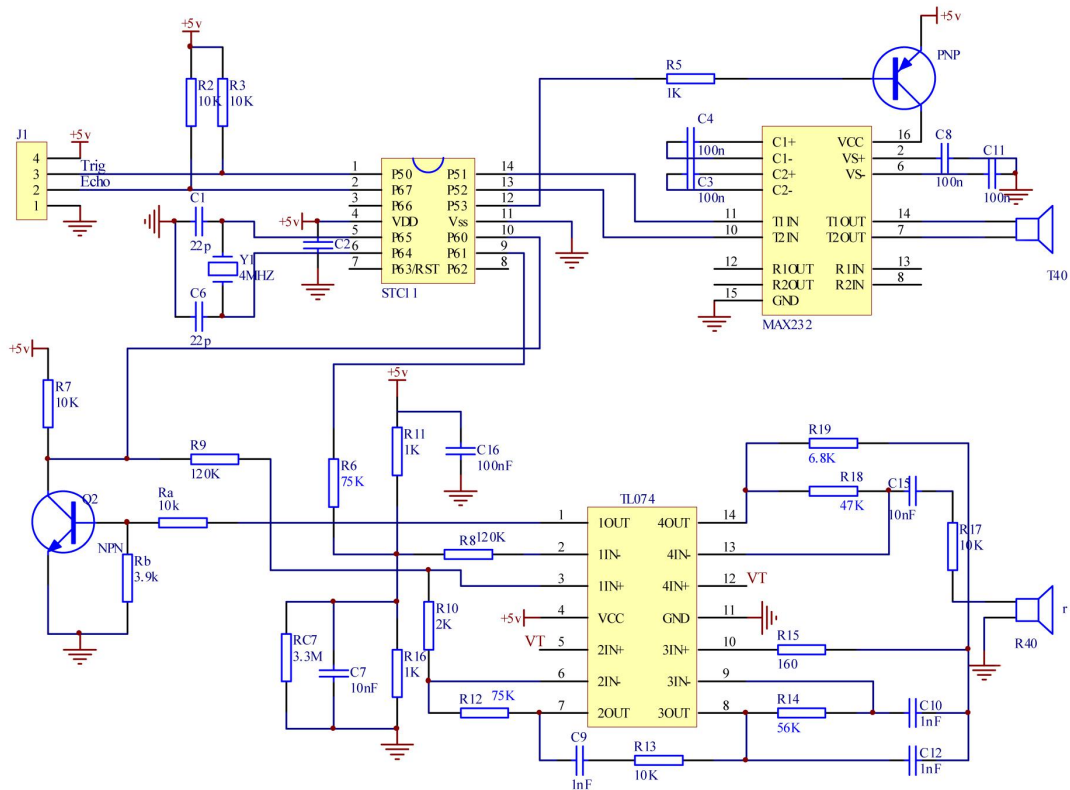
The Fritzing image:



Pin definition:

Trig	Digital output
Echo	Digital output
Vcc	VCC
Gnd	GND

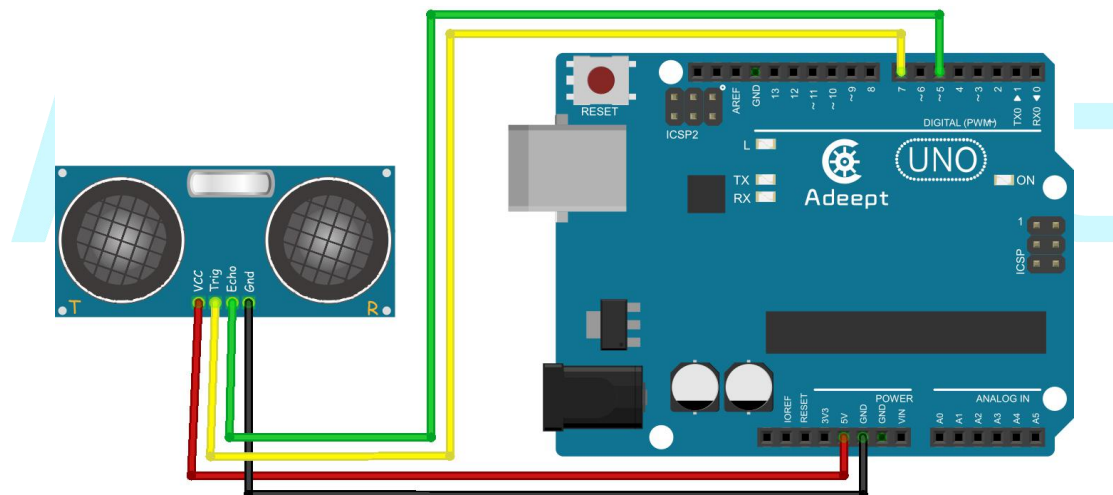
The schematic diagram:



This experiment uses the Ultrasonic Distance Sensor module to detect the distance between the obstacle and module and show the data sensed on Serial Monitor via the serial port.

Experimental Procedures

Step 1: Build the circuit

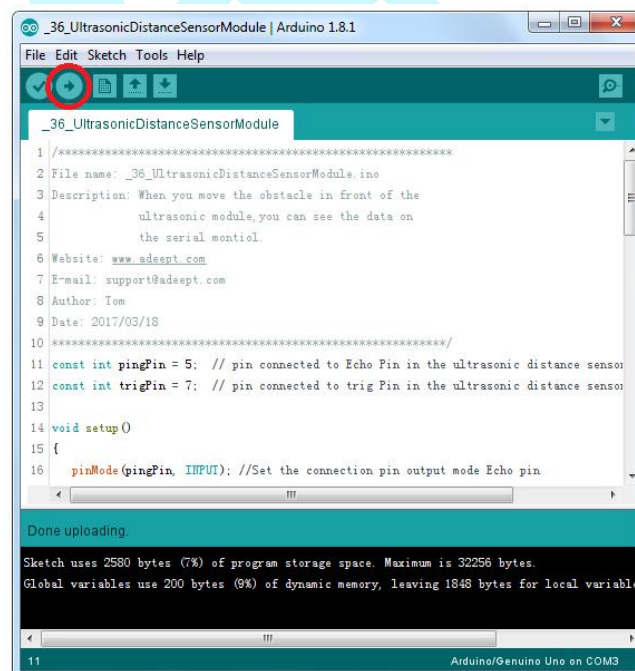
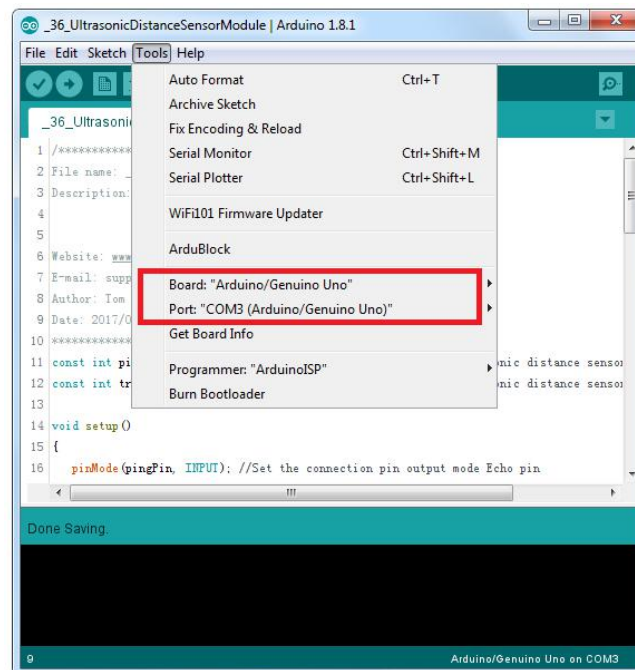


Adeept UNO R3 Board	Ultrasonic Distance Sensor Module
D7	Trig
D5	Echo
5V	Vcc

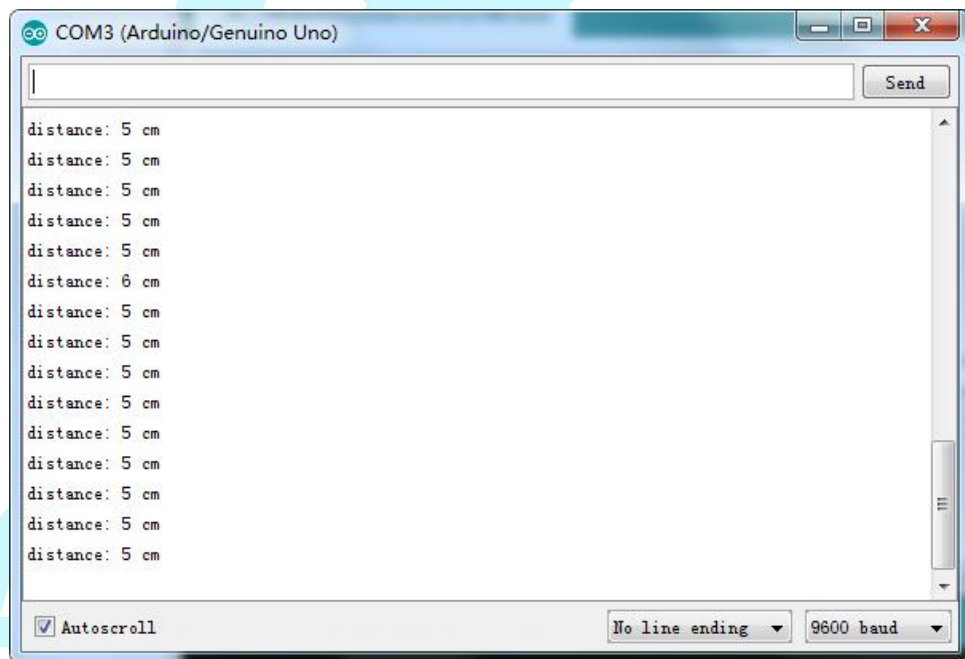
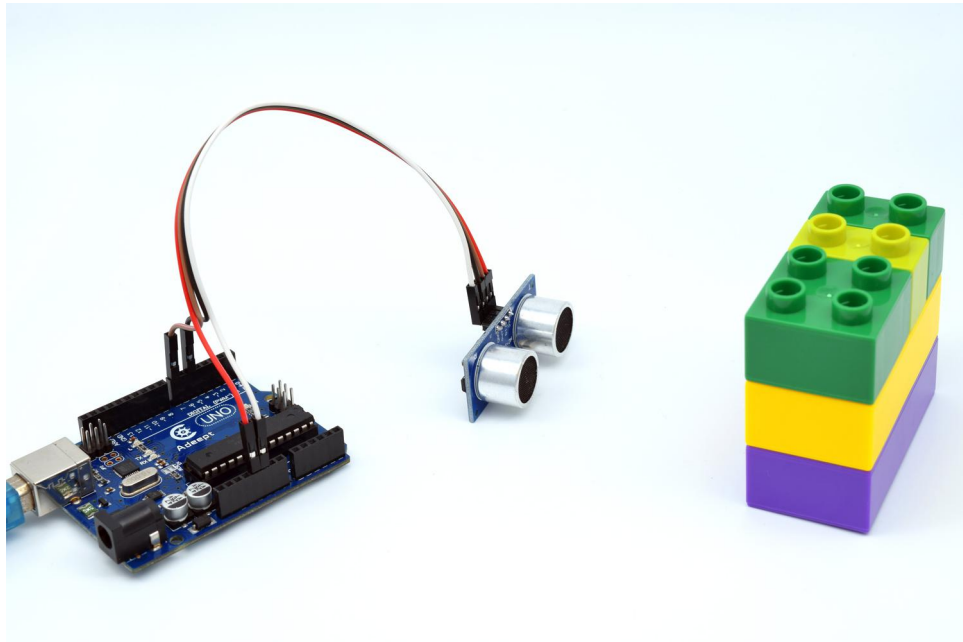
GND	Gnd
-----	-----

Step 2: Program _36_UltrasonicDistanceSensorModule.ino

Step 3: Compile and download the sketch to the UNO R3 board.



Open Serial Monitor of the Arduino IDE. You will see the distance to the obstacle at front of the Ultrasonic Distance Sensor module displayed on the window.



Lesson 37 Control An LED by PC

Introduction

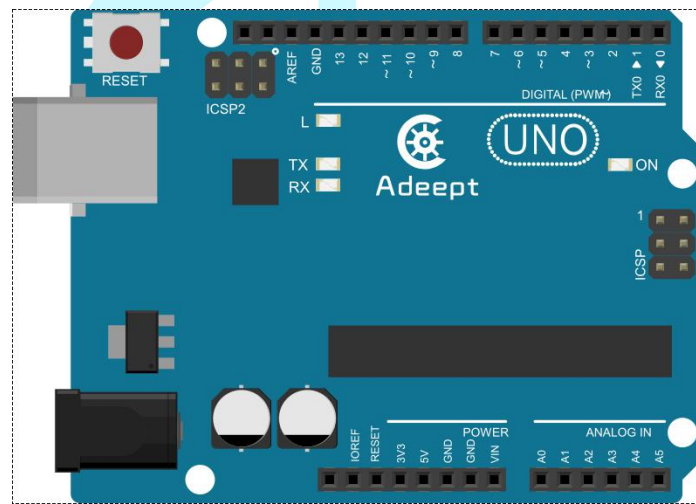
In this lesson, we will control an LED by PC. We send '0' or '1' to Arduino UNO via serial port, the state of the LED on the Arduino UNO board will be toggled.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * USB Cable

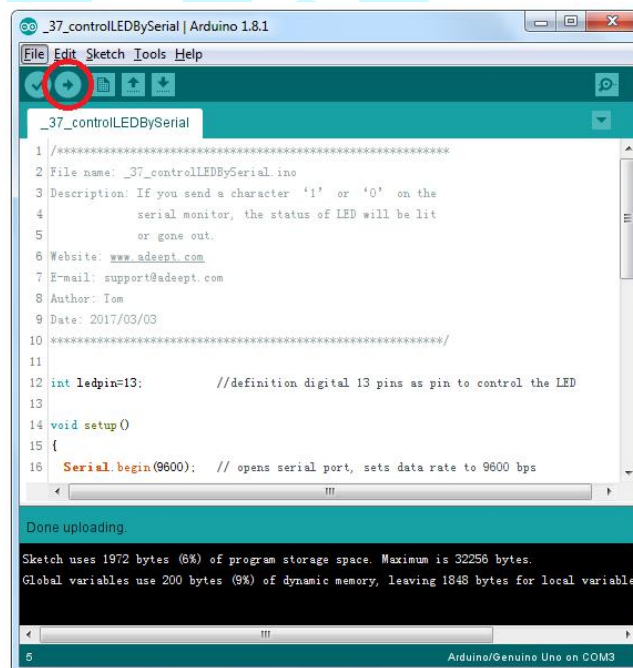
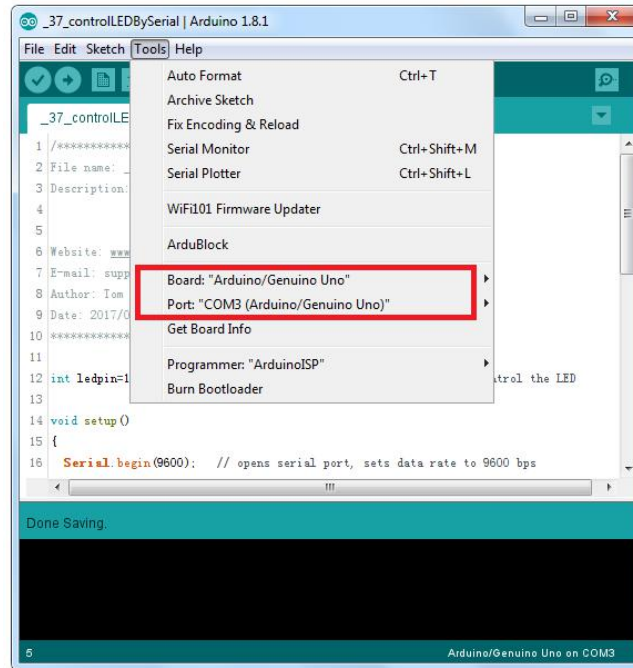
Experimental Procedures

Step 1: Build the circuit

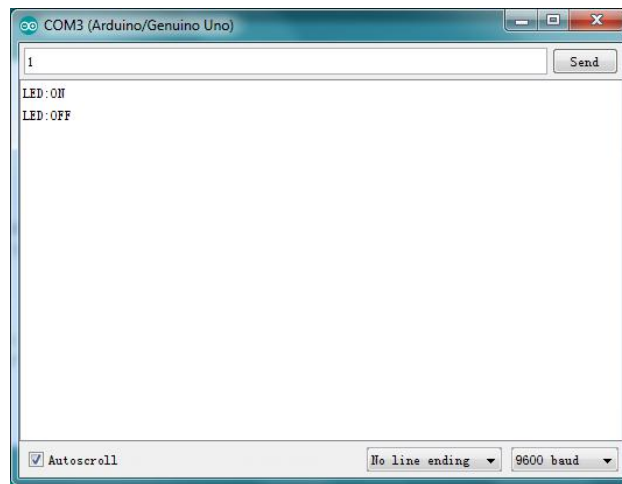


Step 2: Program `_37_controlLEDBySerial.ino`

Step 3: Compile and download the sketch to the UNO R3 board.



Open Serial Monitor of the Arduino IDE. Send 1 or 0, you can control the LED light or off.



Adeept

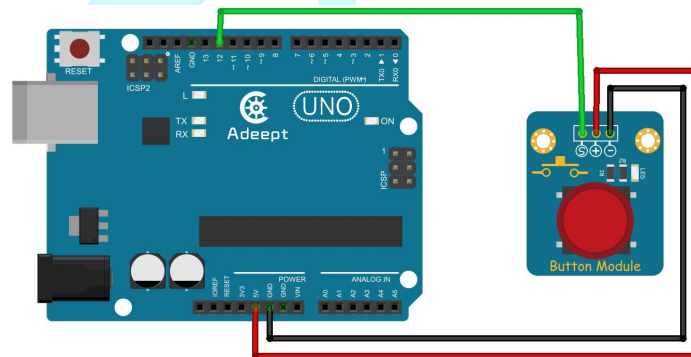
Lesson 38 Upload The State of A Button to PC

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * USB Cable
- 1 * Button Module
- 1 * 3-Pin Wires

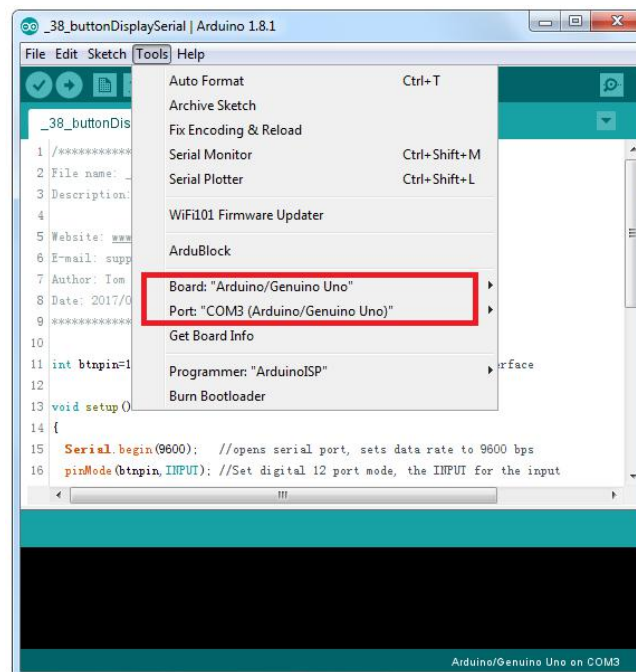
Experimental Procedures

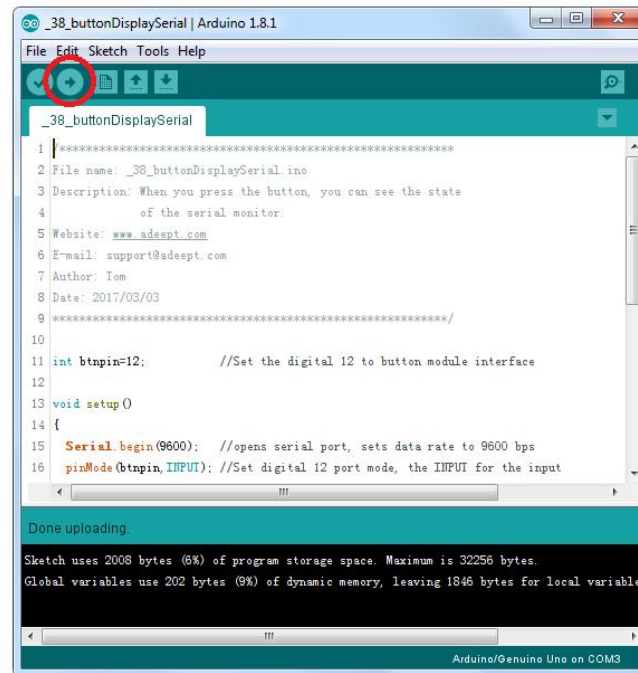
Step 1: Build the circuit



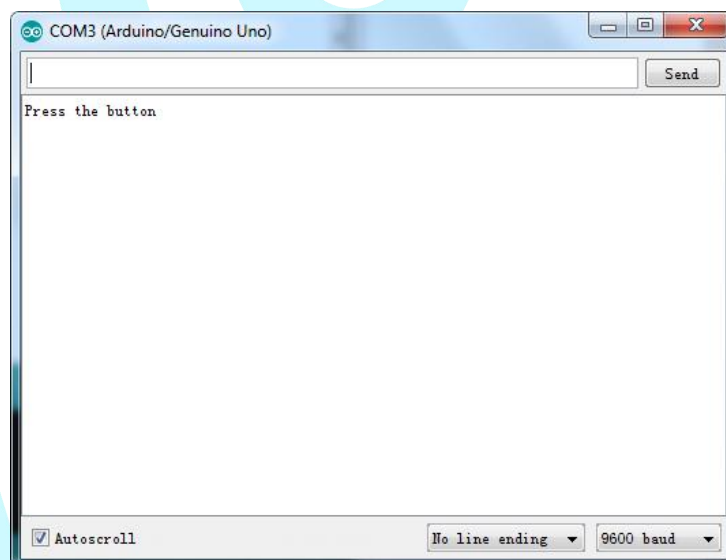
Step 2: Program _38_buttonDisplaySerial.ino

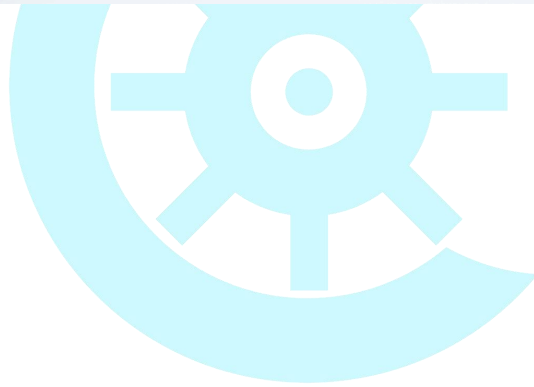
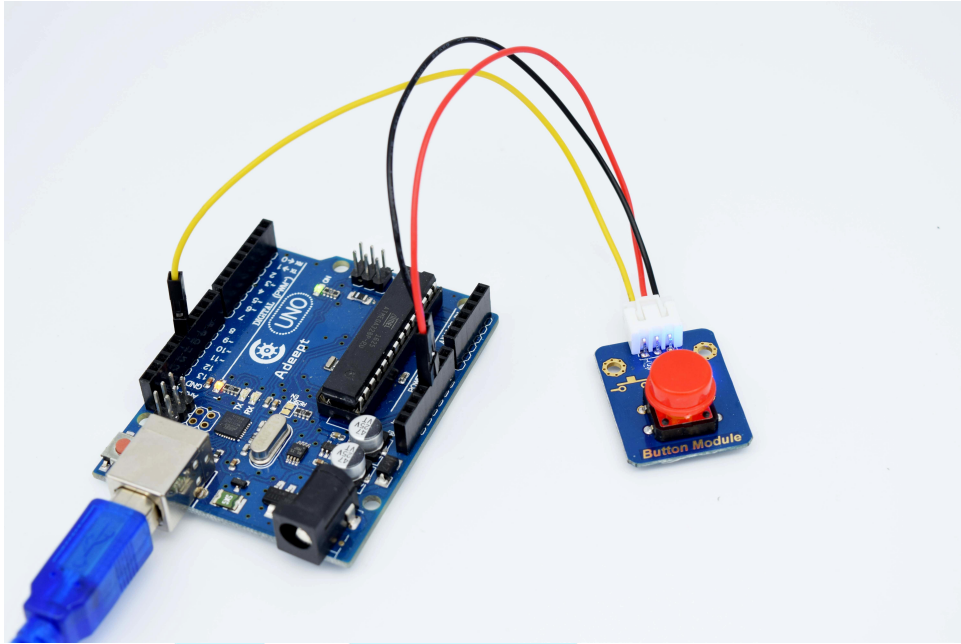
Step 3: Compile and download the sketch to the UNO R3 board.





Open the Serial Monitor in Arduino IDE and press the button module. Then the UNO R3 board will upload the data collected to and display in the Serial Monitor.





Adept

Lesson 39 Simple Laser Pen

Introduction

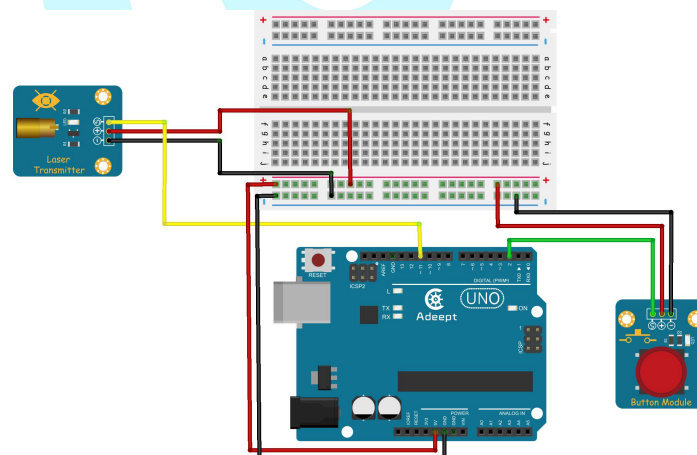
In this lesson, we will make a simple laser pen. The laser pen can be used for teaching or shooting.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Laser Transmitter Module
- 1 * Button Module
- 2 * 3-Pin Wires
- 1 * USB Cable
- 1 * Breadboard
- 2 * Hookup Wire Set

Experimental Procedures

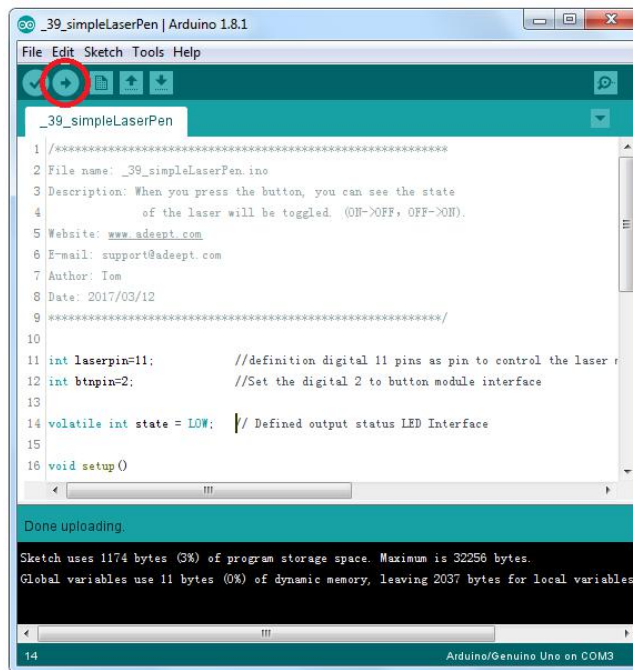
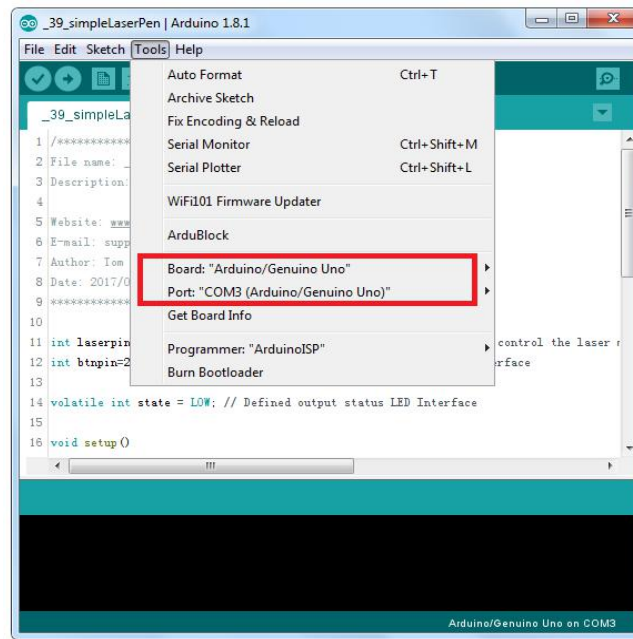
Step 1: Build the circuit



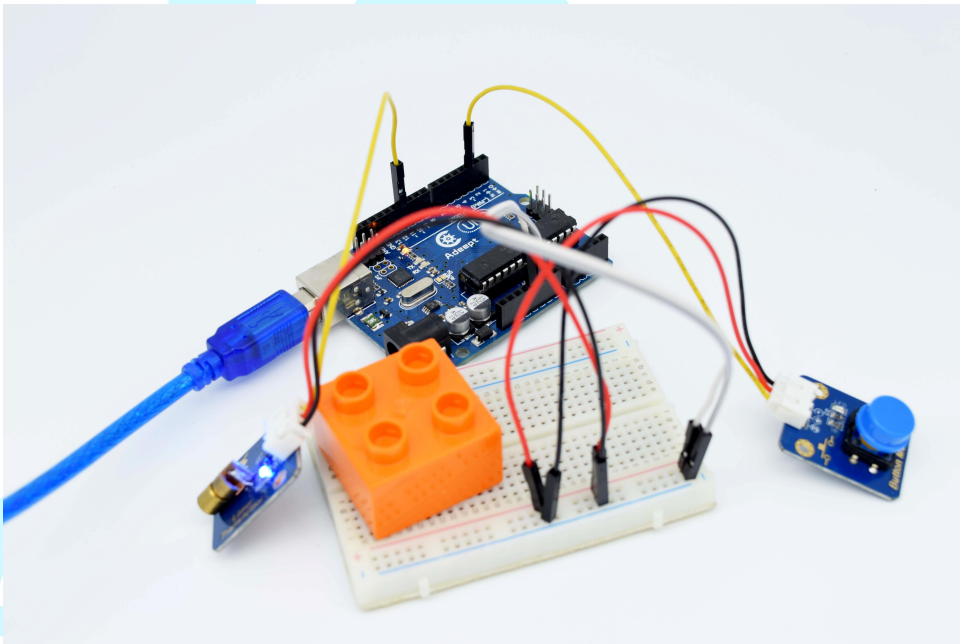
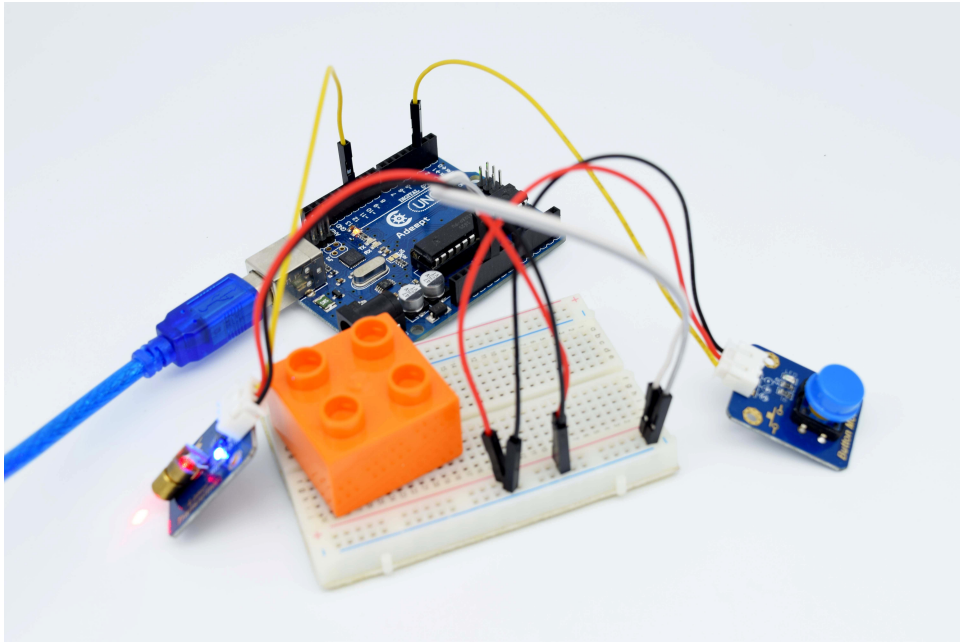
Note: DO NOT look directly into the laser!

Step 2: Program _39_simpleLaserPen.ino

Step 3: Compile and download the sketch to the UNO R3 board.



Use the button to toggle the laser module on and off.



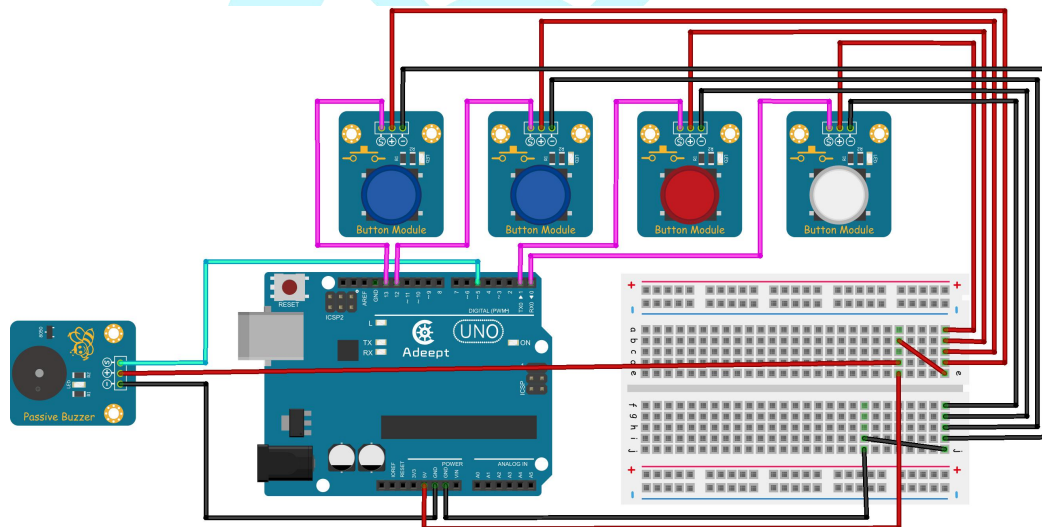
Lesson 40 Control Buzzer by Button

Components

- 1 * Adept Arduino UNO R3 Board
- 4 * Button Module
- 1 * Passive Buzzer Module
- 1 * USB Cable
- 5 * 3-Pin Wires
- 4 * Hookup Wire Set
- 1 * Breadboard

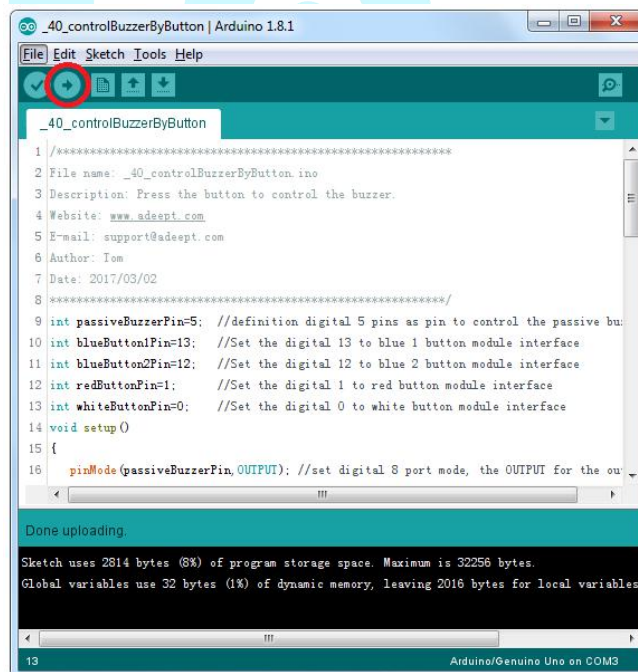
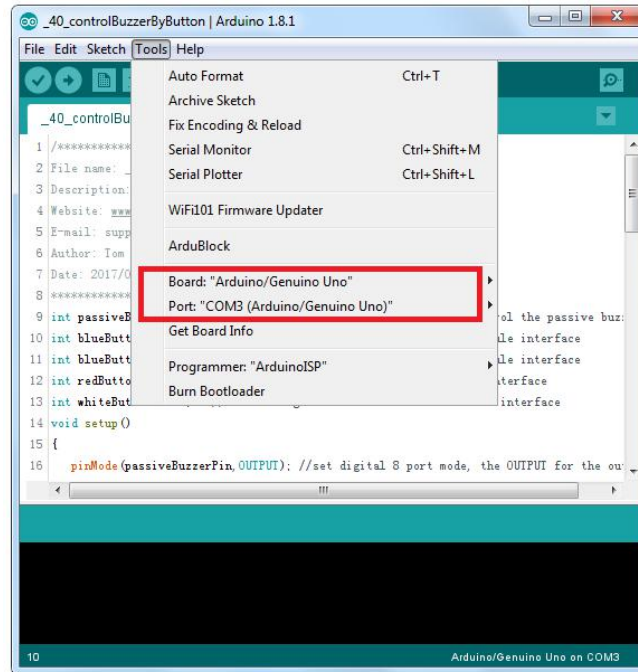
Experimental Procedures

Step 1: Build the circuit

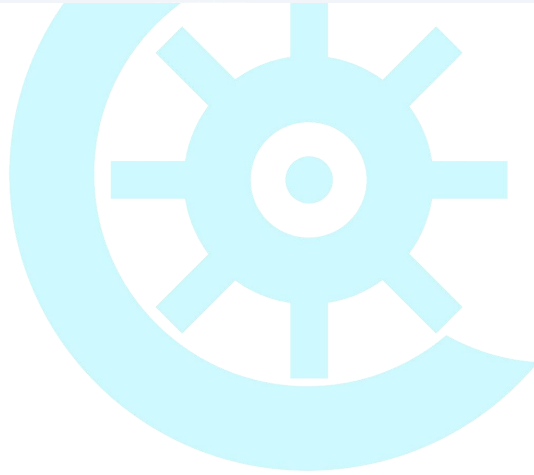
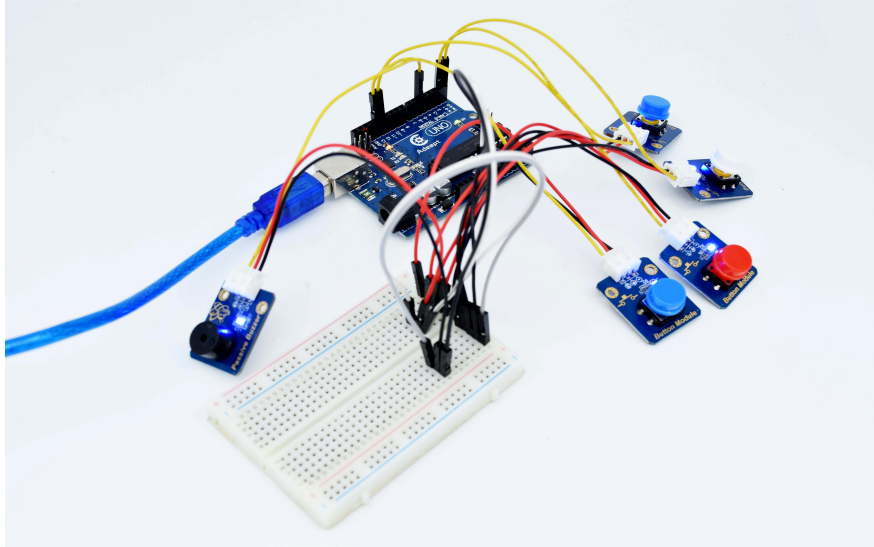


Step 2: Program `_40_controlBuzzerByButton.ino`

Step 3: Compile and download the sketch to the UNO R3 board.



When you press different buttons, the buzzer will make different sounds



Adeept

Lesson 41 A Simple Piano

Introduction

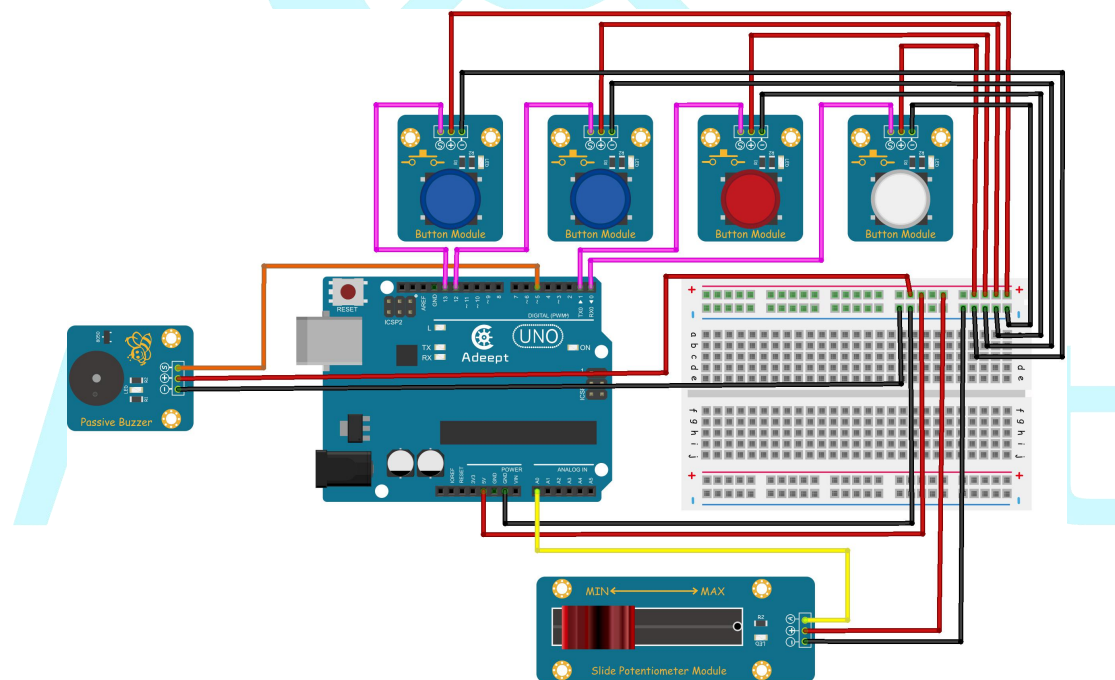
In this lesson, we use the potentiometer and four buttons to make the buzzer with a variety of sounds.

Components

- 1 * Adeept Arduino UNO R3 Board
- 4 * Button Module
- 1 * Passive Buzzer Module
- 1 * Slide Potentiometer Module
- 1 * USB Cable
- 6 * 3-Pin Wires
- 2 * Hookup Wire Set
- 1 * Breadboard

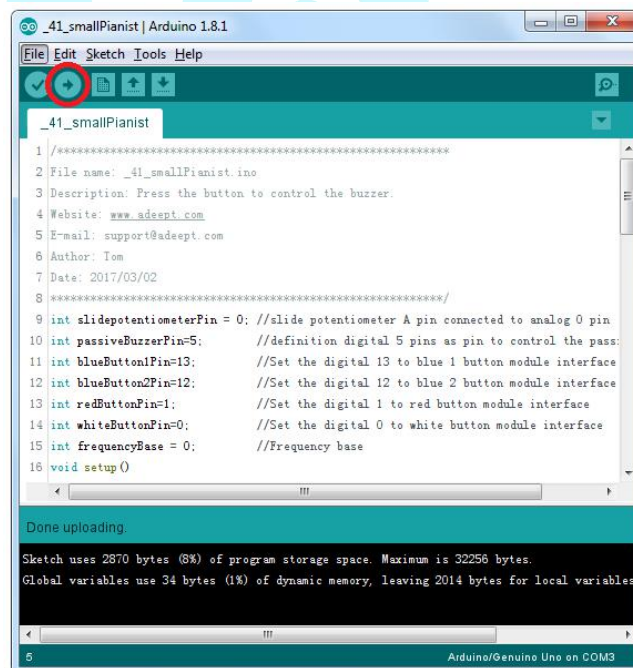
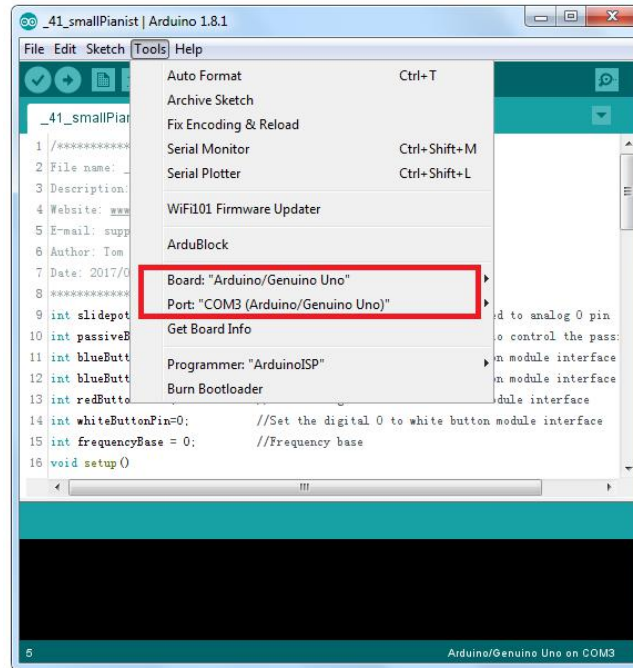
Experimental Procedures

Step 1: Build the circuit

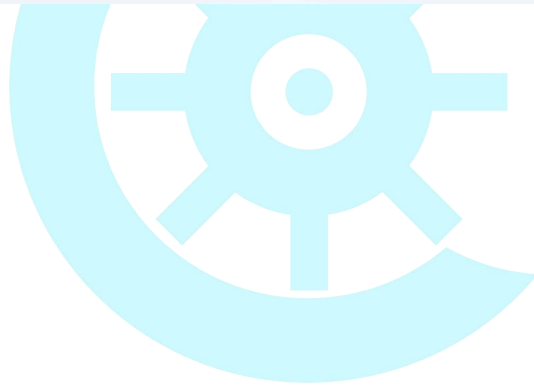
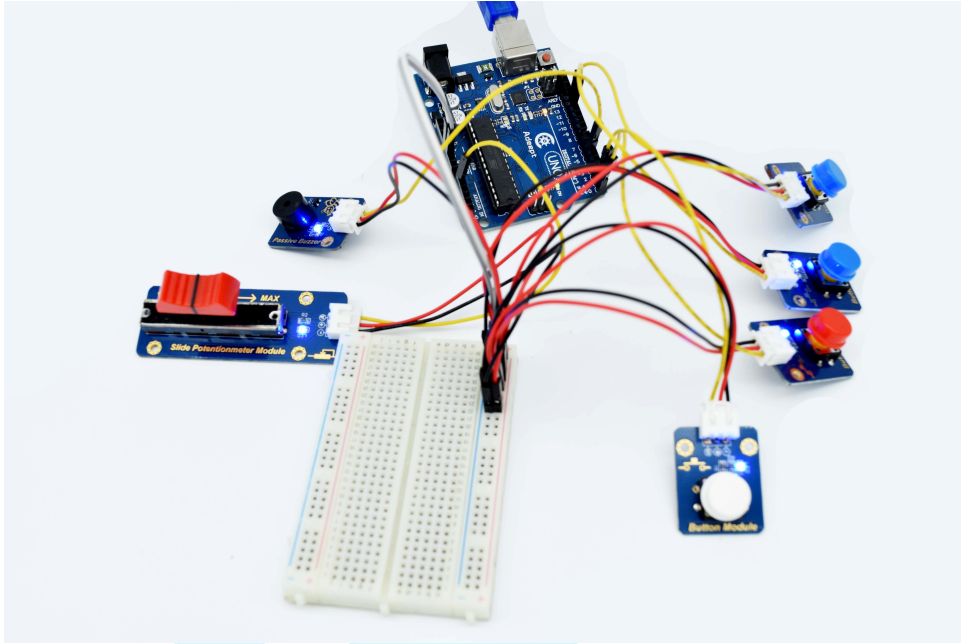


Step 2: Program `_41_smallPianist.ino`

Step 3: Compile and download the sketch to the UNO R3 board.



When you press different buttons, the buzzer makes different sounds. Slide potentiometer module allows the buzzer to make more sounds.



Adeept

Lesson 42 Change The Color of The RGB LED

Introduction

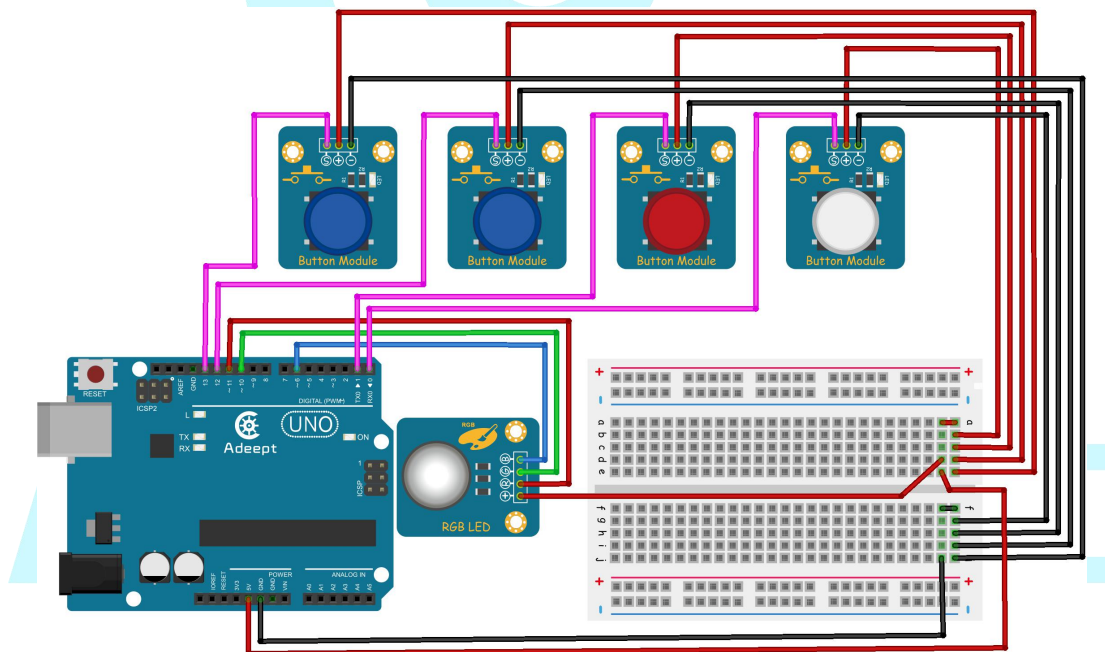
In this lesson, we will control an RGB LED through four buttons to change the color of the LED.

Components

- 1 * Adept Arduino UNO R3 Board
- 4 * Button Module
- 1 * RGB Module
- 1 * USB Cable
- 5 * 3-Pin Wires
- 4 * Hookup Wire Set
- 1 * Breadboard

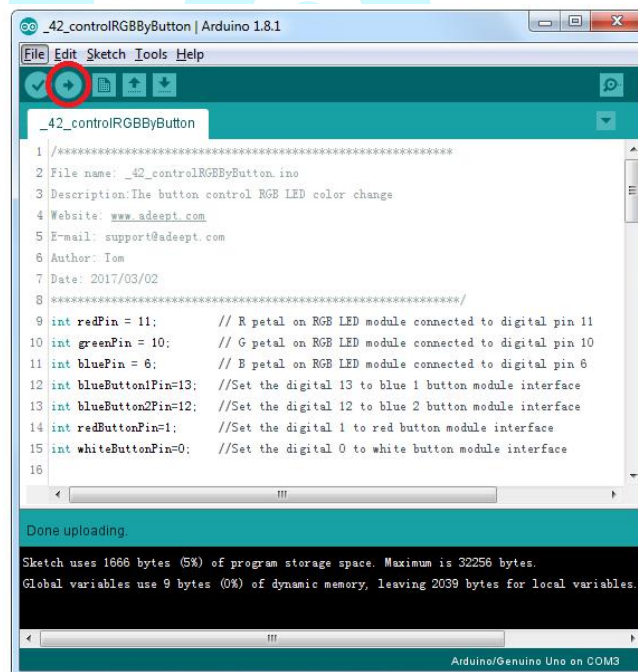
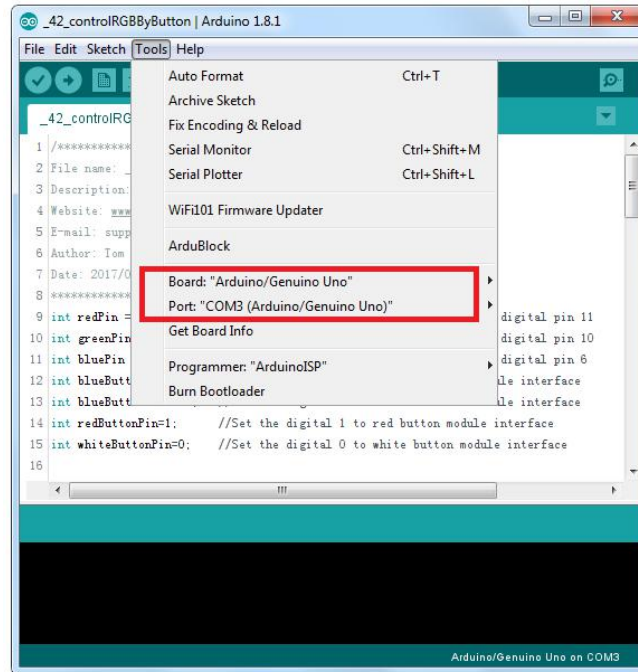
Experimental Procedures

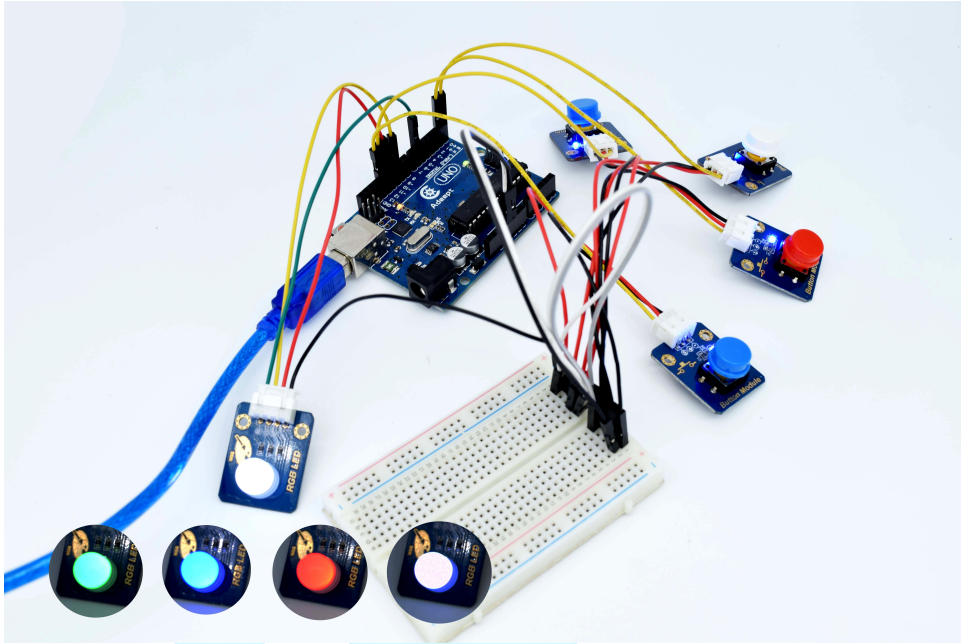
Step 1: Build the circuit



Step 2: Program `_42_controlRGBByButton.ino`

Step 3: Compile and download the sketch to the UNO R3 board.





Adeept

Lesson 43 A Simple Light Control Lamp

Introduction

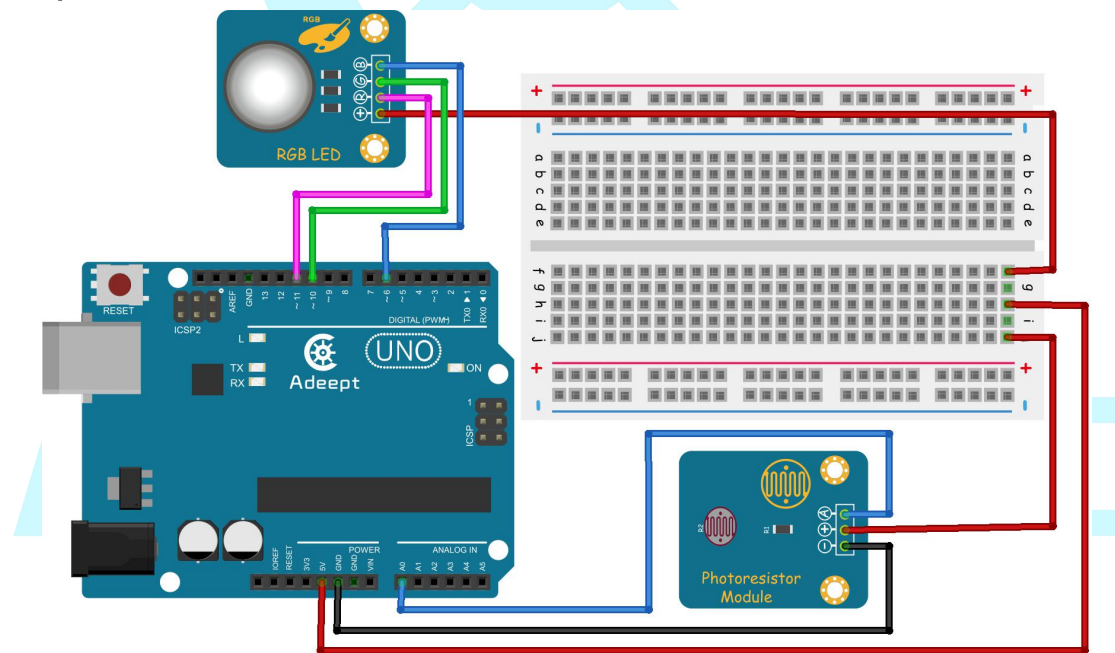
In this lesson, we will use photoresistor module and RGB LED to build a simple light control street lamp model. The lamp will automatically open at night and automatically closed at day.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Photoresistor Module
- 1 * RGB Module
- 1 * USB Cable
- 1 * 4-Pin Wires
- 1 * 3-Pin Wires
- 1 * Hookup Wire Set
- 1 * Breadboard

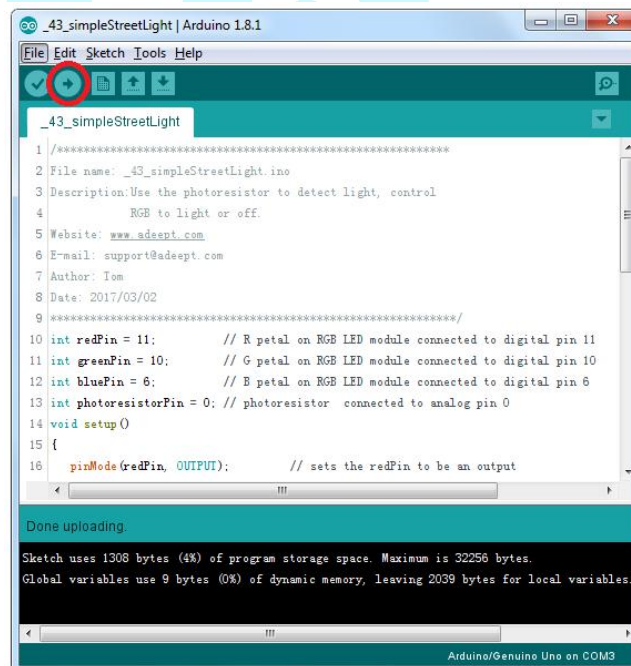
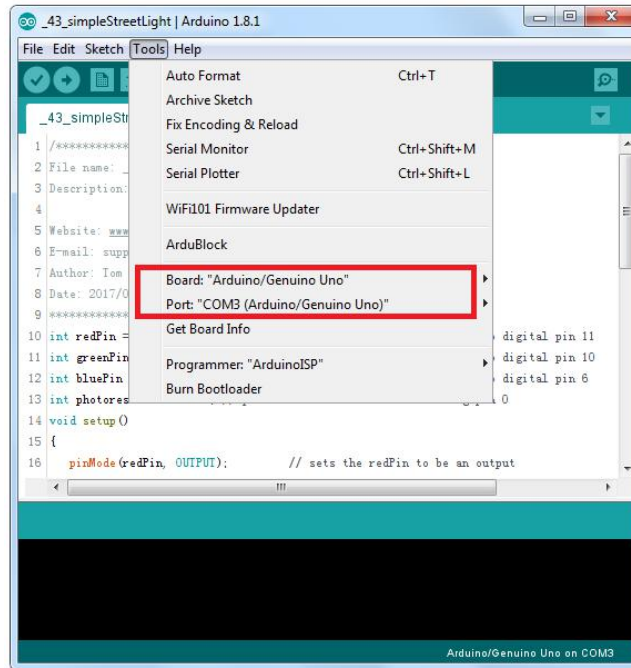
Experimental Procedures

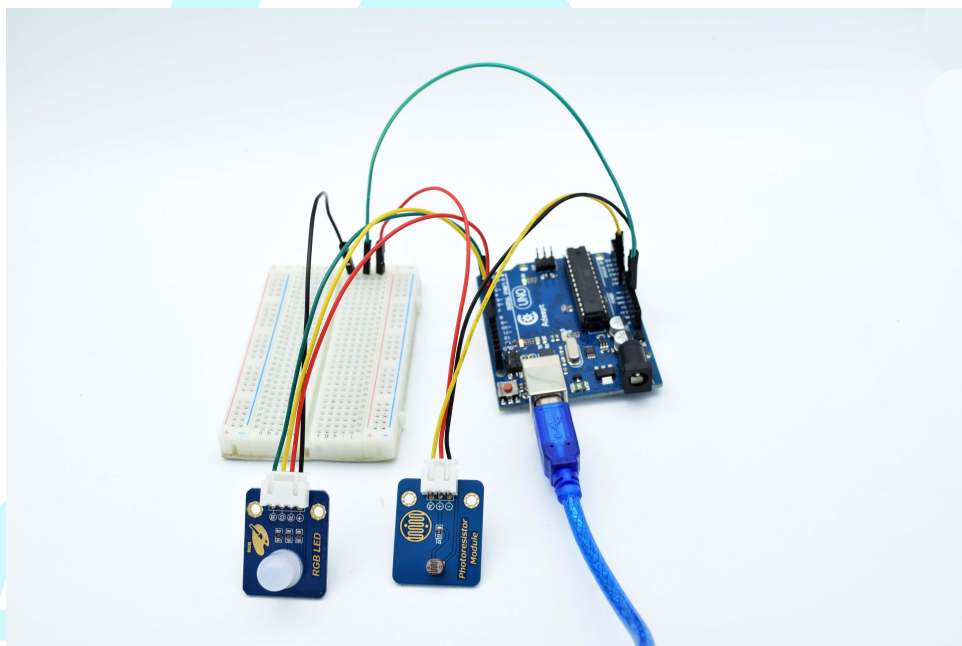
Step 1: Build the circuit



Step 2: Program `_43_simpleStreetLight.ino`

Step 3: Compile and download the sketch to the UNO R3 board.





Lesson 44 Control Segment Display by Rotary Encoder

Introduction

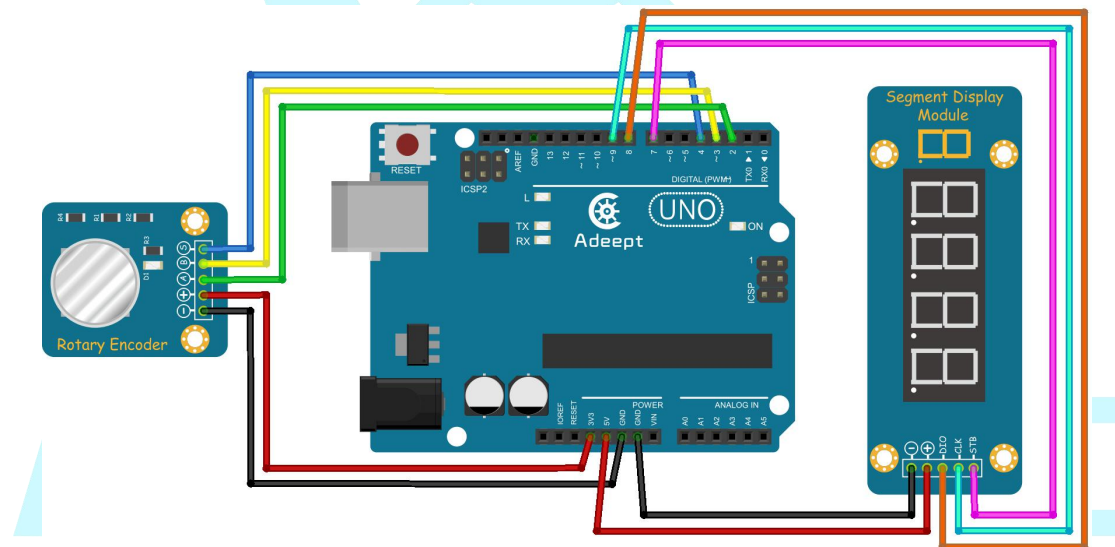
In this lesson, we will rotate the rotary knob to change the data displayed on the segment display.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Segment Display Module
- 1 * Rotary Encoder Module
- 1 * USB Cable
- 2 * 5-Pin Wires
- 1 * Breadboard

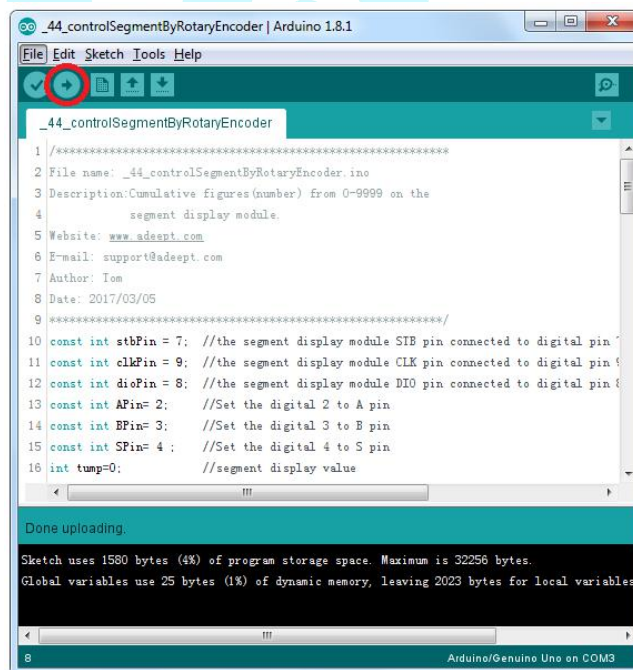
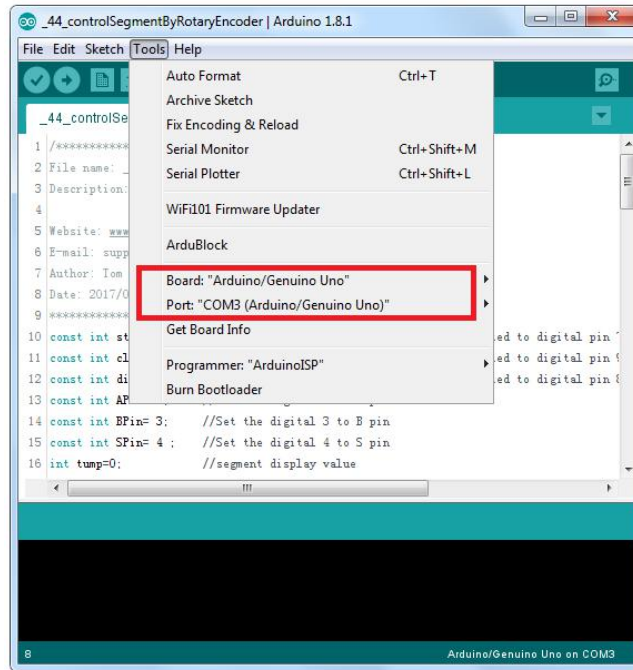
Experimental Procedures

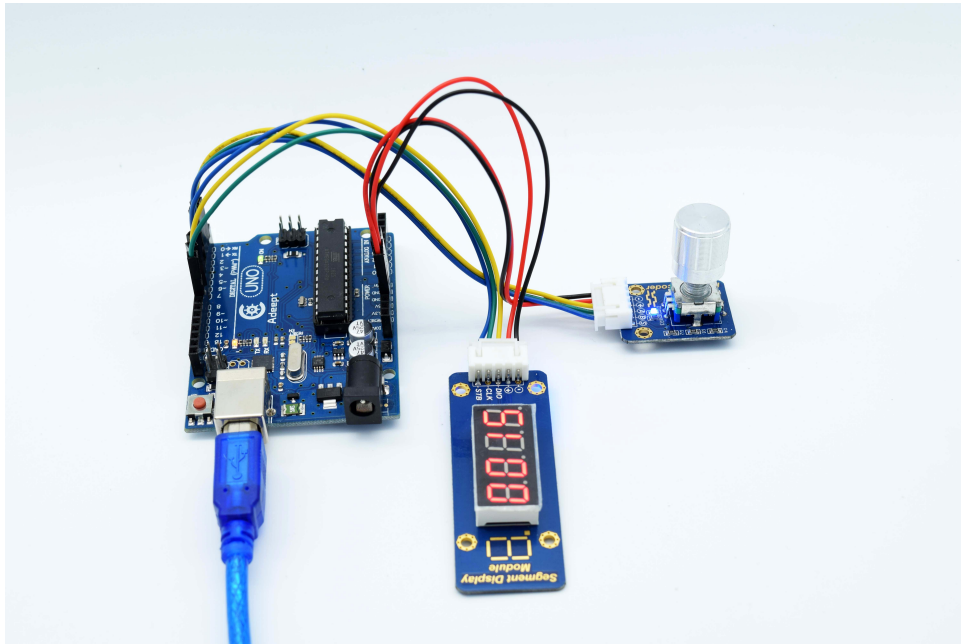
Step 1: Build the circuit



Step 2: Program `_44_controlSegmentByRotaryEncoder.ino`

Step 3: Compile and download the sketch to the UNO R3 board.





Turning the knob clockwise, the number displayed on the segment display will be increased, otherwise, it will be decreased. Press the knob, the data will return to zero.

Adeept

Lesson 45 A Simple Temperature & Humidity Monitoring and Alarm System(1)

Introduction

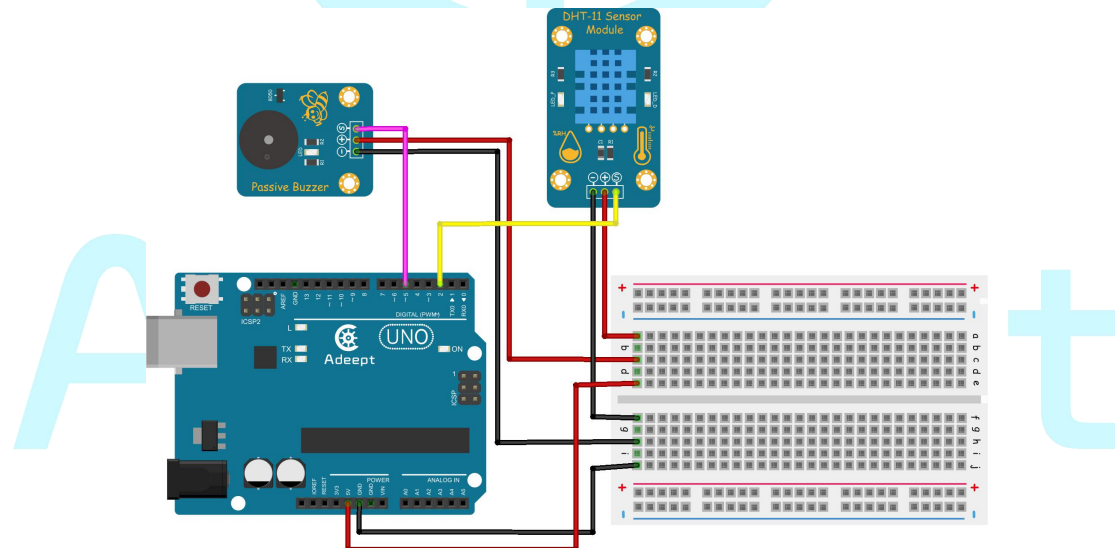
In this lesson, we will build a temperature & humidity monitoring and alarm system based on a passive buzzer and a DHT-11 sensor module.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * DHT-11 Sensor Module
- 1 * Passive Buzzer Module
- 1 * USB Cable
- 2 * 3-Pin Wires
- 1 * Breadboard
- 2 * Hookup Wire Set

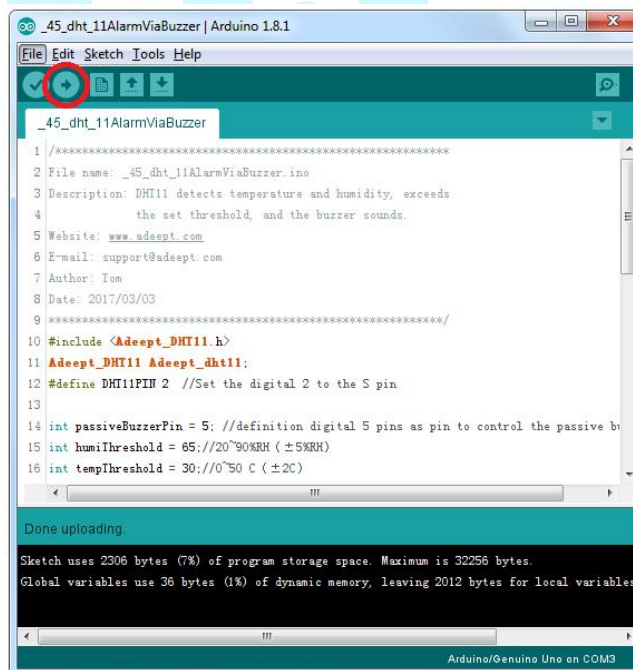
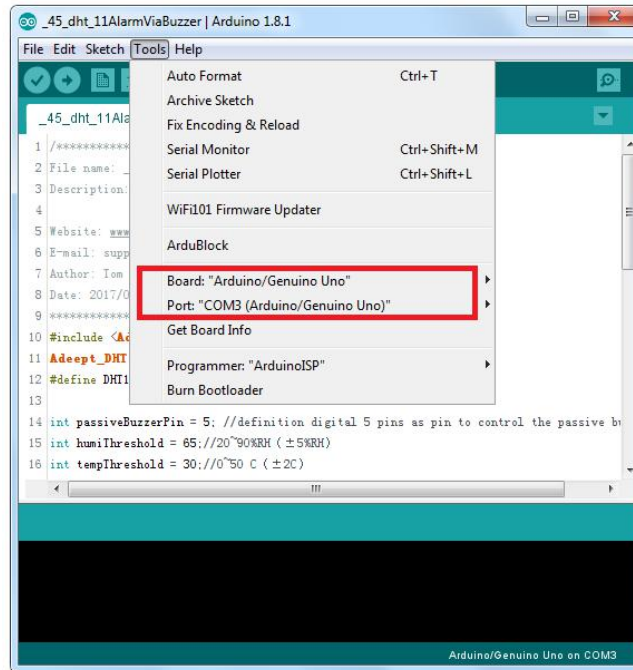
Experimental Procedures

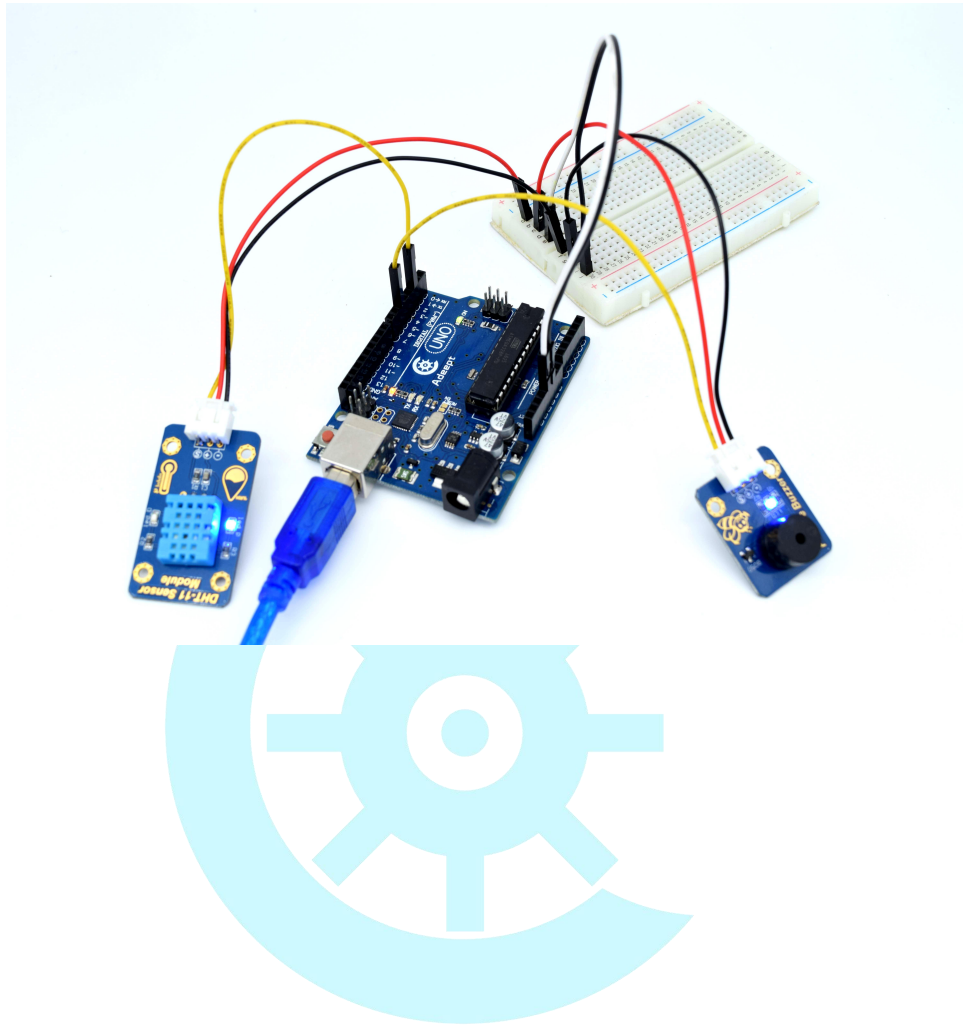
Step 1: Build the circuit



Step 2: Program `_45_dht_11AlarmViaBuzzer.ino`

Step 3: Compile and download the sketch to the UNO R3 board.





Adeept

Lesson 46 A Simple Temperature & Humidity Monitoring and Alarm System(2)

Introduction

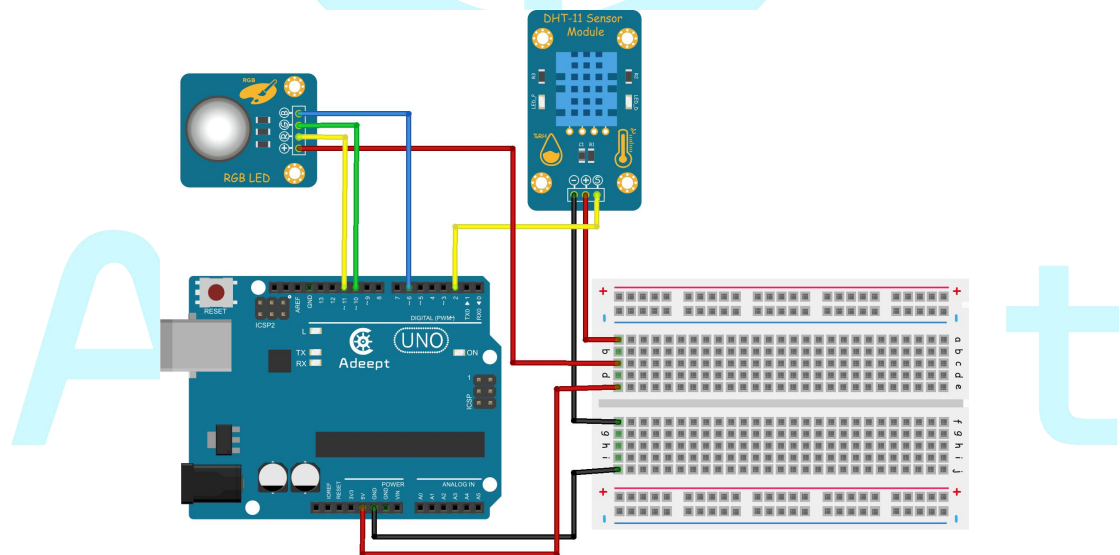
In this lesson, we use the RGB module and DHT-11 sensor module to do a temperature and humidity alarm system.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * DHT-11 Sensor Module
- 1 * RGB Module
- 1 * USB Cable
- 1 * 3-Pin Wires
- 1 * 4-Pin Wires
- 1 * Breadboard
- 2 * Hookup Wire Set

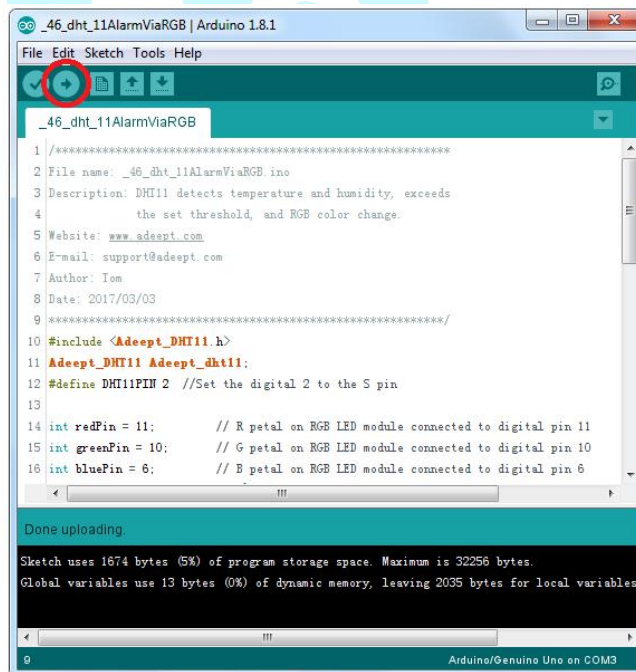
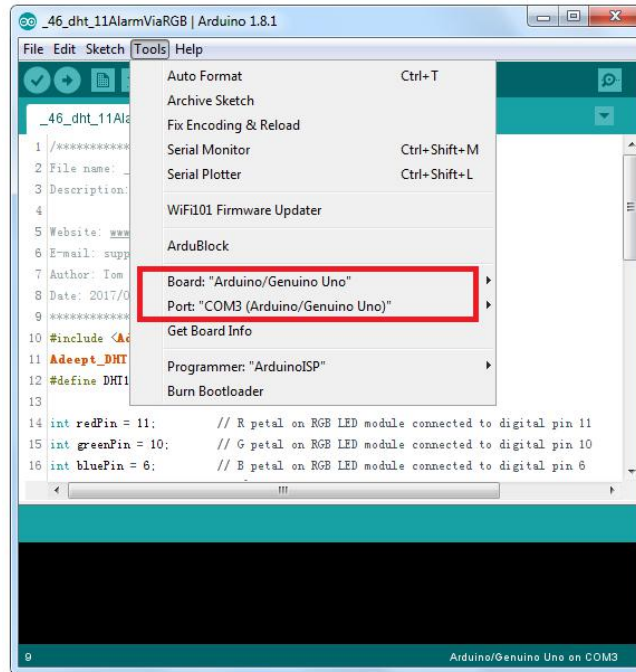
Experimental Procedures

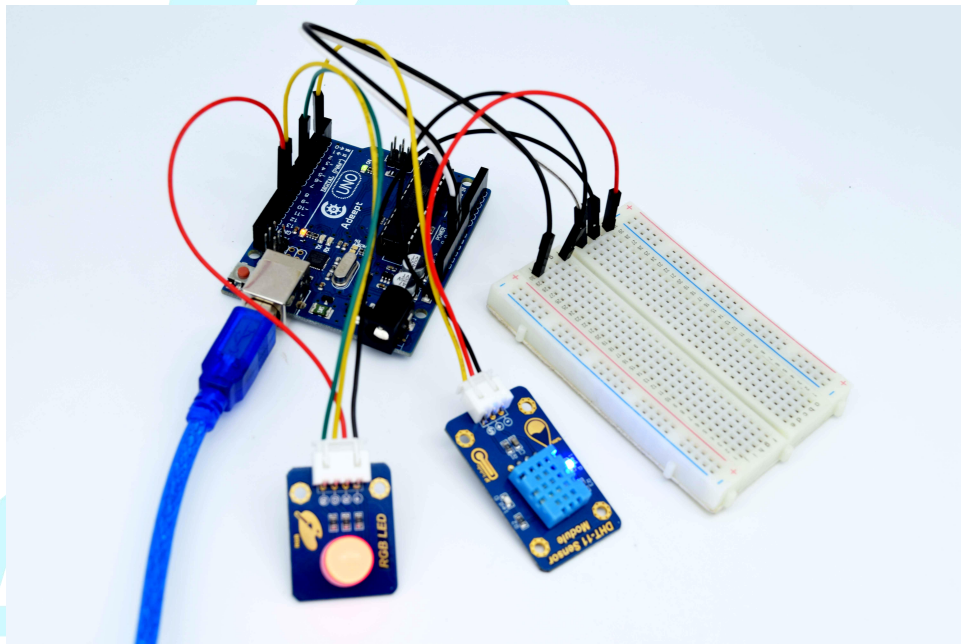
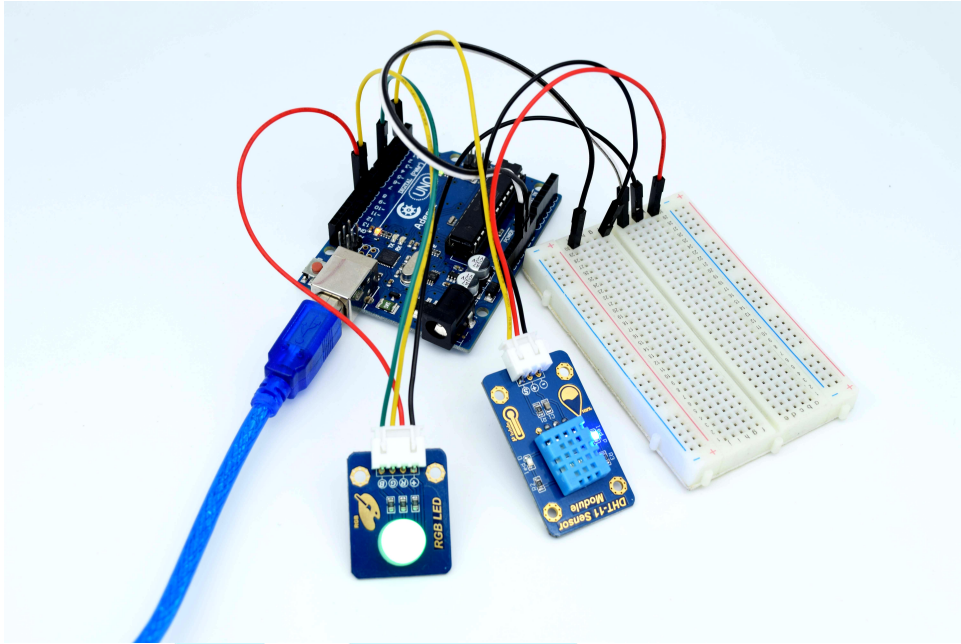
Step 1: Build the circuit



Step 2: Program `_46_dht_11AlarmViaRGB.ino`

Step 3: Compile and download the sketch to the UNO R3 board.





Introduction

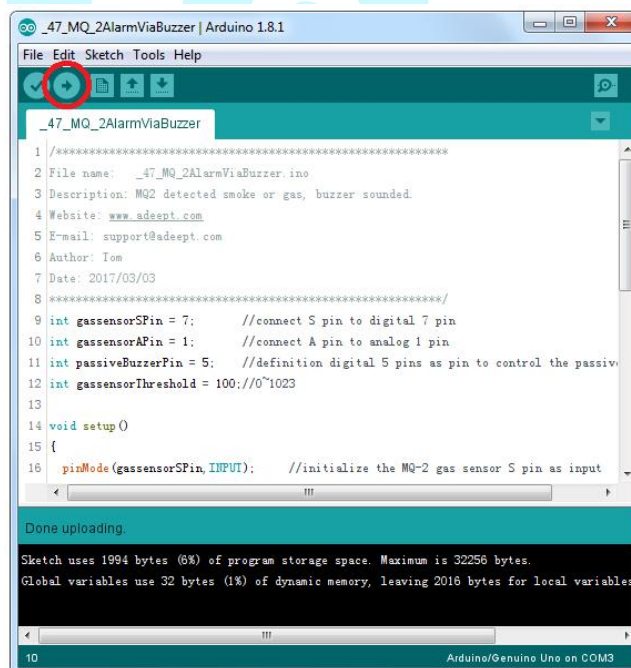
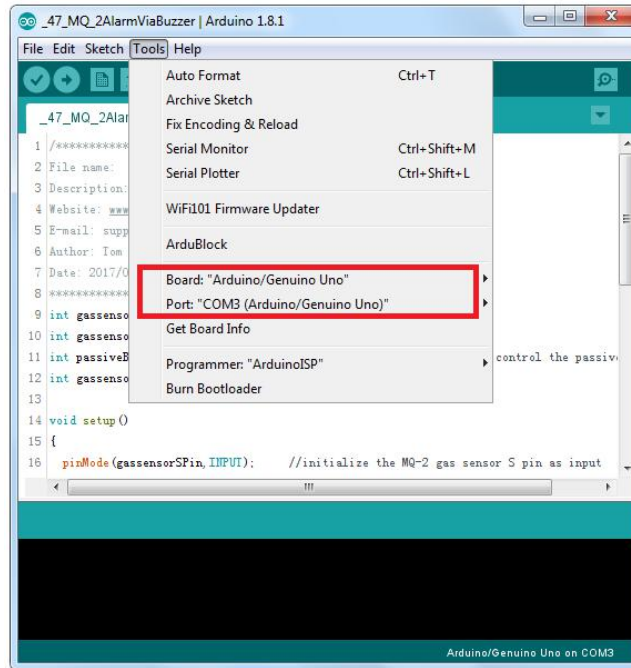
Components

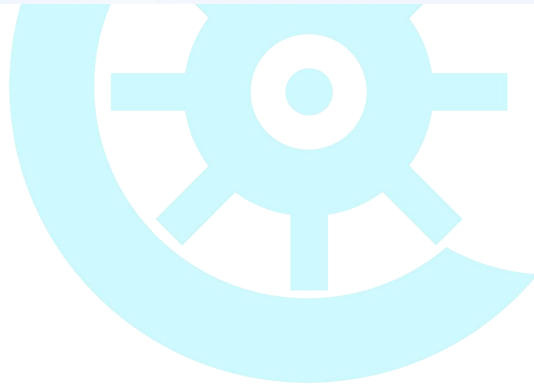
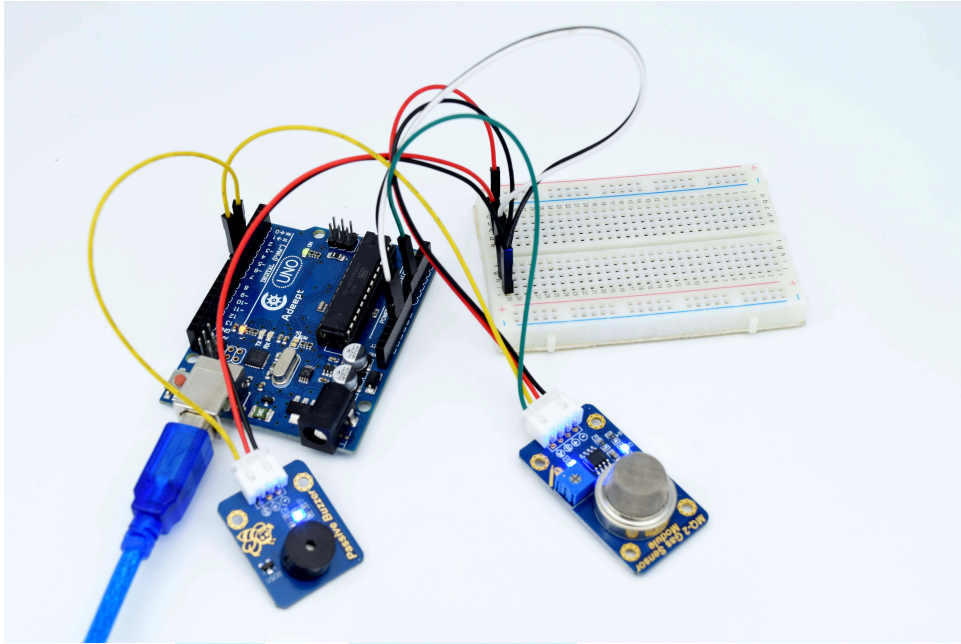
- ## Experimental Procedures

The diagram illustrates the hardware setup for the Arduino Uno R3. The components and their connections are as follows:

- Arduino Uno R3:** The central microcontroller board with pins labeled GND, VCC, and digital/analog pins.
- Breadboard:** Contains a 5V regulator, a 10k pull-down resistor, and a 100k pull-up resistor. The 5V regulator is connected to the Arduino's 5V pin. The 10k resistor is connected between the sensor module's GND and the Arduino's GND. The 100k resistor is connected between the sensor module's VCC and the Arduino's 5V pin.
- MQ-2 Gas Sensor Module:** A blue module with a circular sensor. It is connected to the breadboard and the Arduino. The sensor module's GND is connected to the Arduino's GND. The sensor module's VCC is connected to the Arduino's 5V pin. The sensor module's AO pin is connected to the Arduino's digital pin 2.
- Passive Buzzer:** A small component with two pins. It is connected to the sensor module's AO pin and the Arduino's digital pin 2.

Step 3: Compile and download the sketch to the UNO R3 board.





Adeept

Lesson 48 A Simple Flammable Gases Monitoring and Alarm System(2)

Introduction

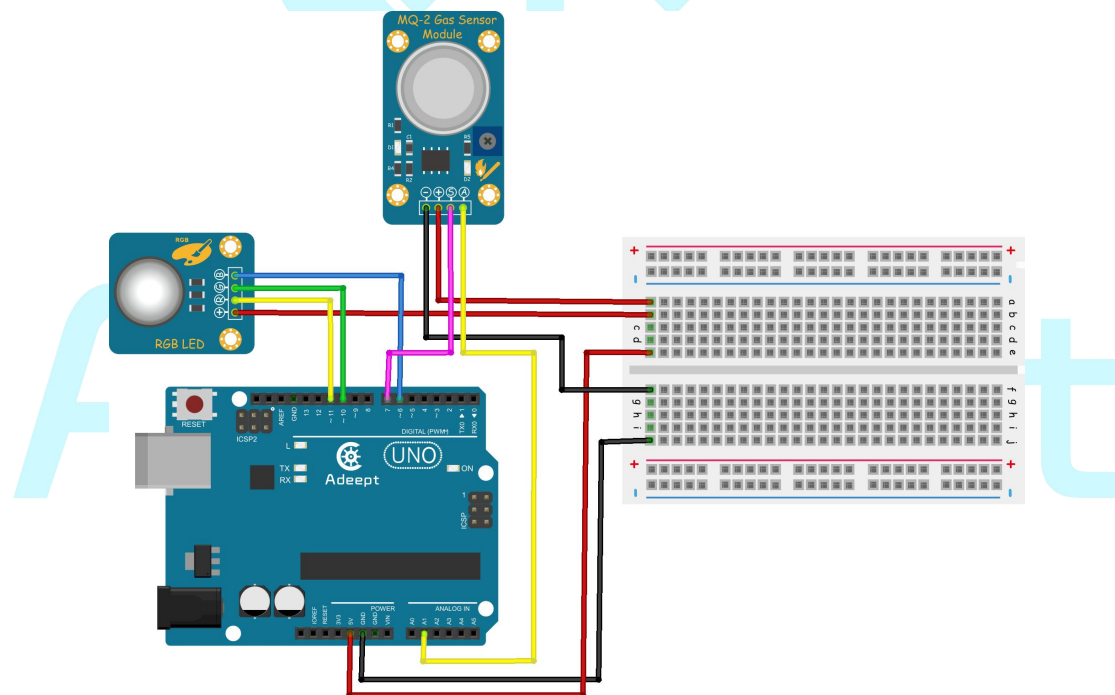
In this lesson, we use the RGB module and MQ-2 gas sensor module to do a MQ-2 Flammable gas alarm system.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * MQ-2 Gas Sensor Module
- 1 * RGB Module
- 1 * USB Cable
- 2 * 4-Pin Wires
- 1 * Breadboard
- 2 * Hookup Wire Set

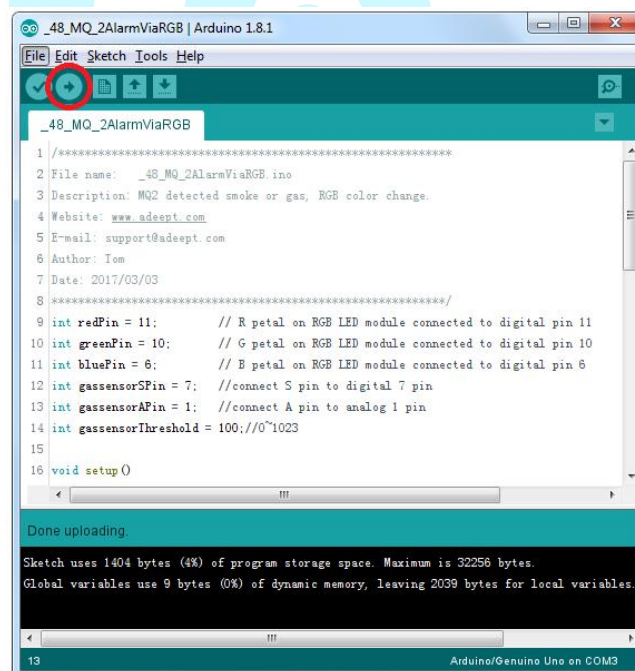
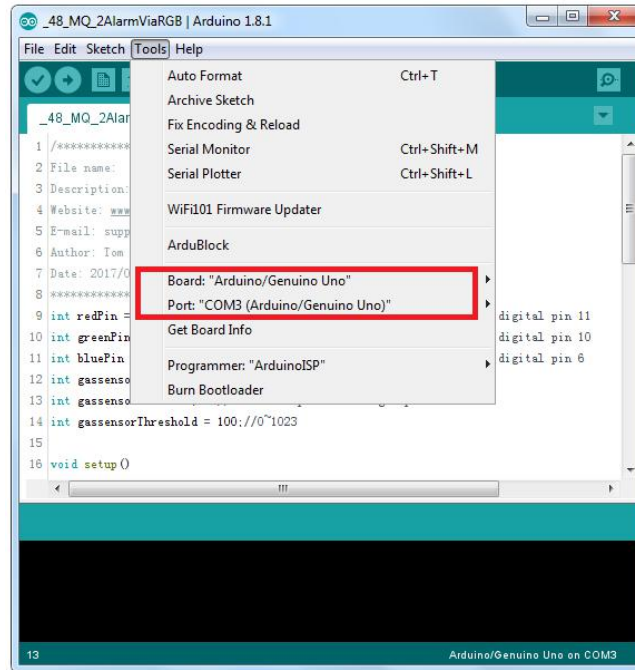
Experimental Procedures

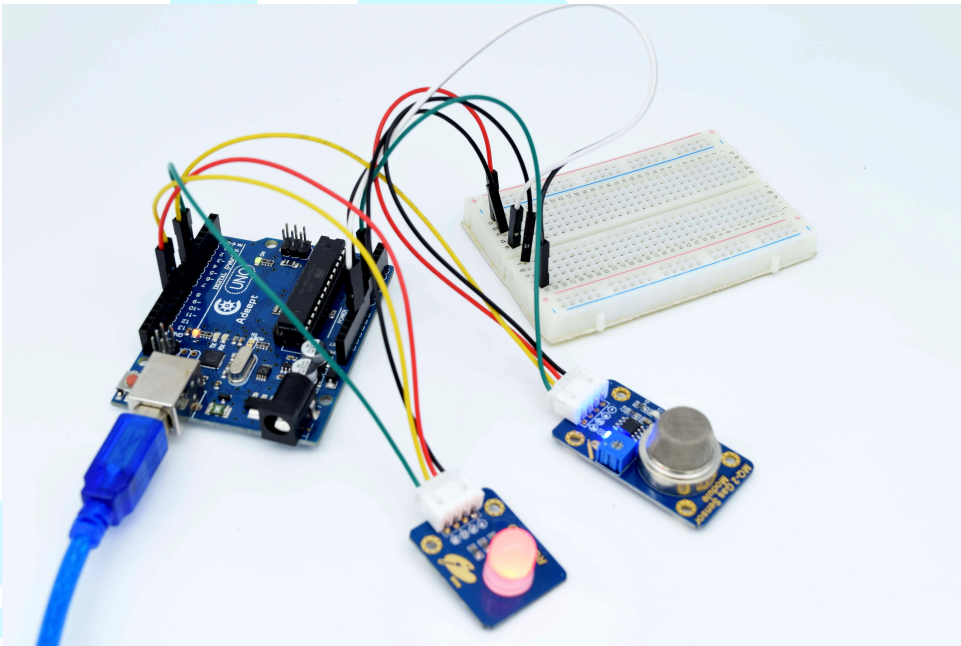
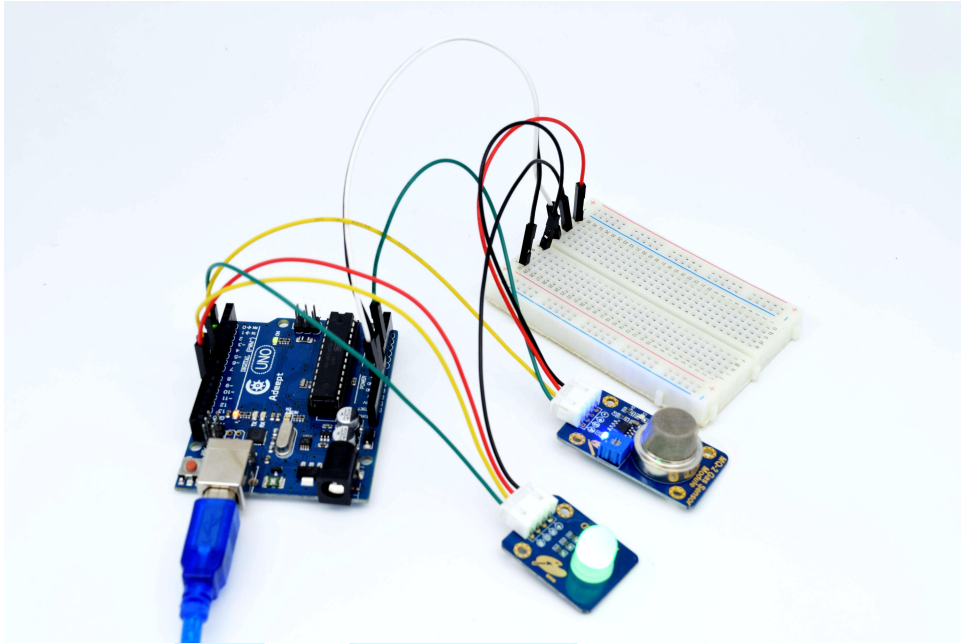
Step 1: Build the circuit



Step 2: Program `_48_MQ_2AlarmViaRGB.ino`

Step 3: Compile and download the sketch to the UNO R3 board.





Lesson 49 A Simple Clock

Introduction

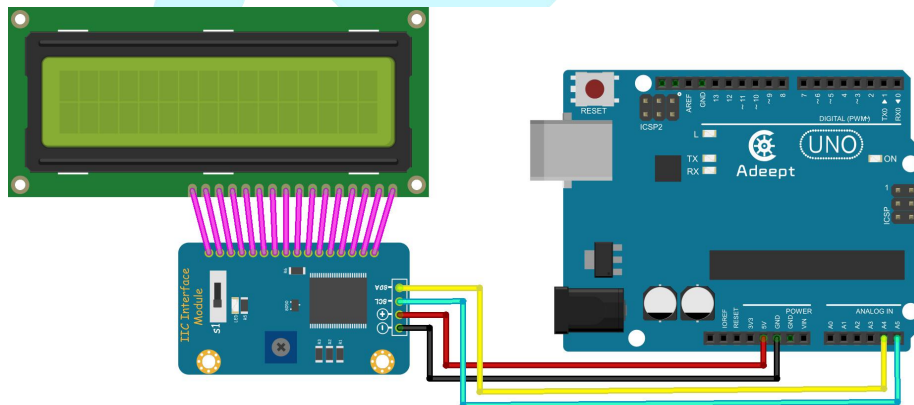
In this lesson, we will build a simple clock based on a LCD1602 and Arduino UNO.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * LCD1602 Module
- 1 * IIC Interface Module
- 1 * USB Cable
- 1 * 4-Pin Wires

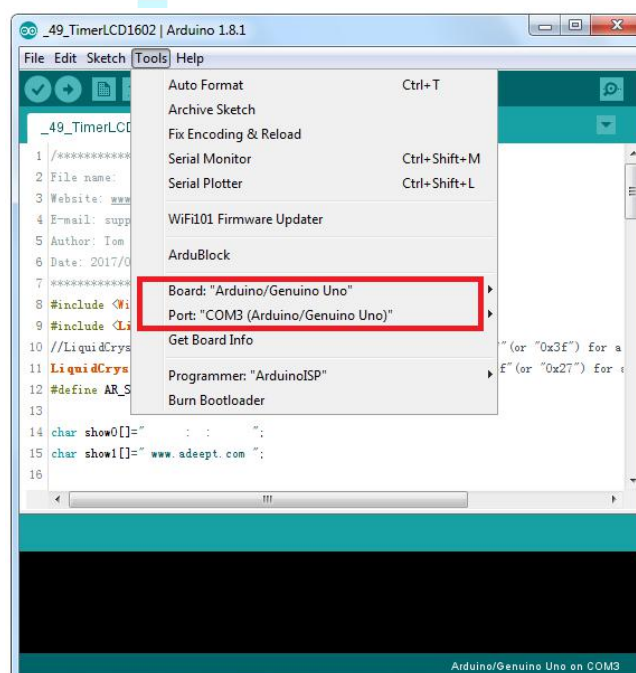
Experimental Procedures

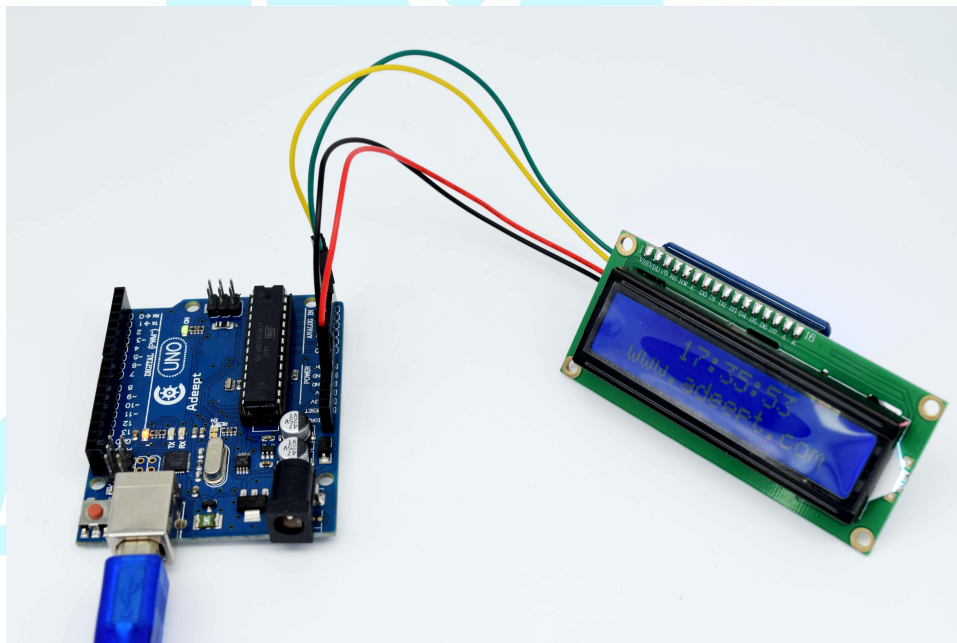
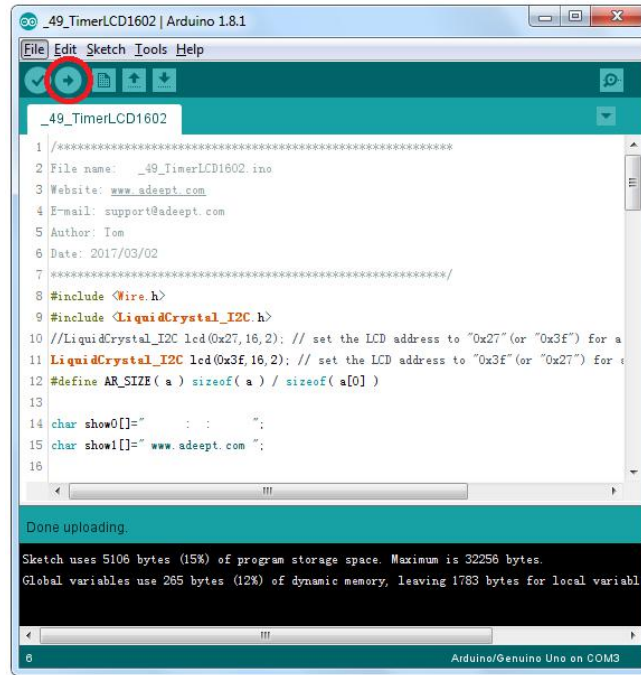
Step 1: Build the circuit



Step 2: Program `_49_TimerLCD1602.ino`

Step 3: Compile and download the sketch to the UNO R3 board.





Lesson 50 Arduino Interacts with Processing(LED Module)

Introduction

Starting from this lesson, we will start experimenting with the interaction between Arduino and Processing. We will create a simple human-computer interaction interface based on Processing, which contains four different colors of the virtual button(LED shape), when the mouse pointer points to a virtual button, the corresponding LED will be turned on.

Components

- 1 * Adeept Arduino UNO R3 Board
- 4 * LED Module
- 1 * USB Cable
- 4 * 3-Pin Wires
- 1 * Breadboard
- 1 * Hookup Wire Set

Principle

Processing key function:

setup()

The setup() function is run once, when the program starts. It's used to define initial environment properties such as screen size and to load media such as images and fonts as the program starts. There can only be one setup() function for each program and it shouldn't be called again after its initial execution.

If the sketch is a different dimension than the default, the size() function or fullscreen() function must be the first line in setup().

Note: Variables declared within setup() are not accessible within other functions, including draw().

draw()

Called directly after setup(), the draw() function continuously executes the lines of code contained inside its block until the program is stopped or noLoop() is called. draw() is called automatically and should never be called explicitly. All Processing programs update the screen at the end of draw(), never earlier.

To stop the code inside of draw() from running continuously, use noLoop(), redraw() and loop(). If noLoop() is used to stop the code in draw() from running, then redraw() will cause the code inside draw() to run a single time, and loop() will cause the code inside draw() to resume running continuously.

The number of times draw() executes in each second may be controlled with the frameRate() function.

It is common to call background() near the beginning of the draw() loop to clear the contents of the window, as shown in the first example above. Since pixels drawn to the

window are cumulative, omitting `background()` may result in unintended results.

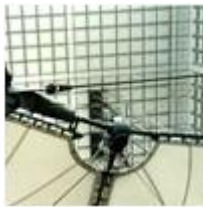
There can only be one `draw()` function for each sketch, and `draw()` must exist if you want the code to run continuously, or to process events such as `mousePressed()`. Sometimes, you might have an empty call to `draw()` in your program, as shown in the second example above.

The Serial library reads and writes data to and from external devices one byte at a time. It allows two computers to send and receive data. This library has the flexibility to communicate with custom microcontroller devices and to use them as the input or output to Processing programs. The serial port is a nine pin I/O port that exists on many PCs and can be emulated through USB.

Name

PImage

Examples



```
PImage photo;
void setup() {
  size(100, 100);
  photo = loadImage("laDefense.jpg");
}
void draw() {
  image(photo, 0, 0);
}
```

Description Datatype for storing images. Processing can display .gif, .jpg, .tga, and .png images. Images may be displayed in 2D and 3D space. Before an image is used, it must be loaded with the `loadImage()` function. The `PImage` class contains fields for the width and height of the image, as well as an array called `pixels[]` that contains the values for every pixel in the image. The methods described below allow easy access to the image's pixels and alpha channel and simplify the process of compositing.

Before using the `pixels[]` array, be sure to use the `loadPixels()` method on the image to make sure that the pixel data is properly loaded.

To create a new image, use the `createImage()` function. Do not use the syntax `new PImage()`.

Fields

`pixels[]` Array containing the color of every pixel in the image

`width` Image width

`height` Image height

Methods

loadPixels() Loads the pixel data for the image into its pixels[] array
updatePixels() Updates the image with the data in its pixels[] array
resize() Changes the size of an image to a new width and height
get() Reads the color of any pixel or grabs a rectangle of pixels
set() writes a color to any pixel or writes an image into another
mask() Masks part of an image with another image as an alpha channel
filter() Converts the image to grayscale or black and white
copy() Copies the entire image
blend() Copies a pixel or rectangle of pixels using different blending modes
save() Saves the image to a TIFF, TARGA, PNG, or JPEG file

Constructor

PImage(width, height, format, factor)

Name

PFont

Examples



```
PFont font;  
// The font must be located in the sketch's  
// "data" directory to load successfully  
font = createFont("LetterGothicStd.ttf", 32);  
textFont(font);  
text("word", 10, 50);
```

Description PFont is the font class for Processing. To create a font to use with Processing, select "Create Font..." from the Tools menu. This will create a font in the format Processing requires and also adds it to the current sketch's data directory. Processing displays fonts using the .vlw font format, which uses images for each letter, rather than defining them through vector data. The loadFont() function constructs a new font and textFont() makes a font active. The list() method creates a list of the fonts installed on the computer, which is useful information to use with the createFont() function for dynamically converting fonts into a format to use with Processing.

To create a new font dynamically, use the createFont() function. Do not use the syntax new PFont().

Methods

list() Gets a list of the fonts installed on the system

Name

size()

Examples


```
size(200, 100);
background(153);
line(0, 0, width, height);
void setup() {
  size(320, 240);
}
void draw() {
  background(153);
  line(0, 0, width, height);
}
```

```
size(150, 200, P3D); // Specify P3D renderer
background(153);
// With P3D, we can use z (depth) values...
line(0, 0, 0, width, height, -100);
line(width, 0, 0, width, height, -100);
line(0, height, 0, width, height, -100);
//...and 3D-specific functions, like box()
translate(width/2, height/2);
rotateX(PI/6);
rotateY(PI/6);
box(35);
```

Description Defines the dimension of the display window width and height in units of pixels. In a program that has the `setup()` function, the `size()` function must be the first line of code inside `setup()`.

The built-in variables `width` and `height` are set by the parameters passed to this function. For example, running `size(640, 480)` will assign 640 to the `width` variable and 480 to the `height` variable. If `size()` is not used, the window will be given a default size of 100 x 100 pixels.

The `size()` function can only be used once inside a sketch, and it cannot be used for resizing.

As of Processing 3, to run a sketch at the full dimensions of a screen, use the `fullScreen()` function, rather than the older way of using `size(displayWidth, displayHeight)`.

The maximum width and height is limited by your operating system, and is usually the width and height of your actual screen. On some machines it may simply be the number of pixels on your current screen, meaning that a screen of 800 x 600 could support `size(1600, 300)`, since that is the same number of pixels. This varies widely, so you'll have to try different rendering modes and sizes until you get what you're looking for. If you need something larger, use `createGraphics` to create a non-visible drawing surface.

The minimum width and height is around 100 pixels in each direction. This is the smallest that is supported across Windows, macOS, and Linux. We enforce the minimum size so that sketches will run identically on different machines.

The renderer parameter selects which rendering engine to use. For example, if you will be drawing 3D shapes, use P3D. In addition to the default renderer, other renderers are:

P2D (Processing 2D): 2D graphics renderer that makes use of OpenGL-compatible graphics hardware.

P3D (Processing 3D): 3D graphics renderer that makes use of OpenGL-compatible graphics hardware.

FX2D (JavaFX 2D): A 2D renderer that uses JavaFX, which may be faster for some applications, but has some compatibility quirks.

PDF: The PDF renderer draws 2D graphics directly to an Acrobat PDF file. This produces excellent results when you need vector shapes for high-resolution output or printing. You must first use Import Library → PDF to make use of the library. More information can be found in the PDF library reference.

As of Processing 3.0, to use variables as the parameters to size() function, place the size() function within the settings() function (instead of setup()). There is more information about this on the settings() reference page.

Syntax

size(width, height)

size(width, height, renderer)

Parameters

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

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

Returns void

Name

noStroke()

Examples



```
noStroke();
```

```
rect(30, 20, 55, 55);
```

Description Disables drawing the stroke (outline). If both noStroke() and noFill() are called, nothing will be drawn to the screen.

Syntax

noStroke()

Returns void

Name

smooth()

Examples

```
void setup() {  
  size(100, 100);  
  smooth(2);  
  noStroke();  
}  
  
void draw() {  
  background(0);  
  ellipse(30, 48, 36, 36);  
  ellipse(70, 48, 36, 36);  
}
```

```
void setup() {  
  fullScreen(P2D, SPAN);  
  smooth(4);  
}  
  
void draw() {  
  background(0);  
  ellipse(x, height/2, height/4, height/4);  
  x += 1;  
}
```

Description Draws all geometry with smooth (anti-aliased) edges. This behavior is the default, so smooth() only needs to be used when a program needs to set the smoothing in a different way. The level parameter increases the level of smoothness. This is the level of over sampling applied to the graphics buffer.

With the P2D and P3D renderers, smooth(2) is the default, this is called "2x anti-aliasing." The code smooth(4) is used for 4x anti-aliasing and smooth(8) is specified for 8x anti-aliasing. The maximum anti-aliasing level is determined by the hardware of the machine that is running the software, so smooth(4) and smooth(8) will not work with every computer.

The default renderer uses smooth(3) by default. This is bicubic smoothing. The other option for the default renderer is smooth(2), which is bilinear smoothing.

With Processing 3.0, smooth() is different than before. It was common to use smooth() and noSmooth() to turn on and off antialiasing within a sketch. Now, because of how the

software has changed, `smooth()` can only be set once within a sketch. It can be used either at the top of a sketch without a `setup()`, or after the `size()` function when used in a sketch with `setup()`. The `noSmooth()` function also follows the same rules.

Syntax

`smooth(level)`

Parameters

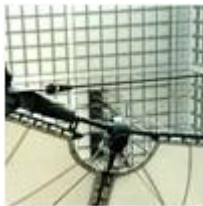
level int: either 2, 3, 4, or 8 depending on the renderer

Returns void

Name

`loadImage()`

Examples



```
PImage img;  
img = loadImage("laDefense.jpg");  
image(img, 0, 0);
```



```
PImage img;  
void setup() {  
  img = loadImage("laDefense.jpg");  
}  
void draw() {  
  image(img, 0, 0);  
}
```



```
PImage webImg;  
void setup() {  
  String url = "https://processing.org/img/processing-web.png";  
  // Load image from a web server  
  webImg = loadImage(url, "png");  
}
```

```
}  
void draw() {  
  background(0);  
  image(webImg, 0, 0);  
}
```

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.

In most cases, load all images in setup() to preload them at the start of the program. Loading images inside draw() will reduce the speed of a program. Images cannot be loaded outside setup() unless they're inside a function that's called after setup() has already run.

Alternatively, the file maybe be loaded from anywhere on the local computer using an absolute path (something that starts with / on Unix and Linux, or a drive letter on Windows), or the filename parameter can be a URL for a file found on a network.

If the file is not available or an error occurs, null will be returned and an error message will be printed to the console. The error message does not halt the program, however the null value may cause a NullPointerException if your code does not check whether the value returned is null.

The extension parameter is used to determine the image type in cases where the image filename does not end with a proper extension. Specify the extension as the second parameter to loadImage(), as shown in the third example on this page. Note that CMYK images are not supported.

Depending on the type of error, a PImage object may still be returned, but the width and height of the image will be set to -1. This happens if bad image data is returned or cannot be decoded properly. Sometimes this happens with image URLs that produce a 403 error or that redirect to a password prompt, because loadImage() will attempt to interpret the HTML as image data.

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()

Examples

```
PFont myFont;
void setup() {
  size(200, 200);
  // Uncomment the following two lines to see the available fonts
  //String[] fontList = PFont.list();
  //printArray(fontList);
  myFont = createFont("Georgia", 32);
  textFont(myFont);
  textAlign(CENTER, CENTER);
  text("!@#$$", width/2, height/2);
}
```

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.

This function allows Processing to work with the font natively in the default renderer, so the letters are defined by vector geometry and are rendered quickly. In the P2D and P3D renderers, the function sets the project to render the font as a series of small textures. For instance, when using the default renderer, the actual native version of the font will be employed by the sketch, improving drawing quality and performance. With the P2D and P3D renderers, the bitmapped version will be used to improve speed and appearance, but the results are poor when exporting if the sketch does not include the .otf or .ttf file, and the requested font is not available on the machine running the sketch.

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

background()

Examples



```
background(51);
```



```
background(255, 204, 0);
```



```
PImage img;  
img = loadImage("laDefense.jpg");  
background(img);
```

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 background 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

grayfloat: 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

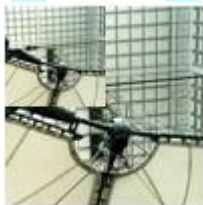
Name

image()

Examples



```
PImage img;  
void setup() {  
  // Images must be in the "data" directory to load correctly  
  img = loadImage("laDefense.jpg");  
}  
void draw() {  
  image(img, 0, 0);  
}
```



```
PImage img;  
void setup() {  
  // Images must be in the "data" directory to load correctly  
  img = loadImage("laDefense.jpg");  
}  
void draw() {
```

```
image(img, 0, 0);  
image(img, 0, 0, width/2, height/2);  
}
```

Description The `image()` function draws an image to the display window. Images must be in the sketch's "data" directory to load correctly. Select "Add file..." from the "Sketch" menu to add the image to the data directory, or just drag the image file onto the sketch window. Processing currently works with GIF, JPEG, and PNG images.

The `img` parameter specifies the image to display and by default the `a` and `b` parameters define the location of its upper-left corner. The image is displayed at its original size unless the `c` and `d` parameters specify a different size. The `imageMode()` function can be used to change the way these parameters draw the image.

The color of an image may be modified with the `tint()` function. This function will maintain transparency for GIF and PNG images.

Syntax

`image(img, a, b)`

`image(img, a, b, c, d)`

Parameters

`img` PImage: the image to display

`a` float: x-coordinate of the image

`b` float: y-coordinate of the image

`c` float: width to display the image

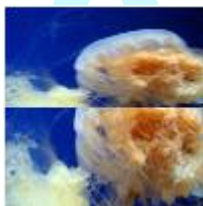
`d` float: height to display the image

Returns void

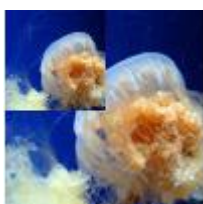
Name

`resize()`

Examples



```
PImage jelly = loadImage("jelly.jpg");  
image(jelly, 0, 0);  
jelly.resize(100, 50);  
image(jelly, 0, 0);
```



```
PImage jelly = loadImage("jelly.jpg");
image(jelly, 0, 0);
jelly.resize(0, 50);
image(jelly, 0, 0);
```

Description Resize the image to a new width and height. To make the image scale proportionally, use 0 as the value for the wide or high parameter. For instance, to make the width of an image 150 pixels, and change the height using the same proportion, use `resize(150, 0)`.

Even though a `PGraphics` is technically a `PImage`, it is not possible to rescale the image data found in a `PGraphics`. (It's simply not possible to do this consistently across renderers: technically infeasible with P3D, or what would it even do with PDF?) If you want to resize `PGraphics` content, first get a copy of its image data using the `get()` method, and call `resize()` on the `PImage` that is returned.

Syntax

`pimg.resize(w, h)`

Parameters

`pimg` `PImage`: any object of type `PImage`

`w` `int`: the resized image width

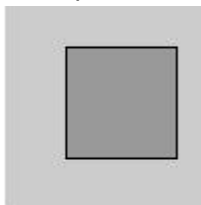
`h` `int`: the resized image height

Returns `void`

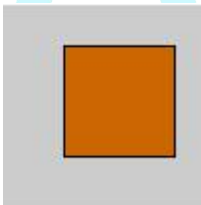
Name

`fill()`

Examples



```
fill(153);
rect(30, 20, 55, 55);
```



```
fill(204, 102, 0);
rect(30, 20, 55, 55);
```

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.

To change the color of an image or a texture, use tint().

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()

Examples



```
PFont mono;
```

```
// The font "andalemo.ttf" must be located in the
```

```
// current sketch's "data" directory to load successfully
```

```
mono = loadFont("andalemo.ttf", 32);
```

```
background(0);
```

```
textFont(mono);
```

```
text("word", 12, 60);
```

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 specified, the font size defaults to the original size (the size in which it was created with the "Create Font..." tool) overriding any previous calls to `textFont()` or `textSize()`.

When fonts are rendered as an image texture (as is the case with the P2D and P3D renderers as well as with `loadFont()` and `vlw` files), 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.

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()`


Examples



```
textSize(32);  
text("word", 10, 30);  
fill(0, 102, 153);  
text("word", 10, 60);  
fill(0, 102, 153, 51);  
text("word", 10, 90);
```



```
size(100, 100, P3D);  
textSize(32);  
fill(0, 102, 153, 204);  
text("word", 12, 45, -30); // Specify a z-axis value  
text("word", 12, 60); // Default depth, no z-value specified
```

The quick
brown fox
jumped
over the
lazy dog.

```
String s = "The quick brown fox jumped over the lazy dog.";  
fill(50);  
text(s, 10, 10, 70, 80); // Text wraps within text box
```

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. (See the third example above.)

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 alphanumeric symbols to be displayed

startint: array index at which to start writing characters

stopint: 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 y-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

numint, or float: the numeric value to be displayed

Returns void

Name

mouseX

Examples

```
void draw() {  
  background(204);  
  line(mouseX, 20, mouseX, 80);  
}
```

Description The system variable mouseX always contains the current horizontal coordinate of the mouse.

Note that Processing can only track the mouse position when the pointer is over the current window. The default value of mouseX is 0, so 0 will be returned until the mouse moves in front of the sketch window. (This typically happens when a sketch is first run.) Once the mouse moves away from the window, mouseX will continue to report its most recent position.

Name

mouseY

Examples

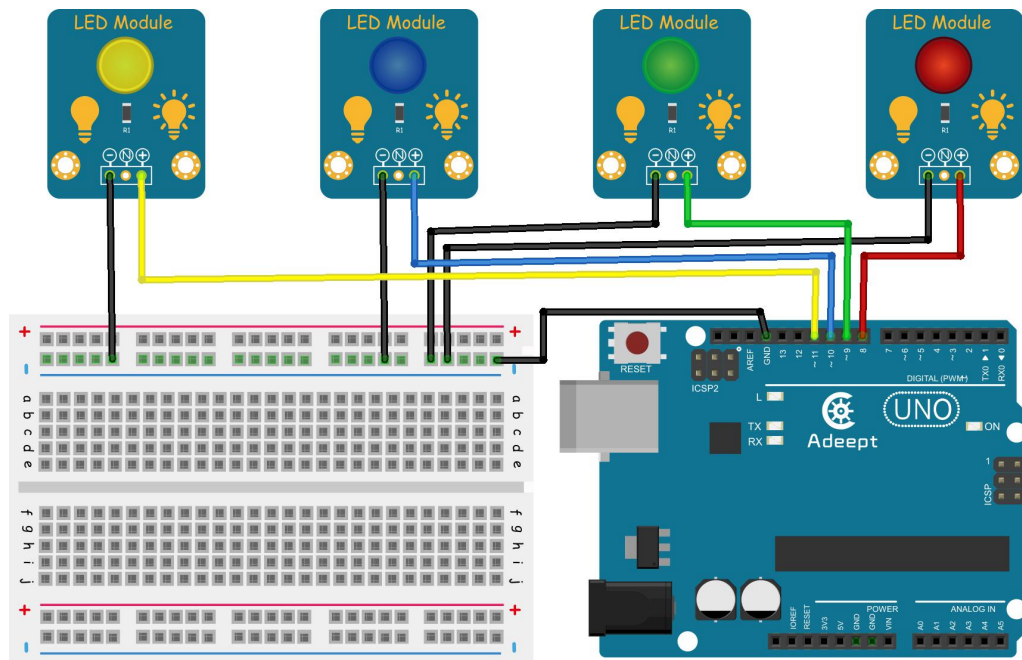
```
void draw() {  
  background(204);  
  line(20, mouseY, 80, mouseY);  
}
```

Description The system variable mouseY always contains the current vertical coordinate of the mouse.

Note that Processing can only track the mouse position when the pointer is over the current window. The default value of mouseY is 0, so 0 will be returned until the mouse moves in front of the sketch window. (This typically happens when a sketch is first run.) Once the mouse moves away from the window, mouseY will continue to report its most recent position.

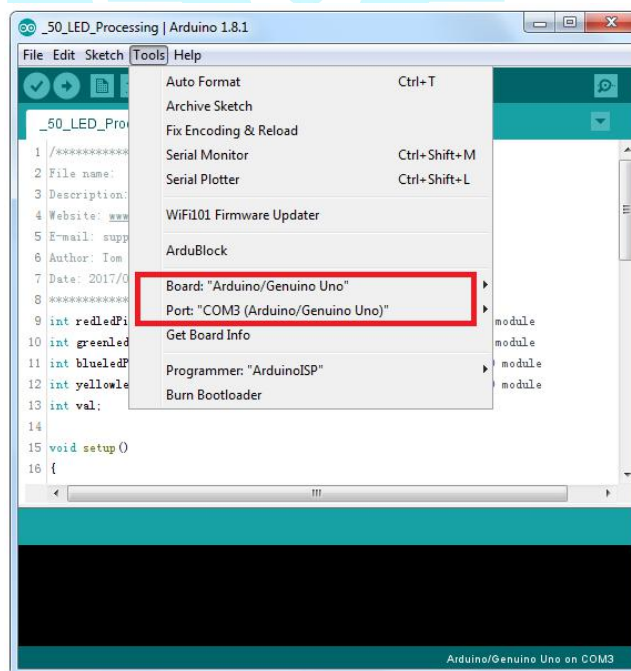
Experimental Procedures

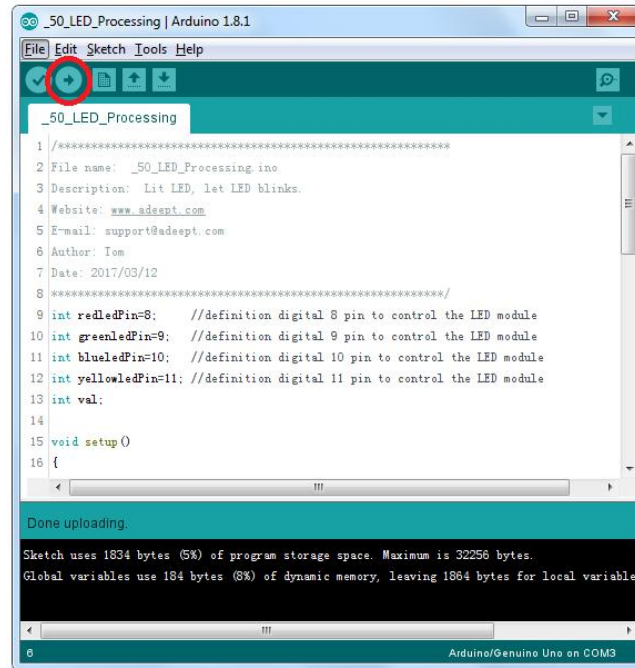
Step 1: Build the circuit

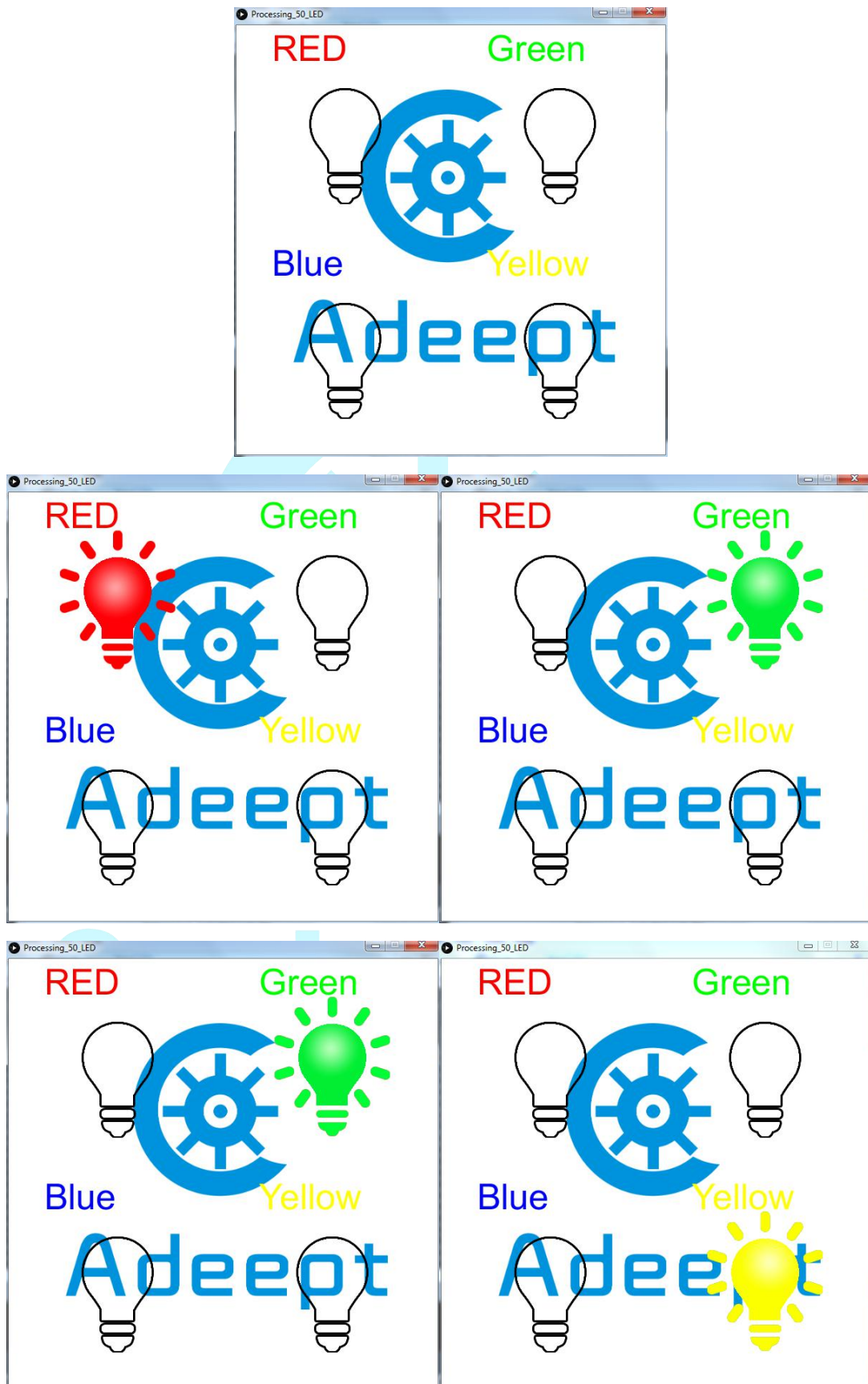


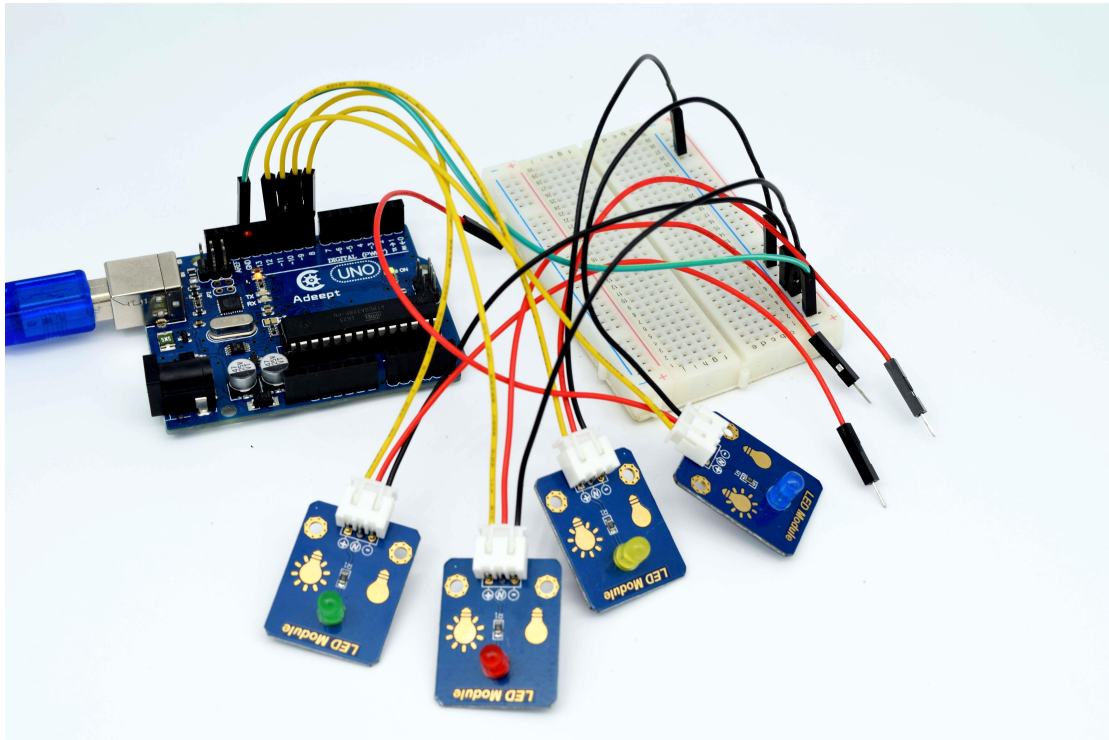
Step 2: Program _50_LED_Processing.ino

Step 3: Compile and download the sketch to the UNO R3 board.









Adeept

Lesson 51 Arduino Interacts with Processing(Button Module)

Introduction

In this lesson, we will create a interface based on Processing, which contains four different colors of virtual buttons. When you press the solid button connected to the Arduino, the corresponding button in the Processing interface will be pressed.

Components

- 1 * Adept Arduino UNO R3 Board
- 4 * Button Module
- 1 * USB Cable
- 4 * 3-Pin Wires
- 1 * Breadboard
- 2 * Hookup Wire Set

Principle

Processing key function:

Name

println()

Examples

```
String s = "The size is ";
int w = 1920;
int h = 1080;
println(s);
println(w, "x", h);

// This program writes to the console:
// The size is
// 1920 x 1080
print("begin- ");
float f = 0.3;
int i = 1024;
print("f is " + f + " and i is " + 1024);
String s = " -end";
println(s);

// This program writes to the console:
// "begin- f is 0.3 and i is 1024 -end"
```

Note that the console is relatively slow. It works well for occasional messages, but does not support high-speed, real-time output (such as at 60 frames per second). It should also be noted, that a `println()` within a for loop can sometimes lock up the program, and cause the sketch to freeze.

Syntax

println()

```
println(what)
```

println(variables)

Parameters

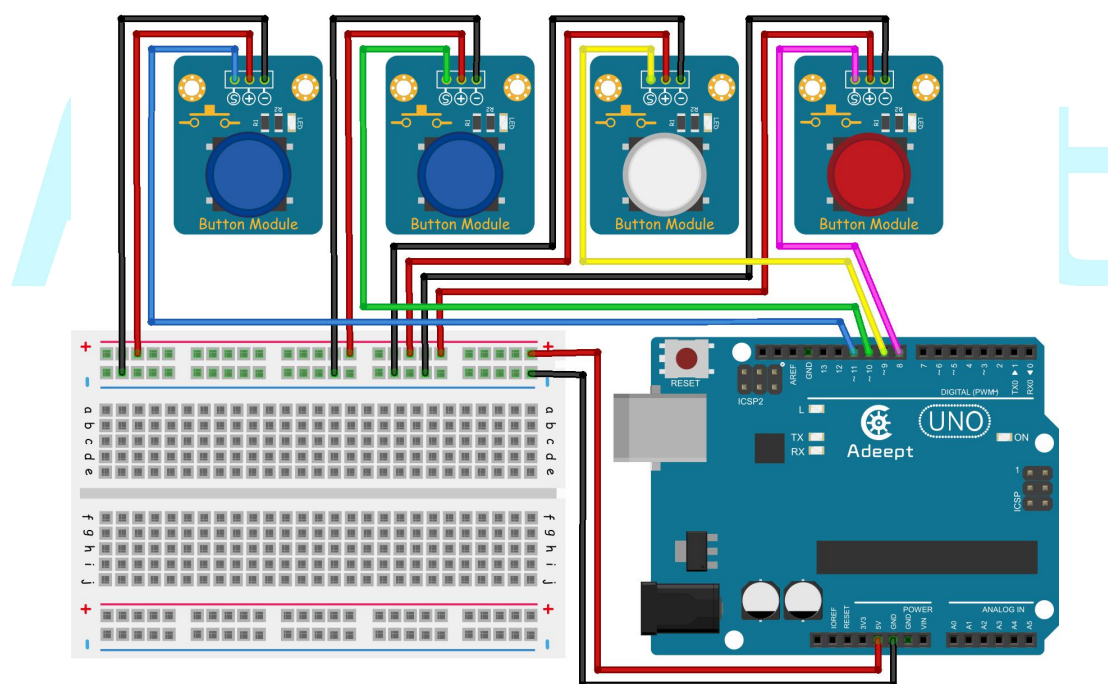
what Object, String, float, char, boolean, or byte: data to print to console

variables Object[]: list of data, separated by commas

Returns void

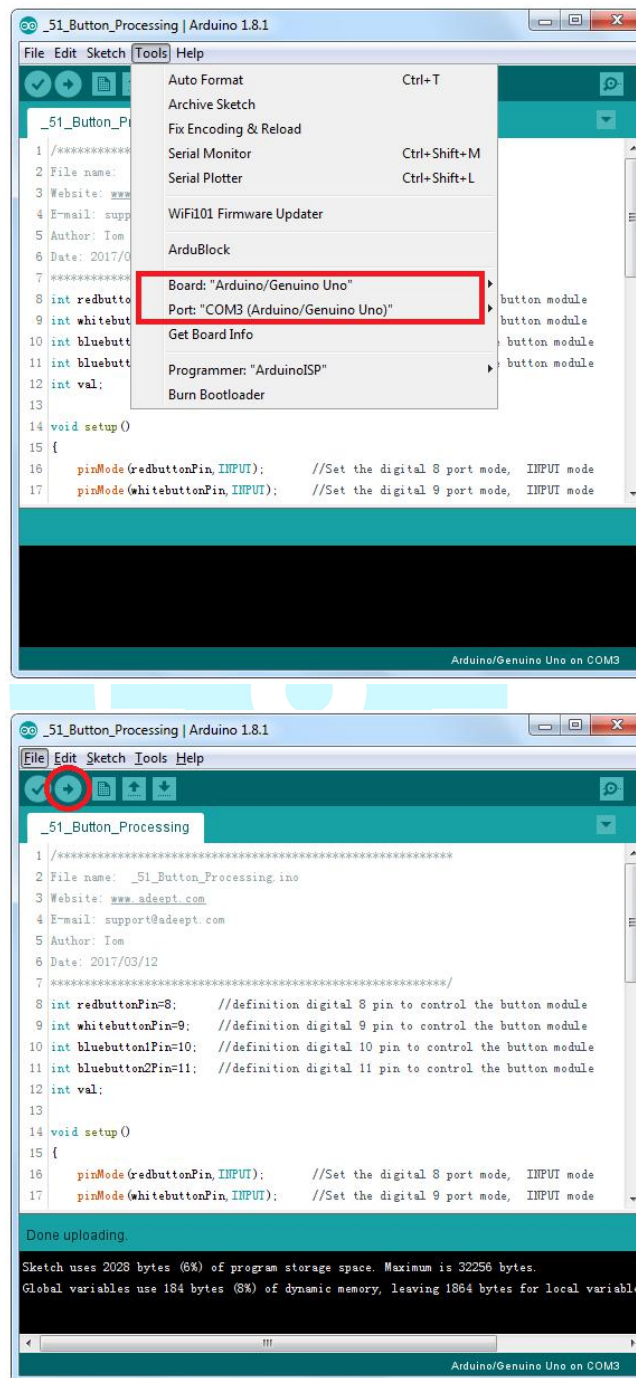
Experimental Procedures

Step 1: Build the circuit

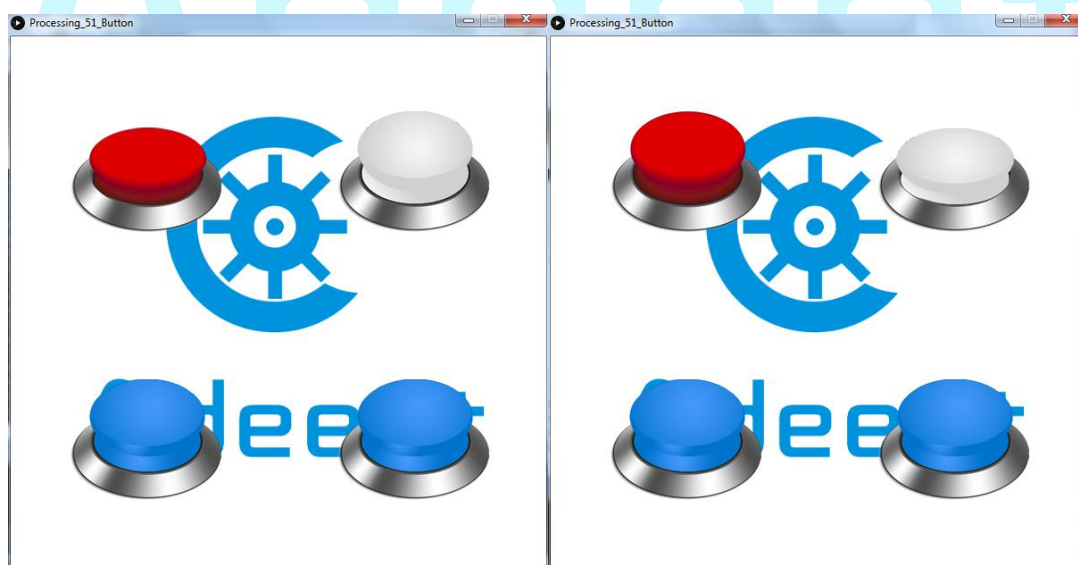
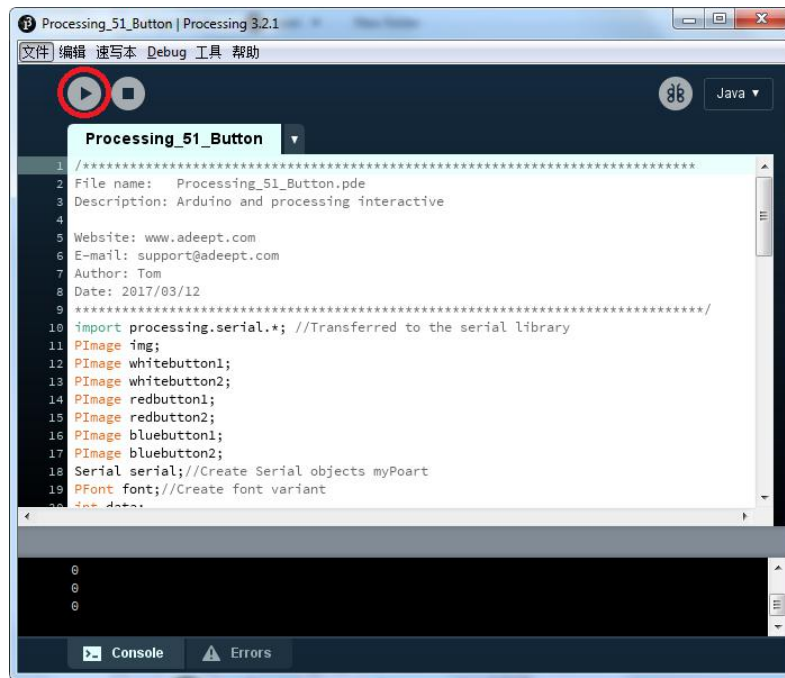


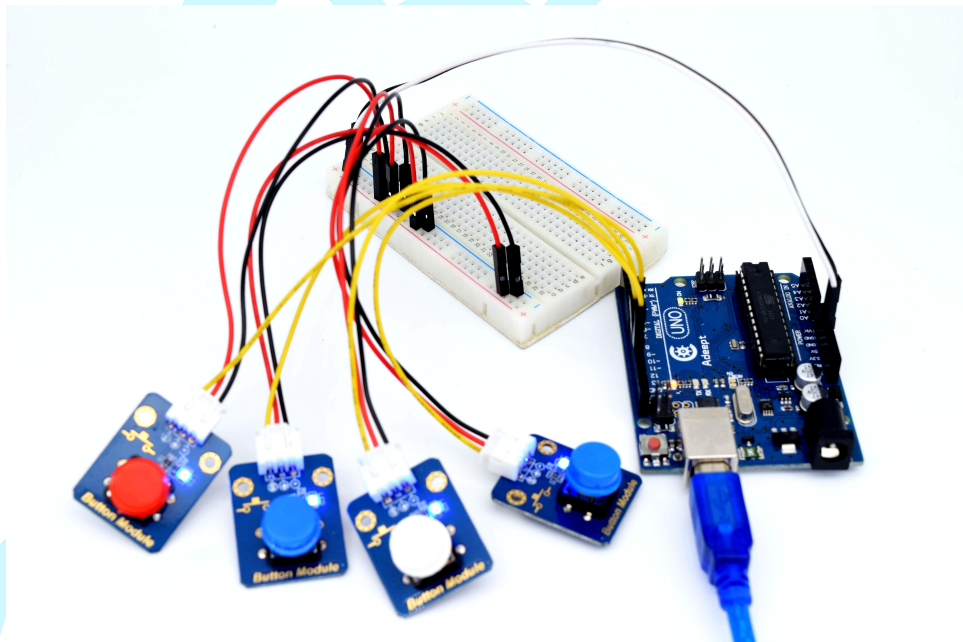
Step 2: Program `51_Button_Processing.ino`

Step 3: Compile and download the sketch to the UNO R3 board.



Step 4: Run the Processing software (Processing_51_Button.pde)





Lesson 52 Arduino Interacts with Processing(RGB Module)

Introduction

In this lesson, we will create a simple interface based on Processing, which contains three different colors of the circle, when the mouse pointer points to a certain color circle, the RGB LED connected to the Arduino will emit the corresponding color of light, when the mouse pointer to the white background, the RGB LED will emit white light.

Components

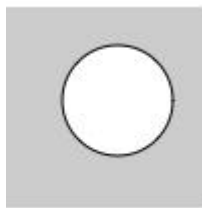
- 1 * Aadept Arduino UNO R3 Board
- 1 * RGB Module
- 1 * USB Cable
- 1 * 4-Pin Wires

Principle

Processing key function:

Name

ellipse()



`ellipse(56, 46, 55, 55);`

Draws an ellipse (oval) to the screen. An ellipse with equal width and height is a circle. By default, the first two parameters set the location, and the third and fourth parameters set the shape's width and height. The origin may be changed with the `ellipseMode()` function.

Syntax

`ellipse(a, b, c, d)`

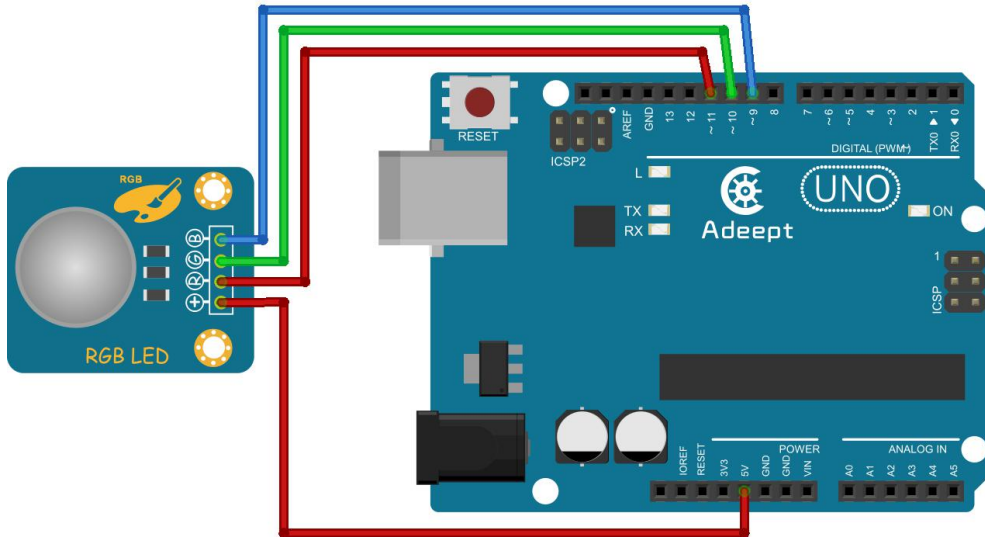
Parameters

- a float: x-coordinate of the ellipse
- b float: y-coordinate of the ellipse
- c float: width of the ellipse by default
- d float: height of the ellipse by default

Returns void

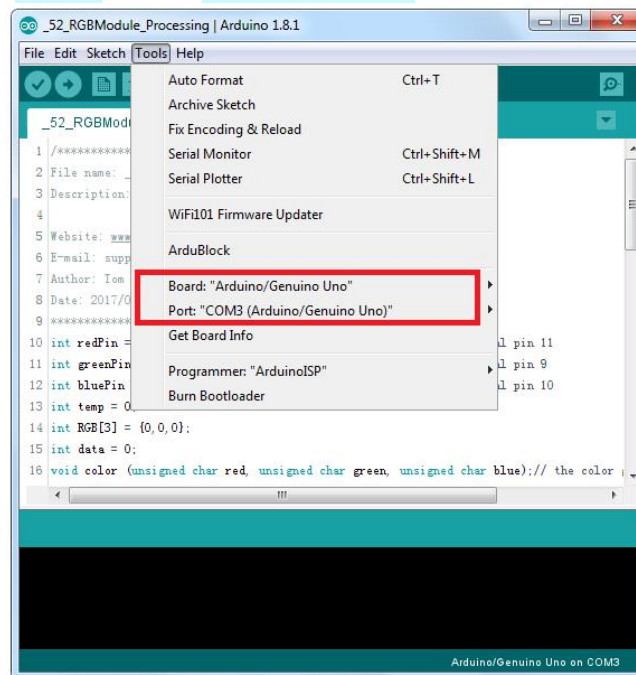
Experimental Procedures

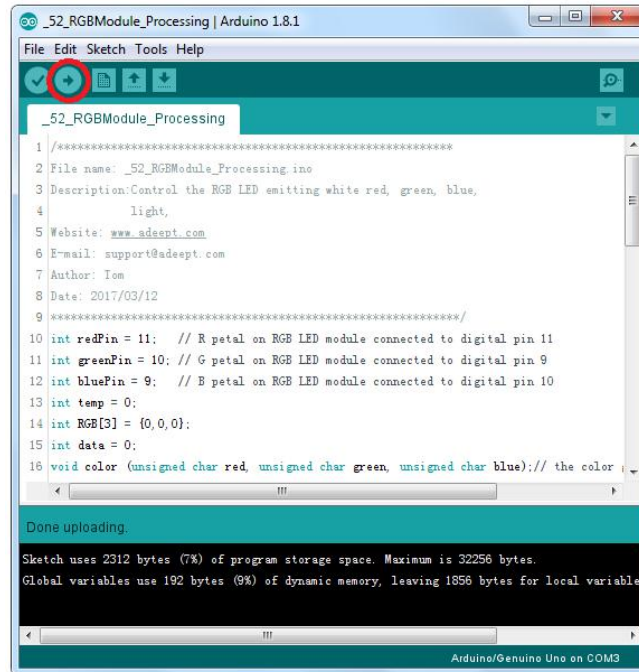
Step 1: Build the circuit



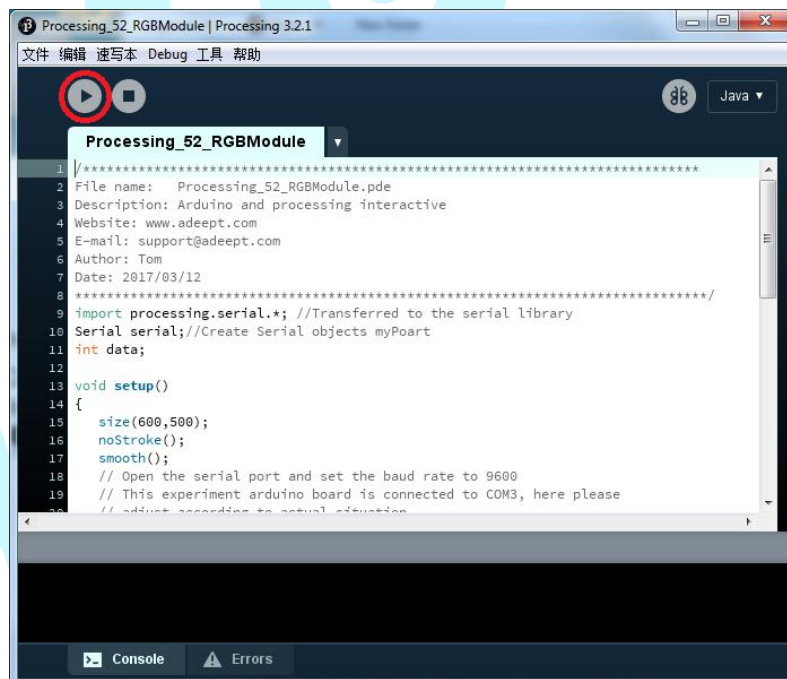
Step 2: Program _52_RGBModule_Processing.ino

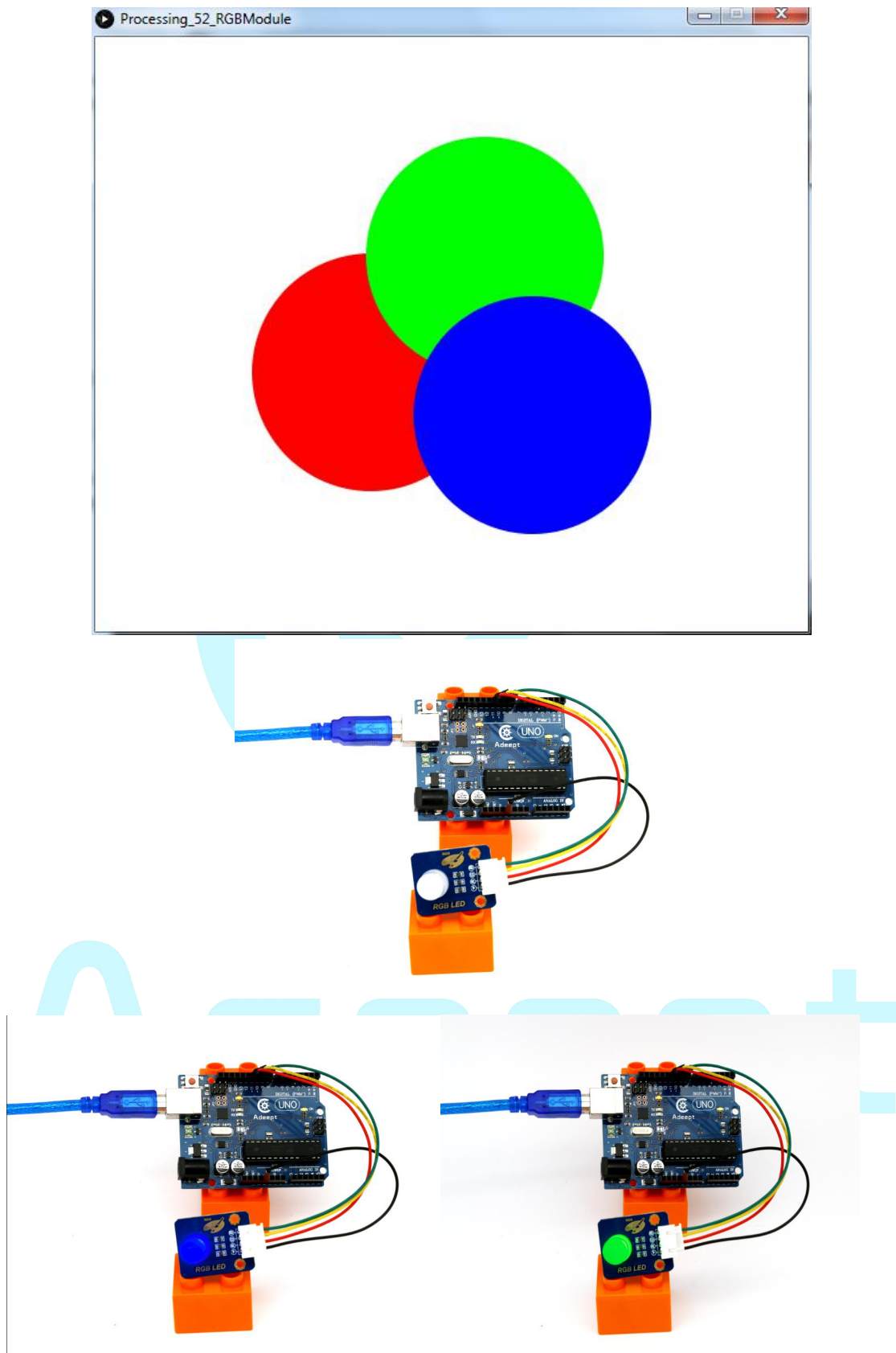
Step 3: Compile and download the sketch to the UNO R3 board.

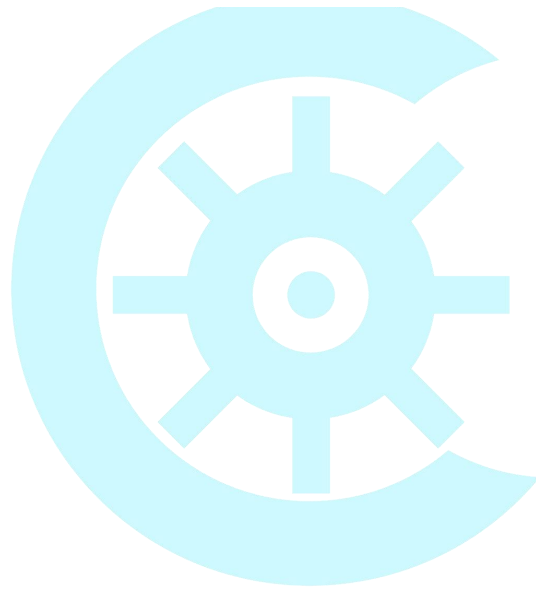
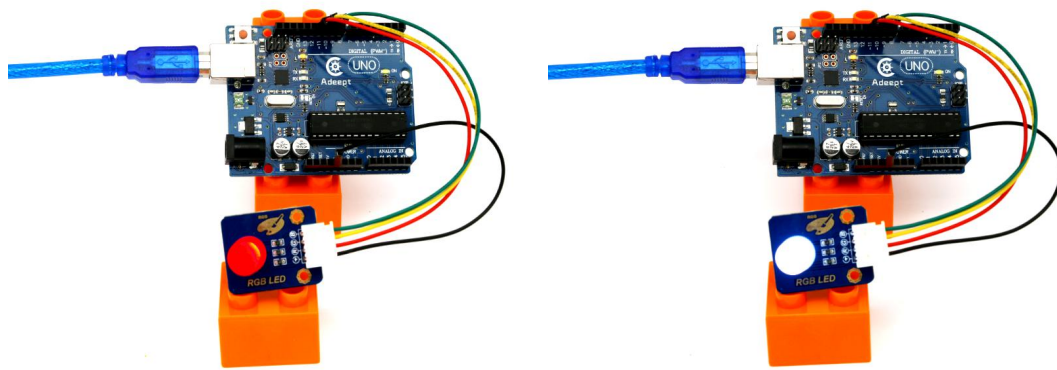




Step 4: Run the Processing software (Processing_52_RGBModule.pde)







Adeept

Lesson 53 Arduino Interacts with Processing(Potentiometer)

Introduction

In this lesson, we will create a simple interface based on Processing, which contains a spiral circle, when rotate the potentiometer knob of the potentiometer module connected to the UNO, the diameter of the circle will be changed.

Components

- 1 * Aadept Arduino UNO R3 Board
- 1 * potentiometer Module
- 1 * USB Cable
- 1 * 3-Pin Wires
- 1 * Breadboard
- 2 * Hookup Wire Set

Principle

Processing key function:

Name: **map()**

Examples

```
size(200, 200);
float value = 25;
float m = map(value, 0, 100, 0, width);
ellipse(m, 200, 10, 10);
float value = 110;
float m = map(value, 0, 100, -20, -10);
println(m); // Prints "-9.0"
void setup() {
  size(200, 200);
  noStroke();
}

void draw() {
  background(204);
  float x1 = map(mouseX, 0, width, 50, 150);
  ellipse(x1, 75, 50, 50);
  float x2 = map(mouseX, 0, width, 0, 200);
  ellipse(x2, 125, 50, 50);
}
```

Description:

Re-maps a number from one range to another.

In the first example above, the number 25 is converted from a value in the range of 0 to 100 into a value that ranges from the left edge of the window (0) to the right edge (width). As shown in the second example, numbers outside of the range are not clamped to the minimum and maximum parameters values, because out-of-range values are often intentional and useful.

Syntax: **map(value, start1, stop1, start2, stop2)**

Parameters:

value float: the incoming value to be converted
start1 float: lower bound of the value's current range
stop1 float: upper bound of the value's current range
start2 float: lower bound of the value's target range
stop2 float: upper bound of the value's target range
Returns: float

Name: **radians()**

Examples

```
float deg = 45.0;  
float rad = radians(deg);  
println(deg + " degrees is " + rad + " radians");
```

Description: Converts a degree measurement to its corresponding value in radians. Radians and degrees are two ways of measuring the same thing. There are 360 degrees in a circle and 2π radians in a circle. For example, $90^\circ = \pi/2 = 1.5707964$. All trigonometric functions in Processing require their parameters to be specified in radians.

Syntax: **radians(degrees)**

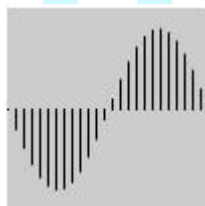
Parameters:

degrees float: degree value to convert to radians

Returns: float

Name: **sin()**

Examples



```
float a = 0.0;  
float inc = TWO_PI/25.0;  
for (int i = 0; i < 100; i=i+4) {  
  line(i, 50, i, 50+sin(a)*40.0);  
  a = a + inc;  
}
```

Description: Calculates the sine of an angle. This function expects the values of the angle parameter to be provided in radians (values from 0 to 6.28). Values are returned in the range -1 to 1.

Syntax: **sin(angle)**

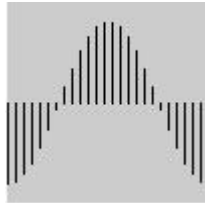
Parameters:

angle float: an angle in radians

Returns: float

Name: **cos()**

Examples



```
float a = 0.0;
float inc = TWO_PI/25.0;
for (int i = 0; i < 25; i++) {
  line(i*4, 50, i*4, 50+cos(a)*40.0);
  a = a + inc;
}
```

Description: Calculates the cosine of an angle. This function expects the values of the angle parameter to be provided in radians (values from 0 to $PI*2$). Values are returned in the range -1 to 1.

Syntax: **cos(angle)**

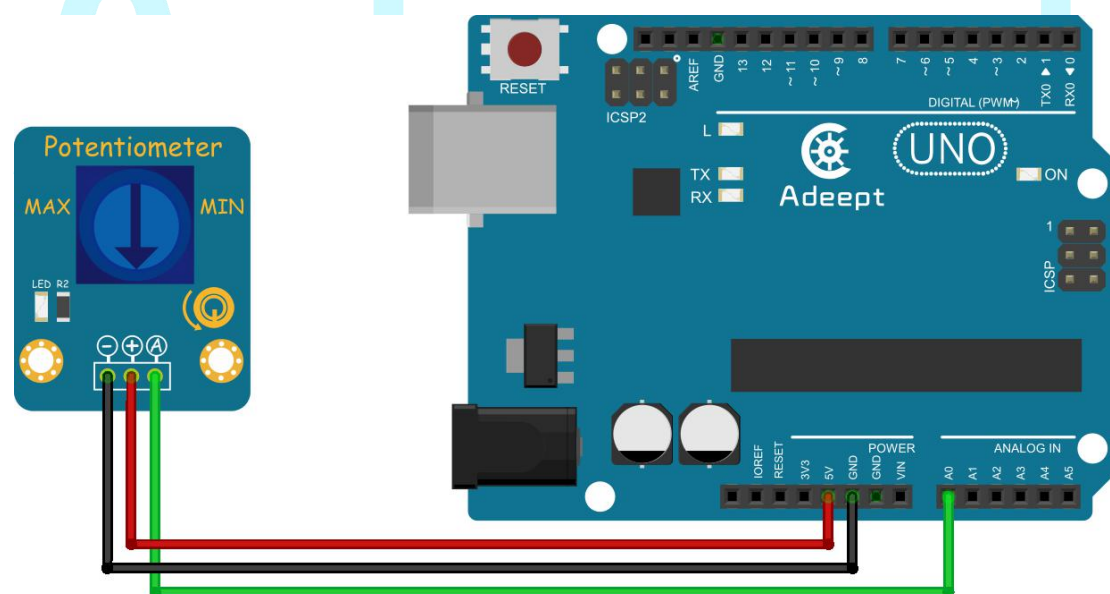
Parameters:

angle float: an angle in radians

Returns: float

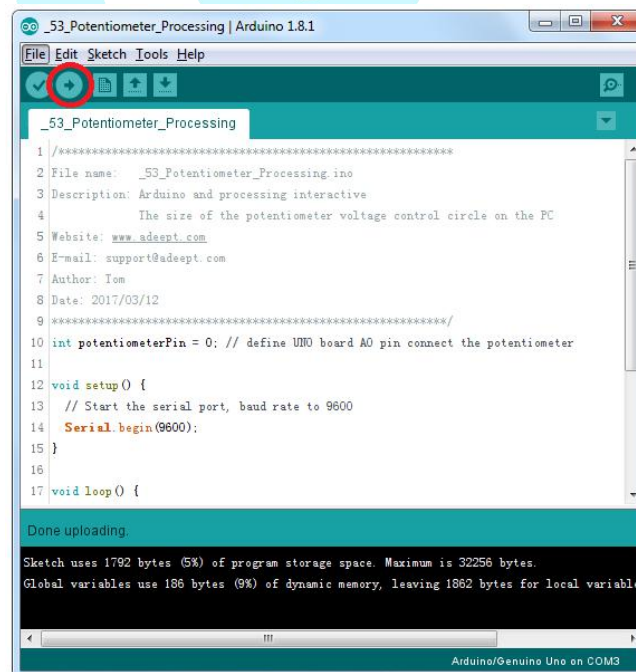
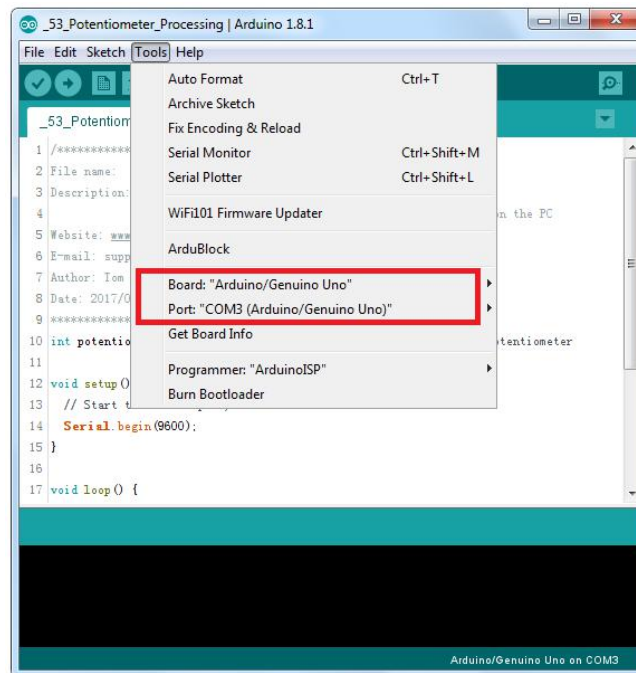
Experimental Procedures

Step 1: Build the circuit

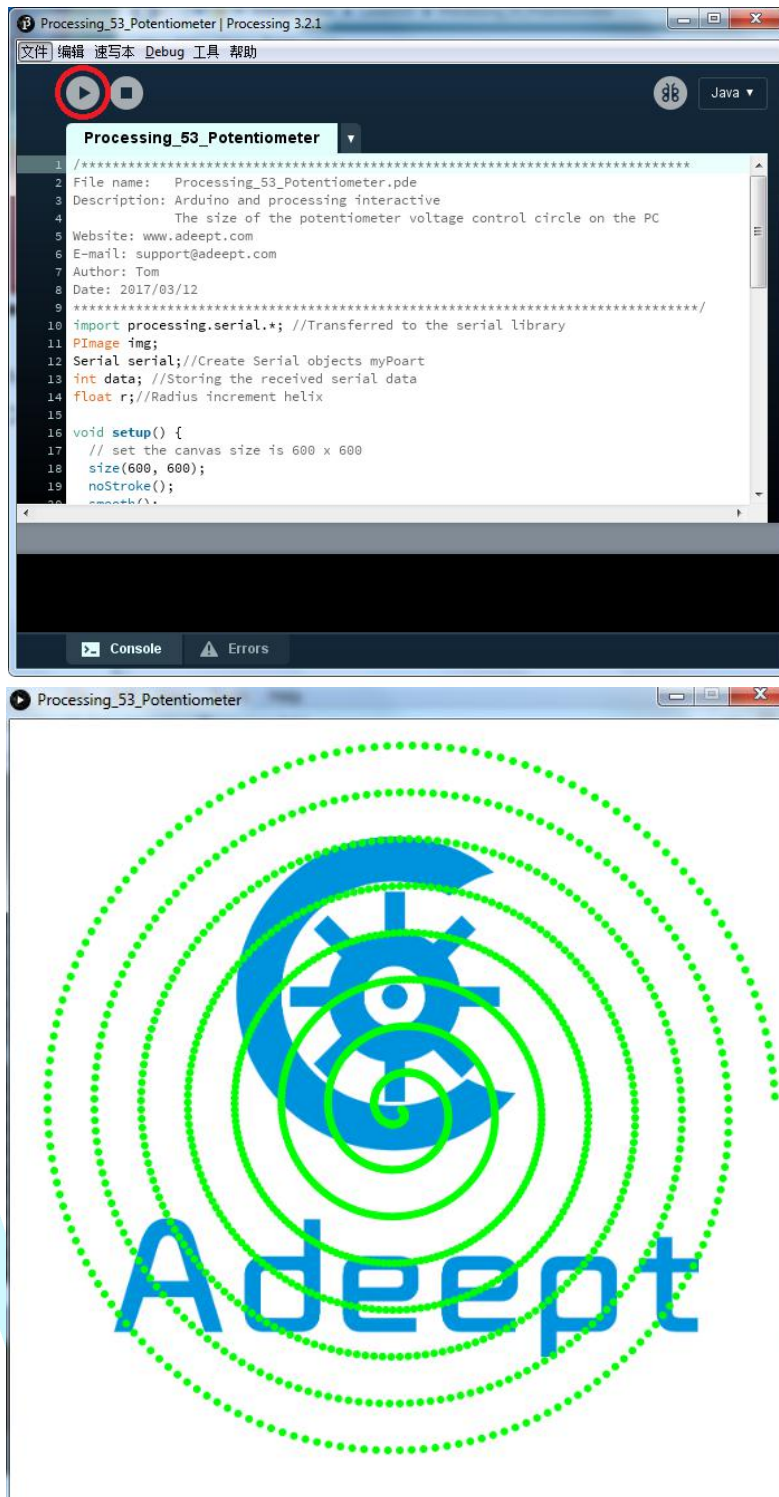


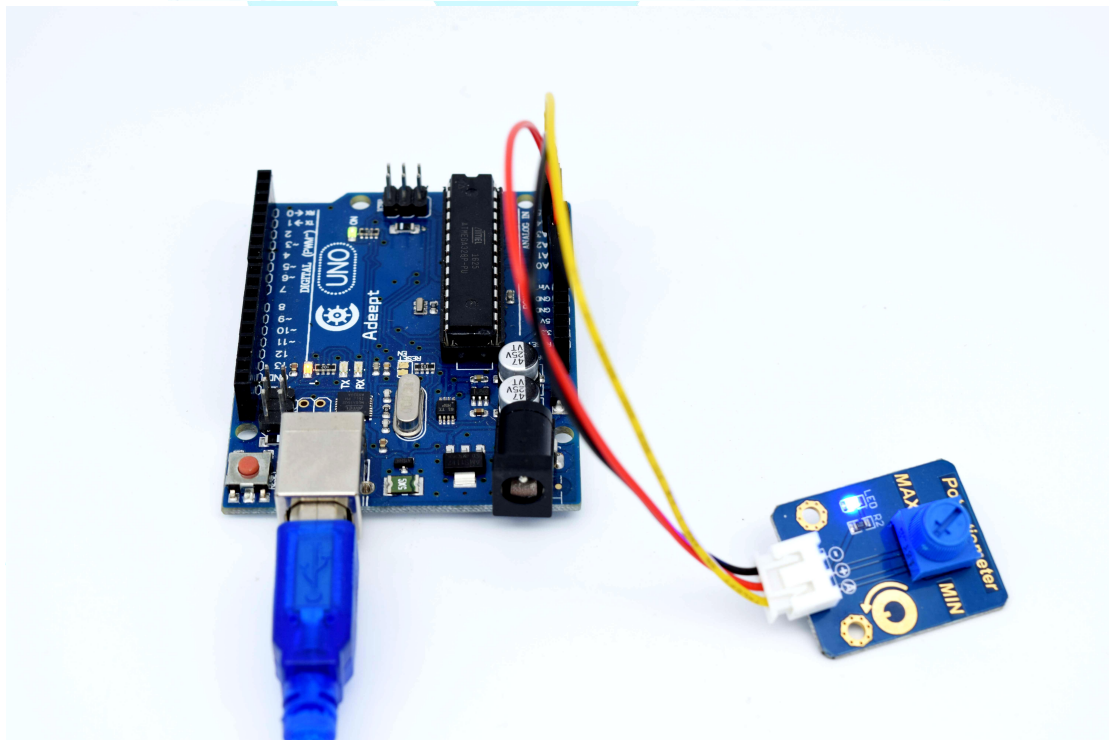
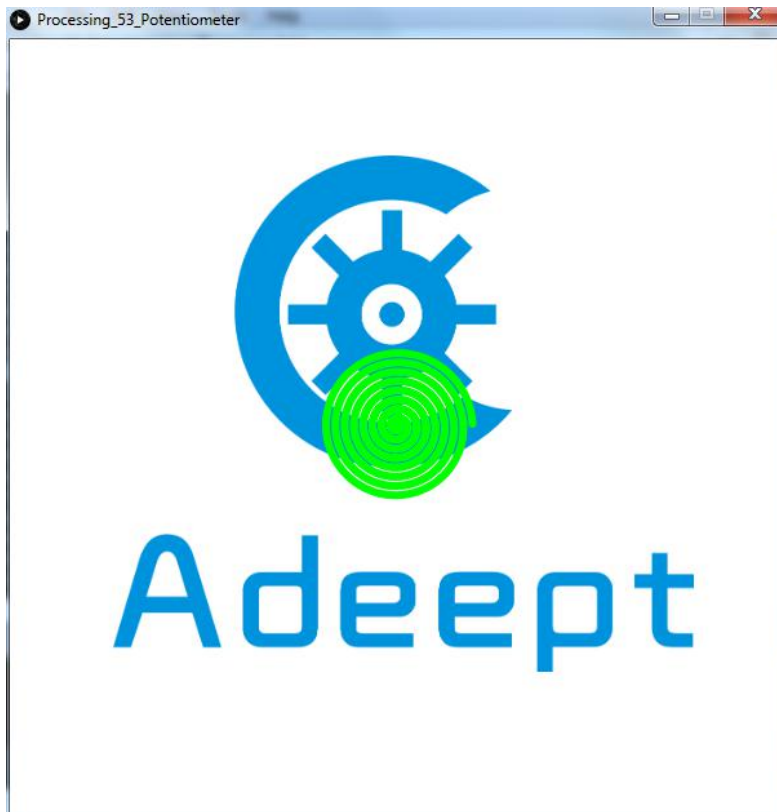
Step 2: Program _53_Potentiometer_Processing.ino

Step 3: Compile and download the sketch to the UNO R3 board.



Step 4: Run the Processing software (Processing_53_Potentiometer.pde)





Lesson 54 Arduino Interacts with Processing(Vibration Module)

Introduction

When you tap the vibration sensor, you will see that the ball in the Processing interface will randomly hit.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Vibration Sensor Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Principle

Processing key function:

Name: **random()**

Examples

```
for (int i = 0; i < 100; i++) {  
  float r = random(50);  
  stroke(r*5);  
  line(50, i, 50+r, i);  
}  
  
for (int i = 0; i < 100; i++) {  
  float r = random(-50, 50);  
  println(r);  
}  
  
// Get a random element from an array  
String[] words = { "apple", "bear", "cat", "dog" };  
int index = int(random(words.length)); // Same as int(random(4))  
println(words[index]); // Prints one of the four words
```

Description: Generates random numbers. Each time the random() function is called, it returns an unexpected value within the specified range. If only one parameter is passed to the function, it will return a float between zero and the value of the high parameter. For example, random(5) returns values between 0 and 5 (starting at zero, and up to, but not including, 5).

If two parameters are specified, the function will return a float with a value between the two values. For example, random(-5, 10.2) returns values starting at -5 and up to (but not including) 10.2. To convert a floating-point random number to an integer, use the int() function.

Syntax:

random(high)

random(low, high)

Parameters:

low float: lower limit

high float: upper limit

Returns: float

Name: **sqrt()**

Examples



```
noStroke();  
float a = sqrt(6561); // Sets 'a' to 81  
float b = sqrt(625);  // Sets 'b' to 25  
float c = sqrt(1);    // Sets 'c' to 1  
rect(0, 25, a, 10);  
rect(0, 45, b, 10);  
rect(0, 65, c, 10);
```

Description: Calculates the square root of a number. The square root of a number is always positive, even though there may be a valid negative root. The square root s of number a is such that $s*s = a$. It is the opposite of squaring.

Syntax:

sqrt(n)

Parameters:

n float: non-negative number

Returns: float

Name: **atan2()**

Examples

```
void draw() {  
  background(204);  
  translate(width/2, height/2);  
  float a = atan2(mouseY-height/2, mouseX-width/2);  
  rotate(a);  
  rect(-30, -5, 60, 10);  
}
```

Description: Calculates the angle (in radians) from a specified point to the coordinate origin as measured from the positive x-axis. Values are returned as a float in the range from π to $-\pi$. The `atan2()` function is most often used for orienting geometry to the position of the cursor. Note: The y-coordinate of the point is the first parameter, and the x-coordinate is the second parameter, due to the structure of calculating the tangent.

Syntax:

atan2(y, x)

Parameters:

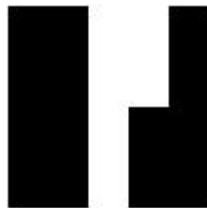
y float: y-coordinate of the point

x float: x-coordinate of the point

Returns: float

Name: **height**

Examples



```
noStroke();  
background(0);  
rect(40, 0, 20, height);  
rect(60, 0, 20, height/2);
```

Description: System variable that stores the height of the display window. This value is set by the second parameter of the `size()` function. For example, the function call `size(320, 240)` sets the height variable to the value 240. The value of height defaults to 100 if `size()` is not used in a program.

Name: **width**

Examples

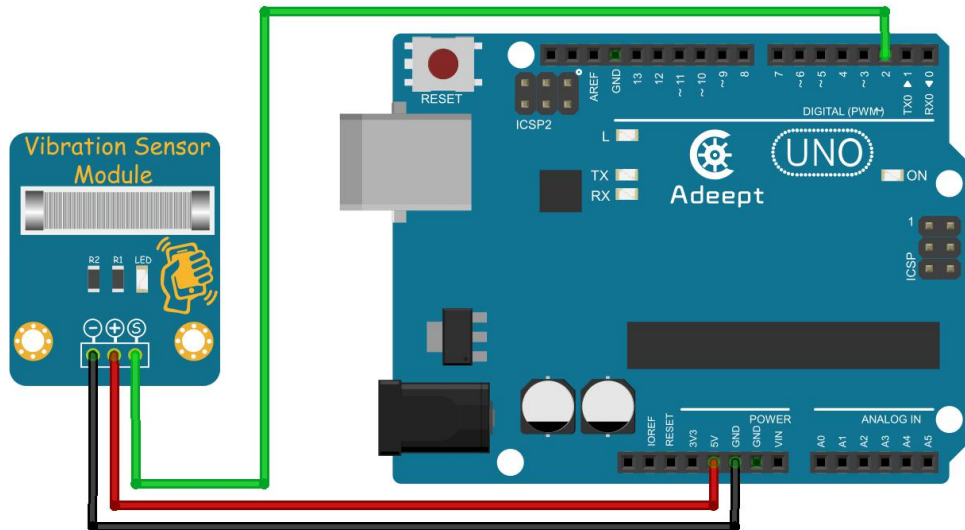


```
noStroke();  
background(0);  
rect(0, 40, width, 20);  
rect(0, 60, width/2, 20);
```

Description: System variable that stores the width of the display window. This value is set by the first parameter of the `size()` function. For example, the function call `size(320, 240)` sets the width variable to the value 320. The value of width defaults to 100 if `size()` is not used in a program.

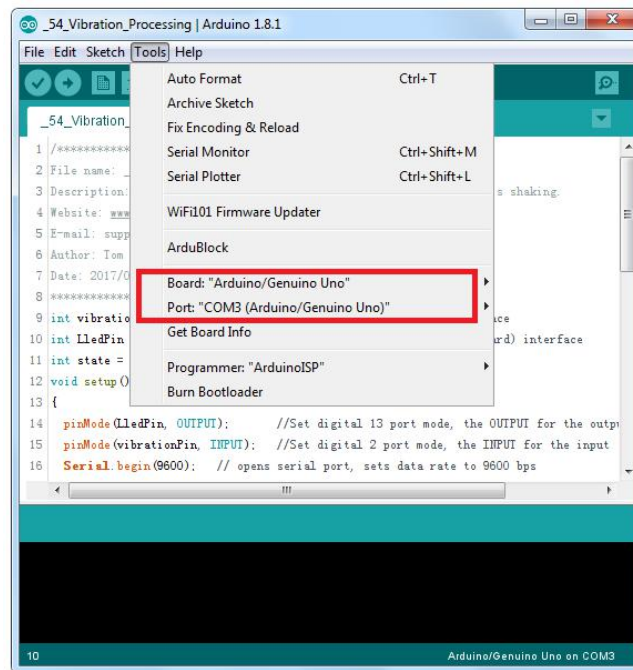
Experimental Procedures

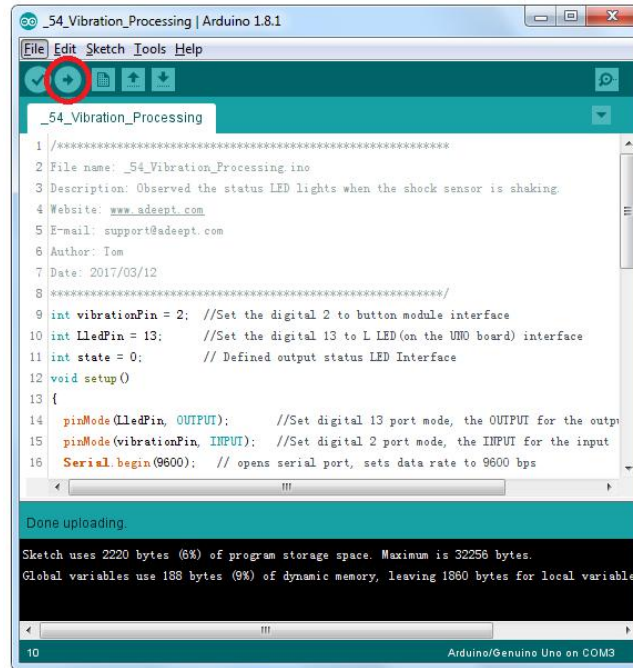
Step 1: Build the circuit



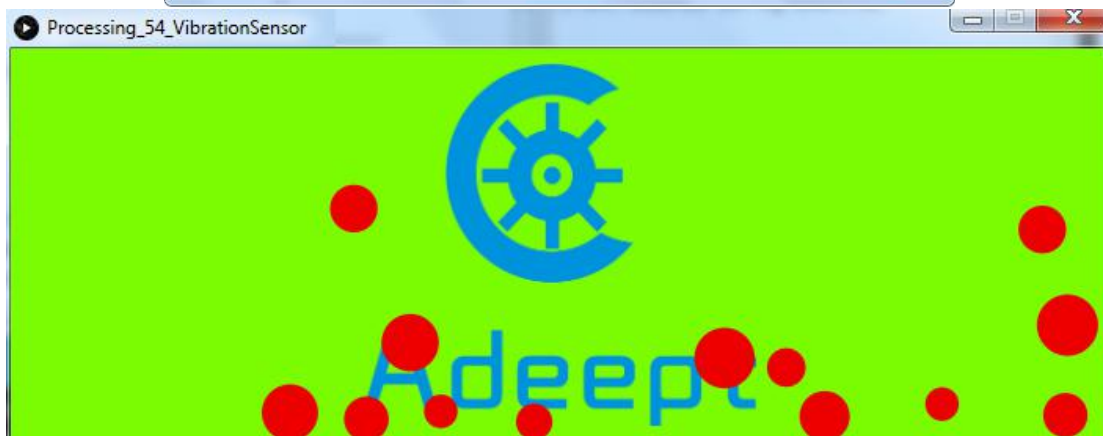
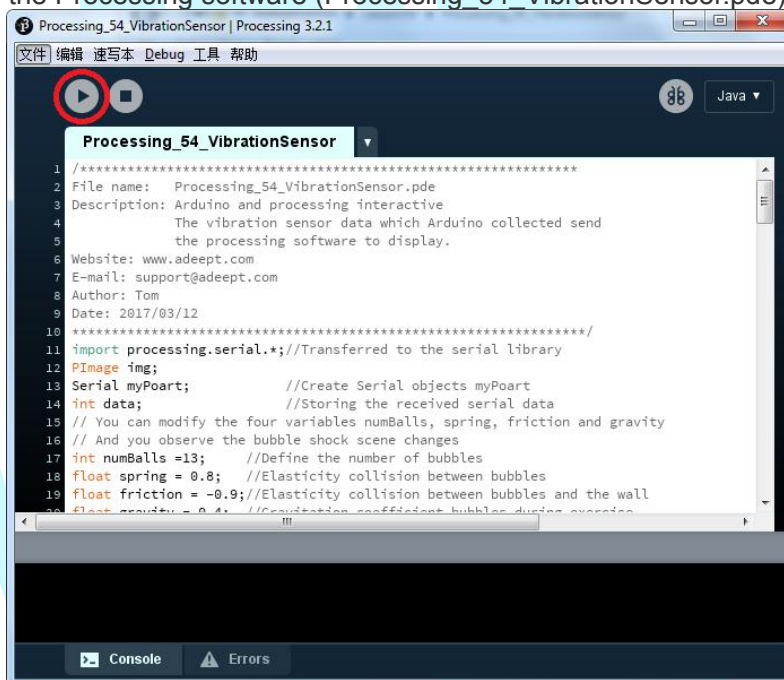
Step 2: Program _54_Vibration_Processing.ino

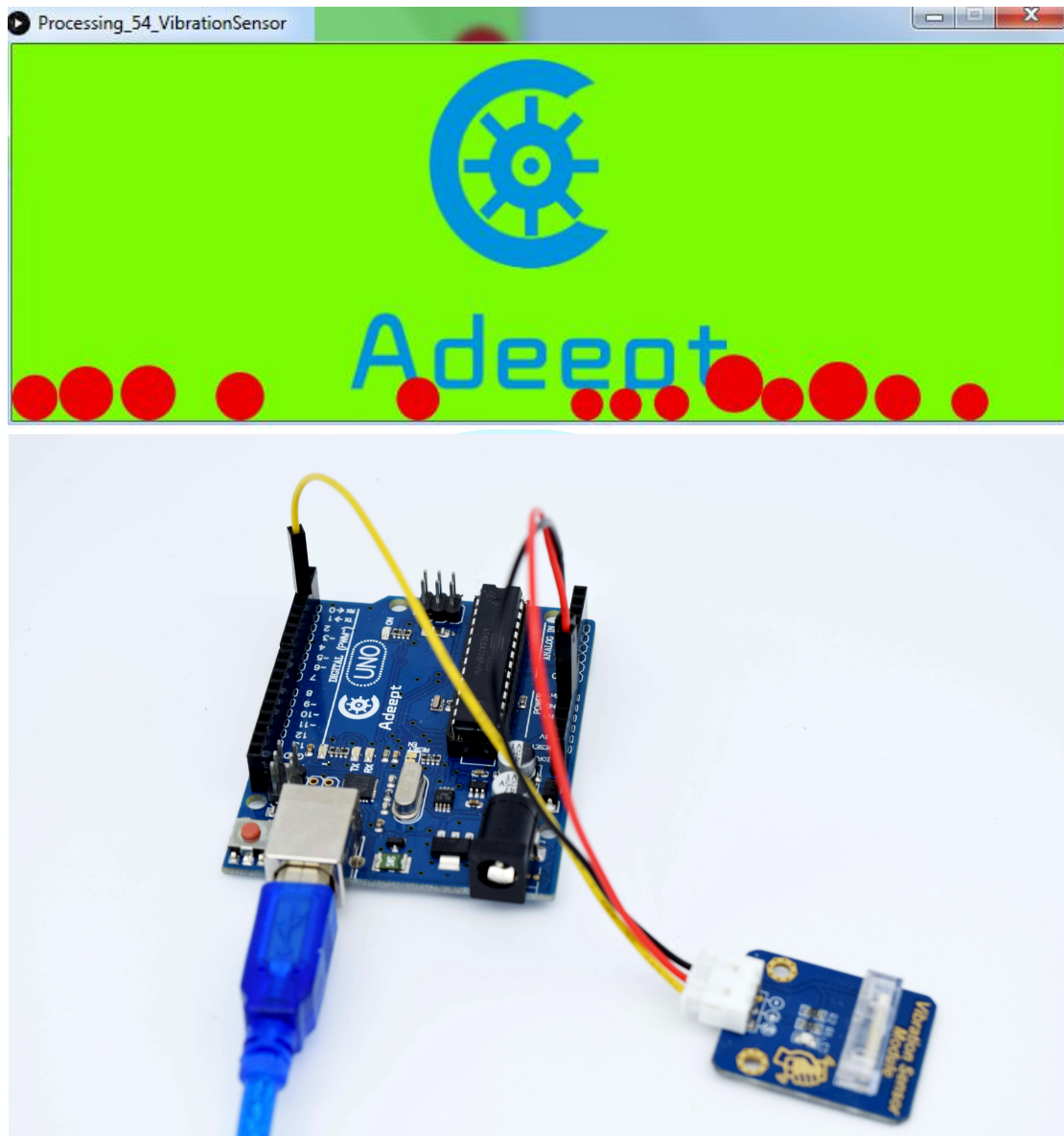
Step 3: Compile and download the sketch to the UNO R3 board.





Step 4: Run the Processing software (Processing_54_VibrationSensor.pde)





Adeept

Lesson 55 Arduino Interacts with Processing(Photoresistor)

Introduction

In this lesson, the brightness of the picture in the Processing interface changes with ambient light intensity.

Components

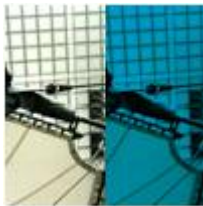
- 1 * Aadept Arduino UNO R3 Board
- 1 * Photoresistor Module
- 1 * USB Cable
- 1 * 3-Pin Wires

Principle

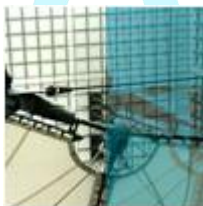
Processing key function:

Name: `tint()`

Examples



```
PImage img;  
img = loadImage("laDefense.jpg");  
image(img, 0, 0);  
tint(0, 153, 204); // Tint blue  
image(img, 50, 0);
```



```
PImage img;  
img = loadImage("laDefense.jpg");  
image(img, 0, 0);  
tint(0, 153, 204, 126); // Tint blue and set transparency  
image(img, 50, 0);
```



```
PImage img;  
img = loadImage("1aDefense.jpg");  
image(img, 0, 0);  
tint(255, 126); // Apply transparency without changing color  
image(img, 50, 0);
```

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

grayfloat: specifies a value between white and black

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

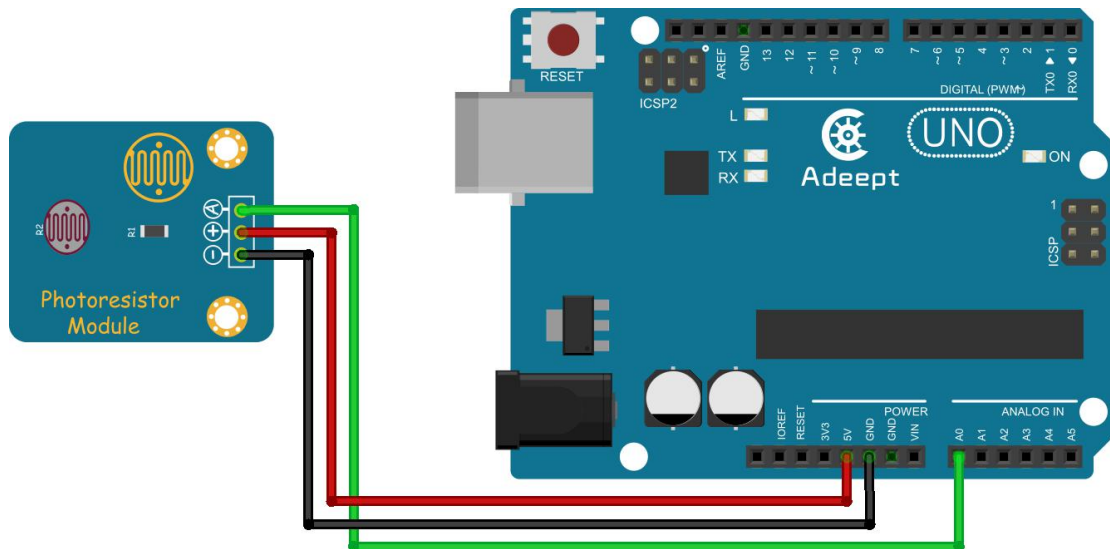
v2 float: green or saturation value (depending on current color mode)

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

Returns void

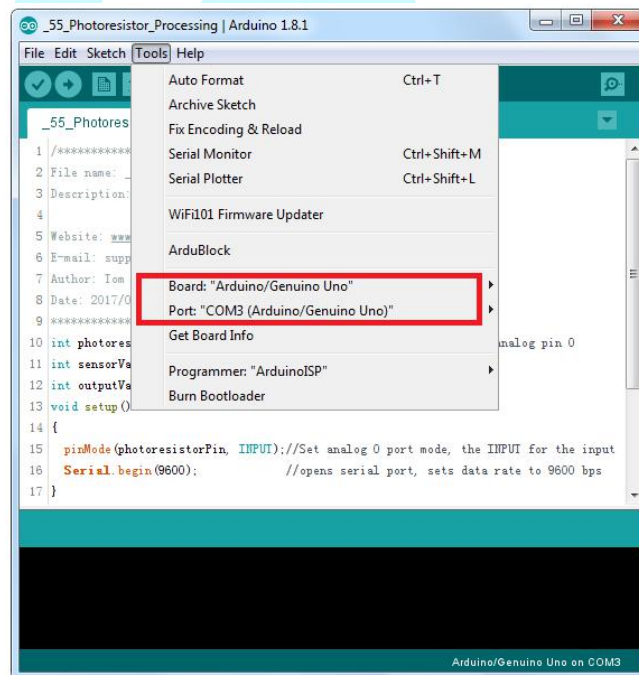
Experimental Procedures

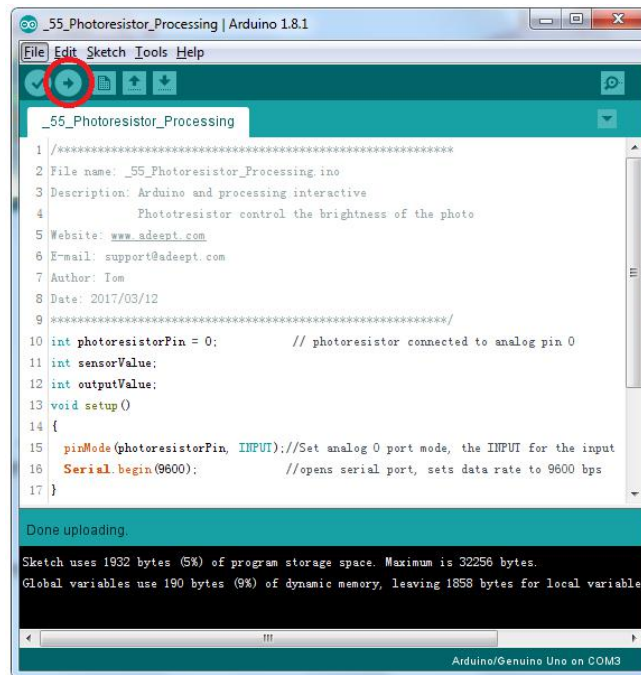
Step 1: Build the circuit



Step 2: Program _55_Photoresistor_Processing.ino

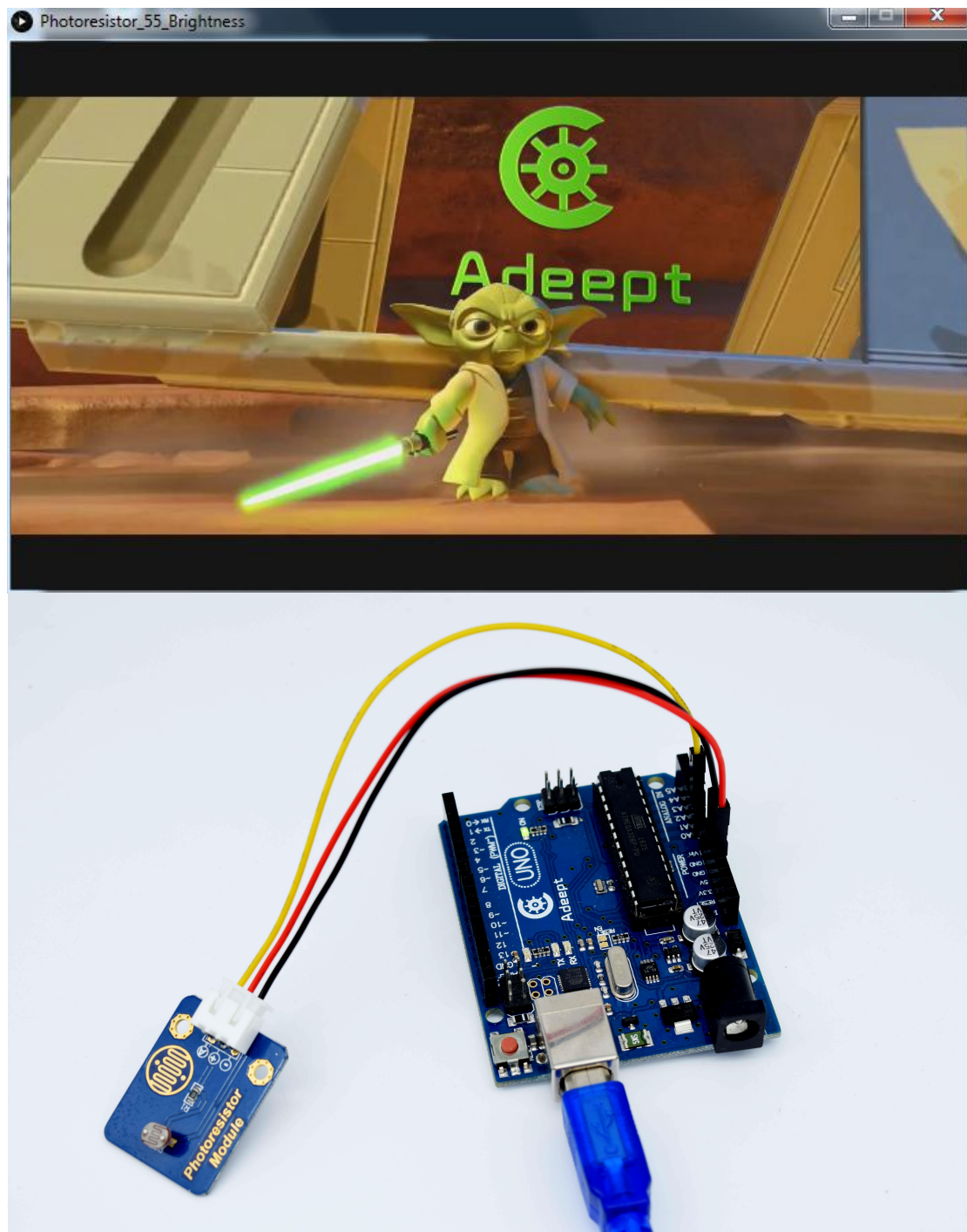
Step 3: Compile and download the sketch to the UNO R3 board.





Step 4: Run the Processing software (Photoresistor_55_Brightness.pde)





Lesson 56 Arduino Interacts with Processing(DS18B20 Module)

Introduction

In this lesson, the Arduino detects the ambient temperature through the DS18B20 and then displays the temperature data as a curve in the Processing interface.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * DS18B20 Module
- 1 * USB Cable
- 1 * 3-Pin Wires

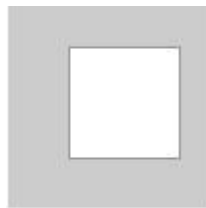
Principle

Processing key function:

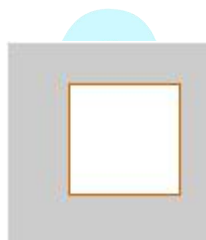
Name

stroke()

Examples



```
stroke(153);  
rect(30, 20, 55, 55);
```



```
stroke(204, 102, 0);  
rect(30, 20, 55, 55);
```

Description Sets the color used to draw lines and borders around shapes. 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.

When drawing in 2D with the default renderer, you may need `hint(ENABLE_STROKE_PURE)` to improve drawing quality (at the expense of performance). See the `hint()` documentation for more details.

Syntax

`stroke(rgb)`

`stroke(rgb, alpha)`

`stroke(gray)`

`stroke(gray, alpha)`

`stroke(v1, v2, v3)`

`stroke(v1, v2, v3, alpha)`

Parameters

`rgb` int: color value in hexadecimal notation

`alpha` float: opacity of the stroke

`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)

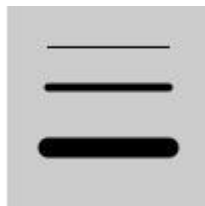
`v3` float: blue or brightness value (depending on current color mode)

Returns void

Name

`strokeWeight()`

Examples



```
strokeWeight(1); // Default
line(20, 20, 80, 20);
strokeWeight(4); // Thicker
line(20, 40, 80, 40);
strokeWeight(10); // Beastly
line(20, 70, 80, 70);
```

Description Sets the width of the stroke used for lines, points, and the border around shapes. All widths are set in units of pixels.

Syntax

`strokeWeight(weight)`

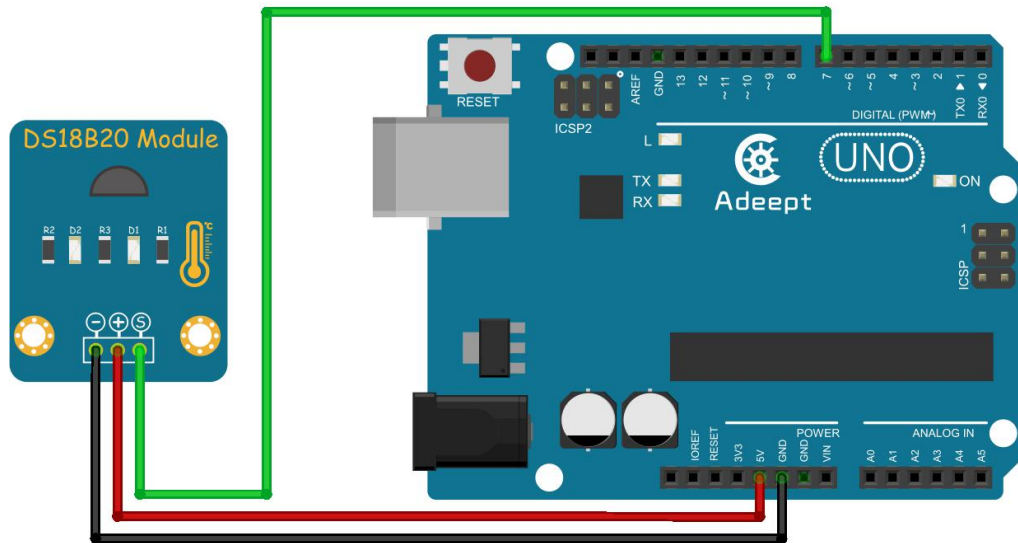
Parameters

`weight` float: the weight (in pixels) of the stroke

Returns void

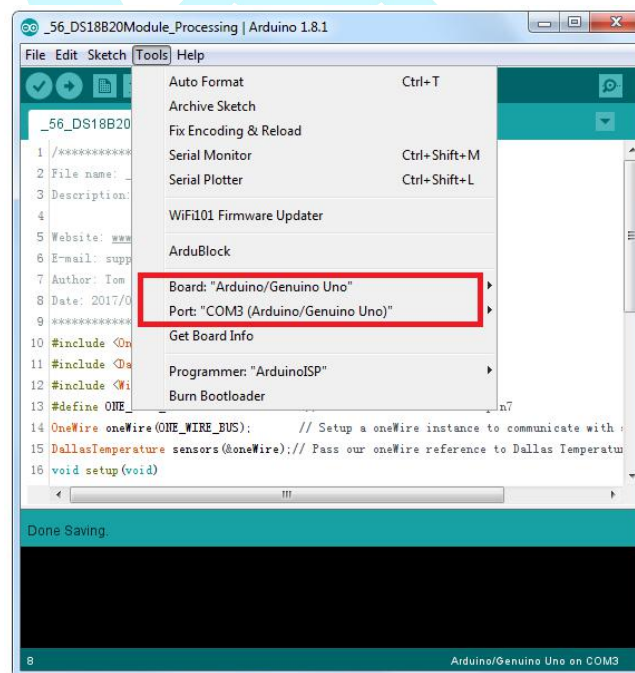
Experimental Procedures

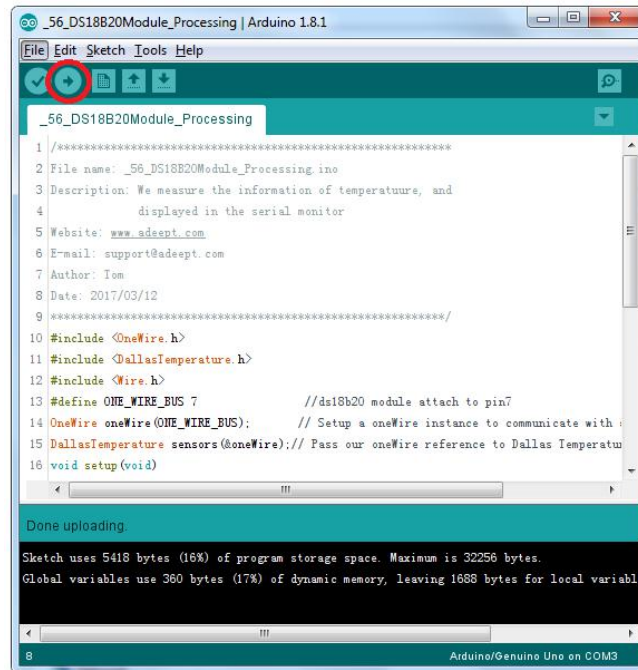
Step 1: Build the circuit



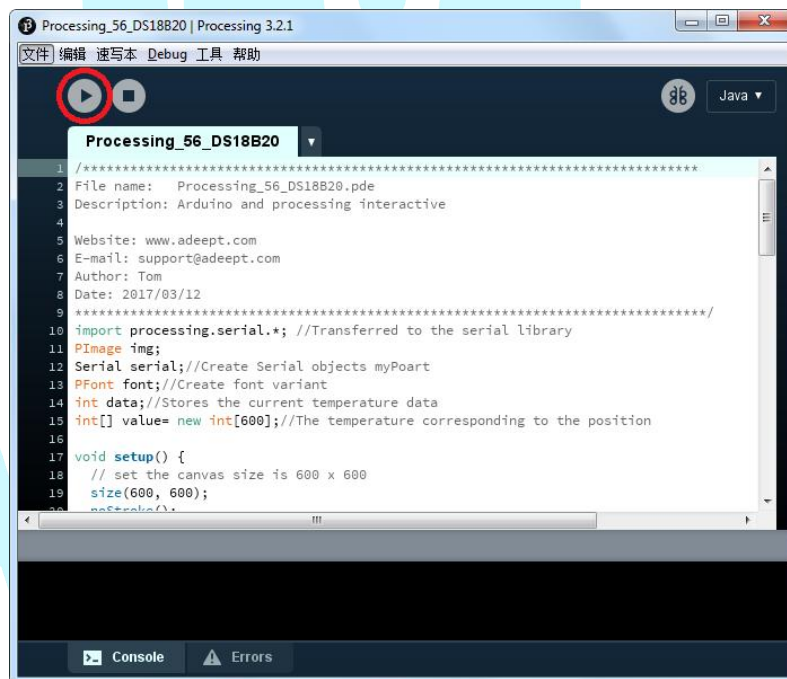
Step 2: Program _56_DS18B20Module_Processing.ino

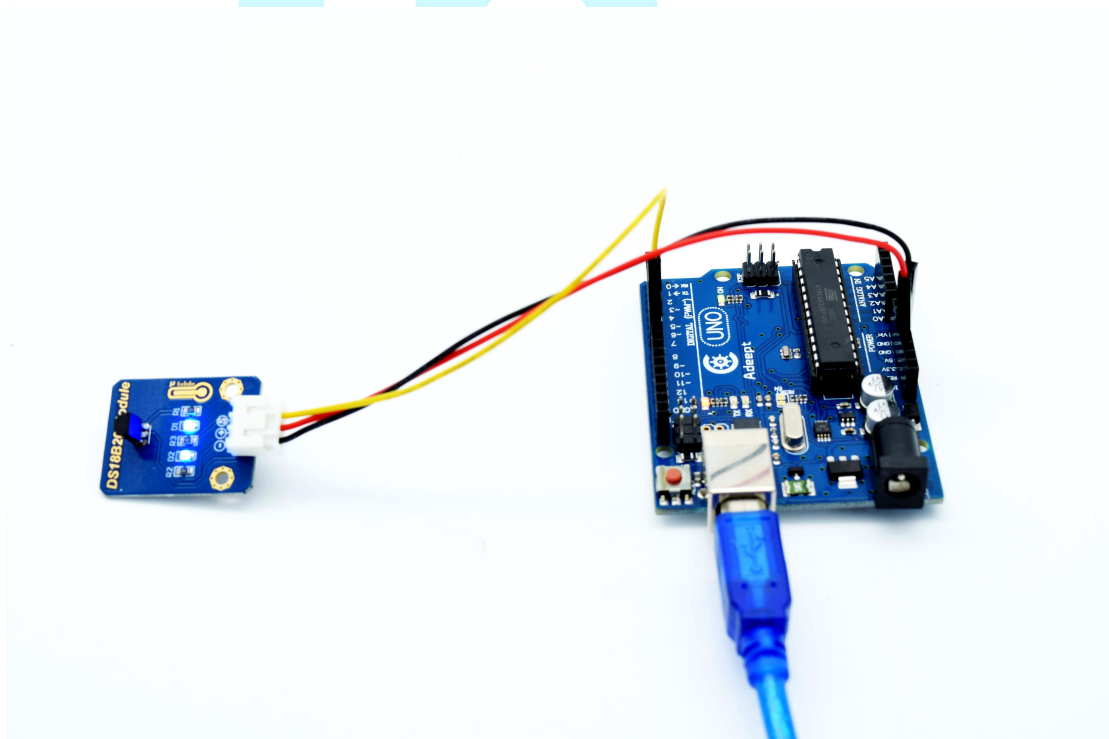
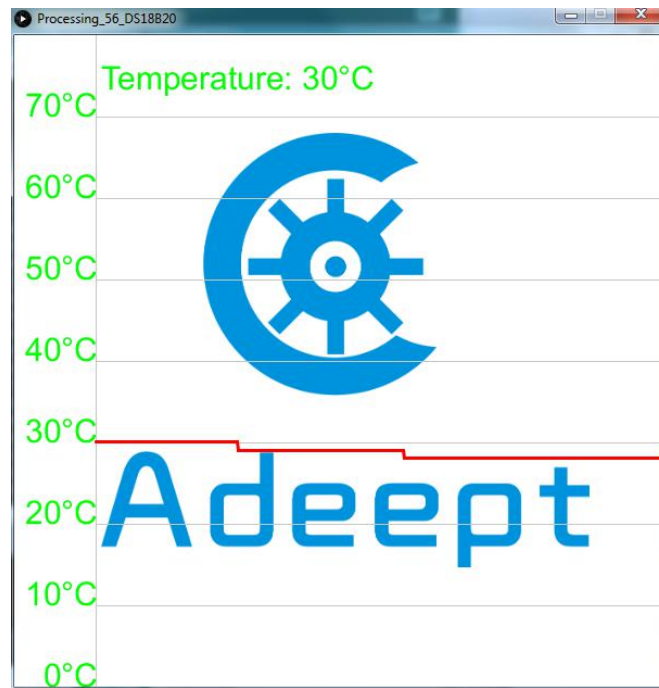
Step 3: Compile and download the sketch to the UNO R3 board.





Step 4: Run the Processing software (Processing_56_DS18B20.pde)





Lesson 57 Arduino Interacts with Processing(Buzzer Module)

Introduction

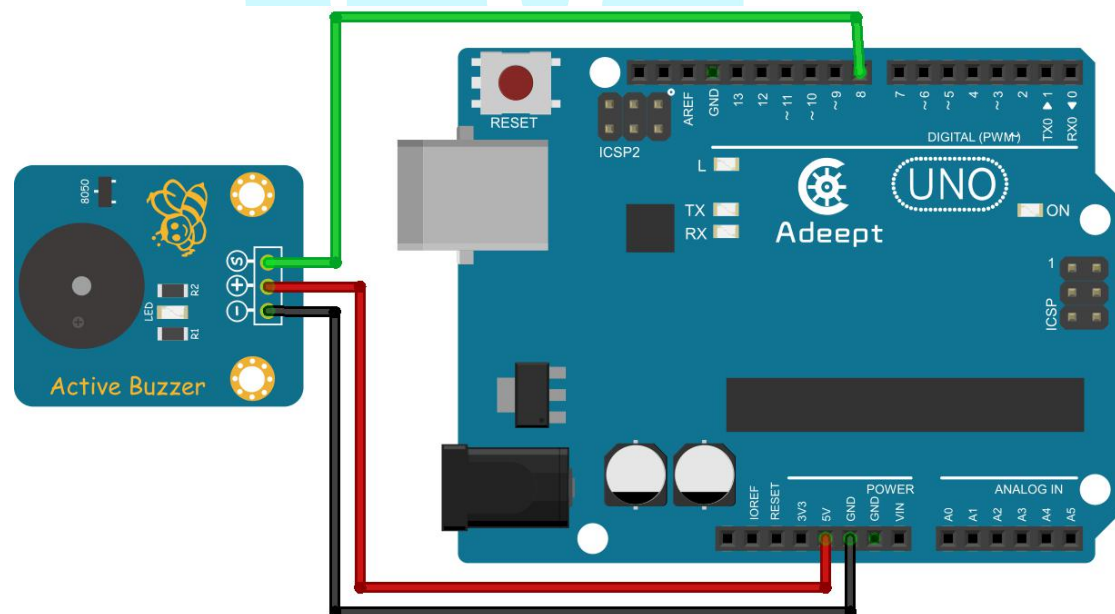
In this lesson, when the mouse pointer points to the buzzer in the Processing interface, the buzzer connected to the Arduino will sound.

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Active Buzzer Module
- 1 * USB Cable
- 1 * 3-Pin Wires

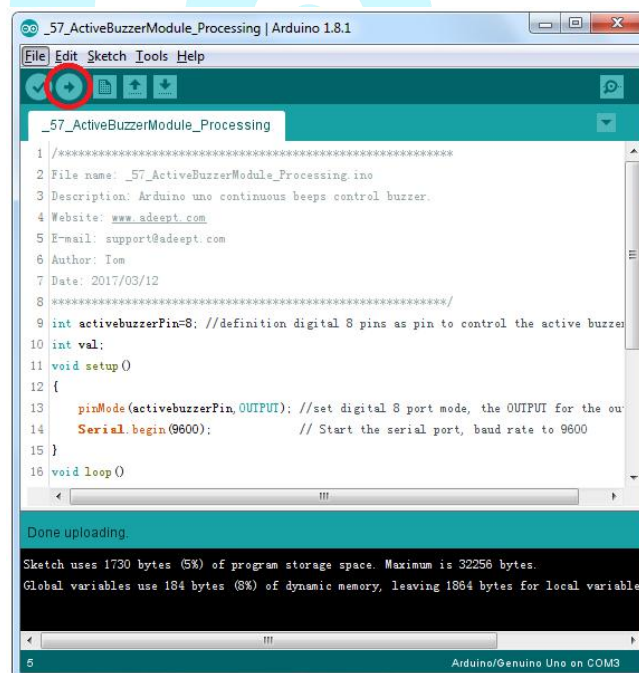
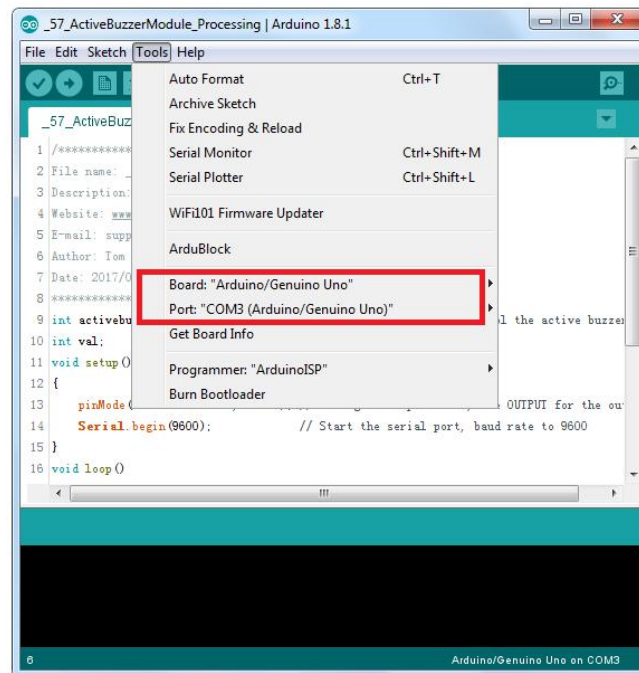
Experimental Procedures

Step 1: Build the circuit

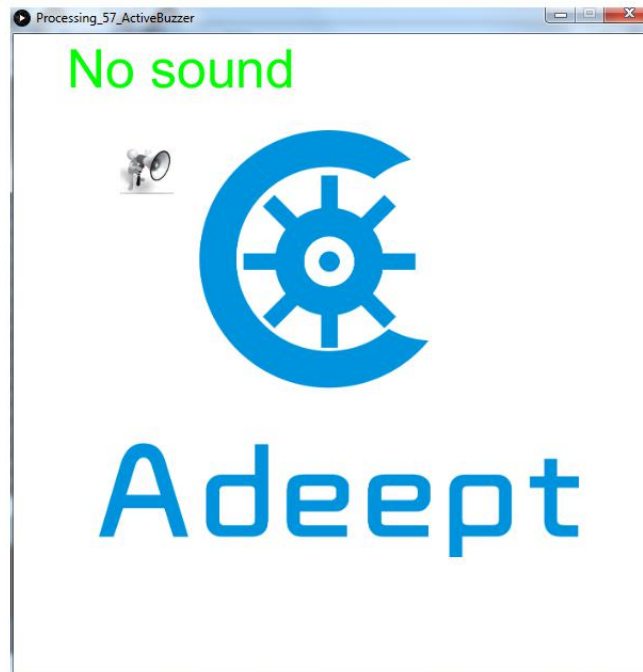
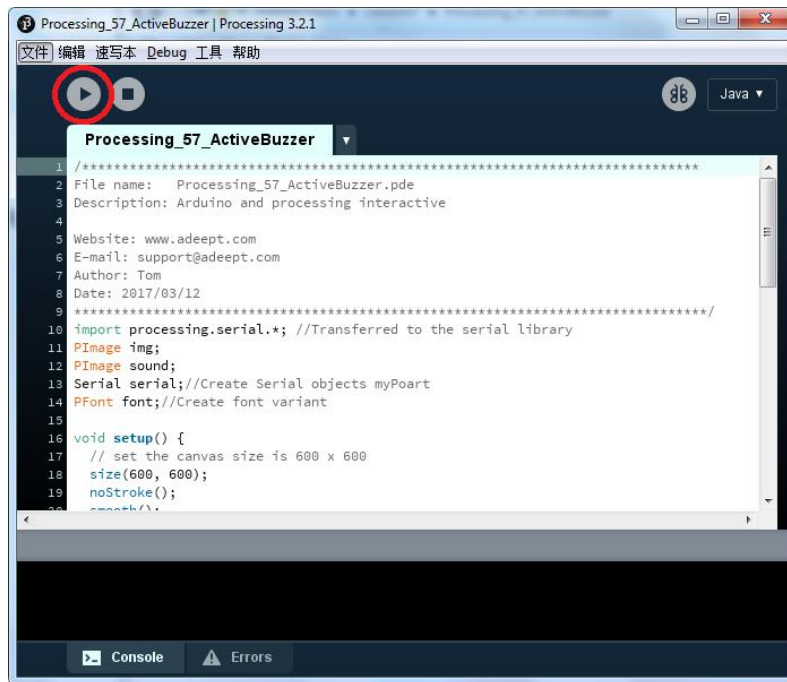


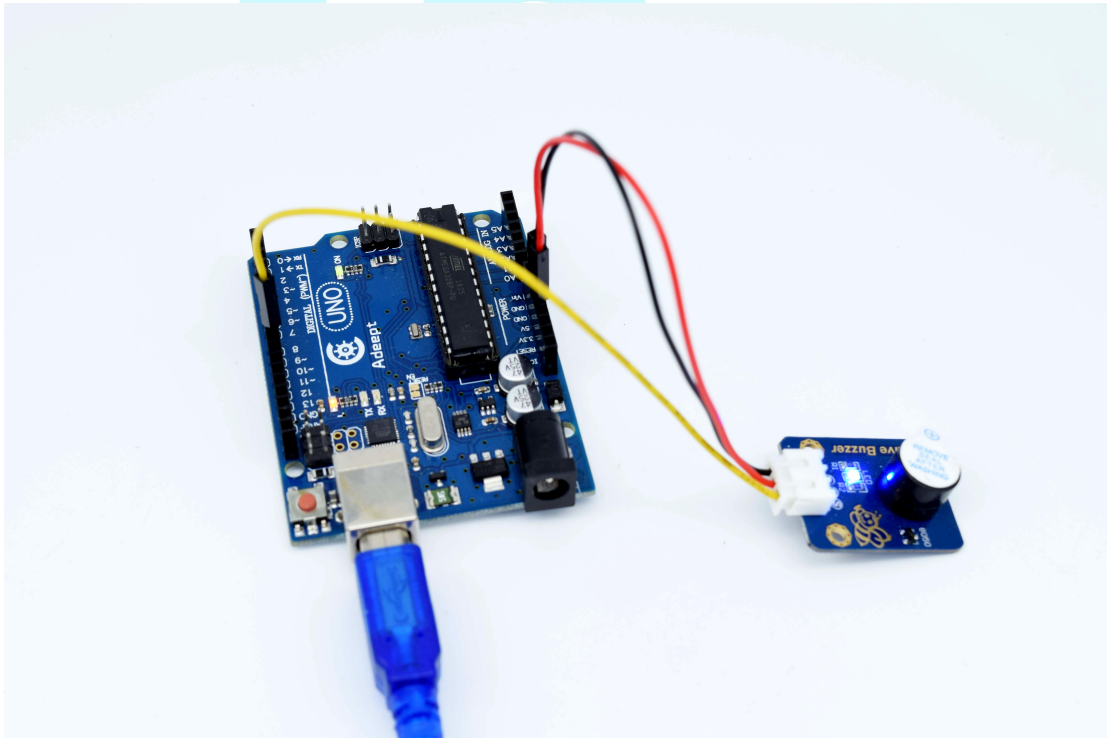
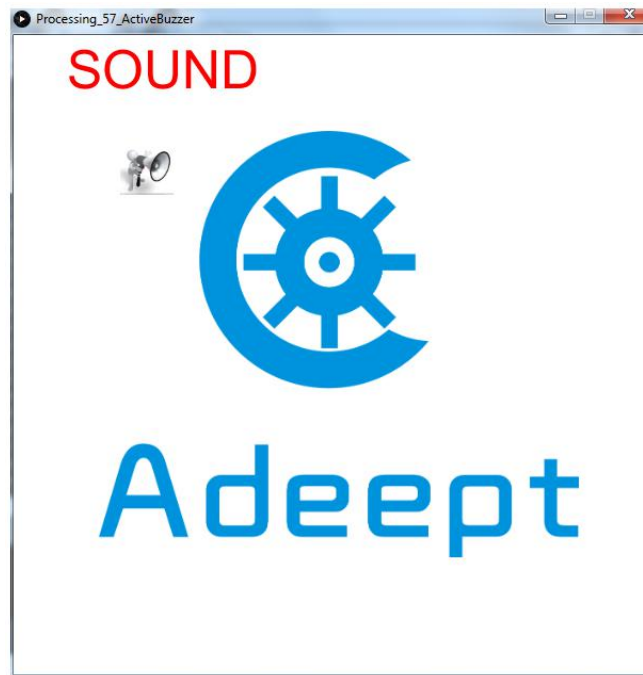
Step 2: Program _57_ActiveBuzzerModule_Processing.ino

Step 3: Compile and download the sketch to the UNO R3 board.



Step 4: Run the Processing software (Processing_57_ActiveBuzzer.pde)





Lesson 58 Arduino Interacts with Processing(Rotary Encoder)

Introduction

When you rotate the knob of the rotary encoder, the stripes in the color bar will be increased or decreased.

Components

- 1 * Aadept Arduino UNO R3 Board
- 1 * Rotary Encoder Module
- 1 * USB Cable
- 1 * 5-Pin Wires

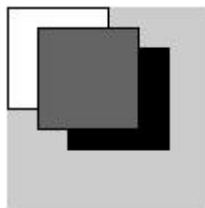
Principle

Processing key function:

Name

pushMatrix()

Examples



```
fill(255);  
rect(0, 0, 50, 50); // White rectangle  
  
pushMatrix();  
translate(30, 20);  
fill(0);  
rect(0, 0, 50, 50); // Black rectangle  
popMatrix();
```

```
fill(100);  
rect(15, 10, 50, 50); // Gray rectangle
```

Description

Pushes the current transformation matrix onto the matrix stack. Understanding pushMatrix() and popMatrix() requires understanding the concept of a matrix stack. The pushMatrix() function saves the current coordinate system to the stack and popMatrix() restores the prior coordinate system. pushMatrix() and popMatrix() are used in conjunction with the other transformation functions and may be embedded to control the scope of the transformations.

Syntax

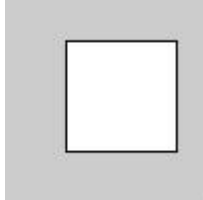
pushMatrix()

Returns void

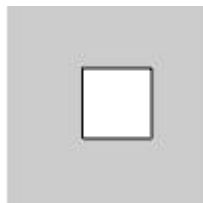
Name

translate()

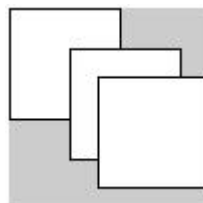
Examples



```
translate(30, 20);  
rect(0, 0, 55, 55);
```



```
// Translating in 3D requires P3D  
// as the parameter to size()  
size(100, 100, P3D);  
// Translate 30 across, 20 down, and  
// 50 back, or "away" from the screen.  
translate(30, 20, -50);  
rect(0, 0, 55, 55);
```



```
rect(0, 0, 55, 55); // Draw rect at original 0,0  
translate(30, 20);  
rect(0, 0, 55, 55); // Draw rect at new 0,0  
translate(14, 14);  
rect(0, 0, 55, 55); // Draw rect at new 0,0
```

Description Specifies an amount to displace objects within the display window. The x parameter specifies left/right translation, the y parameter specifies up/down translation, and the z parameter specifies translations toward/away from the screen. Using this function with the z parameter requires using P3D as a parameter in combination with size as shown in the above example.

Transformations are cumulative and apply to everything that happens after and subsequent calls to the function accumulates the effect. For example, calling translate(50,

0) and then `translate(20, 0)` is the same as `translate(70, 0)`. If `translate()` is called within `draw()`, the transformation is reset when the loop begins again. This function can be further controlled by using `pushMatrix()` and `popMatrix()`.

Syntax

`translate(x, y)`

`translate(x, y, z)`

Parameters

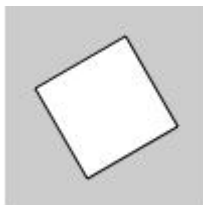
x float: left/right translation
y float: up/down translation
z float: forward/backward translation

Returns void

Name

`rotate()`

Examples



```
translate(width/2, height/2);  
rotate(PI/3.0);  
rect(-26, -26, 52, 52);
```

Description Rotates the amount specified by the angle parameter. Angles must be specified in radians (values from 0 to TWO_PI), or they can be converted from degrees to radians with the `radians()` function.

The coordinates are always rotated around their relative position to the origin. Positive numbers rotate objects in a clockwise direction and negative numbers rotate in the counterclockwise direction. Transformations apply to everything that happens afterward, and subsequent calls to the function compound the effect. For example, calling `rotate(PI/2.0)` once and then calling `rotate(PI/2.0)` a second time is the same as a single `rotate(PI)`. All transformations are reset when `draw()` begins again.

Technically, `rotate()` multiplies the current transformation matrix by a rotation matrix. This function can be further controlled by `pushMatrix()` and `popMatrix()`.

Syntax

`rotate(angle)`

Parameters

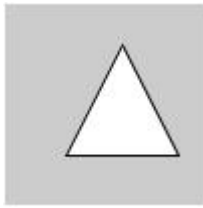
angle float: angle of rotation specified in radians

Returns void

Name

`triangle()`

Examples



```
triangle(30, 75, 58, 20, 86, 75);
```

Description A triangle is a plane created by connecting three points. The first two arguments specify the first point, the middle two arguments specify the second point, and the last two arguments specify the third point.

Syntax

triangle(x1, y1, x2, y2, x3, y3)

Parameters

x1 float: x-coordinate of the first point

y1 float: y-coordinate of the first point

x2 float: x-coordinate of the second point

y2 float: y-coordinate of the second point

x3 float: x-coordinate of the third point

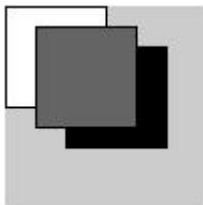
y3 float: y-coordinate of the third point

Returns void

Name

popMatrix()

Examples



```
fill(255);  
rect(0, 0, 50, 50); // White rectangle  
  
pushMatrix();  
translate(30, 20);  
fill(0);  
rect(0, 0, 50, 50); // Black rectangle  
popMatrix();
```

```
fill(100);  
rect(15, 10, 50, 50); // Gray rectangle
```

Description Pops the current transformation matrix off the matrix stack. Understanding pushing and popping requires understanding the concept of a matrix stack. The pushMatrix() function saves the current coordinate system to the stack and popMatrix() restores the prior coordinate system. pushMatrix() and popMatrix() are used in conjunction with the other transformation functions and may be embedded to control the scope of the

transformations.

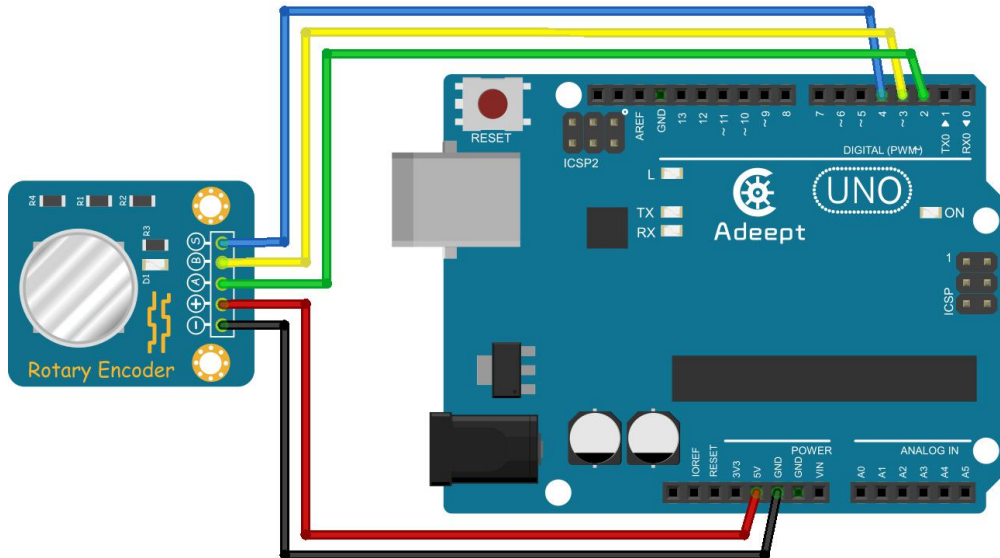
Syntax

popMatrix()

Returns void

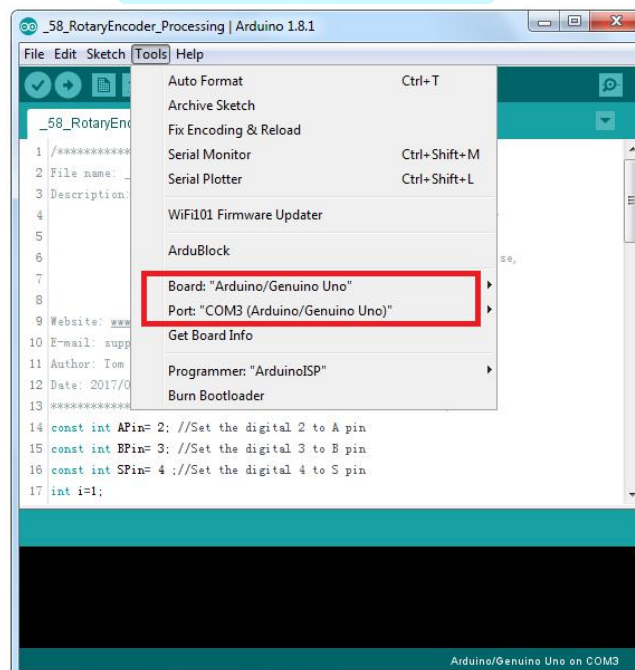
Experimental Procedures

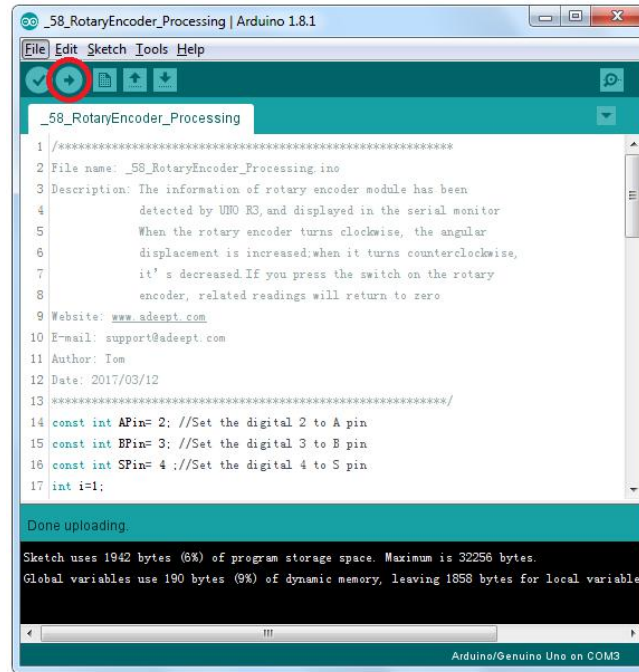
Step 1: Build the circuit



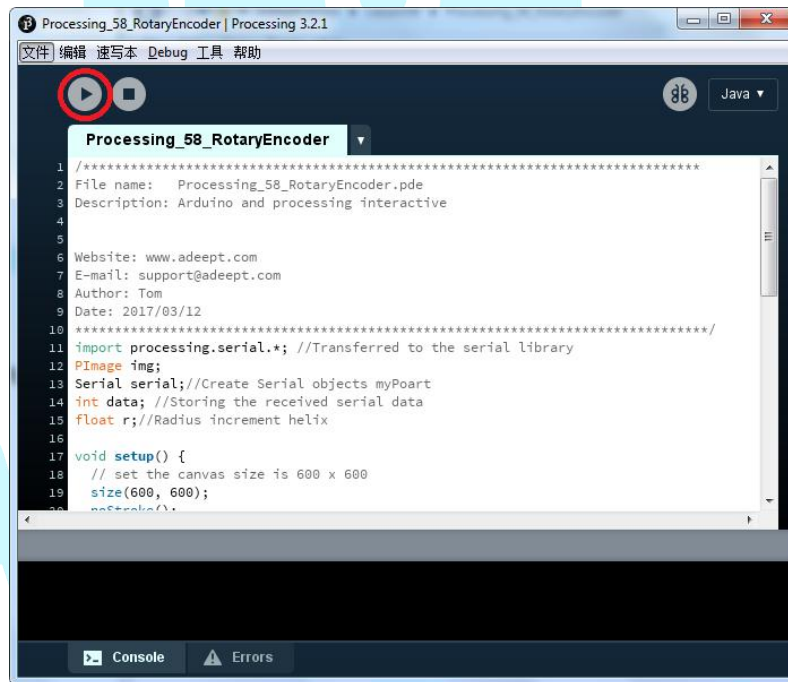
Step 2: Program _58_RotaryEncoder_Processing.ino

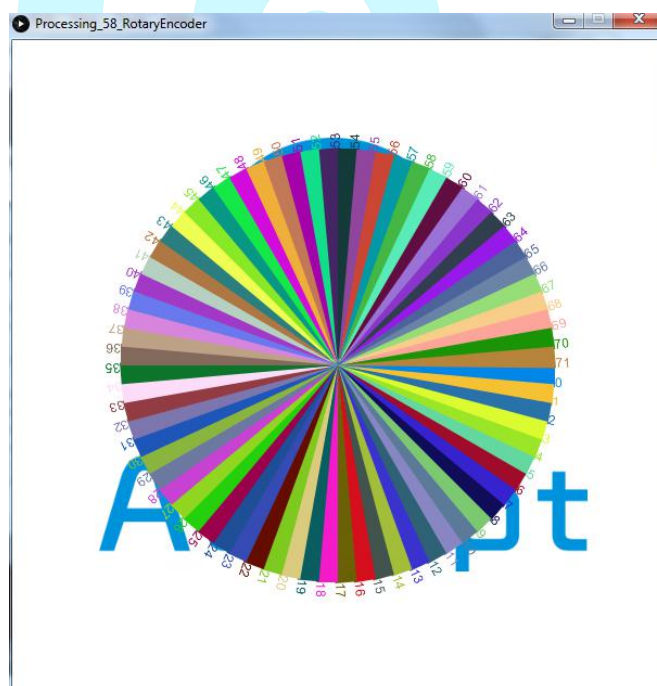
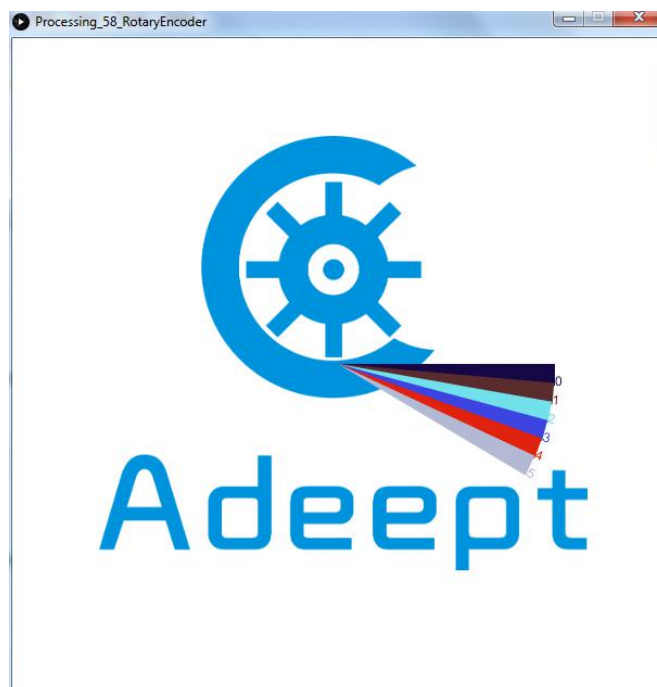
Step 3: Compile and download the sketch to the UNO R3 board.

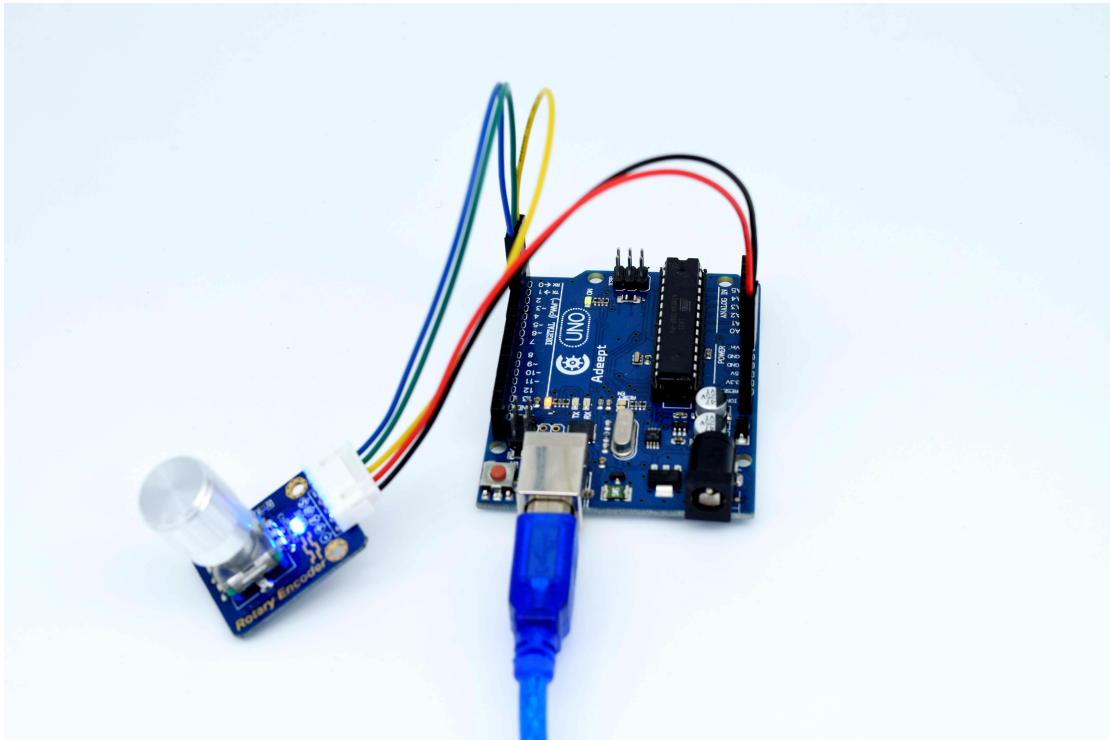




Step 4: Run the Processing software (Processing_58_RotaryEncoder.pde)







Adeept

Lesson 59 Arduino Interacts with Processing(Joystick Module)

Introduction

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

Components

- 1 * Adept Arduino UNO R3 Board
- 1 * Joystick Module
- 1 * USB Cable
- 1 * 5-Pin Wires

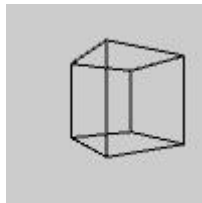
Principle

Processing key function:

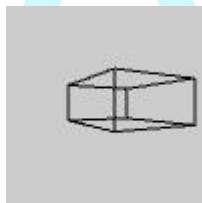
Name

box()

Examples



```
size(100, 100, P3D);  
translate(58, 48, 0);  
rotateY(0.5);  
noFill();  
box(40);
```



```
size(100, 100, P3D);  
translate(58, 48, 0);  
rotateY(0.5);  
noFill();  
box(40, 20, 50);
```

Description A box is an extruded rectangle. A box with equal dimensions on all sides is a cube.

Syntax

box(size)

box(w, h, d)

Parameters

size float: dimension of the box in all dimensions (creates a cube)

w float: dimension of the box in the x-dimension

h float: dimension of the box in the y-dimension

d float: dimension of the box in the z-dimension

Returns void

Name

split()

Examples

```
String men = "Chernenko, Andropov, Brezhnev";  
String[] list = split(men, ',');  
// list[0] is now "Chernenko", list[1] is "Andropov"...  
String numbers = "8 67 5 309";  
int[] nums = int(split(numbers, ' '));  
// nums[0] is now 8, nums[1] is now 67...  
String men = "Chernenko ] Andropov ] Brezhnev";  
String[] list = split(men, " ] ");  
// list[0] is now "Chernenko", list[1] is "Andropov"...
```

Description The `split()` function breaks a String into pieces using a character or string as the delimiter. The `delim` parameter specifies the character or characters that mark the boundaries between each piece. A `String[]` array is returned that contains each of the pieces.

If the result is a set of numbers, you can convert the `String[]` array to a `float[]` or `int[]` array using the datatype conversion functions `int()` and `float()`. (See the second example above.)

The `splitTokens()` function works in a similar fashion, except that it splits using a range of characters instead of a specific character or sequence.

Syntax

split(value, delim)

Parameters

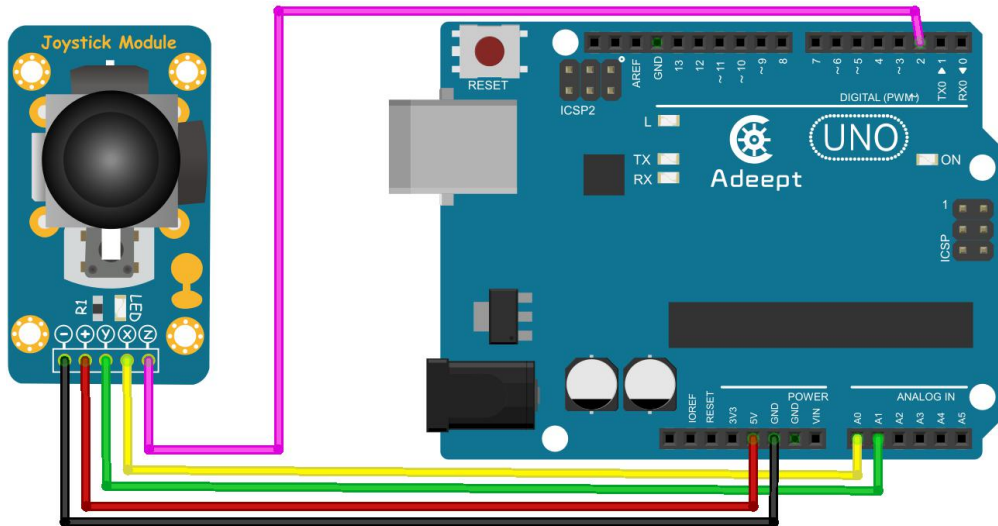
value String: the String to be split

delim char: the character or String used to separate the data

Returns `String[]`

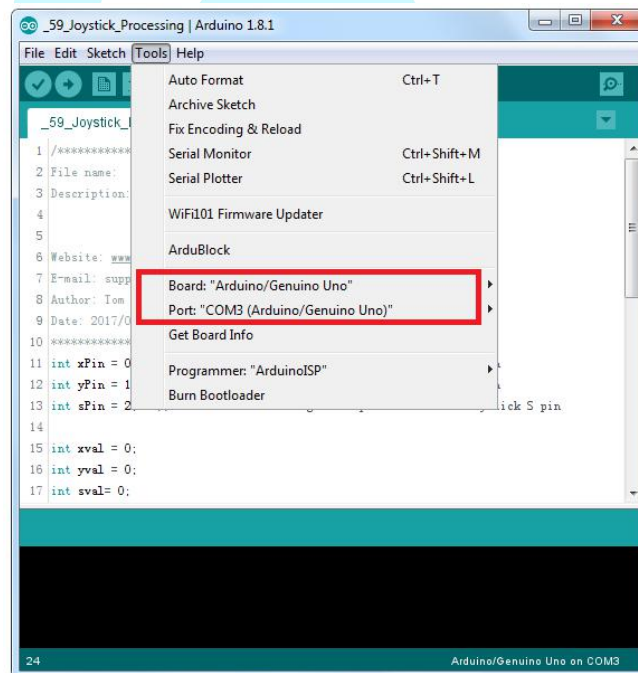
Experimental Procedures

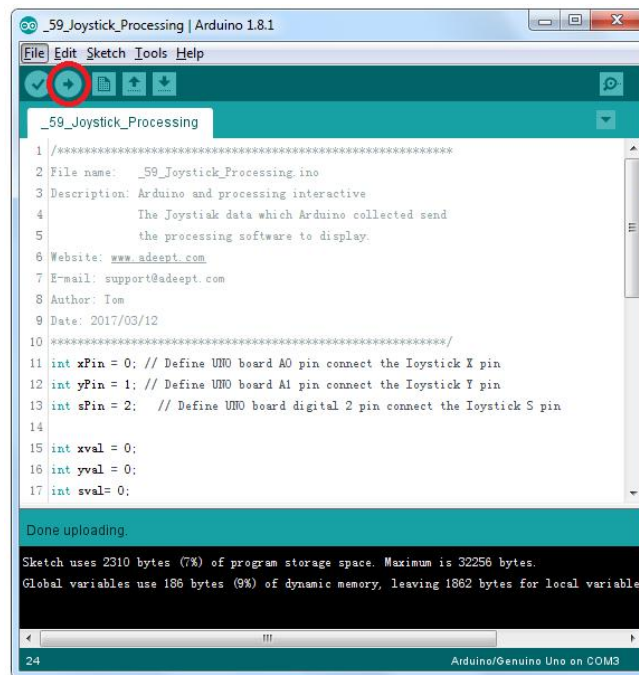
Step 1: Build the circuit



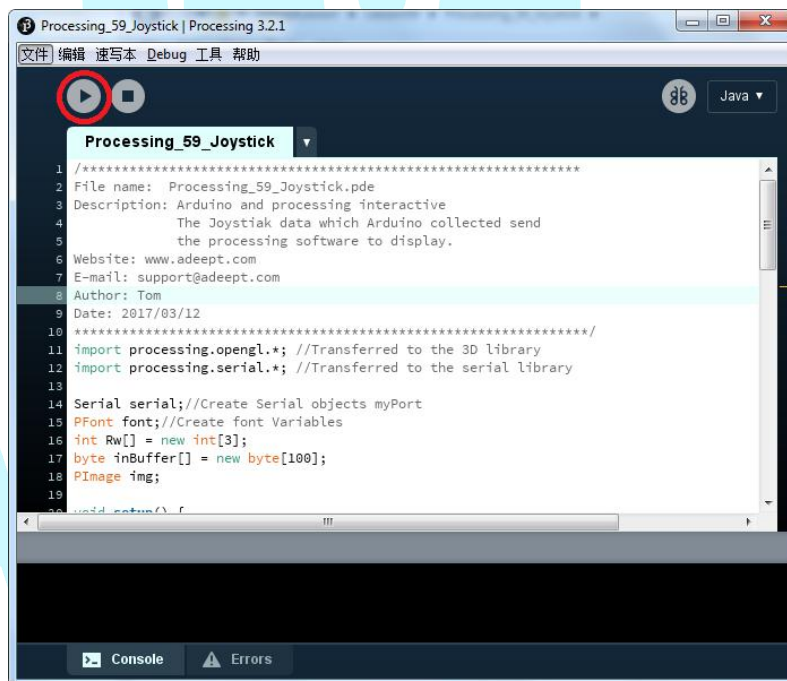
Step 2: Program `_59_Joystick_Processing.ino`

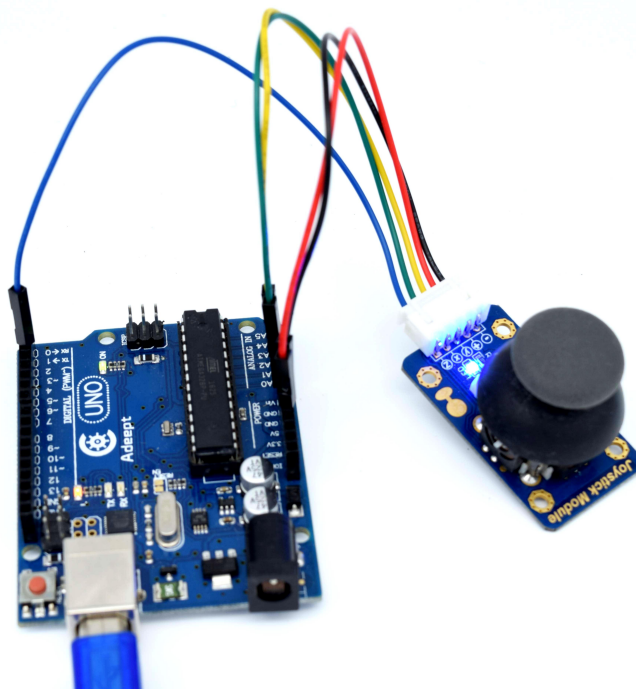
Step 3: Compile and download the sketch to the UNO R3 board.





Step 4: Run the Processing software (Processing_59_Joystick.pde)





Lesson 60 Arduino Interacts with Processing(Ultrasonic Distance Module)

Introduction

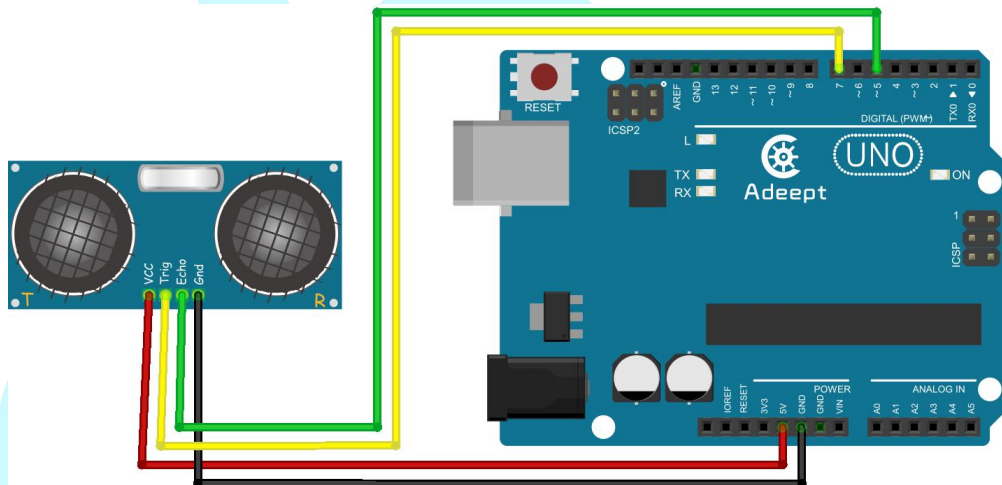
In this lesson, Arduino detects the distance between the ultrasonic module and the front obstacle through the ultrasonic module, the smaller the distance, the greater the radius of the circle in the Processing interface.

Components

- 1 * Adeept Arduino UNO R3 Board
- 1 * Ultrasonic Distance Module
- 1 * USB Cable
- 1 * 4-Pin Wires

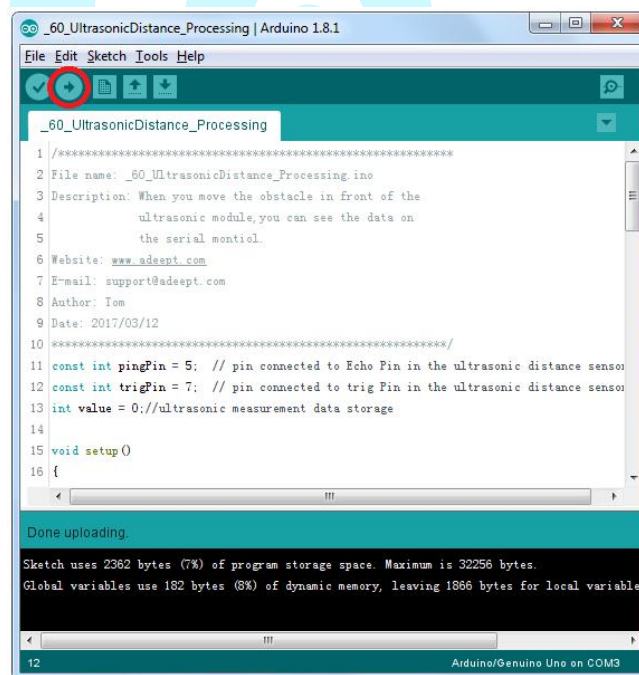
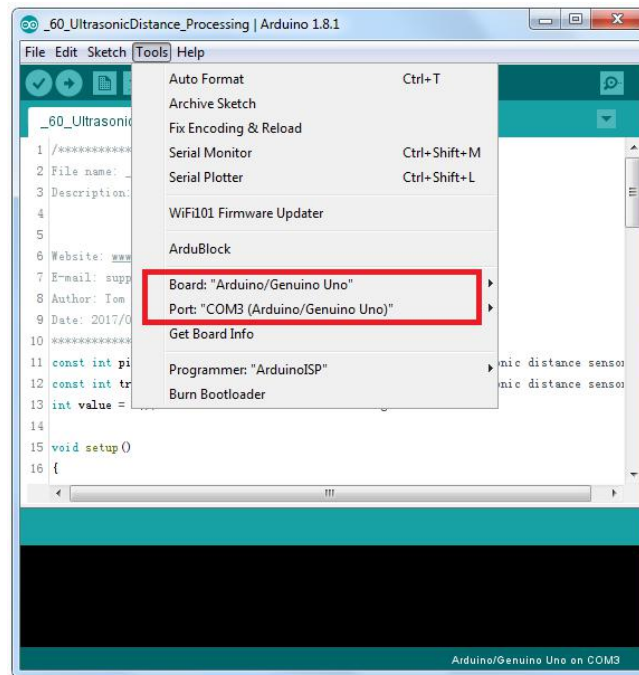
Experimental Procedures

Step 1: Build the circuit

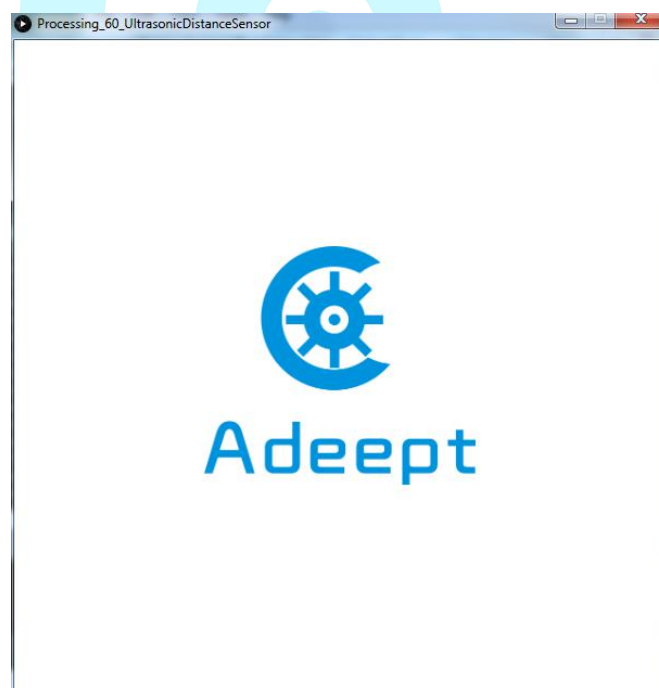
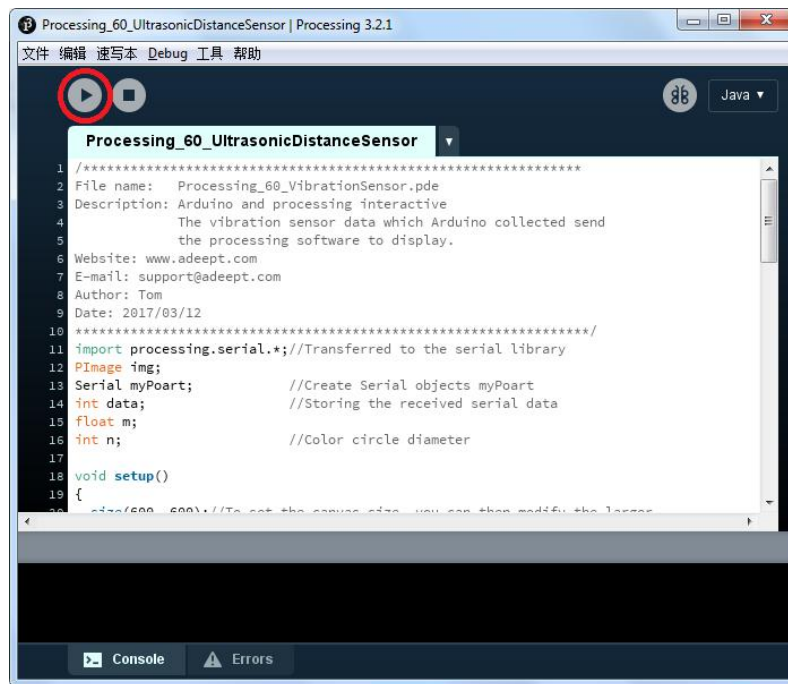


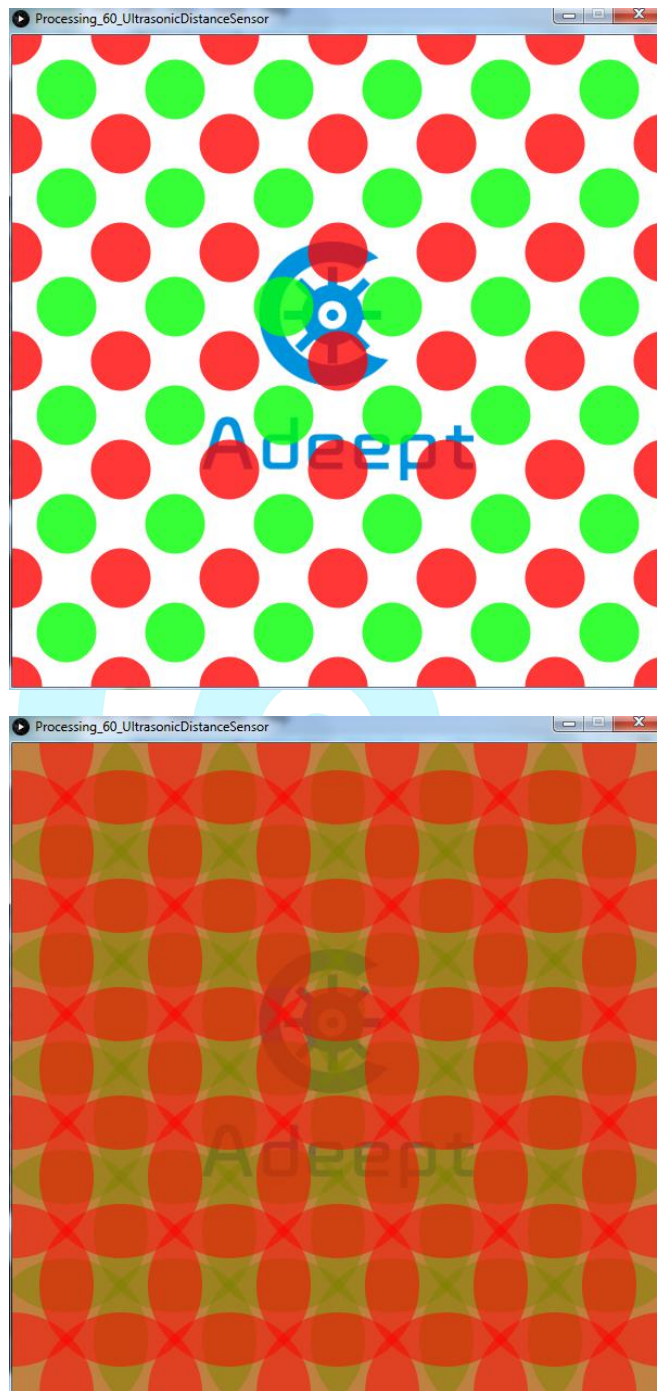
Step 2: Program _60_UltrasonicDistance_Processing.ino

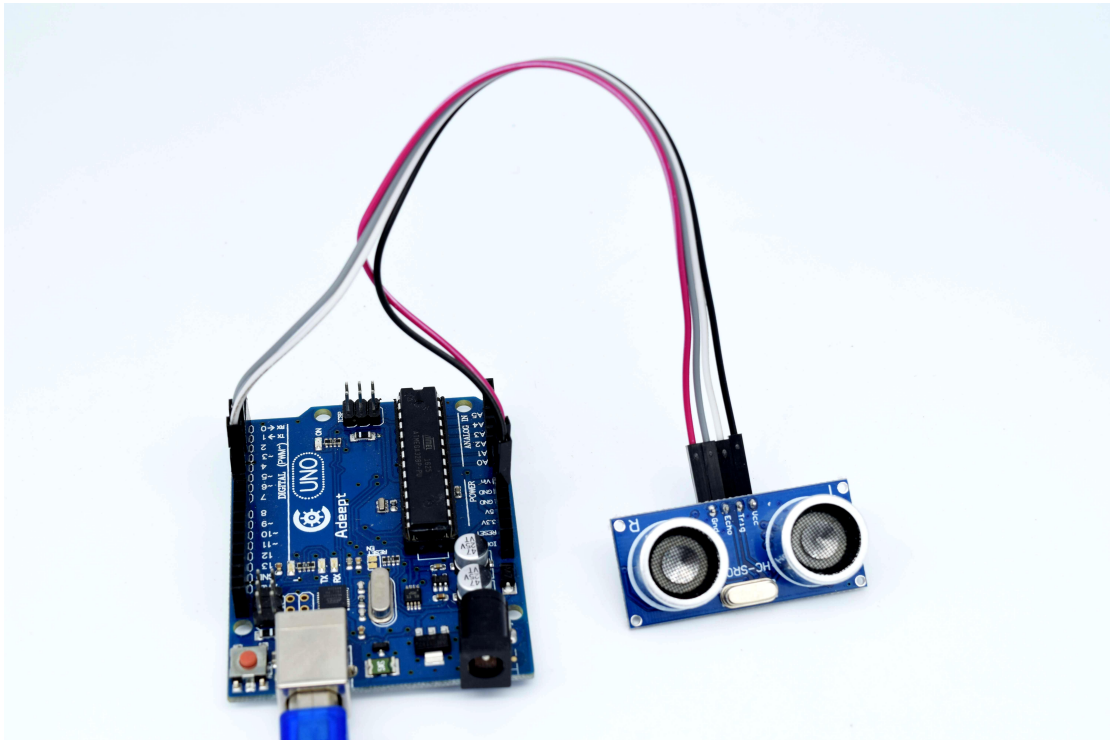
Step 3: Compile and download the sketch to the UNO R3 board.



Step 4: Run the Processing software (Processing_60_UltrasonicDistanceSensor.pde)







Adeept

Afterword

Thanks for purchasing our product and reading the manual! If you spot any errors or have any ideas or questions for the product and this guide, welcome to contact us! We will correct them if any as quickly as possible.

After completing all projects in the guide, you should have some knowledge of the book and Arduino, thus you can try to change the car into other projects by adding more Adeept modules or changing the code for extended functions.

For more information about Arduino, Raspberry Pi, smart car robot, or robotics, etc., please follow our website www.adeept.com. We will introduce more cost-effective, innovative and intriguing products!

Thanks again for choose Adeept product!



Adeept



Adeept

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