Getting Started with Mixly

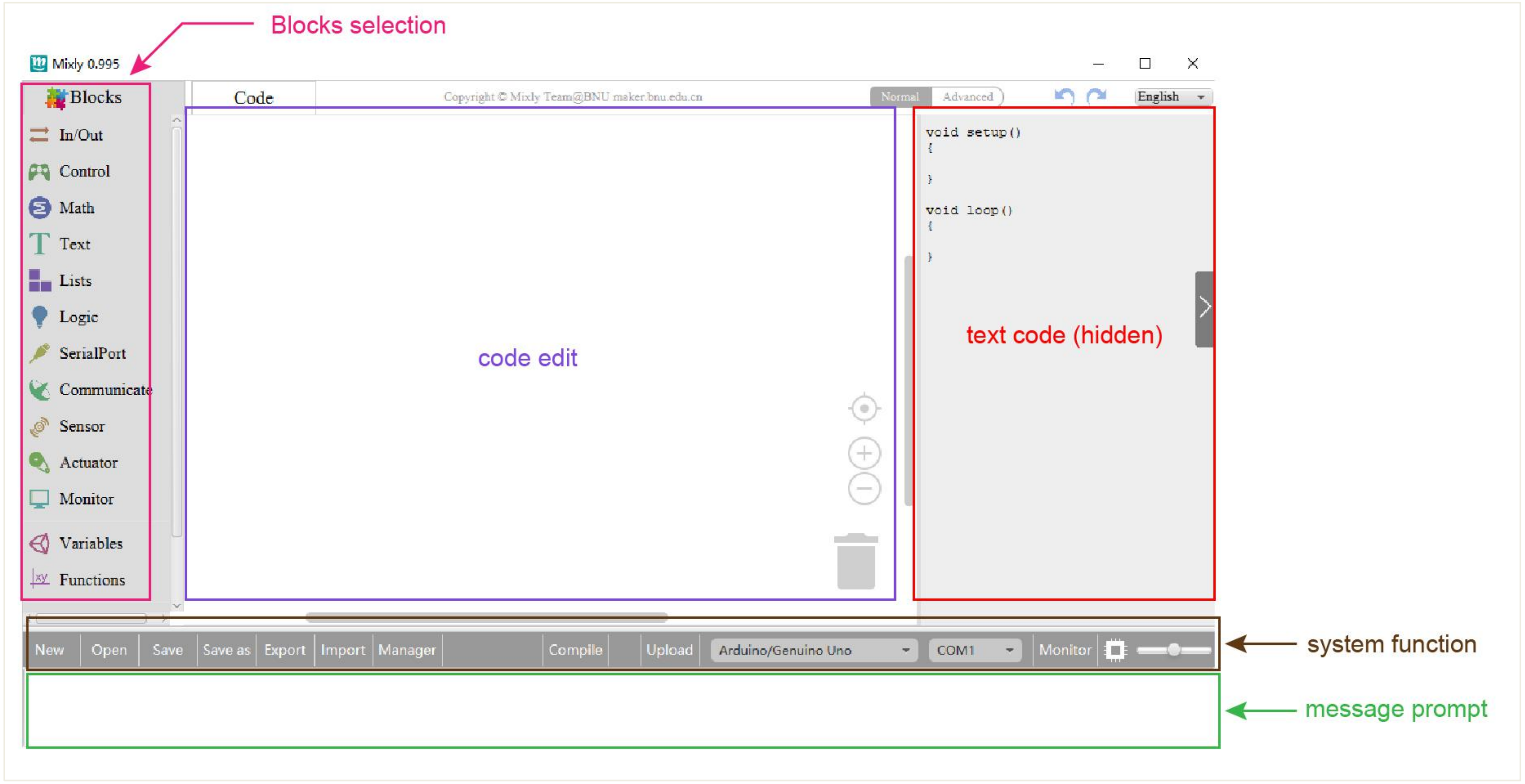
1) Introduction for Mixly

Mixly is a free open-source graphical Arduino programming software, based on Google's Blockly graphical programming framework, and developed by Mixly Team@ BNU.
It is a free open-source graphical programming tool for creative electronic development; a complete support ecosystem for creative e-education; a stage for maker educators to realize their dreams.
Although there is an Ardublock graphical programming software launched by Arduino official, Ardublock is not perfect enough, and many common functions cannot be realized.

2) Interface Functions of Mixly

1.1 System Functions

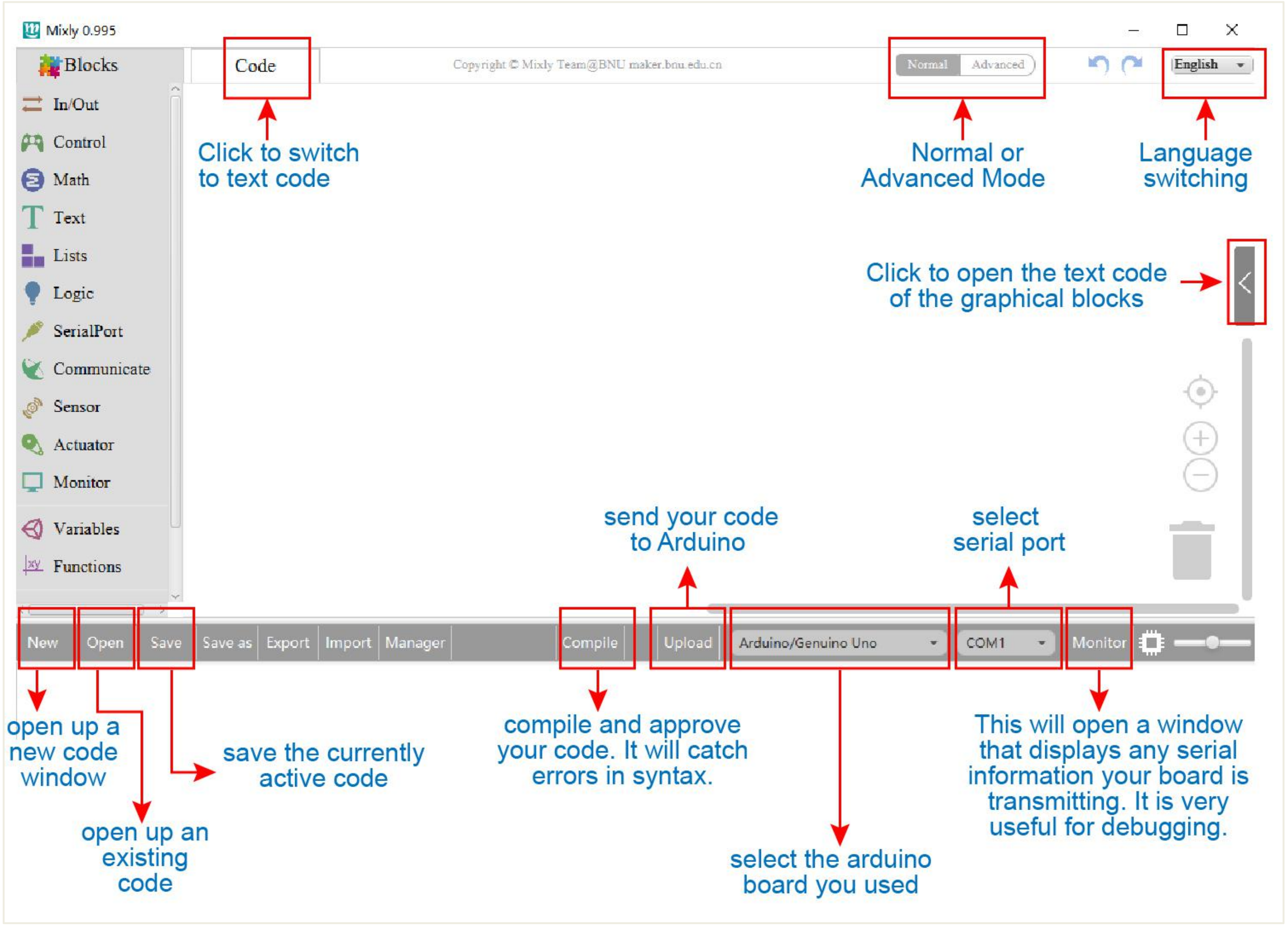
Look at the main interface of Mixly, it includes five parts, that is, Blocks selection, code edit, text code (hidden), system function and message prompt area. Shown below.



```
void setup()
{

}

void loop()
{

}
```

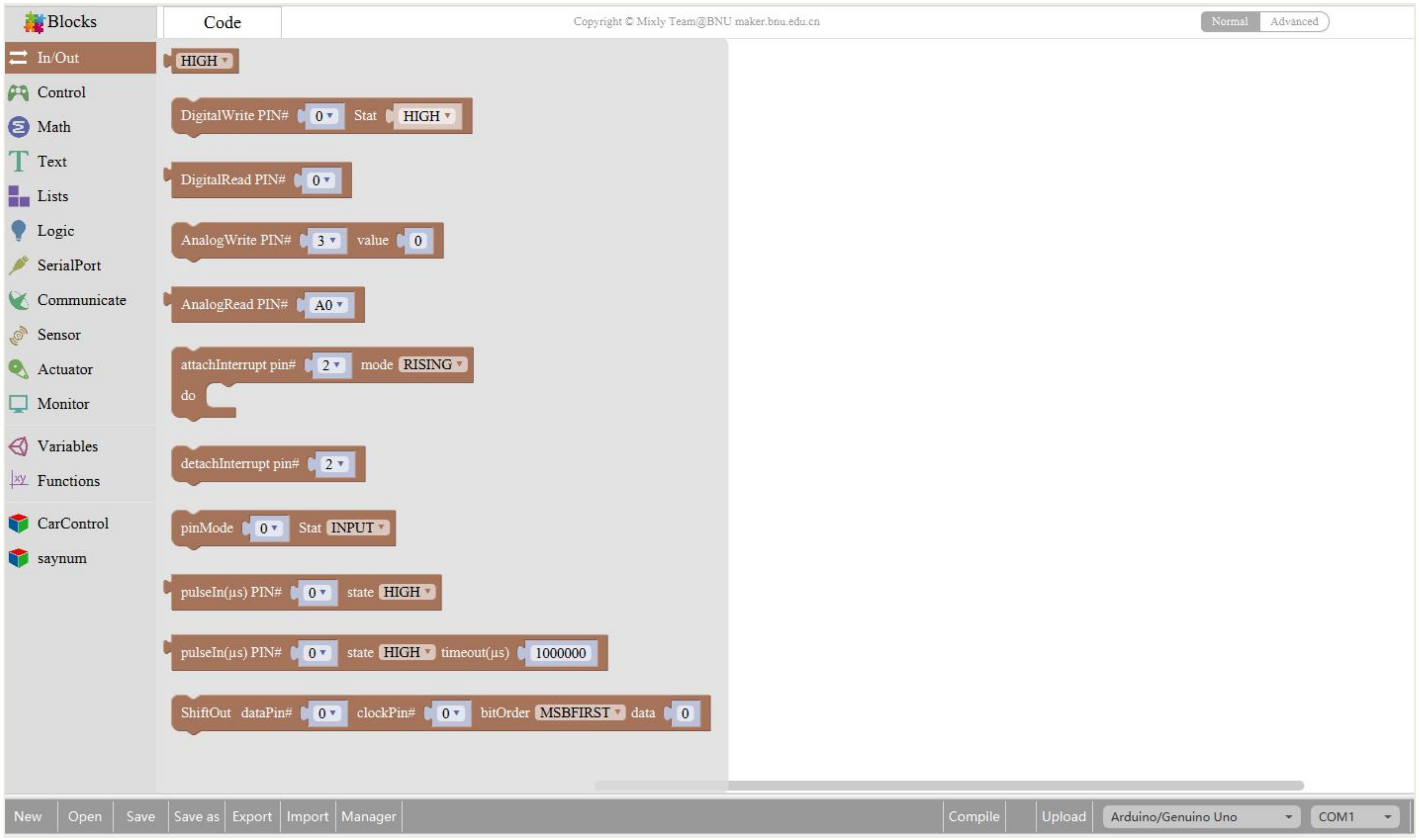
Some common functions:

Through this interface, you can complete the code compile、upload、save and manage. It support four remove methods: drag it left out code window, or drag to Recycle Bin, delete key, or right-click to delete block. It supports four languages: English、Español (Spanish)、中文简体(Chinese Simplified)、中文繁体(Chinese Traditional).



Mixly 0.995
Blocks
In/Out
Control
Math
Text
Lists
Logic
SerialPort
Communicate
Sensor
Actuator
Monitor
Variables
Functions
Code
Copyright © Mixly Team@BNU maker.bnu.edu.cn
Normal
Advanced
English
Click to switch to text code
Normal or Advanced Mode
Language switching
Click to open the text code of the graphical blocks
send your code to Arduino
select serial port
New
Open
Save
Save as
Export
Import
Manager
Compile
Upload
Arduino/Genuino Uno
COM1
Monitor
open up a new code window
open up an existing code
save the currently active code
compile and approve your code. It will catch errors in syntax.
select the arduino board you used
This will open a window that displays any serial information your board is transmitting. It is very useful for debugging.

1.2 In/Out Block



HIGH

DigitalWrite PIN# 0 Stat HIGH

DigitalRead PIN# 0

AnalogWrite PIN# 3 value 0

AnalogRead PIN# A0

attachInterrupt pin# 2 mode RISING
do

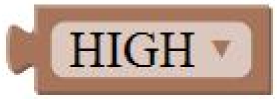
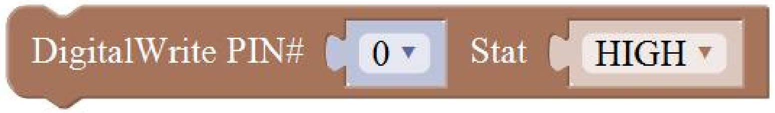
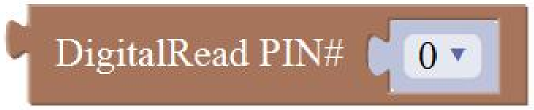
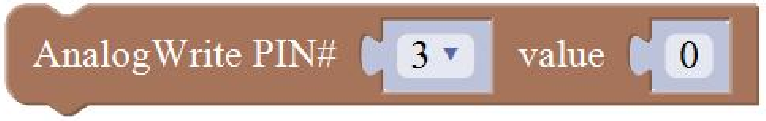
detachInterrupt pin# 2

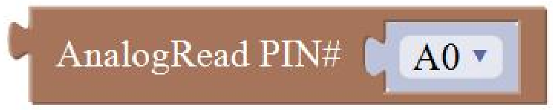
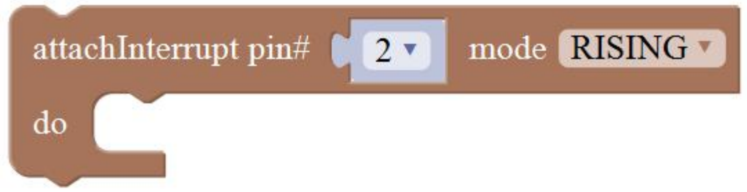
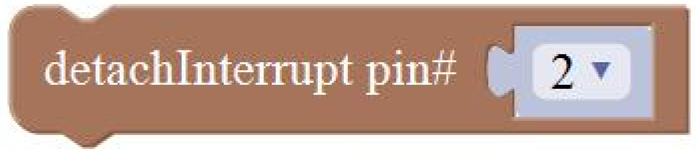
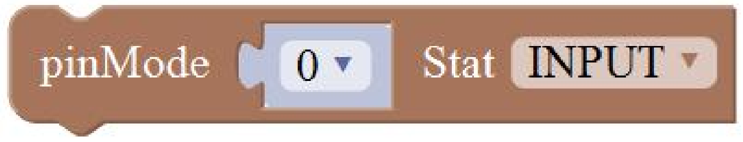
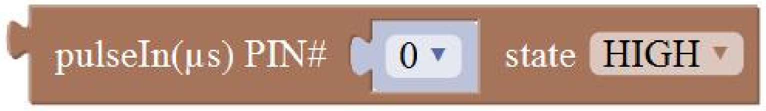
pinMode 0 Stat INPUT

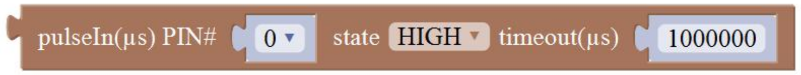
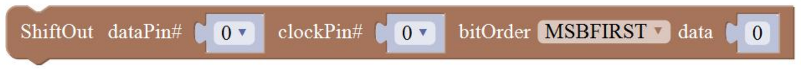
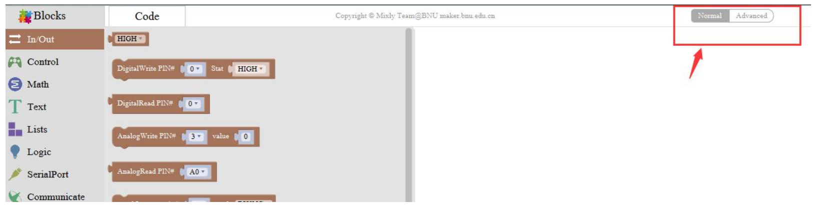
pulseIn(μs) PIN# 0 state HIGH

pulseIn(μs) PIN# 0 state HIGH timeout(μs) 1000000

ShiftOut dataPin# 0 clockPin# 0 bitOrder MSBFIRST data 0

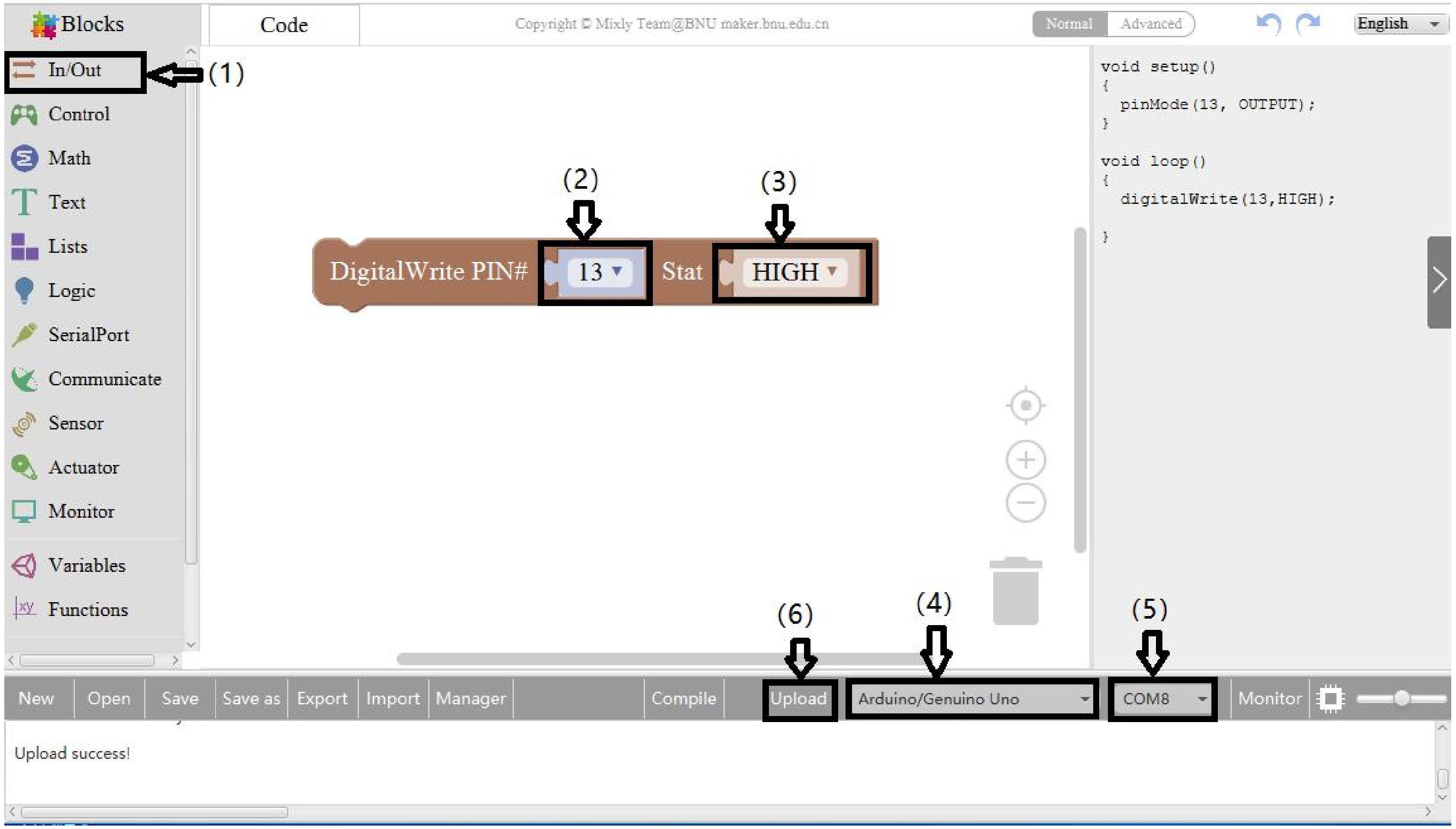
NO.	BLOCK ICON	DEFINITION
1		Returns HIGH or LOW voltage
2		Write digital value to a specific Port. Digital Output: set the HIGH or LOW output for IO pins
3		Returns a digital value of a specific Port. Digital IO Read Pin, generally used to read the HIGH or LOW level detected by Digital sensor
4		Write analog value between 2 and 255 to a specific Port. Analog Output: set the Analog value output by Analog IO pins (0~255).

5		Returns value between 0 and 1023 of a specific Port. Analog IO Read Pin, generally used to read the Analog value detected by Analog sensor.
6		External Interrupts function, with three trigger interrupt modes RISING, FALLING, CHANGE.
7		Detachs interrupt to a specific Port. Turn off the given interrupt function.
8		Set the IO pins as Output or Input state
9		Read the continuous time of HIGH or LOW pulse from IO pins (generally used for ultrasonic ranging)

10		Read a pulse (either HIGH or LOW) on a pin within a time set in timeout.
11		Set the ShiftOut data pin, clock pin. Output the data needed from the bitOrder MSBFIRST or LSBFIRST (Most Significant Bit First, or, Least Significant Bit First). Generally used for controlling the 74HC595 CHIP.
12	 Blocks Code Copyright © Mixly Team@BNU maker.bnu.edu.cn Normal Advanced In/Out Control Math Text Lists Logic SerialPort Communicate HIGH DigitalWrite PIN# 0 Stat HIGH DigitalRead PIN# 0 AnalogWrite PIN# 3 value 0 AnalogRead PIN# A0	This is the function interface under Normal mode. If select Advanced mode, the functions will be more.

For example:

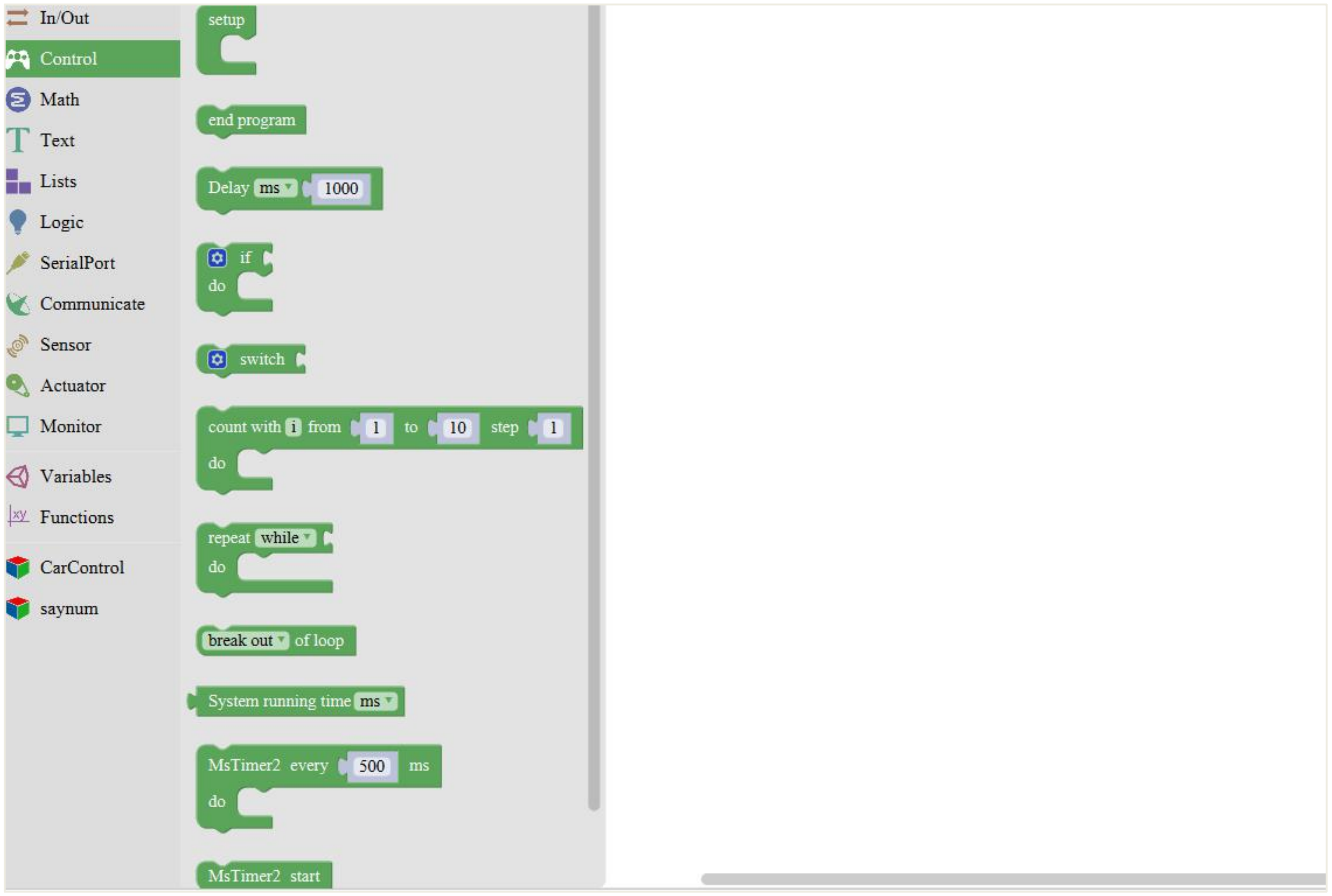
Connect your Arduino Uno board, then follow the steps below to light the Pin13 led on Arduino UNO.

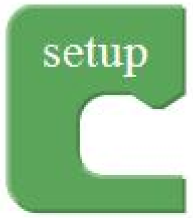
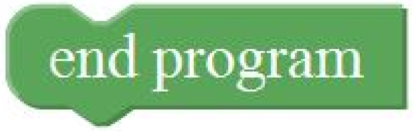
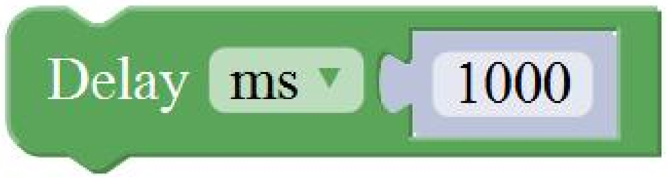


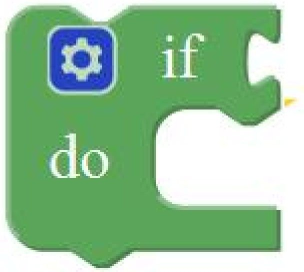
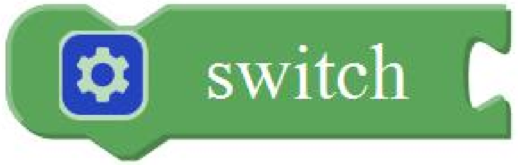
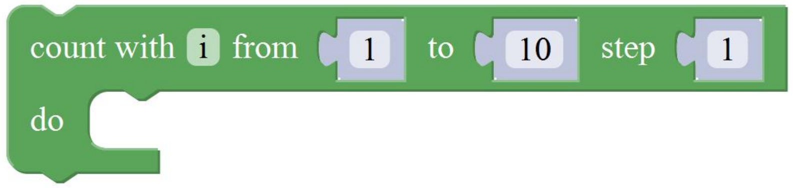
```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13,HIGH);

}
```

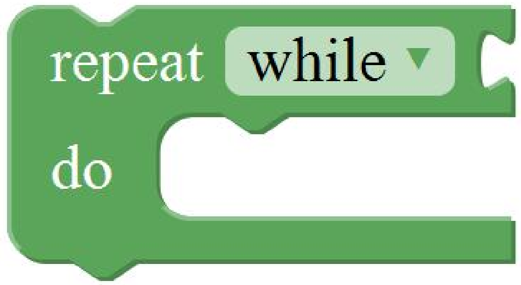
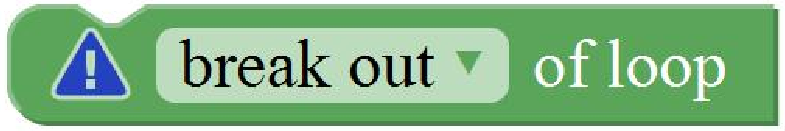
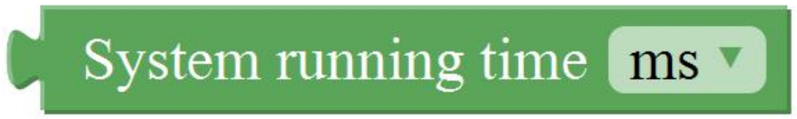
```
Upload success!
```

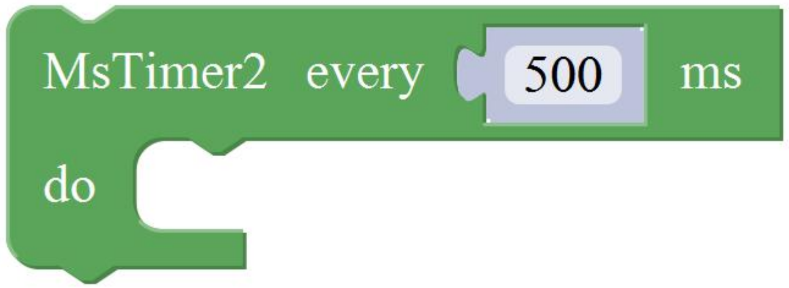
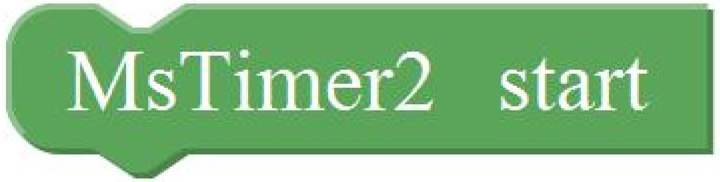
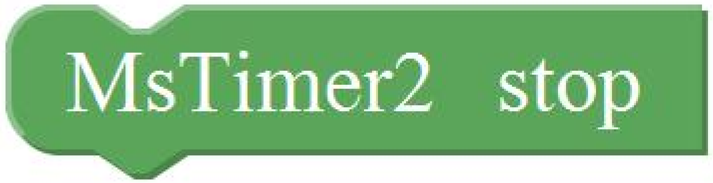
1.3 Control Block



NO.	BLOCK ICON	DEFINITION
1		Initialization (run only once)
2		End the program, means the program will stop running when use this block.
3		Delay function, click to select **ms** or **us** (pause the program for the amount of time (in milliseconds) specified as parameter. There are 1000 milliseconds in a second.)

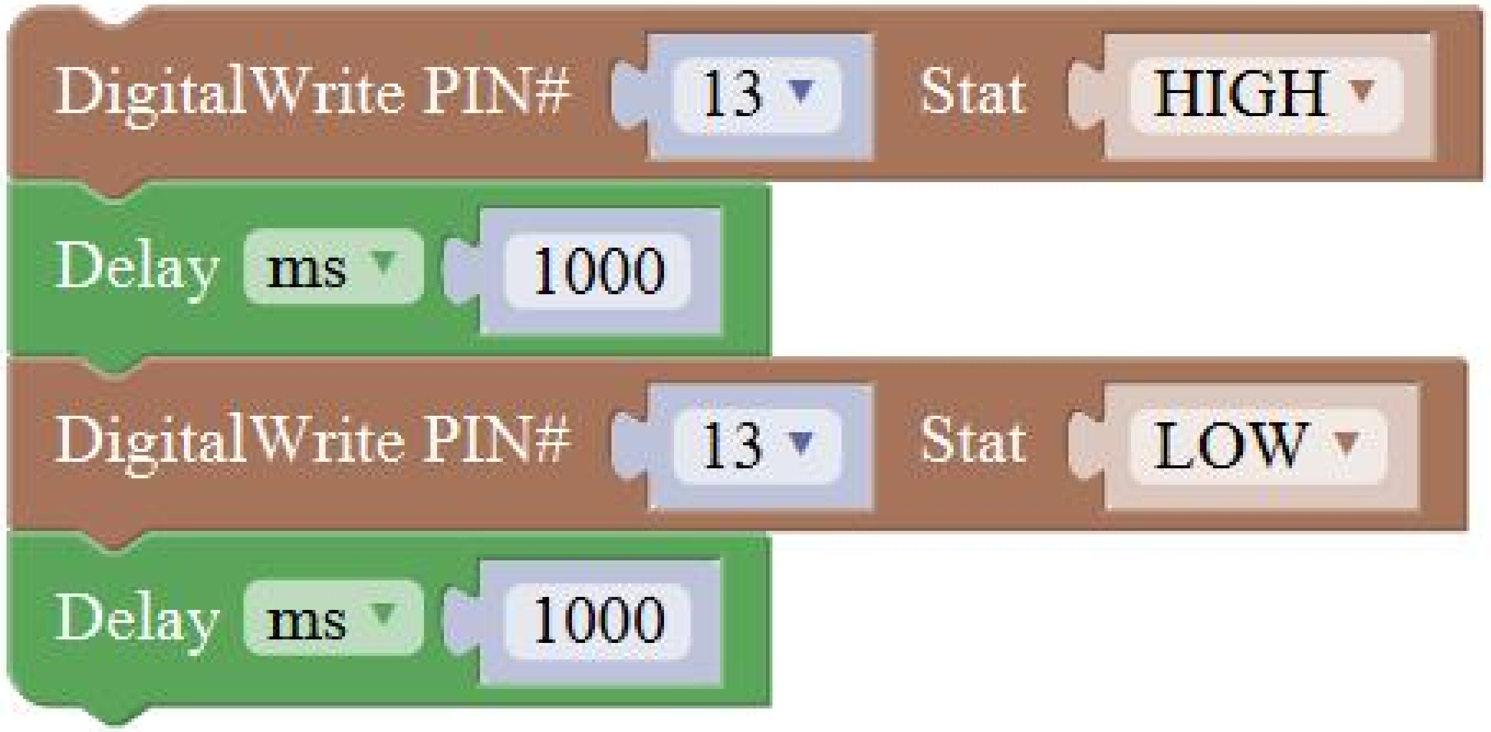
4		**if_do** function (first evaluate a value be true or false, if a value is true, then do some statement. You can click the blue gear icon to select the **else if** block or **else** block.)
5		**switch** function. You can click the blue gear icon to select the **case** block or **default** block. (used to evaluate several programs then execute the corresponding function matched with program.)
6		Equal to **for** statement.

7		A **while** loop statement.
8		**break** function, used to exit from the containing loop.
9		**millis()** function, returns the system running time since the program started. (The unit can be **ms** (milliseconds) or **μs** (microsecond)).

10		Timer interrupt function, that is, set a trigger interrupt for the amount of time (in milliseconds) specified as parameter.
11		Timer interrupt start block
12		Timer interrupt stop block

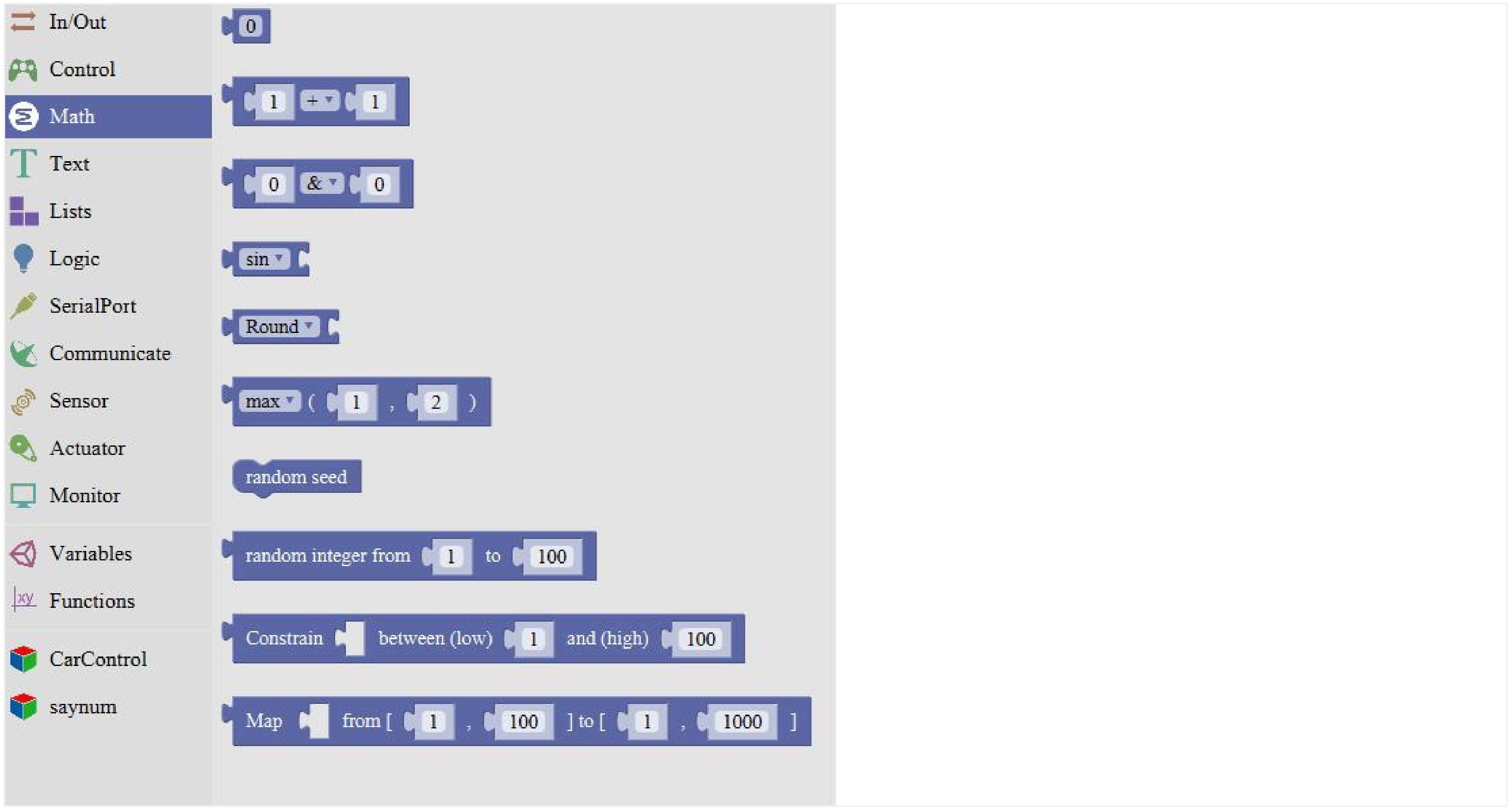
For example:

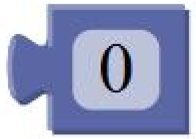
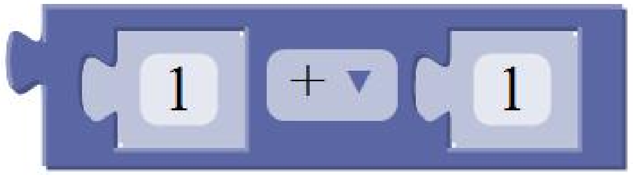
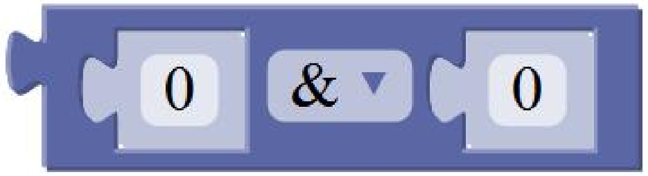
Compile and upload the program below to your Arduino board, you should see Pin13 LED on Arduino UNO continue to flash.(with an interval of 1s, equal to 1000ms)

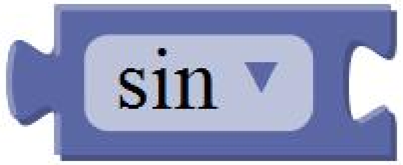
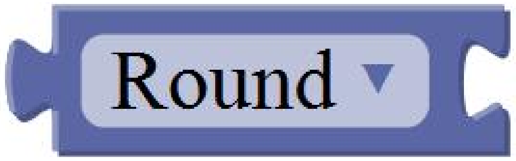


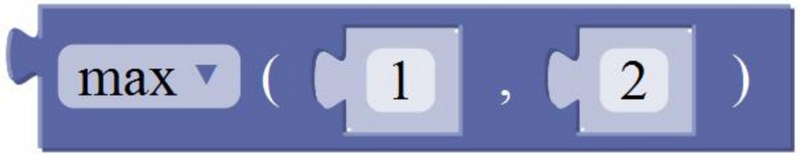
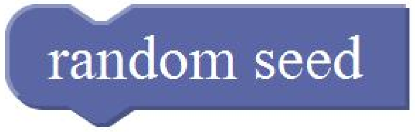
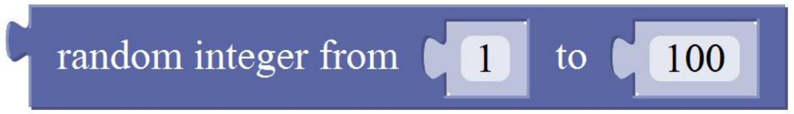
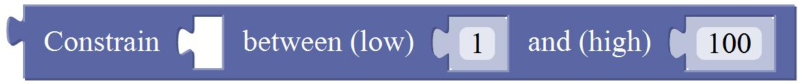
```
DigitalWrite PIN# 13 Stat HIGH
Delay ms 1000
DigitalWrite PIN# 13 Stat LOW
Delay ms 1000
```

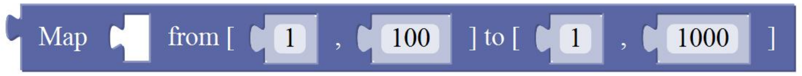
1.4 Math Block



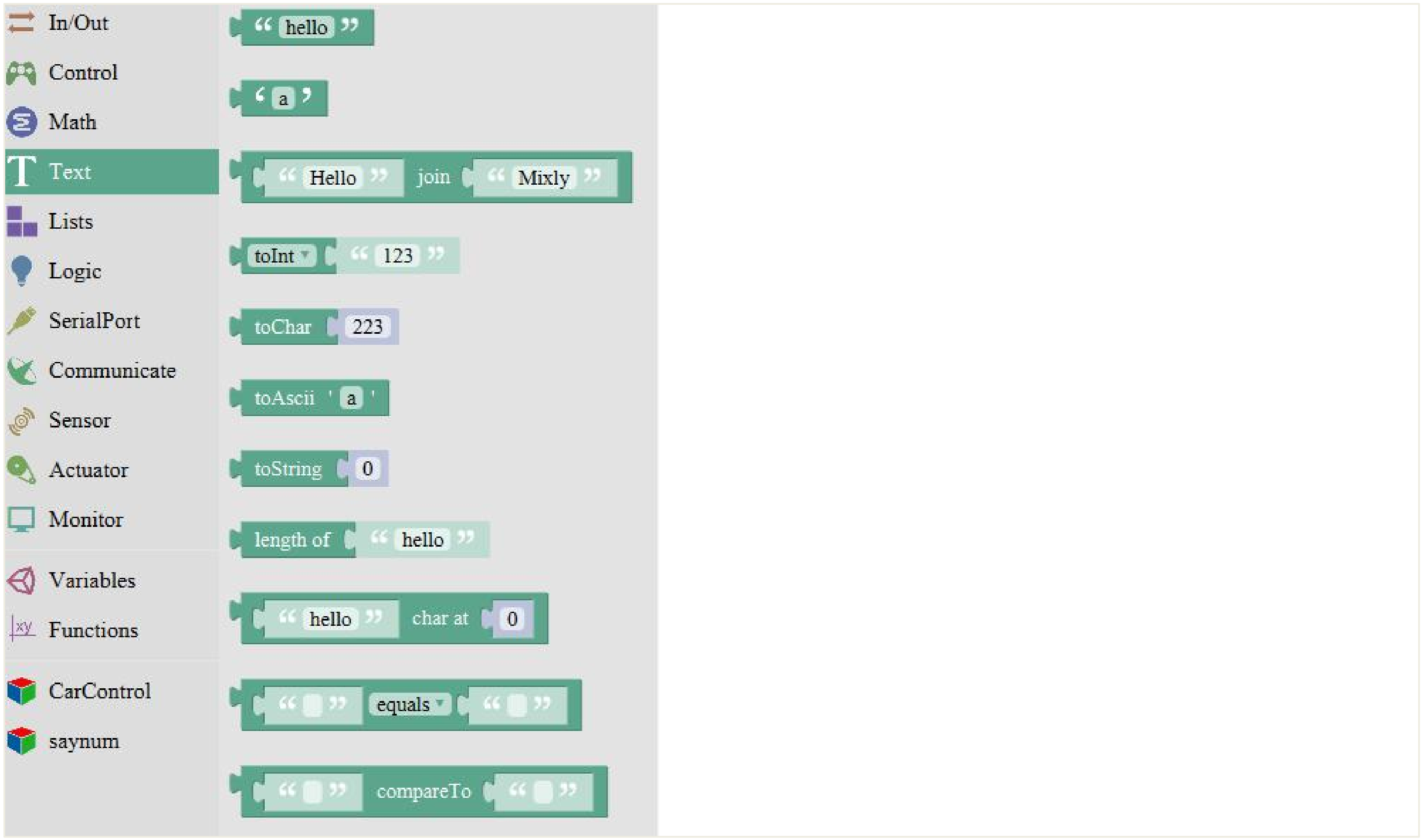
NO.	BLOCK ICON	DEFINITION
1		A number
2		Click to select the Arithmetic Operators: ＋(addition); －(subtraction); x (Multiplication); ÷ (division); % (remainder); ^ (bitwise xor)
3		Click to select the & (bitwise end); I (bitwise or); << (bitshift left); >> (bitshift right)

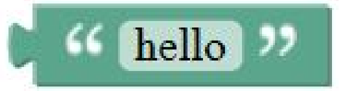
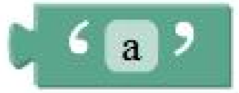
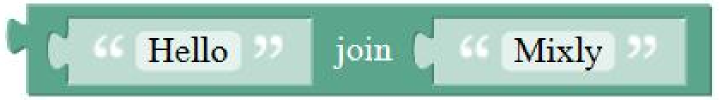
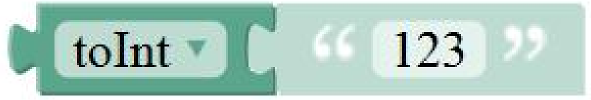
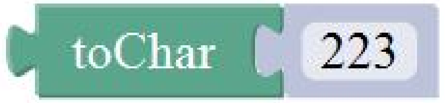
4		Click to select the **sin**; **cos**; **tan**; **asin; acos; atan; ln; log10; e^; 10^; ++ (increment)** ; **-- (decrement)**
5		Click to select the **Round; Ceil; Floor; abs; sq; sqrt** **Round:** Returns the integer part a number using around. **Ceil:** Returns the integer part a number using ceil. **Floor:** Returns the integer part a number using floor. **abs:** Return the absolute value of a number. **sq:** Return the square of a number. **sqrt:** Return the square root of a number.

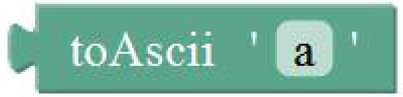
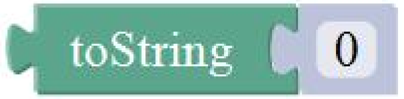
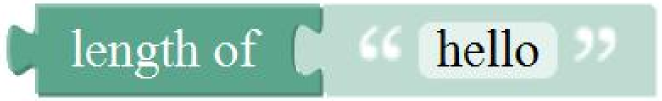
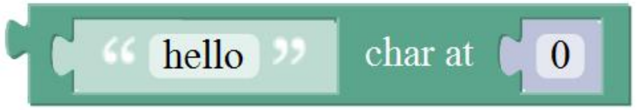
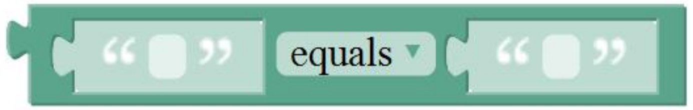
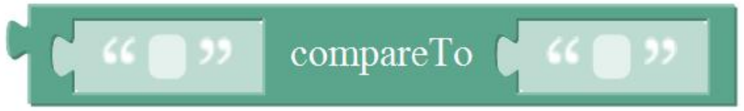
6		If select the **max**, returns the larger number; if select the **min**, returns the smaller number.
7		Initialize the random seed
8		Return a random integer between the two specified limits, inclusive.
9		Constrain a number to be between the specified limits (inclusive). (generally used to constrain an analog value read from sensor)

10	 Map from [1 , 100] to [1 , 1000]	Map a number from the first interval to the second interval. (For instance, potentiometer-controlled servo, map the range of potentiometer (0, 1023) to the angle of servo (0, 180)).

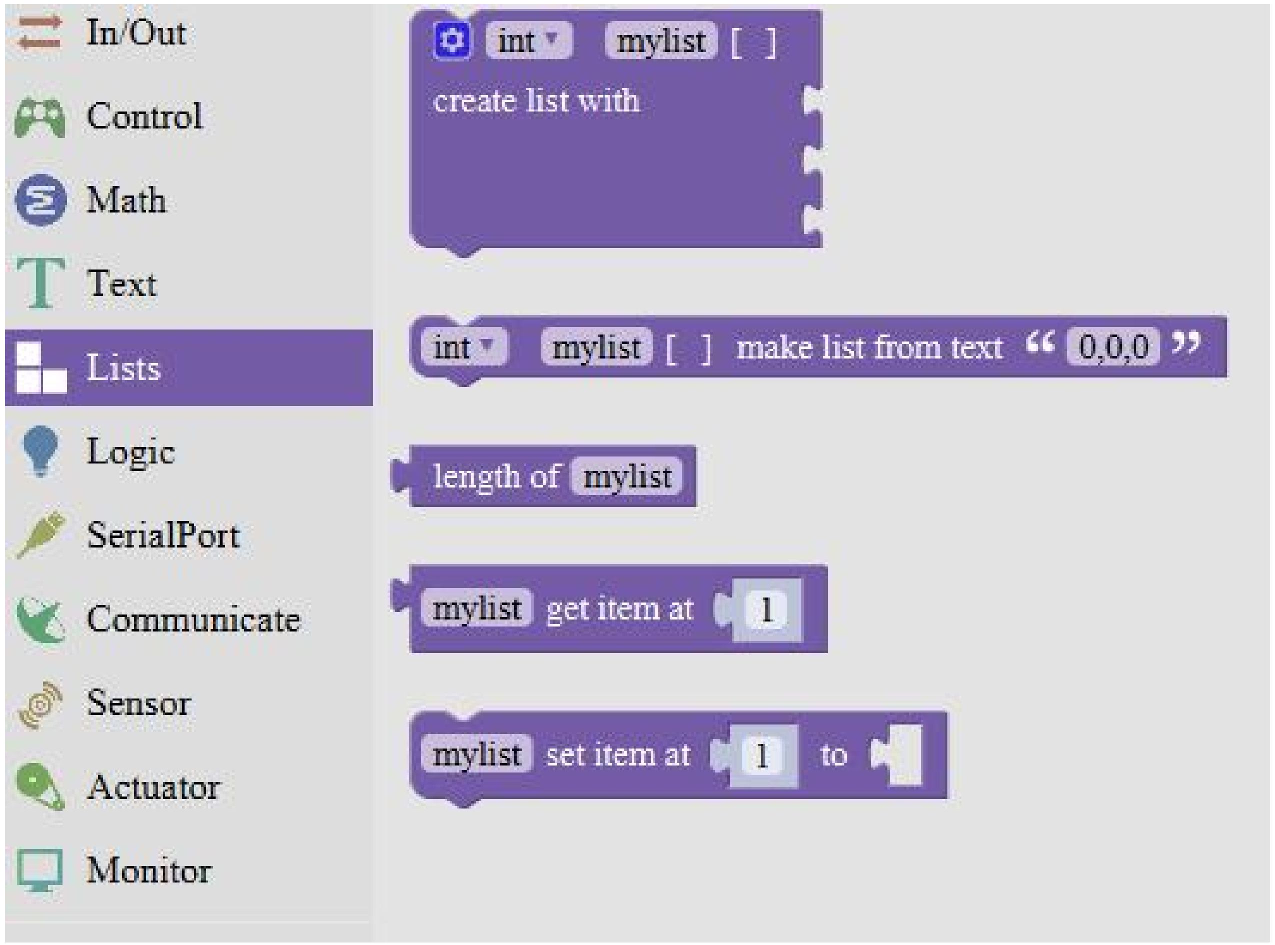
1.5 Text Block

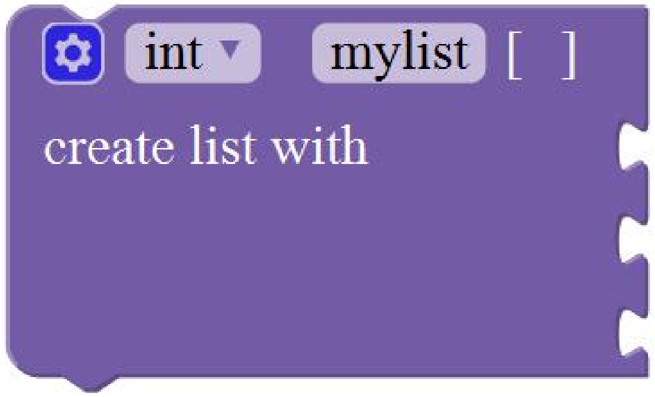
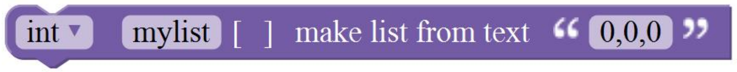
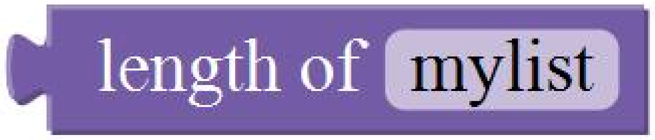


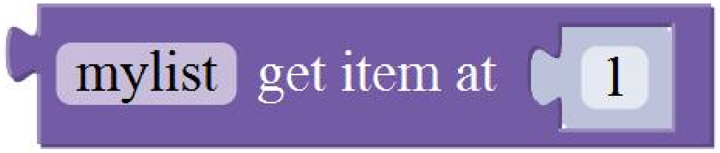
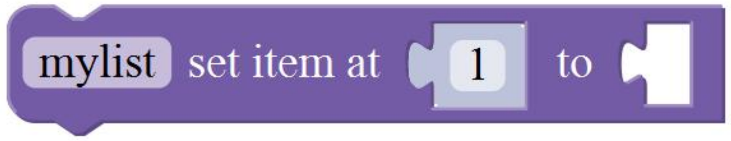
NO.	BLOCK ICON	DEFINITION
1		character string: a letter, word, or line of text.
2		A character
3		Creates a piece of text by joining together two piece of text. (Here Hello join Mixly equals HelloMixly)
4		Converts a string into an integer or an float.
5		Returns the char corresponding to an ASCII code (Decimal number 97 corresponding to a)

6		Returns the ASCII code corresponding to a char.
7		Converts a number into a string.
8		Calculates the length of a string
9		Output the char of a string (the char at 0 of hello is h)
10		The first string equals or startsWith or endsWith the second string, returns 1, otherwise returns 0. (if equals, both strings are abc, returns 1.)
11		Returns a decimal value of the first string subtracts the second string.

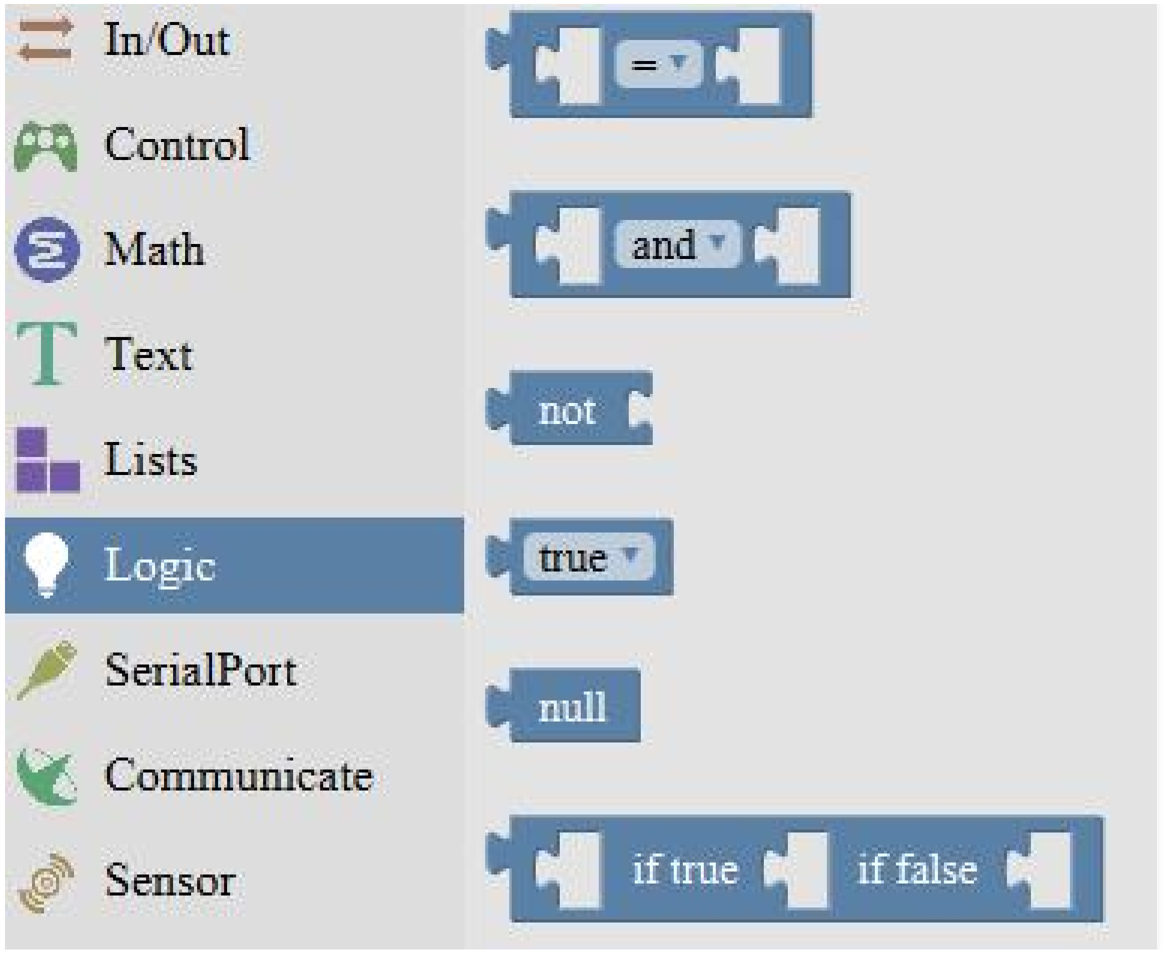
1.6 List Block

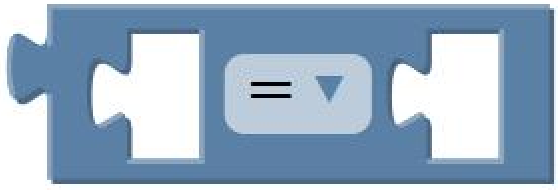


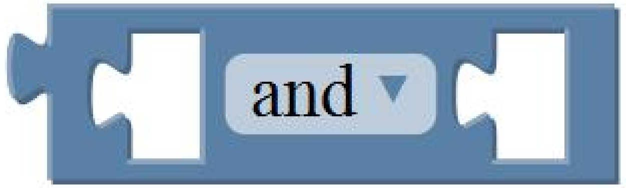
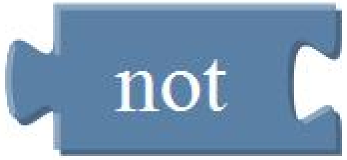
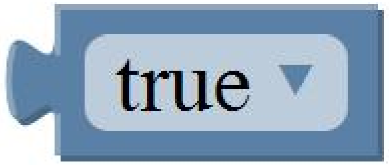
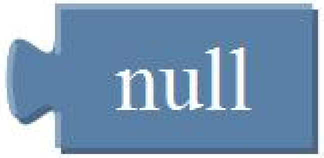
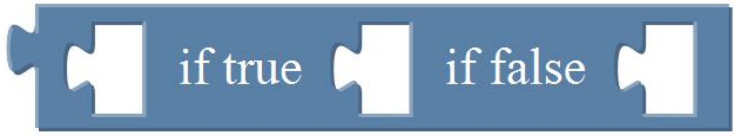
NO.	BLOCK ICON	DEFINITION
1		Create a list with any number of items
2		Creats a list from a text. (int mylist [ ]={0,0,0};)
3		Returns the length of a list

4		Returns the value of at the specified position in a list.
5		Sets the value of at the specified position in a list. Set the first item in mylist to another item.

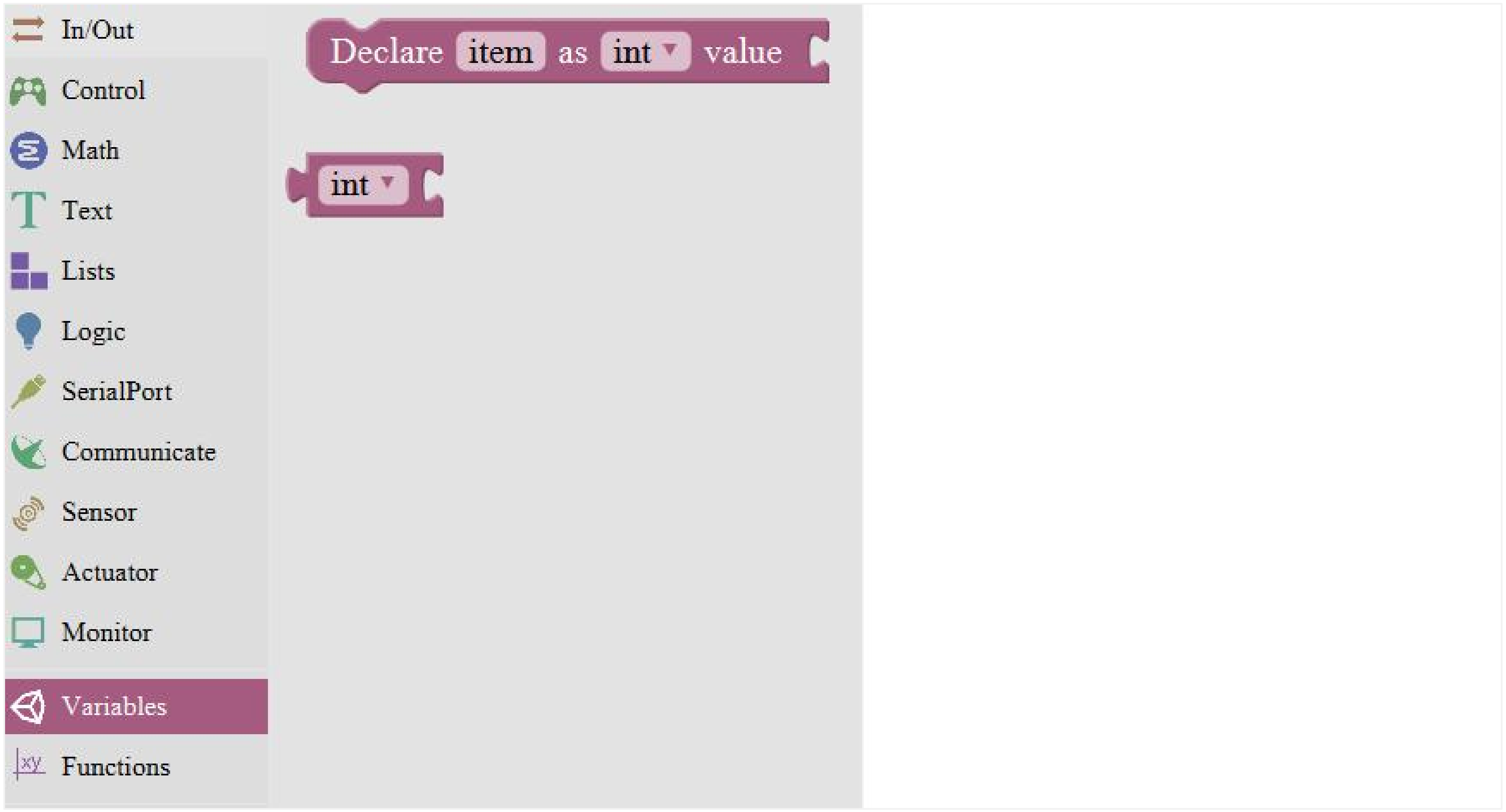
1.7 Logic Block

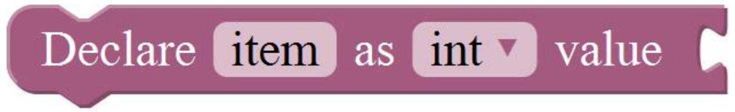
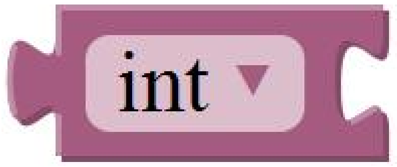


NO.	BLOCK ICON	DEFINITION
1		**logic comparision** =: Return true if both inputs equal each other. **≠** : Return true if both inputs are not equal to each other. <: Return true if the first input is smaller than the second input. **≤** : Return true if the first input is smaller than or equal to the second input. >: Return true if the first input is greater than the second input. **≥ :** Return true if the first input is greater than or equal to the second input.

2		**and:** Return true if both inputs are true; **or:** Return true if at least one of the inputs is true
3		Returns true if the input is false. Returns false if the input is true.
4		Returns either true or false.
5		Returns null
6		If the first number is true, the second number is returned, otherwise the third number.

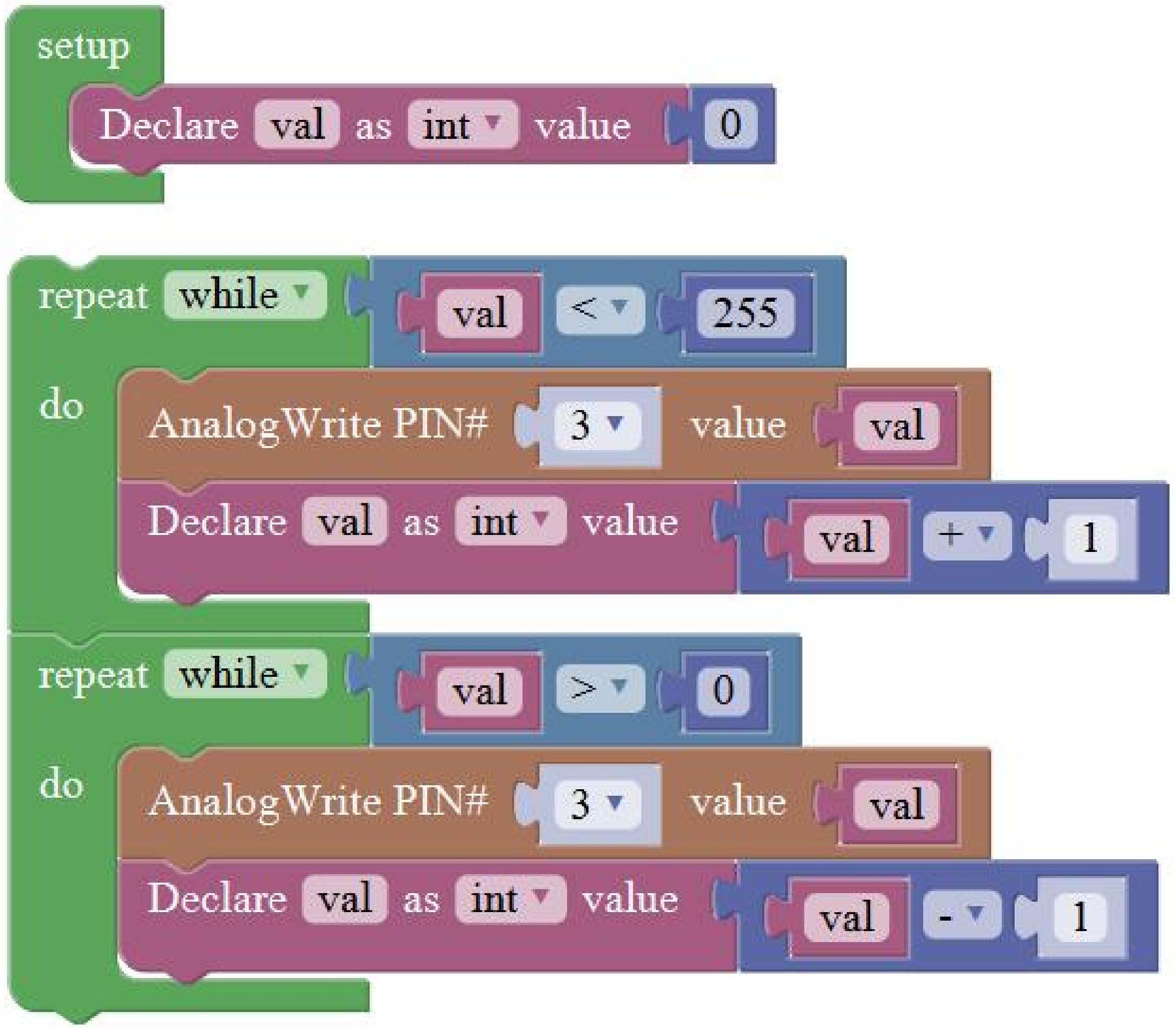
1.8 Variable Block



NO.	BLOCK ICON	DEFINITION
1		Declare and initialize a variable. Click to select int, long, float, boolean, byte, char, string
2		Define the data types

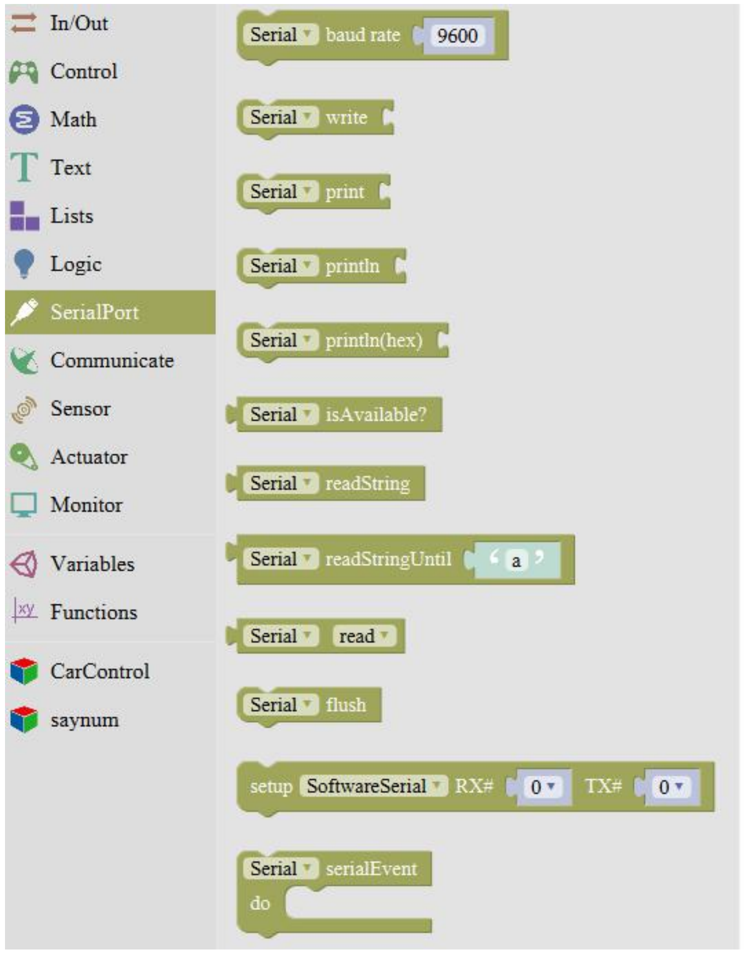
For example: LED breath

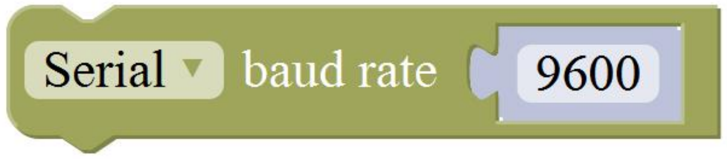
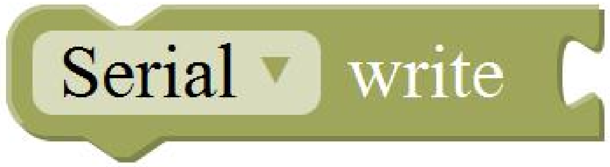
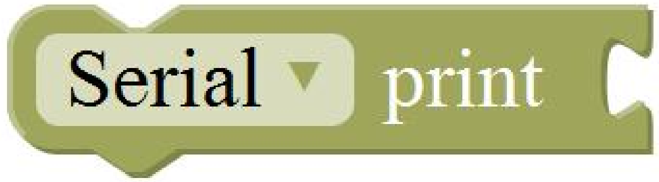
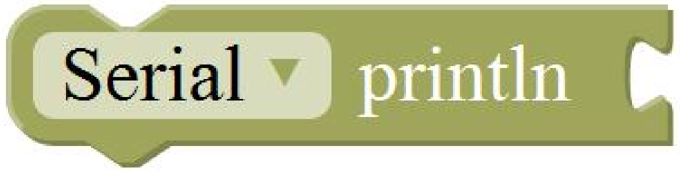
You need an Arduino Uno and one LED module. Connect the control pin of LED module to Pin 3 of Uno board (or other pins with "~", that is, those pins can output PWM signal). LED will gradually light then gradually dim, repeatedly.

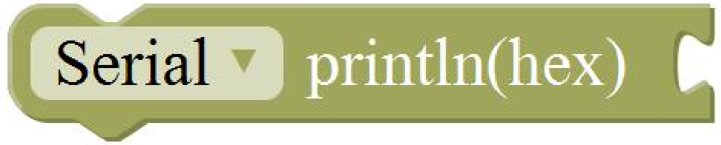
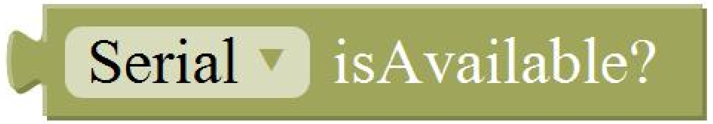
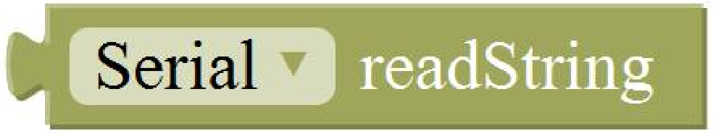
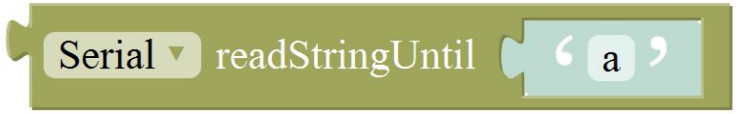
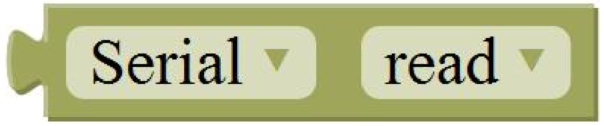


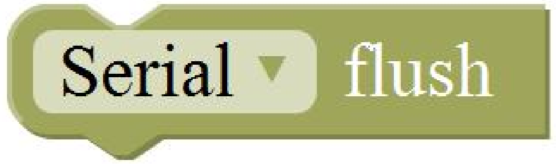
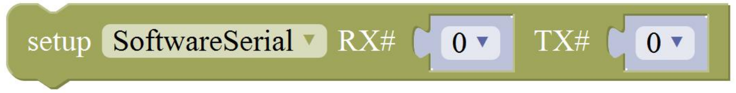
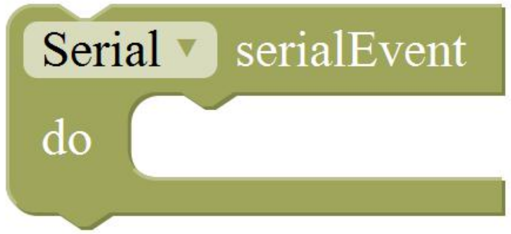
```
setup
  Declare val as int value 0

repeat while  val < 255
do
  AnalogWrite PIN# 3 value val
  Declare val as int value val + 1
repeat while  val > 0
do
  AnalogWrite PIN# 3 value val
  Declare val as int value val - 1
```

1.9 SerialPort Block



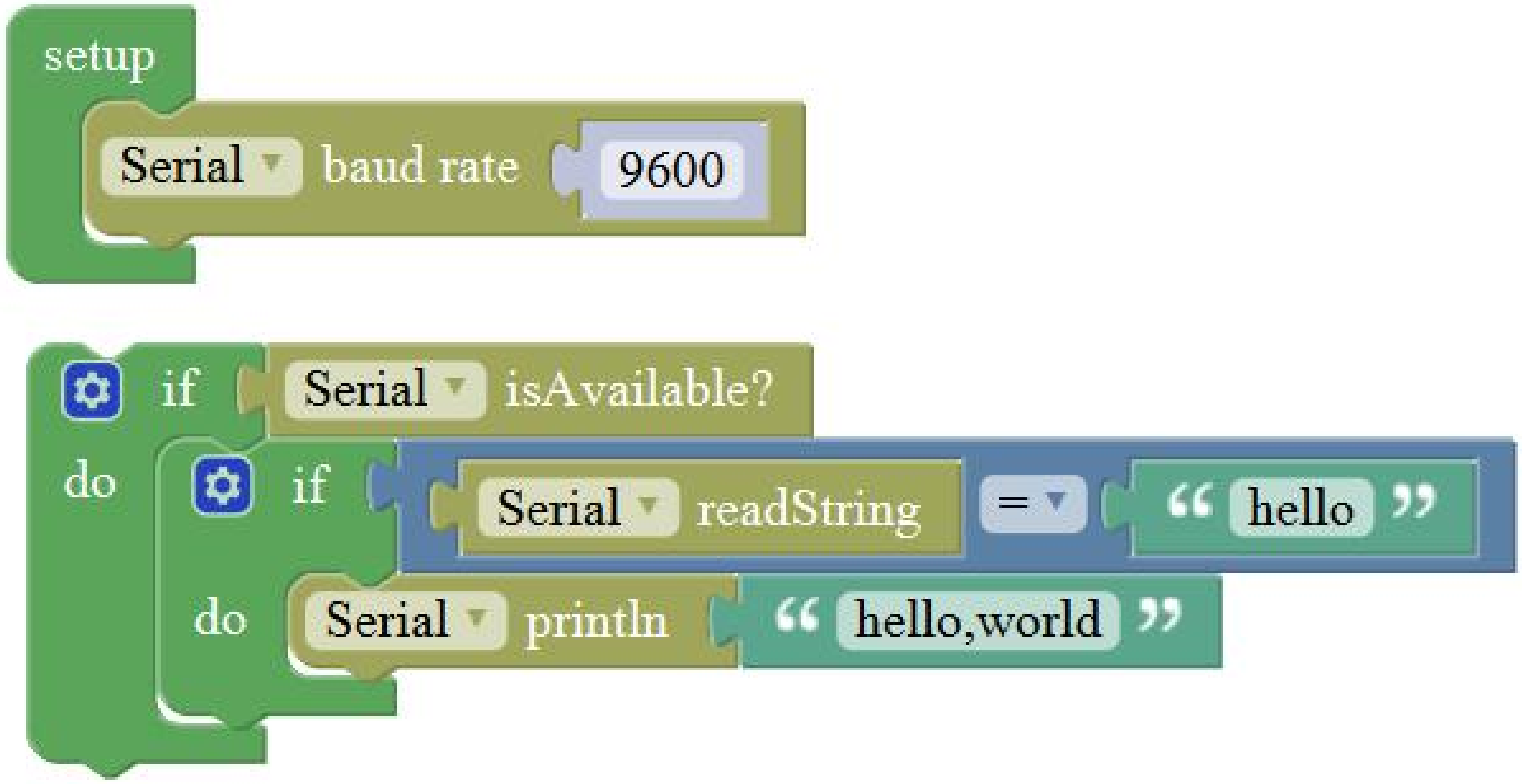
NO.	BLOCK ICON	DEFINITION
1		Set the serial buad rate to 9600
2		Write the specified number, text or other value.
3		Print the specified number, text or other value on monitor.
4		Print the specified number, text or other value on newline of monitor.

5	 Serial ▾ println(hex)	Print the specified number in hexademical format on newline of monitor.
6	 Serial ▾ isAvailable?	If the serial port is available, it returns true, otherwise returns false. (generally used in Bluetooth communication)
7	 Serial ▾ readString	Returns a string in serial port
8	 Serial ▾ readStringUntil “ a ”	A string read from serial port to a string variable, pause until read the specified character.
9	 Serial ▾ read ▾	Read the serial data by byte (generally used to read the value sent from Bluetooth) (delete the data has been read)

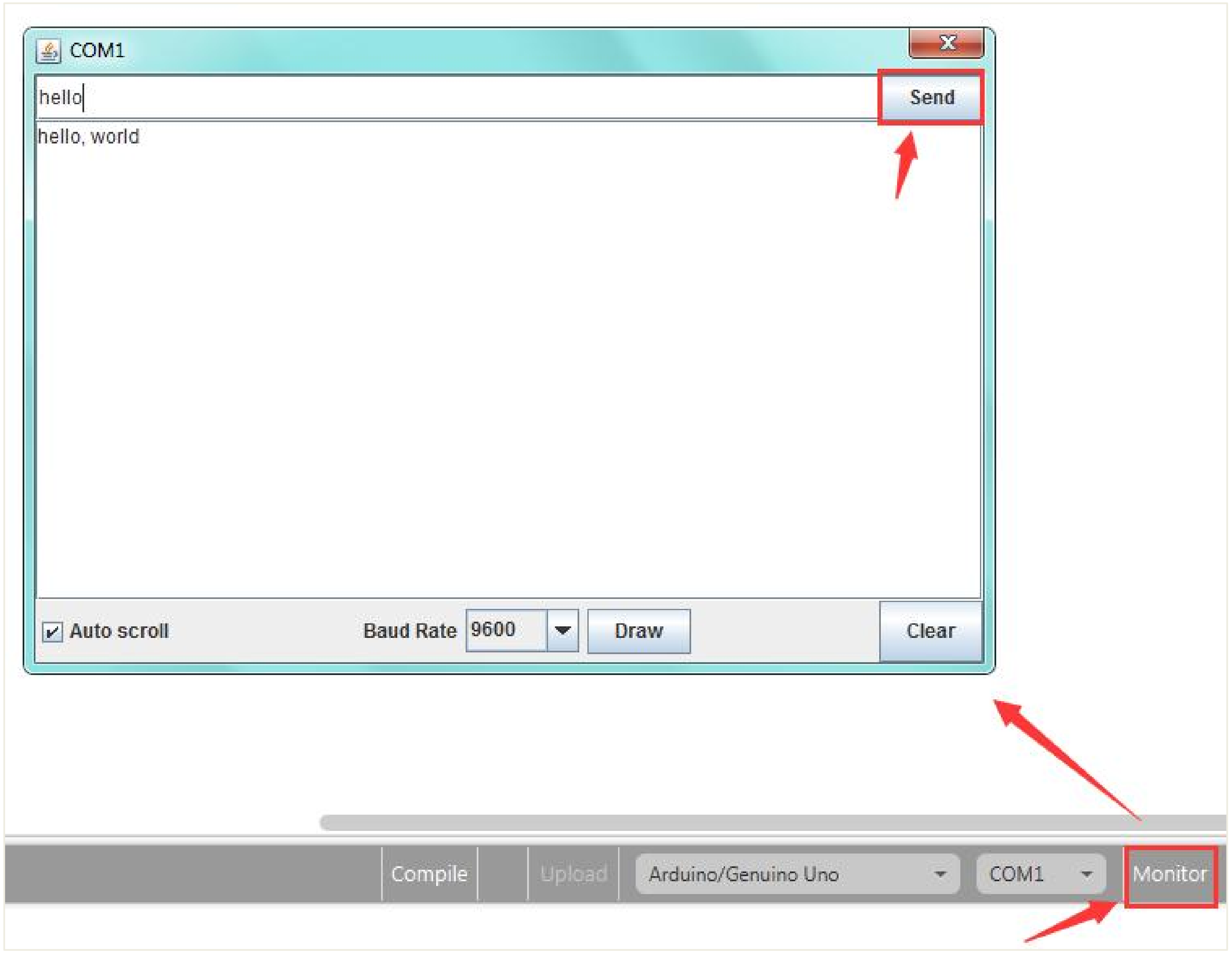
10		Wait for the output data completed
11		Set the software serial port (call this function if need to use several serial ports)
12		Event function trigger by serial port data, that is, serial port is ready to call this function. (equal to an interrupt function)

For example: serial communication

Done uploading the code, open the Arduino monitor, then enter a “hello” on the top bar, and click Send, it will print out “hello,world”.



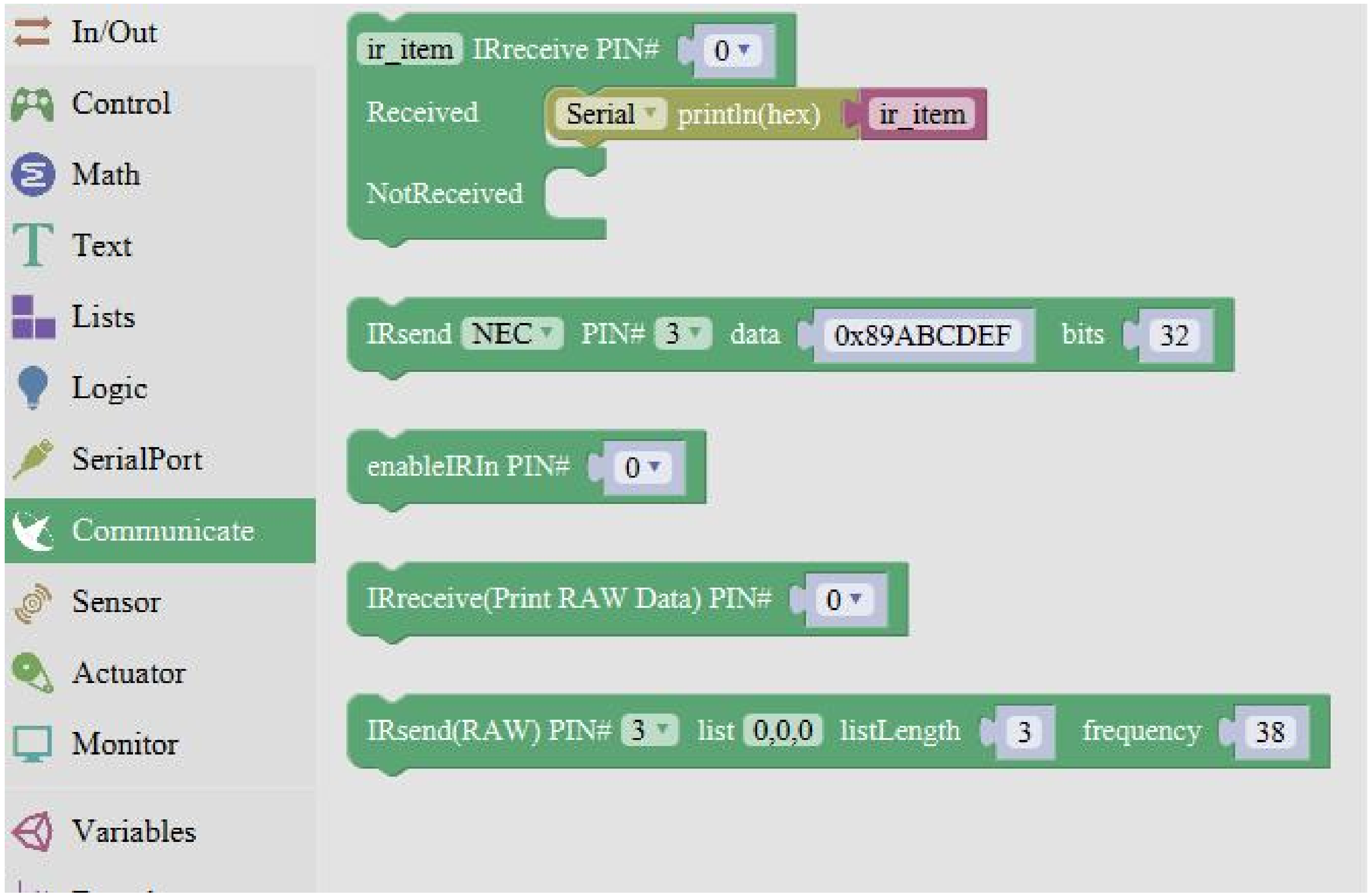
```
setup
    Serial baud rate 9600

if Serial isAvailable?
do  if Serial readString = "hello"
    do  Serial println "hello,world"
```



```
hello, world
```

COM1
hello
Send
Auto scroll
Baud Rate
9600
Draw
Clear
Compile
Upload
Arduino/Genuino Uno
COM1
Monitor

1.10 Communicate Block



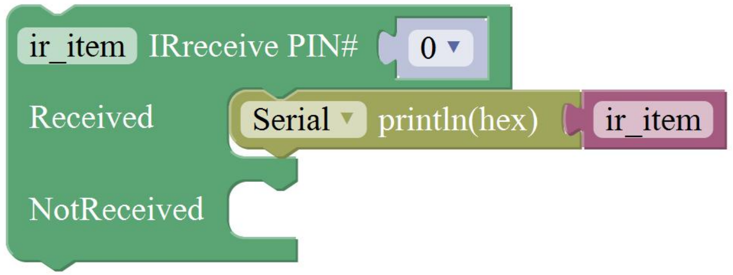
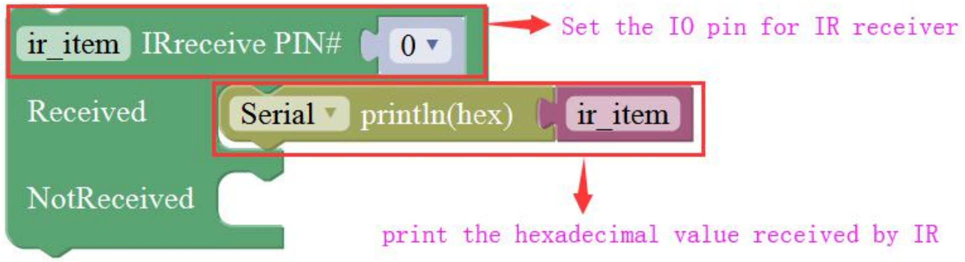
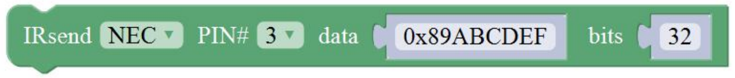
ir_item IRreceive PIN# 0
Received Serial println(hex) ir_item
NotReceived

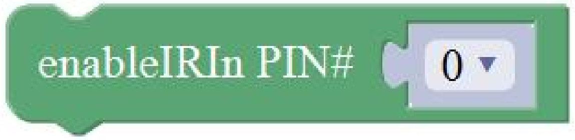
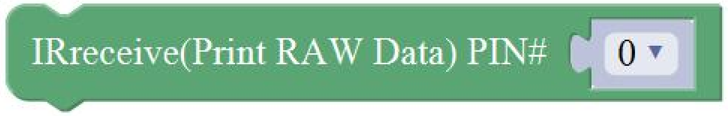
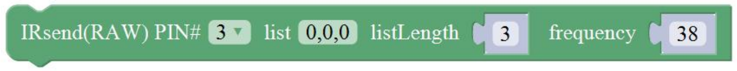
IRsend NEC PIN# 3 data 0x89ABCDEF bits 32

enableIRIn PIN# 0

IRreceive(Print RAW Data) PIN# 0

IRsend(RAW) PIN# 3 list 0,0,0 listLength 3 frequency 38

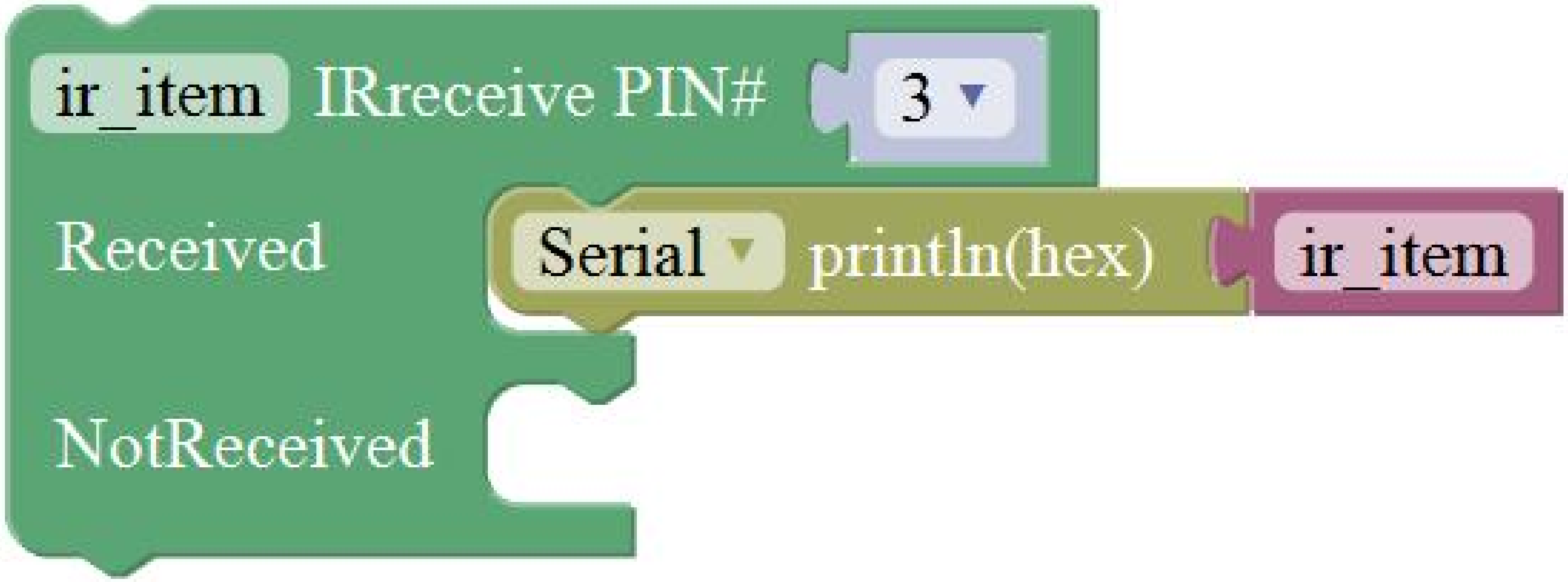
NO.	BLOCK ICON	DEFINITION
1	 ir_item IRreceive PIN# 0 Received Serial println(hex) ir_item NotReceived	Do something when receiving infrared signals.  ir_item IRreceive PIN# 0 Set the IO pin for IR receiver Received Serial println(hex) ir_item NotReceived print the hexadecimal value received by IR
2	 IRsend NEC PIN# 3 data 0x89ABCDEF bits 32	Sends infrared signals of the specified types. IR transmitter sends the data, here use the libraries, only PIN3 port.

3	 	Enable IR decoding
4	 	Print the Infrared signal in RAW types when receiving it.
5	 	Sends RAW infrared signals (set the pin number, list, length of list and IR frequency)

For example:

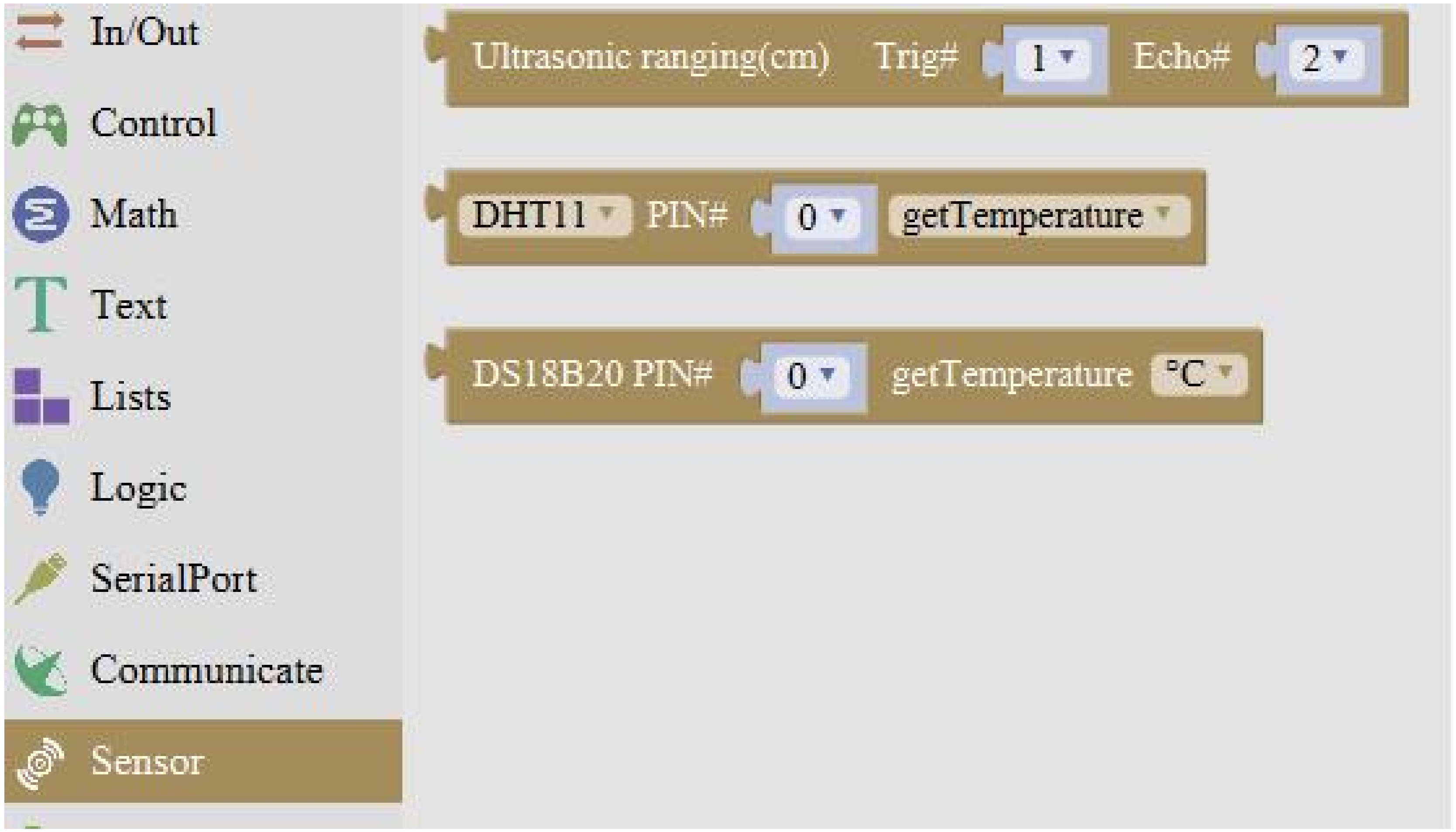
You need an Arduino Uno board, an IR receiver module and an IR remote control.

Connect the signal pin of IR receiver to Digital pin 3 of Uno board, then upload the code and open the monitor. If send a signal to an IR receiver module using an IR remote control, you should see the monitor show the corresponding signal data.



ir_item IRreceive PIN# 3
Received
Serial println(hex) ir_item
NotReceived

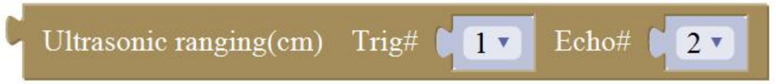
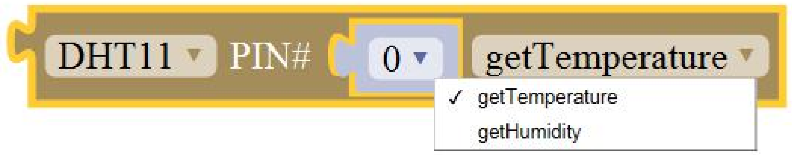
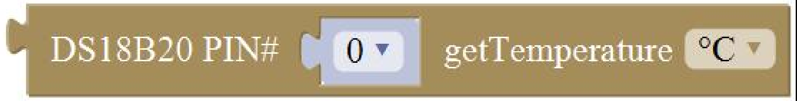
1.11 Sensor Block



Ultrasonic ranging(cm) Trig# 1 Echo# 2

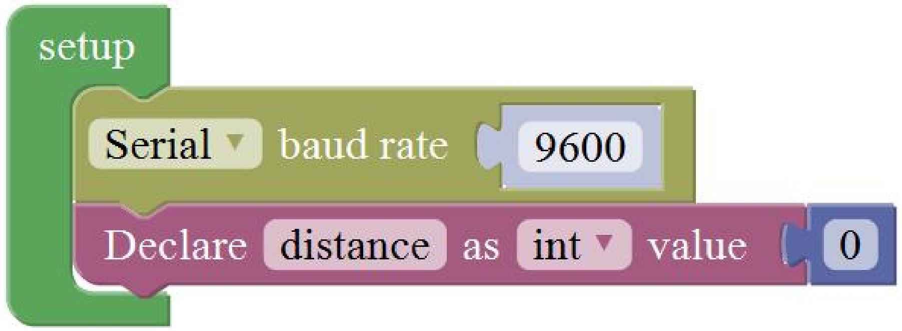
DHT11 PIN# 0 getTemperature

DS18B20 PIN# 0 getTemperature °C

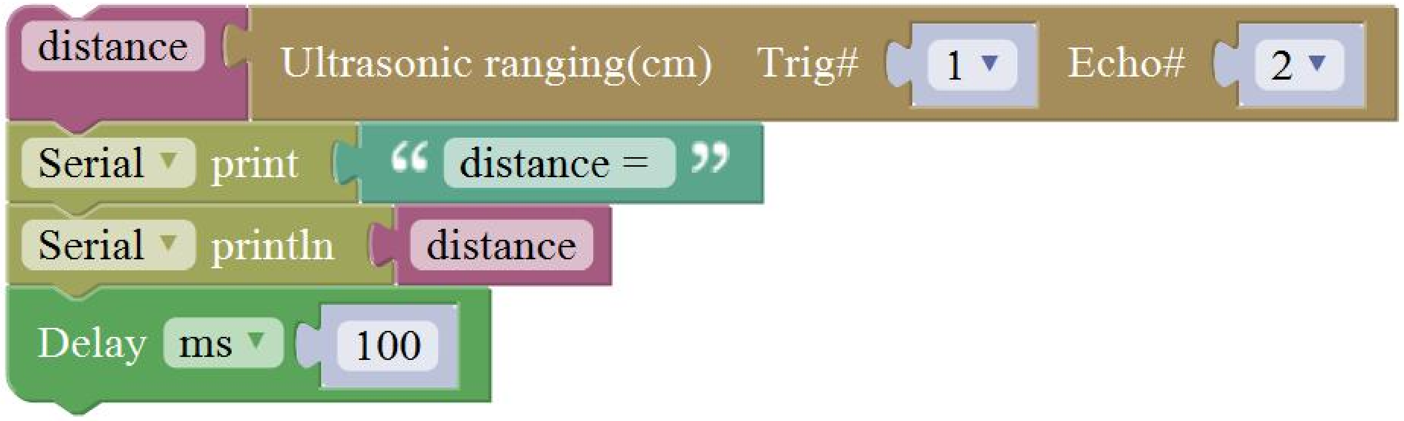
NO.	BLOCK ICON	DEFINITION
1	 Ultrasonic ranging(cm) Trig# 1 Echo# 2	Set the Trig and Echo pin of ultrasonic sensor. Returns the distance of ultrasonic sensor measured. (**unit:** cm)
2	 DHT11 PIN# 0 getTemperature ✓ getTemperature getHumidity	Set the control pin of DHT11 temperature and humidity sensor. Returns the temperature or humidity of DHT 11 sensor measured.
3	 DS18B20 PIN# 0 getTemperature ℃	Set the pin of digital temperature sensor DS18B20. Returns the temperature value of DS18B20 sensor measured.

For example: ultrasonic ranging

Connect the Trig pin of ultrasonic sensor to Digital 1 of Uno, Echo pin to D2, then upload the code and open the monitor, you should see the distance value, updating once per 100ms.

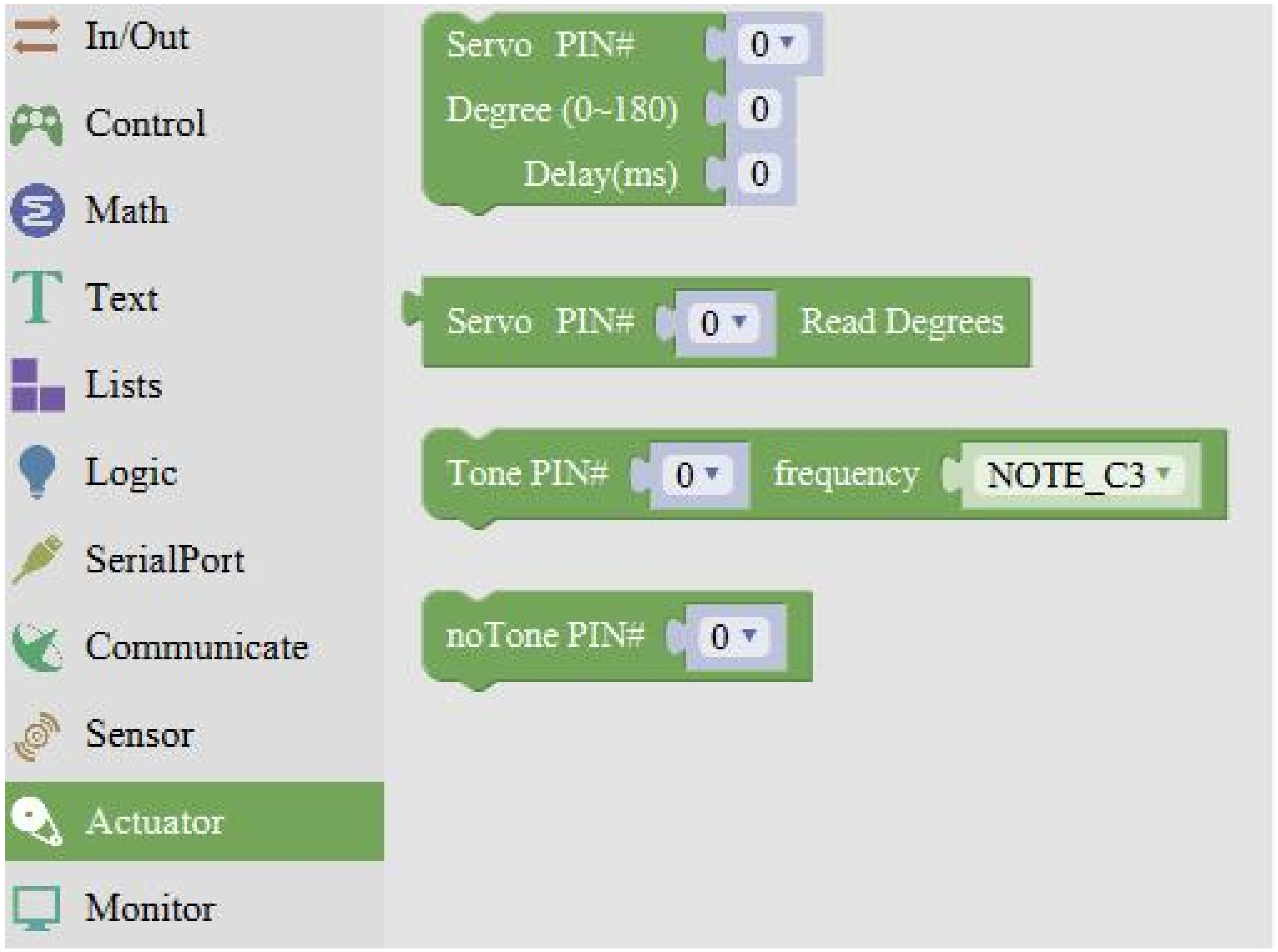


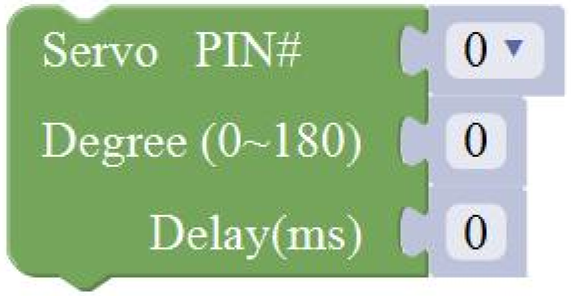
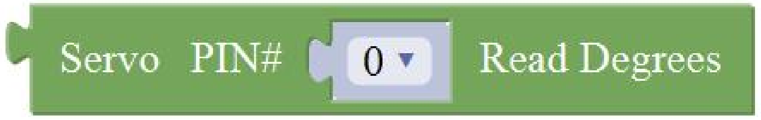
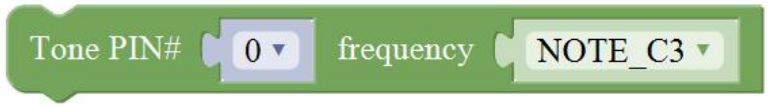
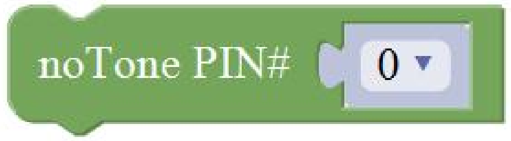
```
setup
    Serial ▼ baud rate 9600
    Declare distance as int ▼ value 0
```



```
distance  Ultrasonic ranging(cm)  Trig# 1 ▼  Echo# 2 ▼
Serial ▼ print “ distance = ”
Serial ▼ println distance
Delay ms ▼ 100
```

1.12 Actuator Block

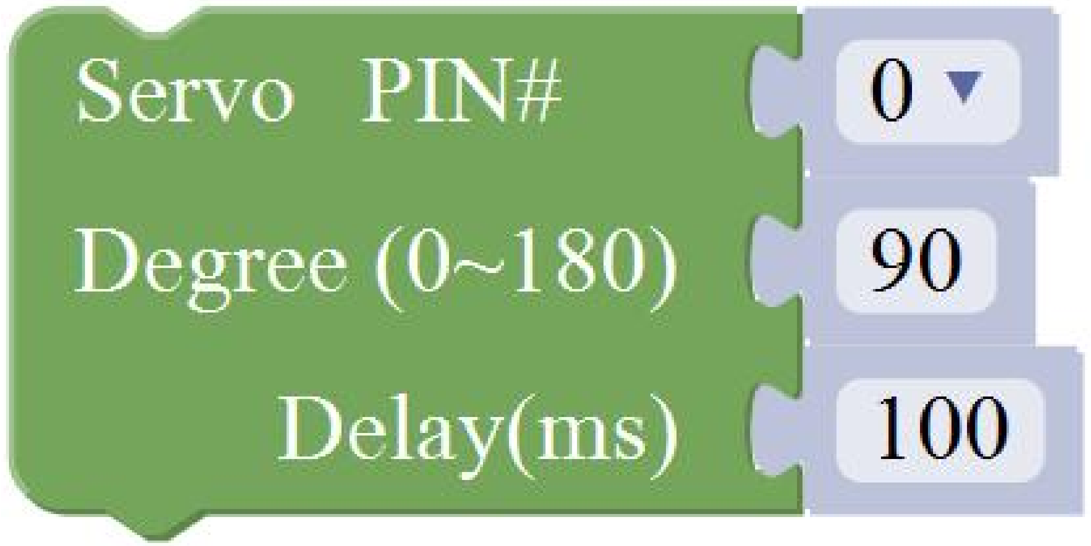


NO.	BLOCK ICON	DEFINITION
1		Sets the servo pin; Moves between 0-180 degree; Delay time for servo to rotate.
2		Returns that degree with the last servo move. Read the degree of servo connected to IO pin set
3		Set the pin and specified frequency for buzzer to play sound.
4		Stop playing sound

For example:

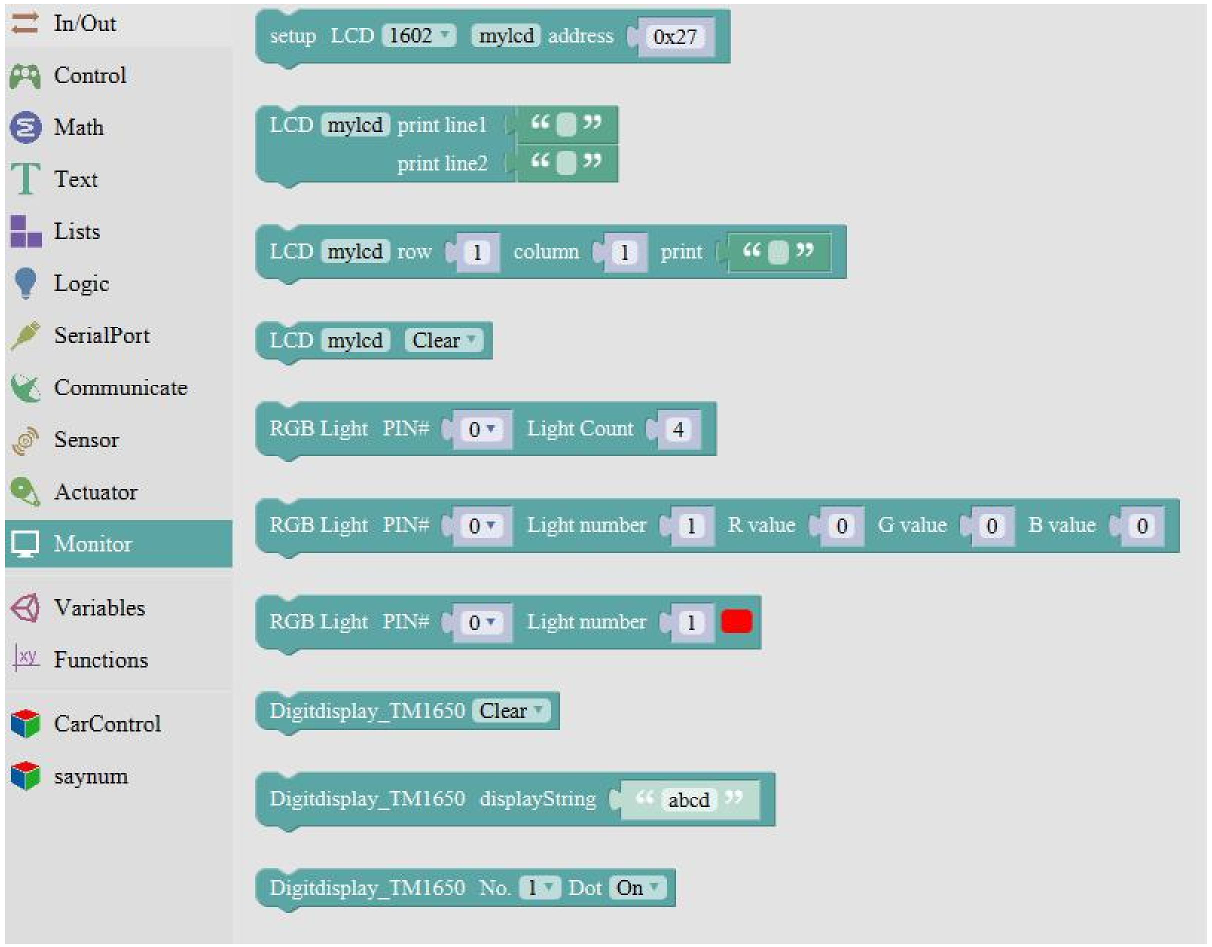
Connect the signal end of servo to Digital 0 of Uno, then upload the code below, servo will rotate 90 degrees.

Note: Delay 100ms is the time required for servo to move.

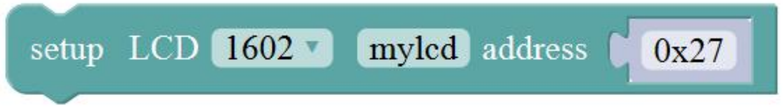
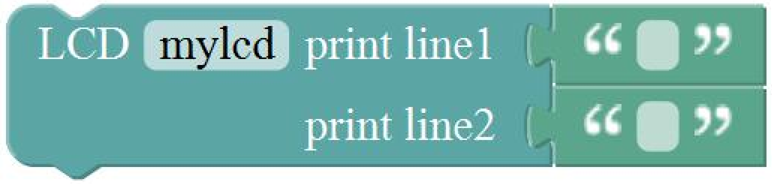
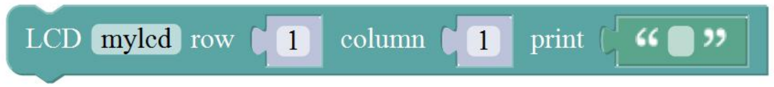
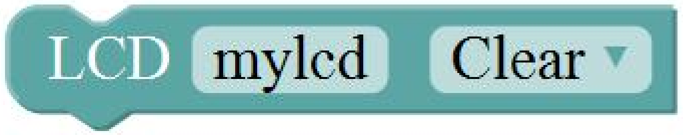
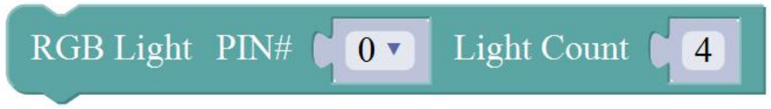


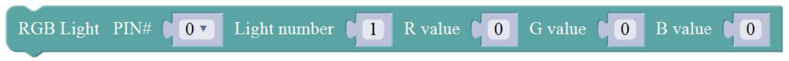
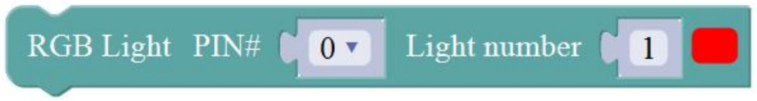
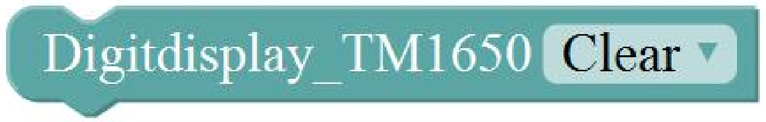
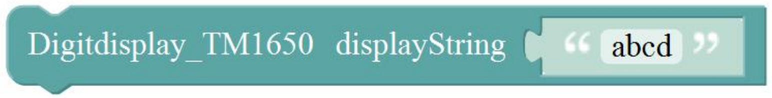
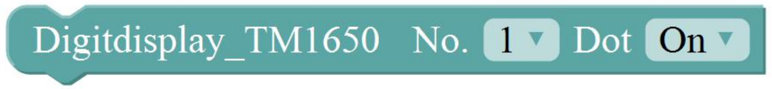
```
Servo PIN# 0
Degree (0~180) 90
Delay(ms) 100
```

1.13 Monitor Block



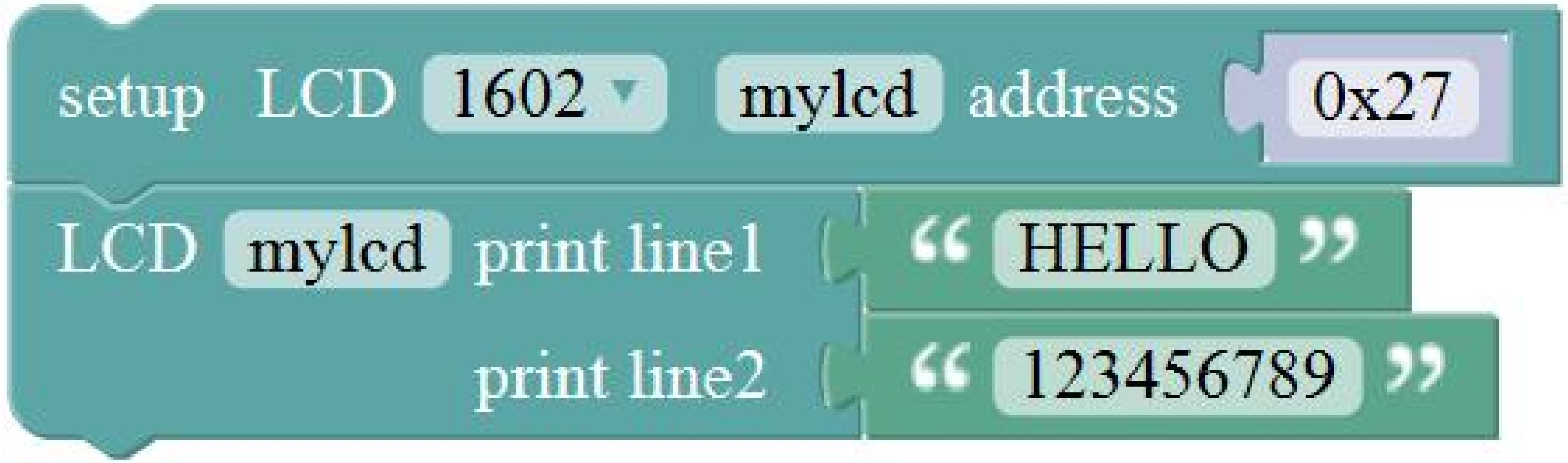
```
setup LCD 1602 mylcd address 0x27
LCD mylcd print line1 " "
          print line2 " "
LCD mylcd row 1 column 1 print " "
LCD mylcd Clear
RGB Light PIN# 0 Light Count 4
RGB Light PIN# 0 Light number 1 R value 0 G value 0 B value 0
RGB Light PIN# 0 Light number 1 🟥
Digitdisplay_TM1650 Clear
Digitdisplay_TM1650 displayString " abcd "
Digitdisplay_TM1650 No. 1 Dot On
```

NO.	BLOCK ICON	DEFINITION
1	 setup LCD 1602 mylcd address 0x27	Set the IIC LCD1602 address
2	 LCD mylcd print line1 " " print line2 " "	Input the value on LCD line 1 and line 2 from left to right.
3	 LCD mylcd row 1 column 1 print " "	Set the row and column of LCD to print the char
4	 LCD mylcd Clear	Clear the LCD screen
5	 RGB Light PIN# 0 Light Count 4	Set the control pin and the number of RGB light.

6	 RGB Light PIN# 0 Light number 1 R value 0 G value 0 B value 0	Set the RGB light pin, light number and brightness
7	 RGB Light PIN# 0 Light number 1 🟥	Set the control pin, light number and color. (click to select the color)
8	 Digitdisplay_TM1650 Clear	Clear the data, namely turn off digital display
9	 Digitdisplay_TM1650 displayString " abcd "	Four-digit display, displaying abcd.
10	 Digitdisplay_TM1650 No. 1 Dot On	Turn on or off the digitdisplay (here turn on the first digitdisplay)

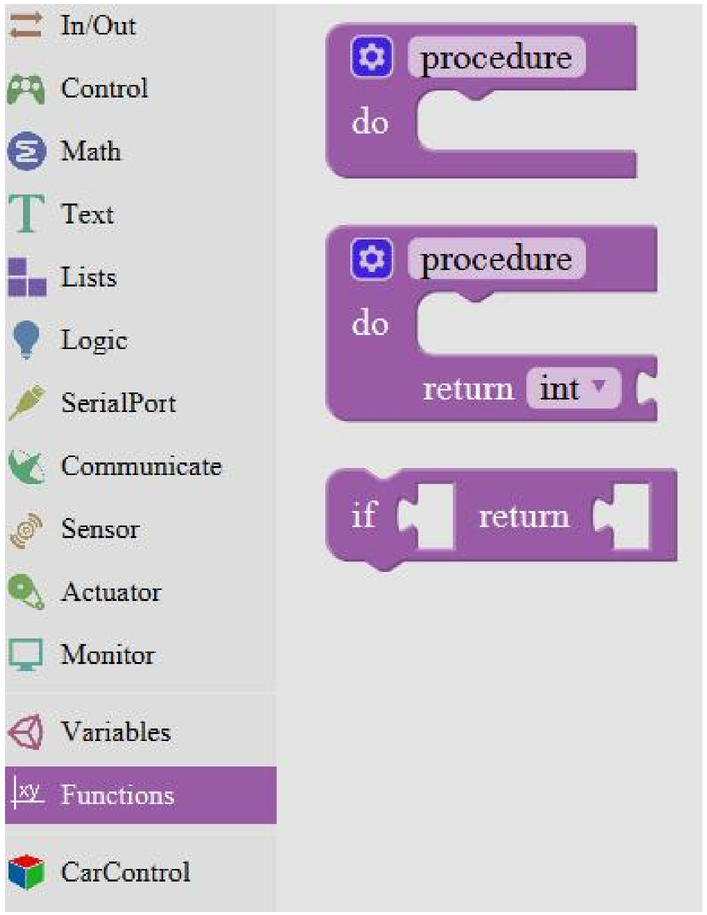
For example:

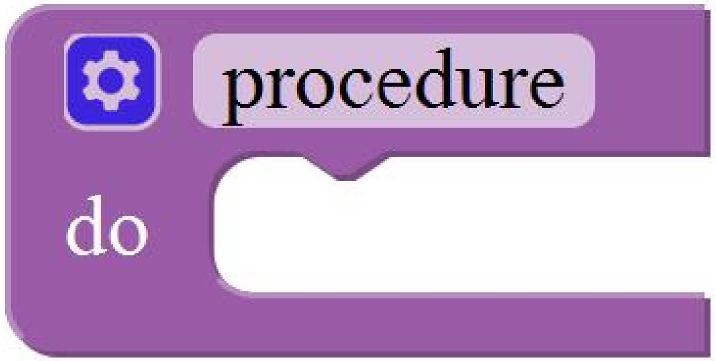
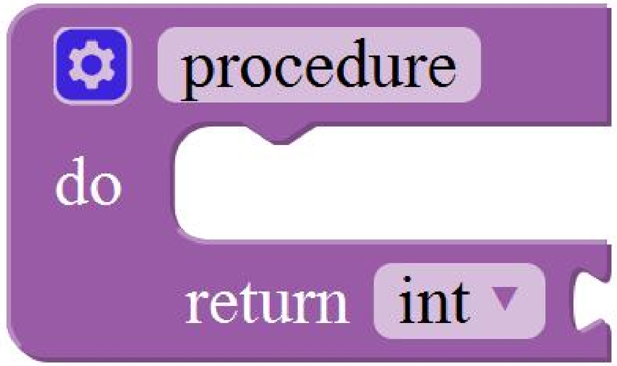
Separately connect the SDA (A4) and SCL (A5) of Arduino Uno to SDA and SCL pins of IIC LCD1602, then set the address of your LCD1602 screen, the LCD address we used here is 0x27. Then upload the code, LCD screen has two lines, you should see the line 1 print HELLO, and line 2 print 123456789.

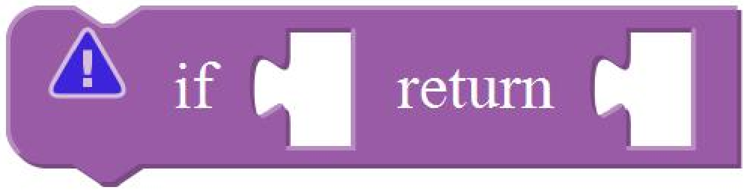


```
setup LCD 1602 ▾ mylcd address 0x27
LCD mylcd print line1 " HELLO "
          print line2 " 123456789 "
```

1.14 Functions Block

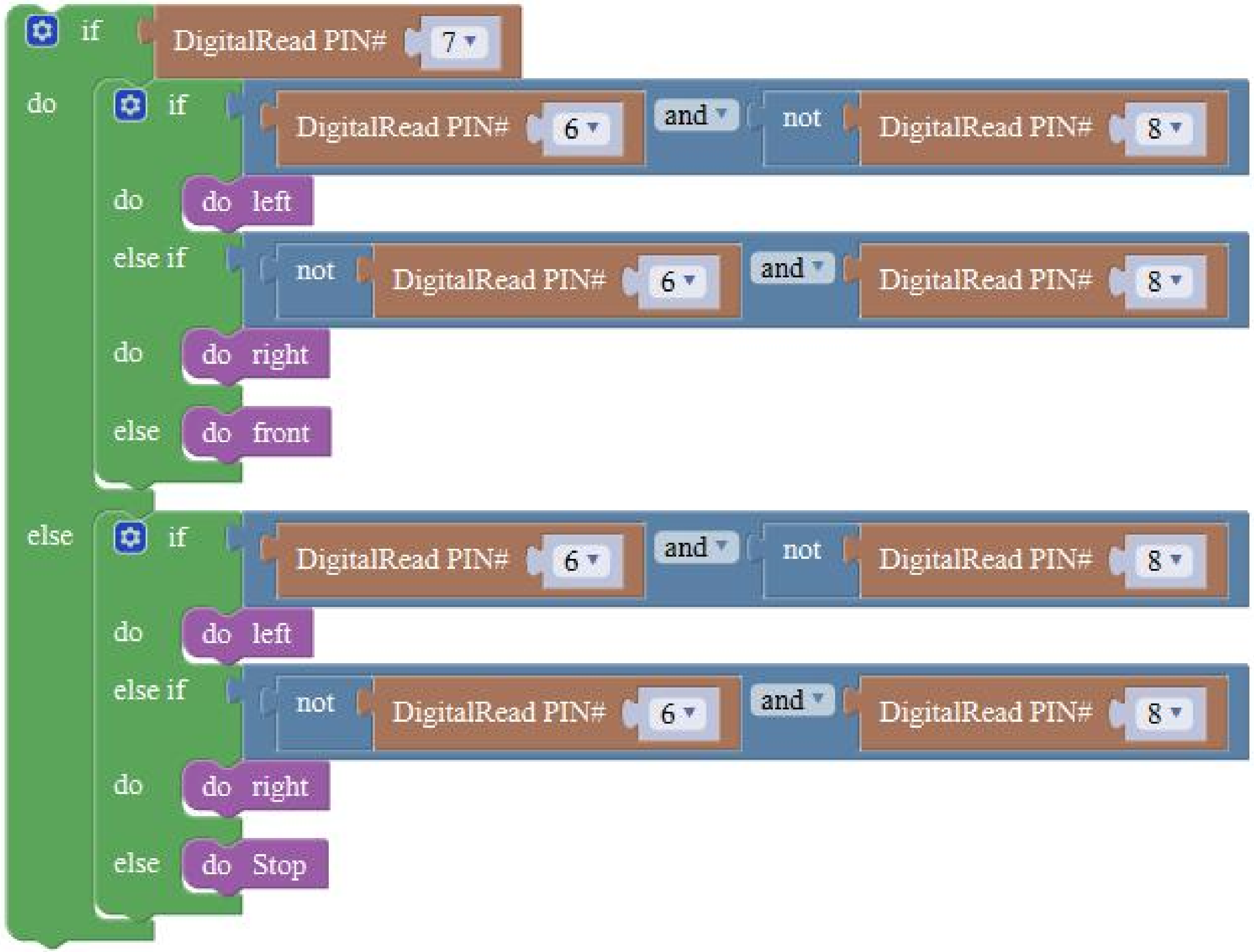


NO.	BLOCK ICON	DEFINITION
1		Creates a function with no output. Click the blue icon to set the procedure parameter. (no return value)
2		Creates a function with an output. Click the blue icon to set the procedure parameter. (with return value and can set the data types)

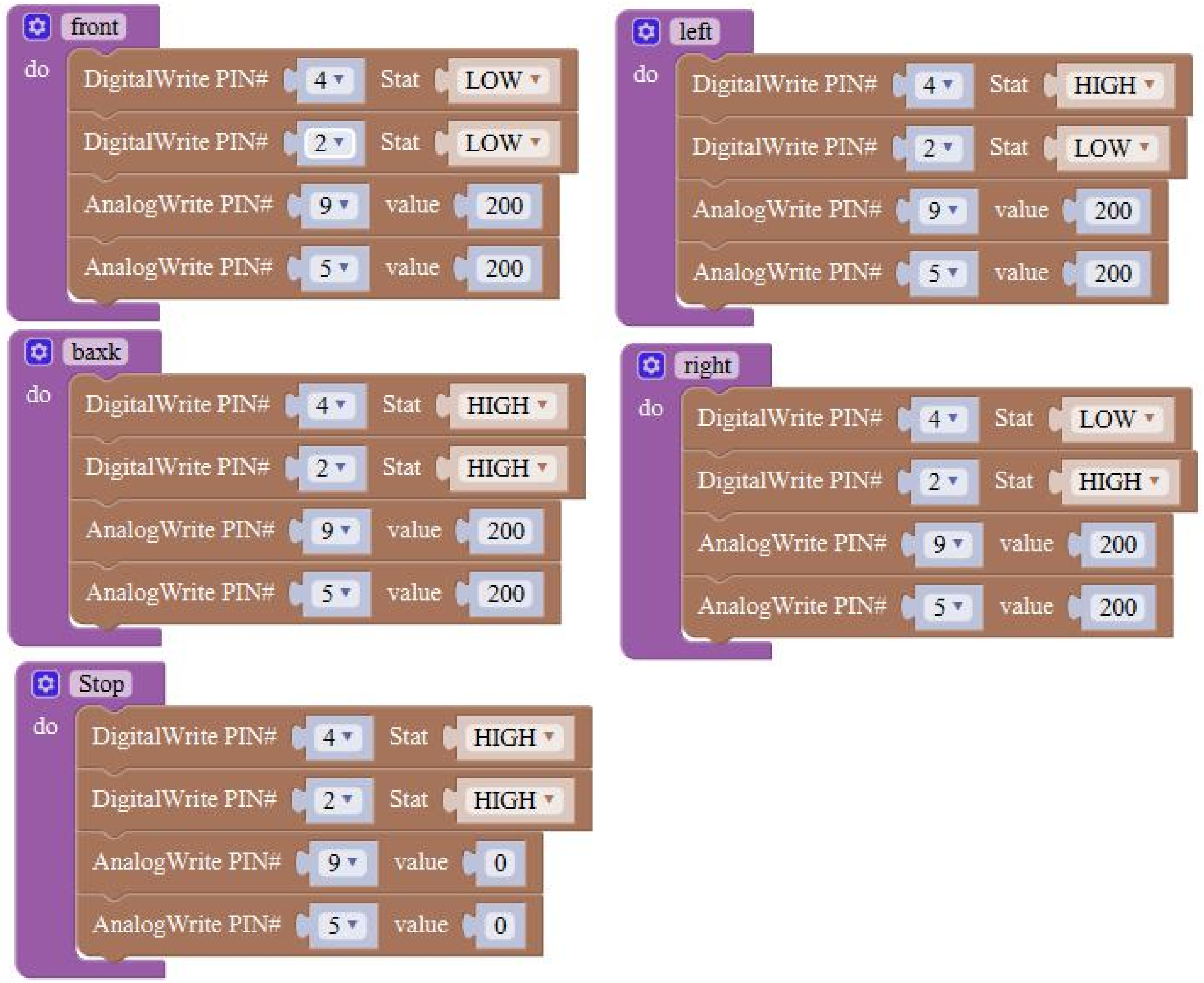
3		If a value is true, then return a second value.

For example:

Below is an example code for line tracking car. We use three tracking modules (left to D6, middle to D7, right to D8). of course you need a tracking car to test it. First edit the forward, backward, turn left, turn right and stop into functions block. Then compile and upload the code below.



```
if DigitalRead PIN# 7
do
    if DigitalRead PIN# 6 and not DigitalRead PIN# 8
    do    do left
    else if not DigitalRead PIN# 6 and DigitalRead PIN# 8
    do    do right
    else  do front
else
    if DigitalRead PIN# 6 and not DigitalRead PIN# 8
    do    do left
    else if not DigitalRead PIN# 6 and DigitalRead PIN# 8
    do    do right
    else  do Stop
```



```
front
do
    DigitalWrite PIN# 4 Stat LOW
    DigitalWrite PIN# 2 Stat LOW
    AnalogWrite PIN# 9 value 200
    AnalogWrite PIN# 5 value 200
```

```
left
do
    DigitalWrite PIN# 4 Stat HIGH
    DigitalWrite PIN# 2 Stat LOW
    AnalogWrite PIN# 9 value 200
    AnalogWrite PIN# 5 value 200
```

```
baxk
do
    DigitalWrite PIN# 4 Stat HIGH
    DigitalWrite PIN# 2 Stat HIGH
    AnalogWrite PIN# 9 value 200
    AnalogWrite PIN# 5 value 200
```

```
right
do
    DigitalWrite PIN# 4 Stat LOW
    DigitalWrite PIN# 2 Stat HIGH
    AnalogWrite PIN# 9 value 200
    AnalogWrite PIN# 5 value 200
```

```
Stop
do
    DigitalWrite PIN# 4 Stat HIGH
    DigitalWrite PIN# 2 Stat HIGH
    AnalogWrite PIN# 9 value 0
    AnalogWrite PIN# 5 value 0
```

2. Resources

Download the Mixly software:

https://drive.google.com/open?id=1oQxF-AZ0Aw6OQhu_8NSvwo3L2OP0Z6cU

Download the Example Code:

https://drive.google.com/open?id=1Fjd3SHHkg_-0IdB6GPX2uuTv3aRGMCSd

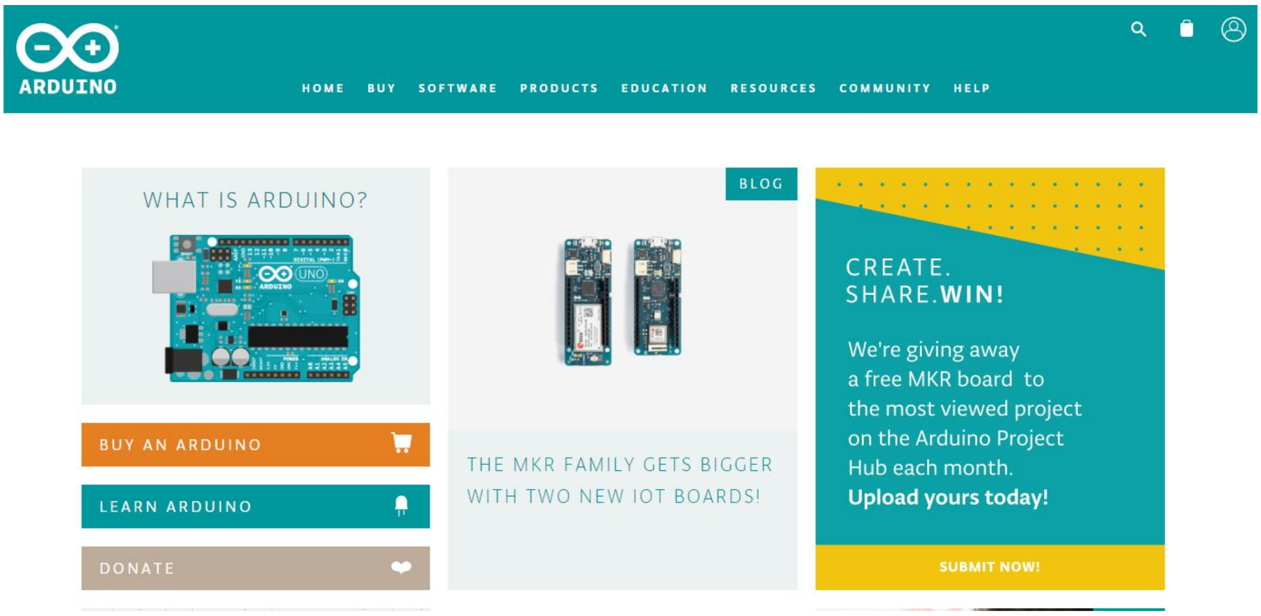
Relevant links:

https://wiki.keyestudio.com/How_to_Import_Mixly_Library

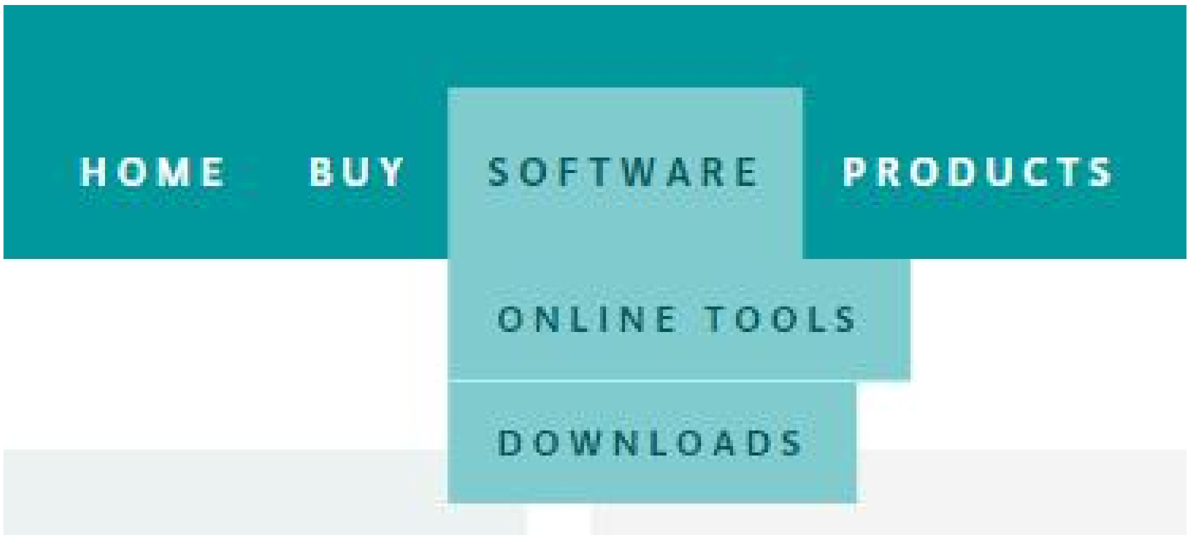
Getting Started with Arduino

Installing Arduino Software

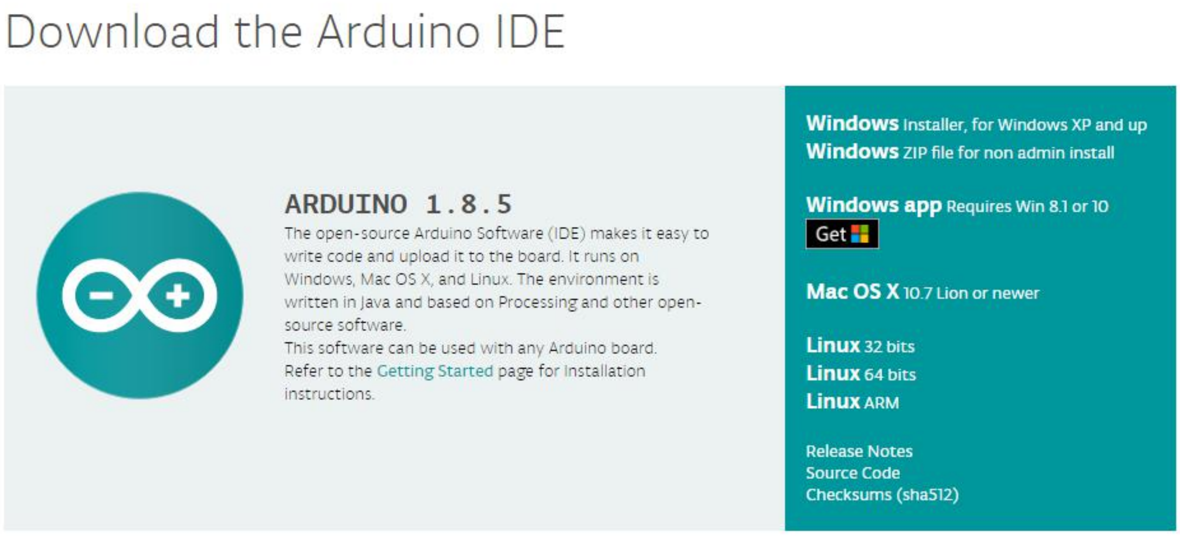
When you get the control board, first you should install the Arduino software and driver. We usually use the software Arduino 1.5.6 version.
You can download it from the link below:
https://www.arduino.cc/en/Main/OldSoftwareReleases#1.5.x
Or you can browse the ARDUINO website to download the latest version from this link, https://www.arduino.cc, pop up the following interface.



Then click the SOFTWARE on the browse bar, you will have two options ONLINE TOOLS and DOWNLOADS.



Click DOWNLOADS, it will appear the latest software version of ARDUINO 1.8.5 shown as below.


Download the Arduino IDE

ARDUINO 1.8.5

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

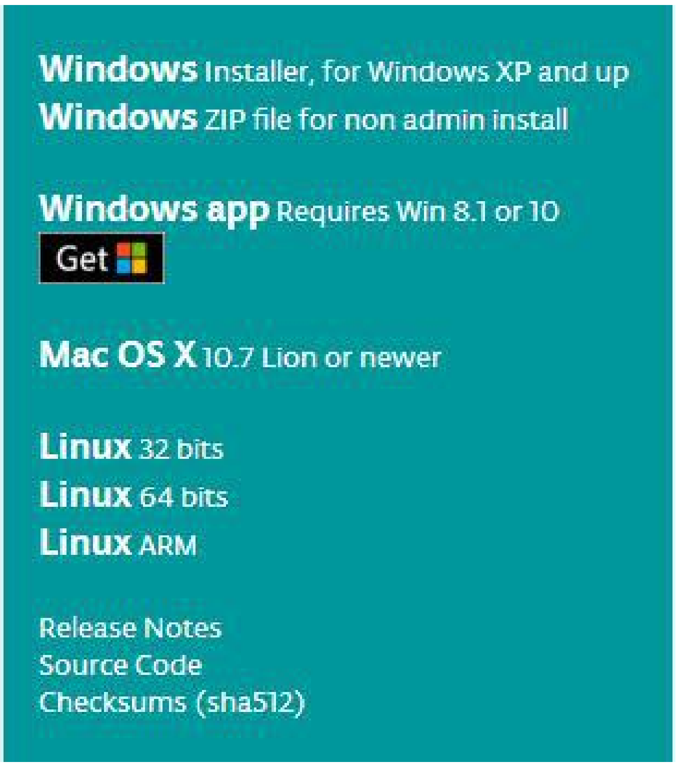
Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

Release Notes
Source Code
Checksums (sha512)

In this software page, on the right side you can see the version of development software for different operating systems. So ARDUINO has a rather powerful compatibility. You should download the software that is compatible with the operating system of your computer.

In our project, we will take WINDOWS system as an example here.
First click Windows Installer, you will get the following page.

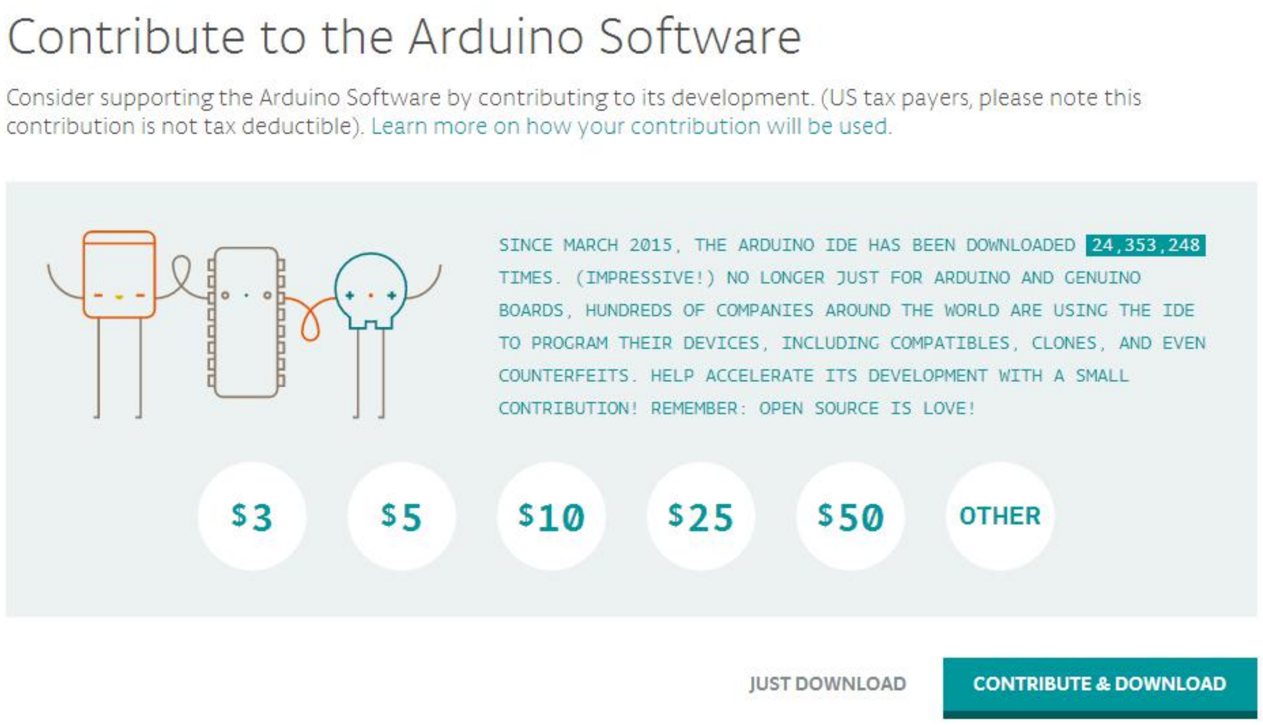

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

Release Notes
Source Code
Checksums (sha512)


Contribute to the Arduino Software

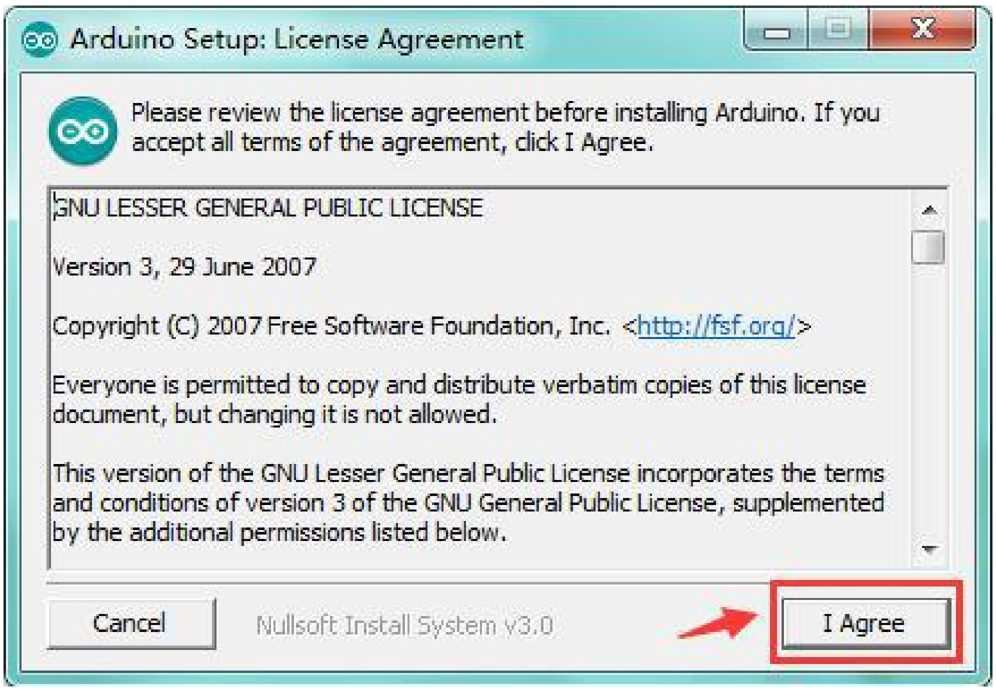
Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.

SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED 24,353,248 TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

Click JUST DOWNLOAD, and when the ZIP file is downloaded well to your computer, you can directly unzip the file and then click the icon of ARDUINO program to start it.

Installing Arduino (Windows)

Install Arduino with the exe. Installation package. Here we provide you with Arduino-1.5.6-r2-windows package, you can directly click the icon to install it.


Arduino Setup: License Agreement

Please review the license agreement before installing Arduino. If you accept all terms of the agreement, click I Agree.

GNU LESSER GENERAL PUBLIC LICENSE

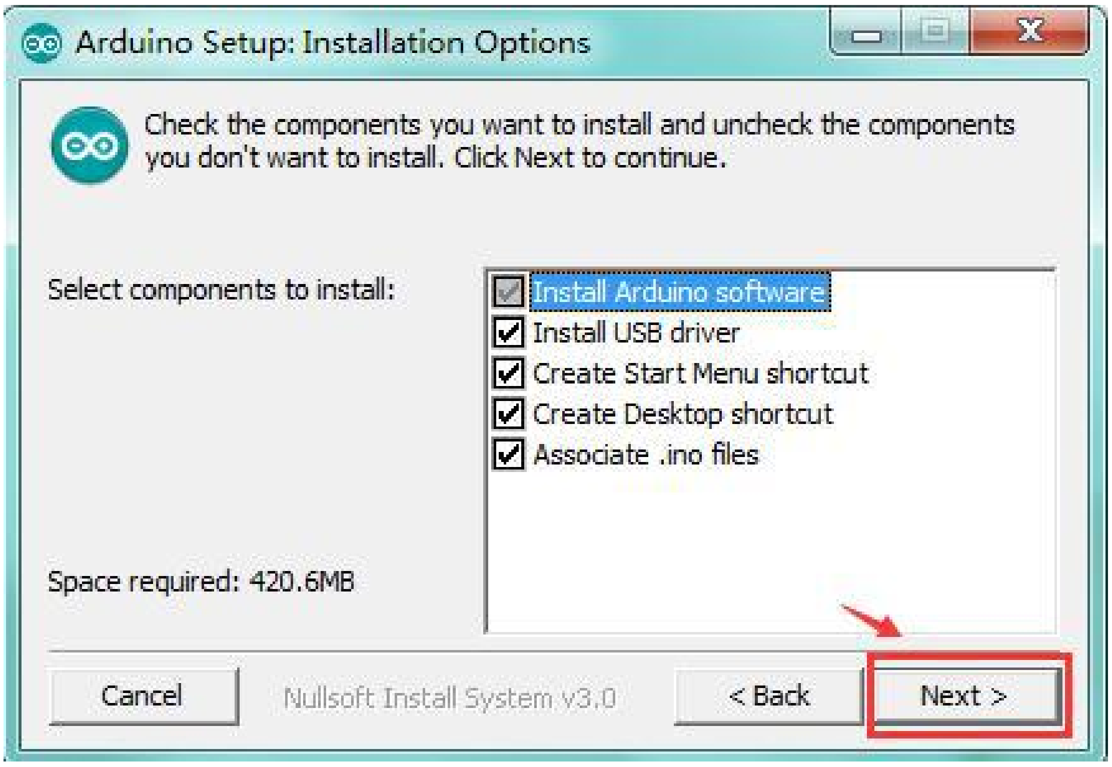
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

Click *"I Agree"* to see the following interface.


Arduino Setup: Installation Options

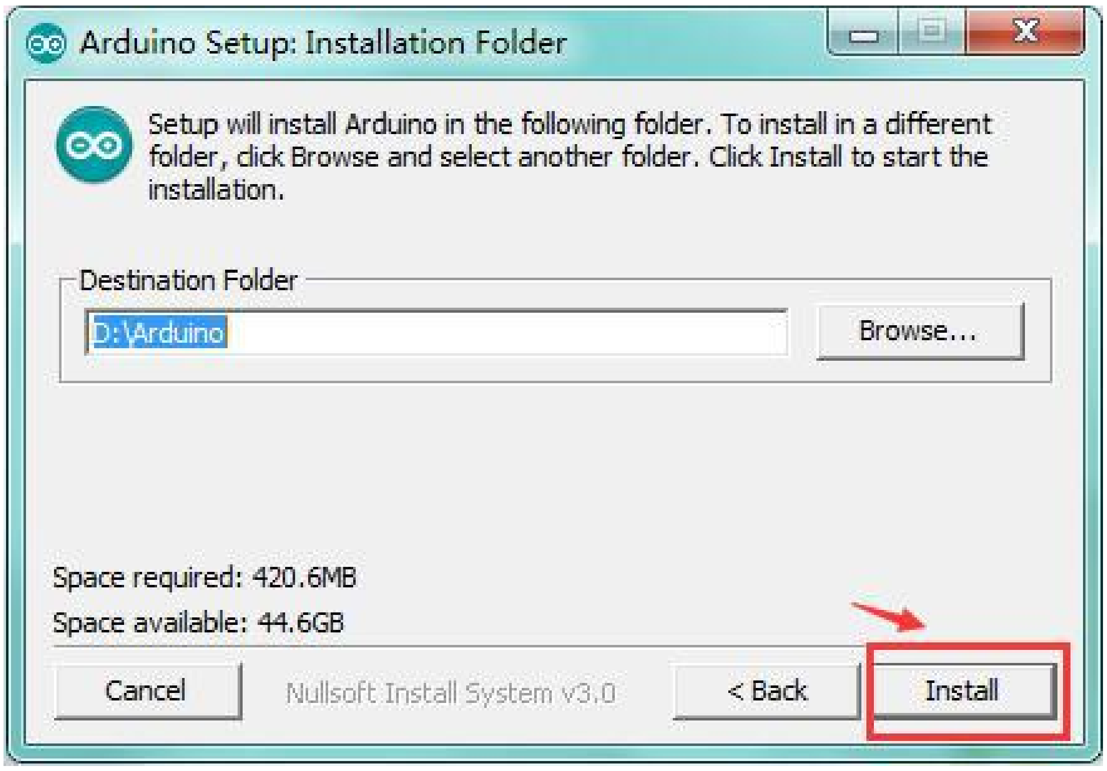
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [✓] Install Arduino software
- [✓] Install USB driver
- [✓] Create Start Menu shortcut
- [✓] Create Desktop shortcut
- [✓] Associate .ino files

Space required: 420.6MB

Click *"Next"*. Pop up the interface below.


Arduino Setup: Installation Folder

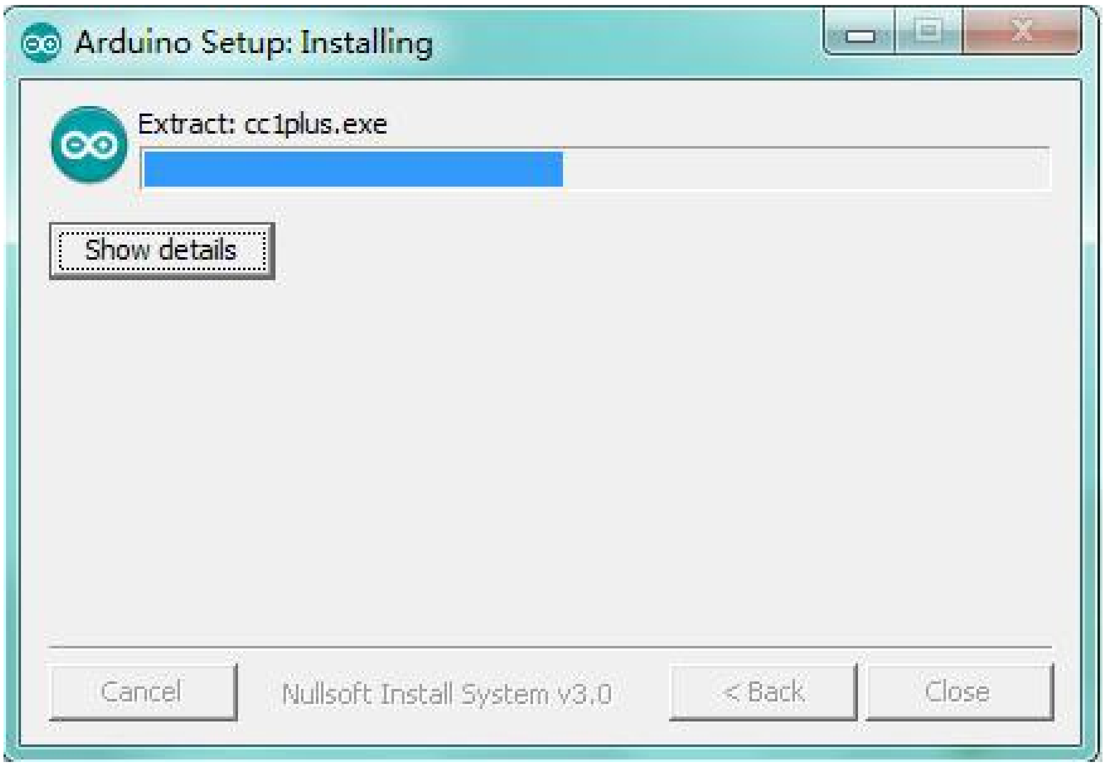
Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: D:\Arduino

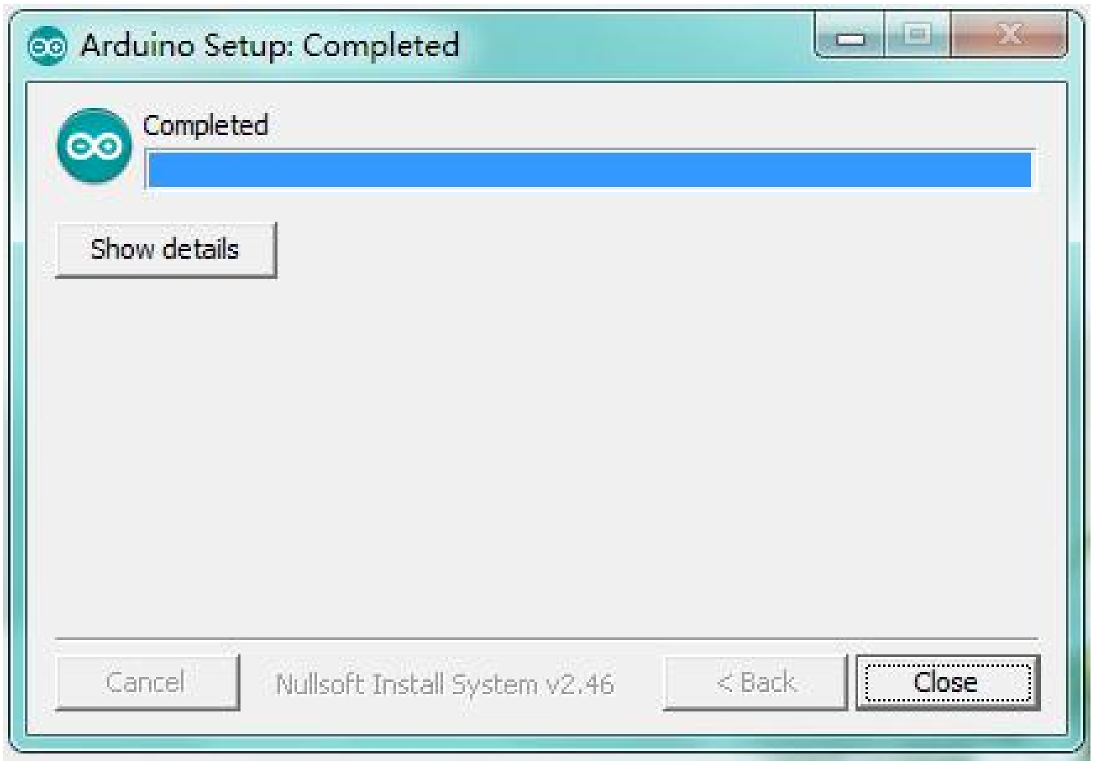
Space required: 420.6MB
Space available: 44.6GB

You can press Browse… to choose an installation path or directly type in the directory you want.

Then click "Install" to initiate installation.

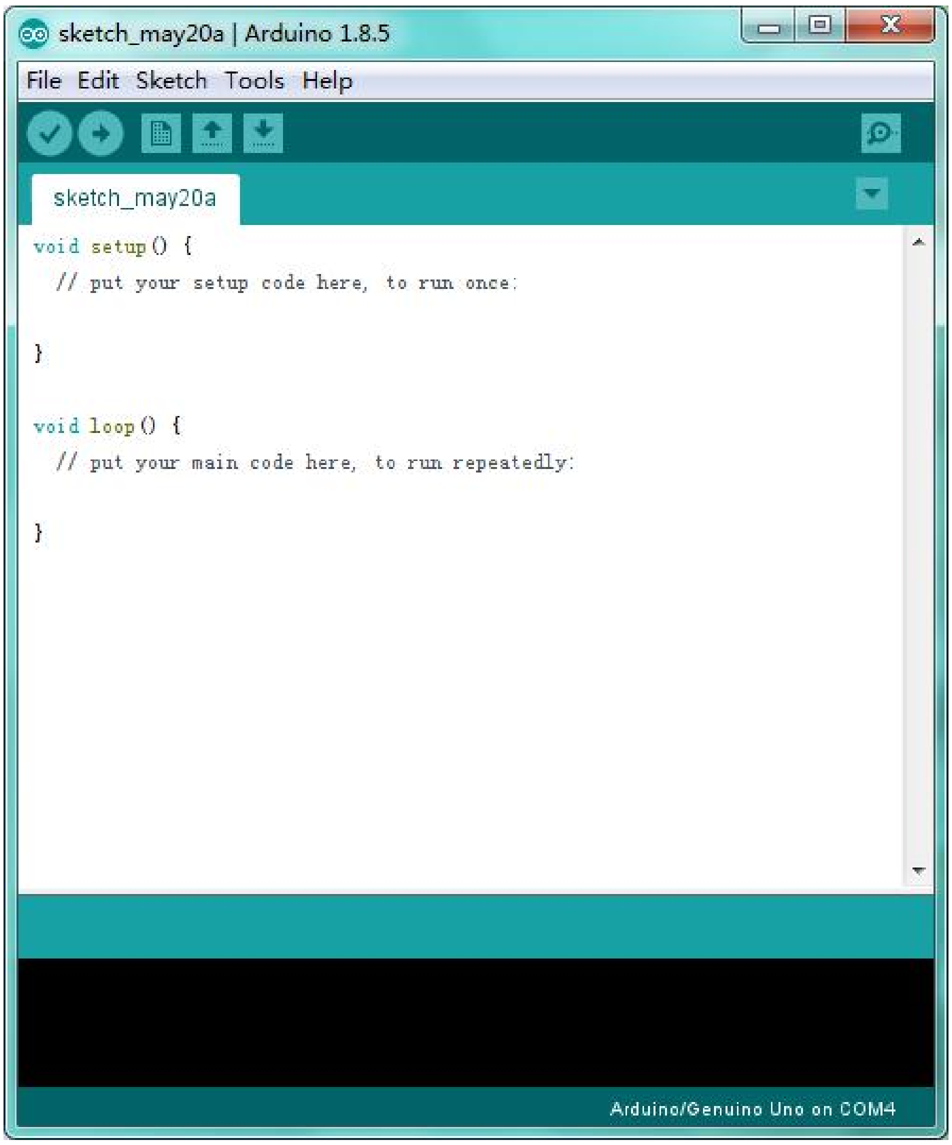


Wait for the installing process, if appear the interface of Window Security, just continue to click Install to finish the installation.



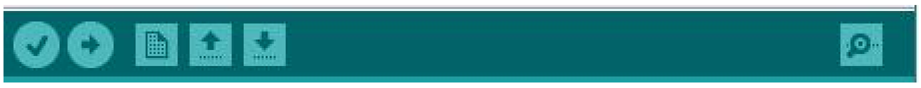
All right, up to now, you have completed the Arduino setup! The following icon will appear on your PC desktop.

Double-click the icon of Arduino to enter the desired development environment shown as below.


sketch_may20a | Arduino 1.8.5
File Edit Sketch Tools Help
sketch_may20a
Arduino/Genuino Uno on COM4

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

The functions of each button on the Toolbar are listed below:



Verify/Compile	Check the code for errors
Upload	Upload the current Sketch to the Arduino
New	Create a new blank Sketch
Open	Show a list of Sketches
Save	Save the current Sketch
Serial Monitor	Display the serial data being sent from the Arduino

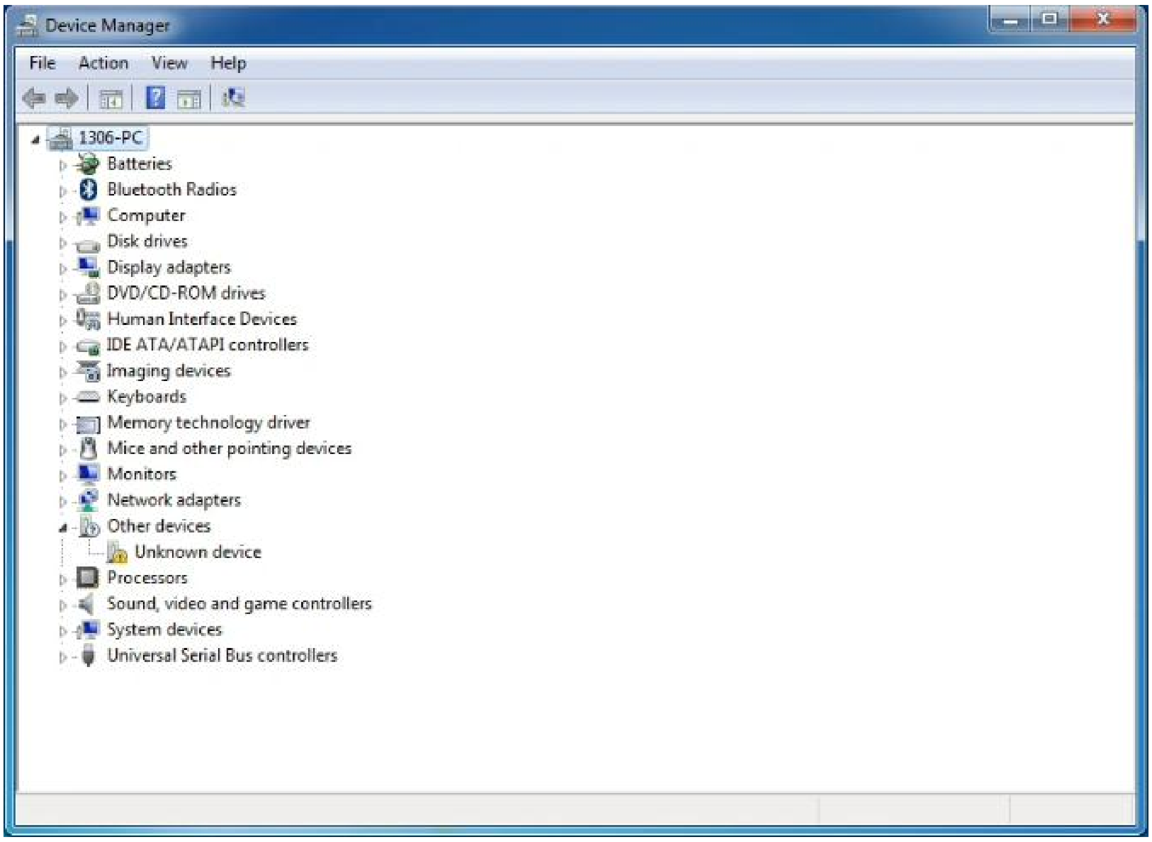
Installing Driver

Next, we will introduce the driver installation for development board. The driver installation may have slight differences in different computer systems. So in the following let’s move on to the driver installation in the WIN 7 system.

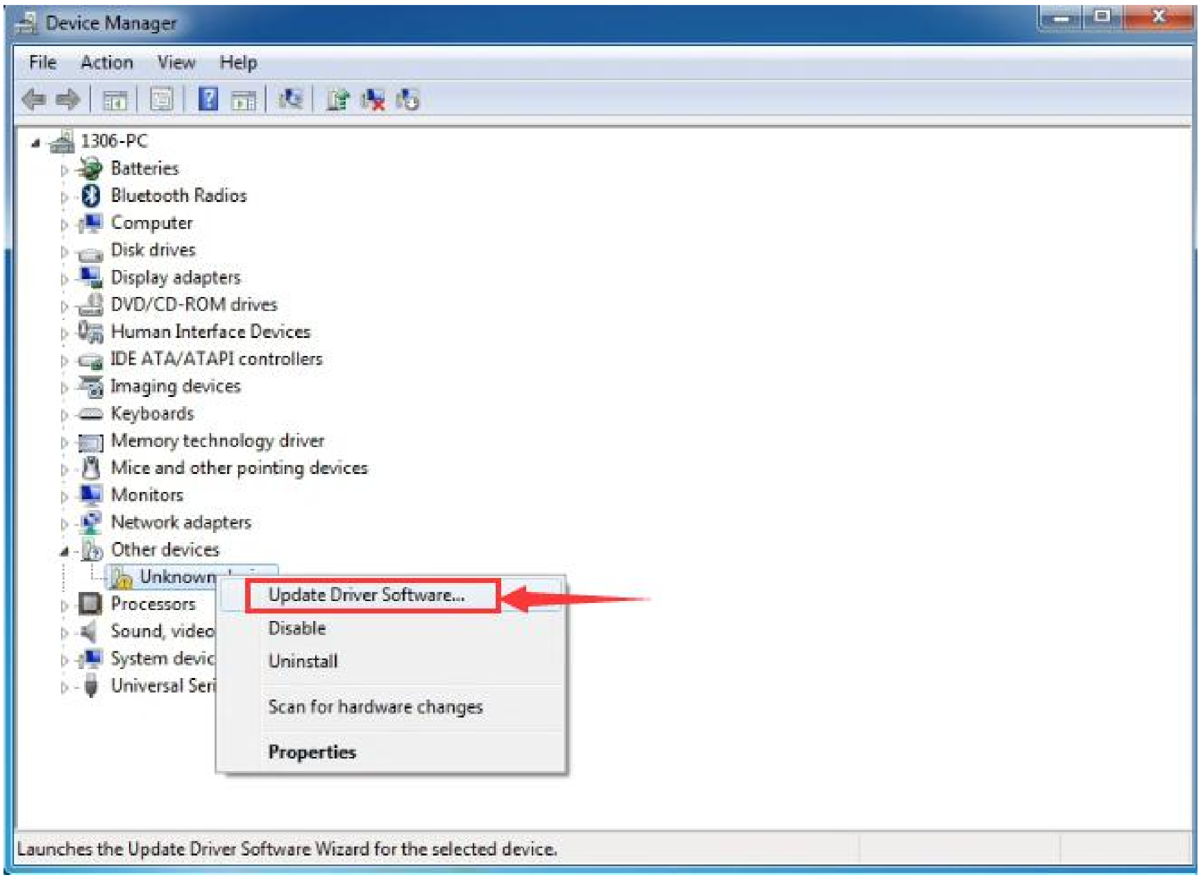
The Arduino folder contains both the Arduino program itself and the drivers that allow the Arduino to be connected to your computer by a USB cable. Before we launch the Arduino software, you are going to install the USB drivers.

Plug one end of your USB cable into the Arduino and the other into a USB socket on your computer.

When you connect the UNO board to your computer at the first time, right click the icon of your *"Computer" —>for "Properties"—> click "Device manager"*, under “Other Devices”, you should see an icon for “Unknown device” with a little yellow warning triangle next to it. This is your Arduino.

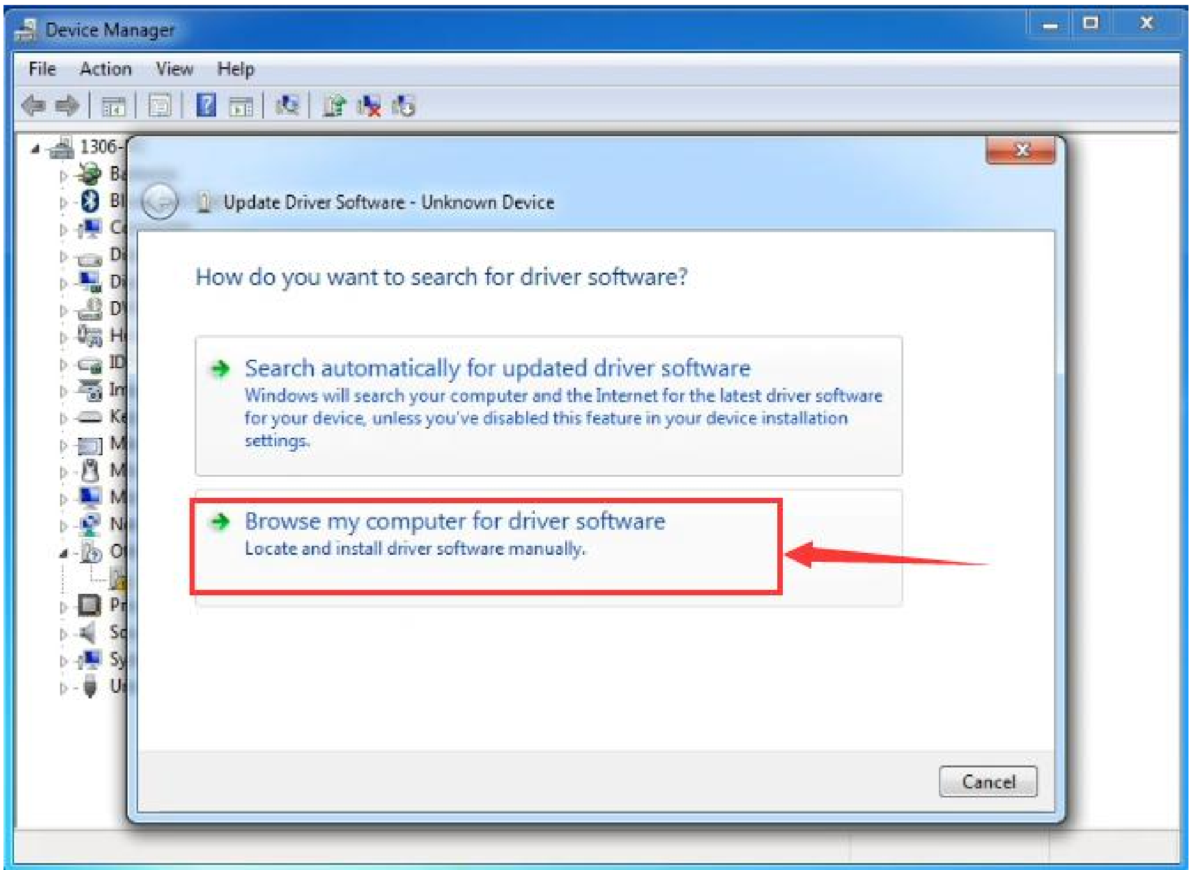


Then right-click on the device and select the top menu option (Update Driver Software...) shown as the figure below.



Launches the Update Driver Software Wizard for the selected device.

It will then be prompted to either "Search Automatically for updated driver software" or "Browse my computer for driver software". Shown as below. In this page, select "Browse my computer for driver software".



How do you want to search for driver software?

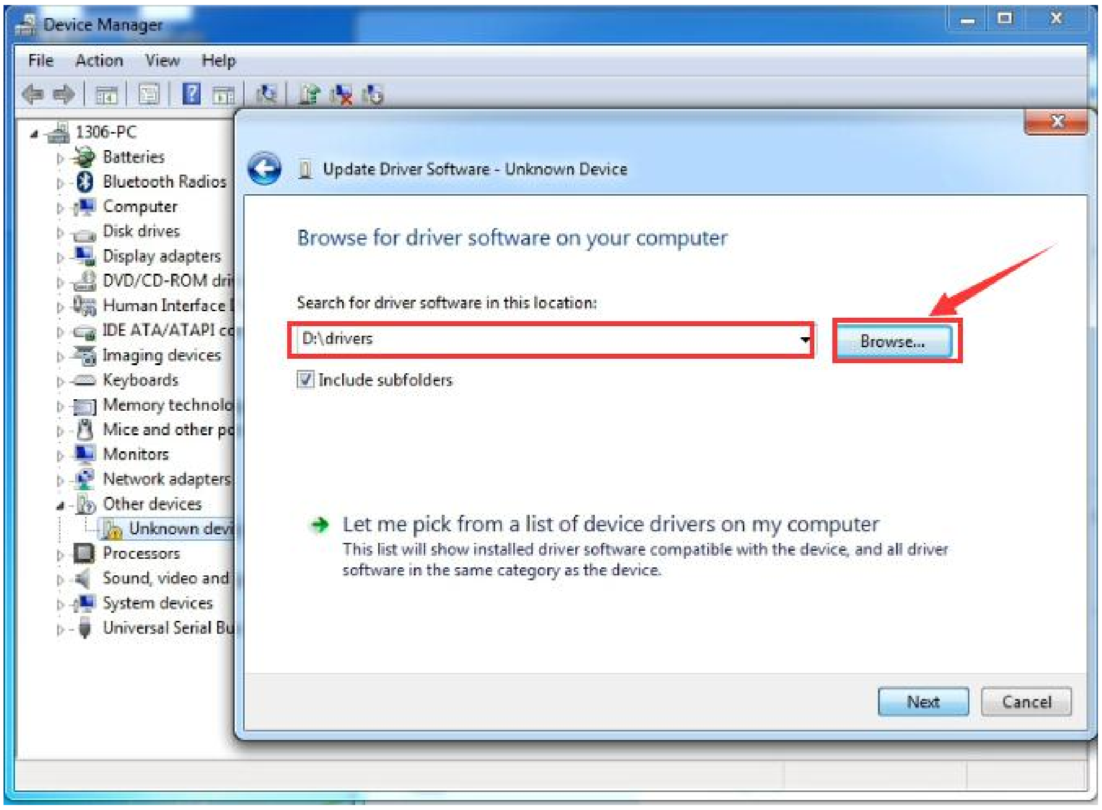
Search automatically for updated driver software

Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

Browse my computer for driver software

Locate and install driver software manually.

After that, select the option to browse and navigate to the “drivers” folder of Arduino installation.



Update Driver Software - Unknown Device

Browse for driver software on your computer

Search for driver software in this location:

D:\drivers

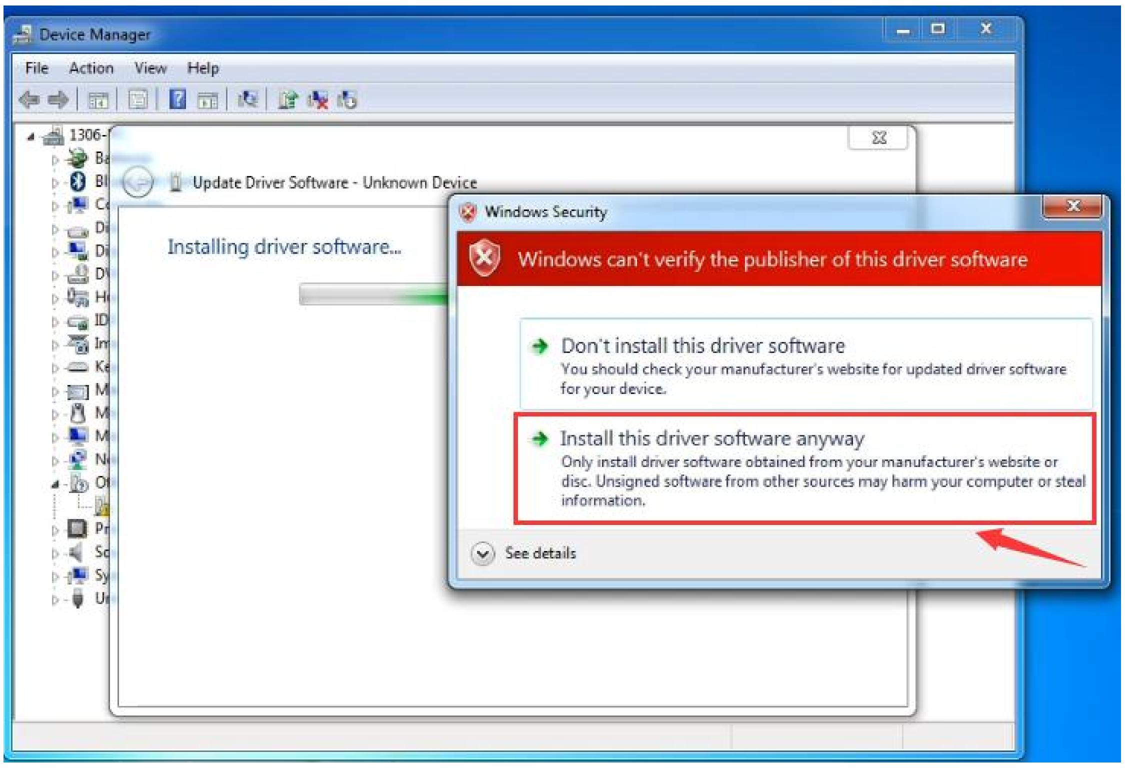
Browse...

☑ Include subfolders

Let me pick from a list of device drivers on my computer

This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

Click “Next” and you may get a security warning, if so, allow the software to be installed. Shown as below.



Installing driver software...

Windows Security

Windows can't verify the publisher of this driver software

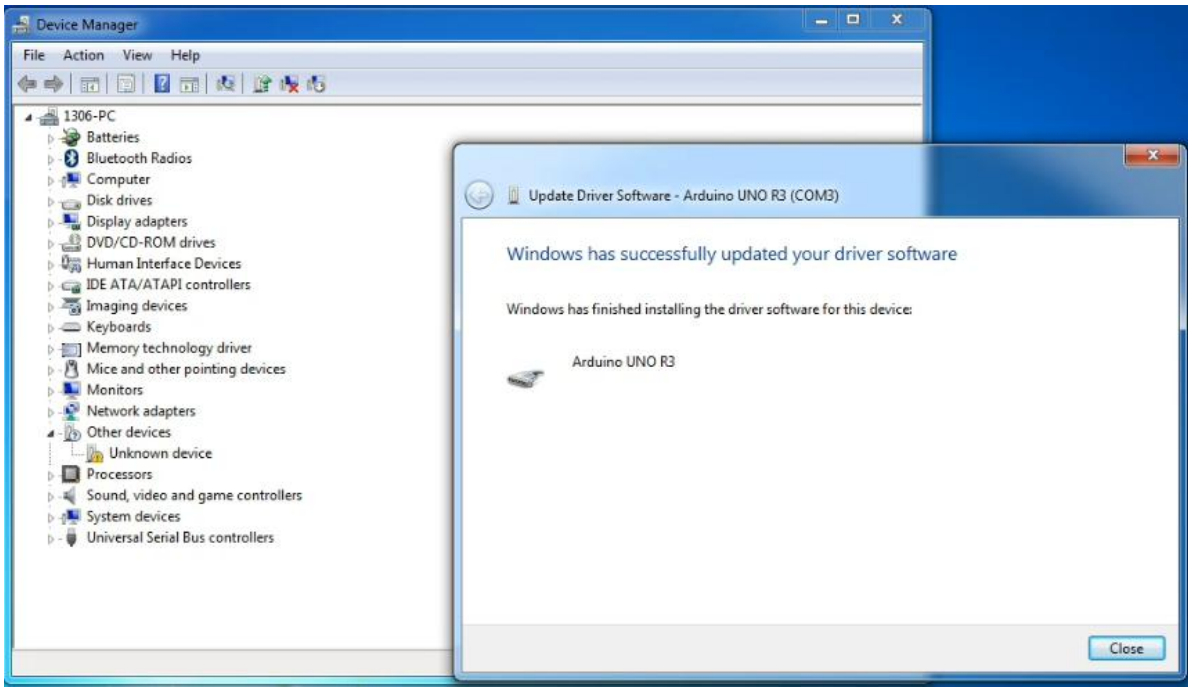
Don't install this driver software

You should check your manufacturer's website for updated driver software for your device.

Install this driver software anyway

Only install driver software obtained from your manufacturer's website or disc. Unsigned software from other sources may harm your computer or steal information.

Once the software has been installed, you will get a confirmation message. Installation completed, click "Close".



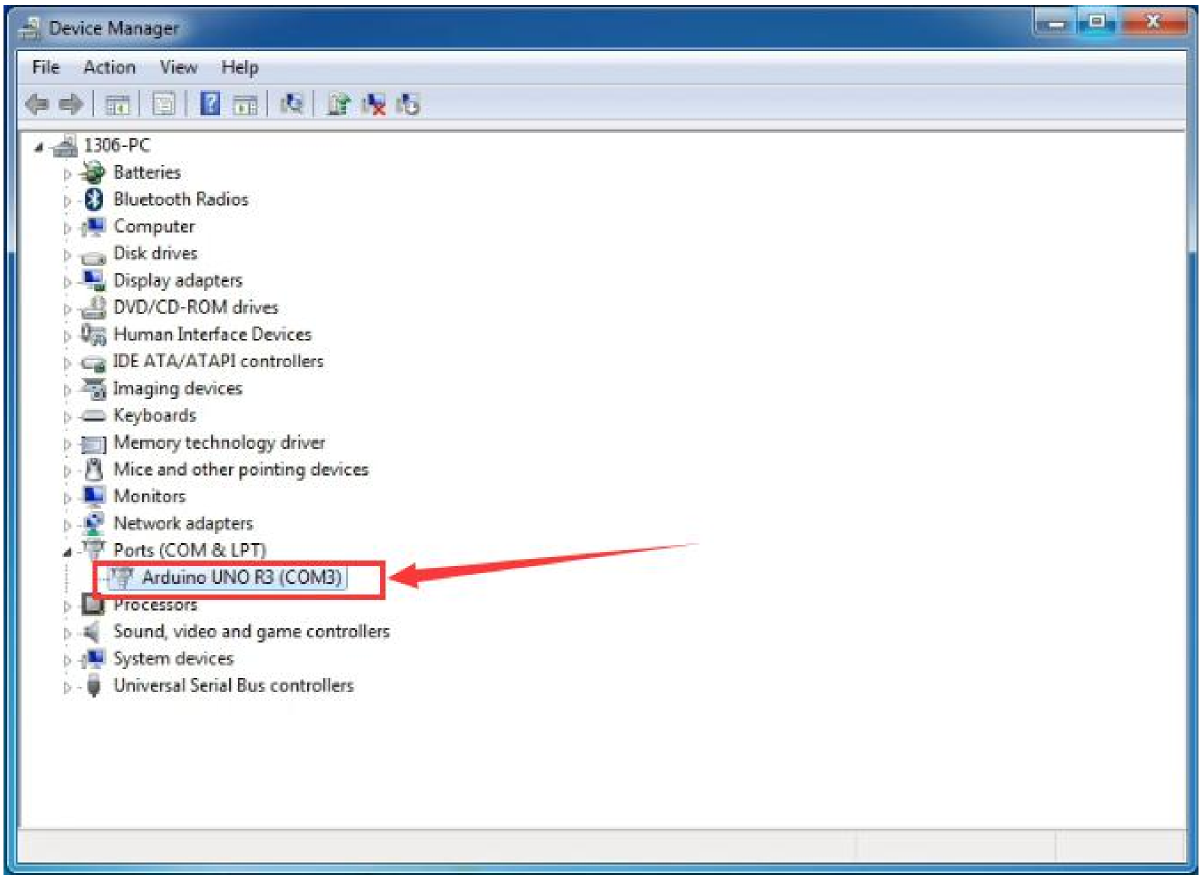
Update Driver Software - Arduino UNO R3 (COM3)

Windows has successfully updated your driver software

Windows has finished installing the driver software for this device:

Arduino UNO R3

Up to now, the driver is installed well. Then you can right click *"Computer"* *—>"Properties"—>"Device manager"*, you should see the device as the figure shown below.



Displaying Hello World

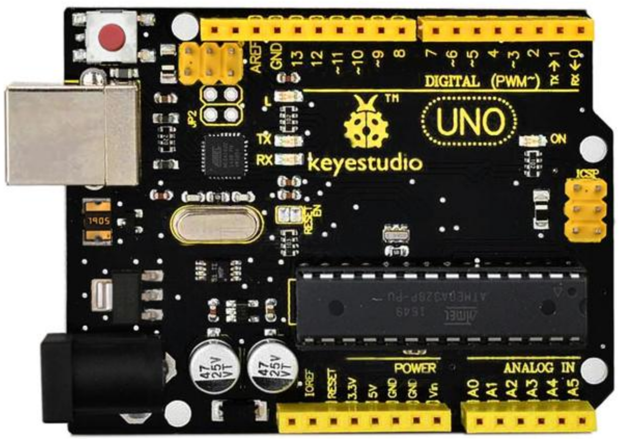
Overview

It is very simple. You can use only a main board and a USB cable to display the "Hello World!". It is a communication experiment between the control board and PC. This is an entry experiment for you to enter the Arduino programming world. Note that need to use a serial communication software, Arduino IDE. In the above part, you can check the detailed use of Arduino IDE.

Component Required:

- UNO R3 control board*1
- USB cable*1

Component Introduction:



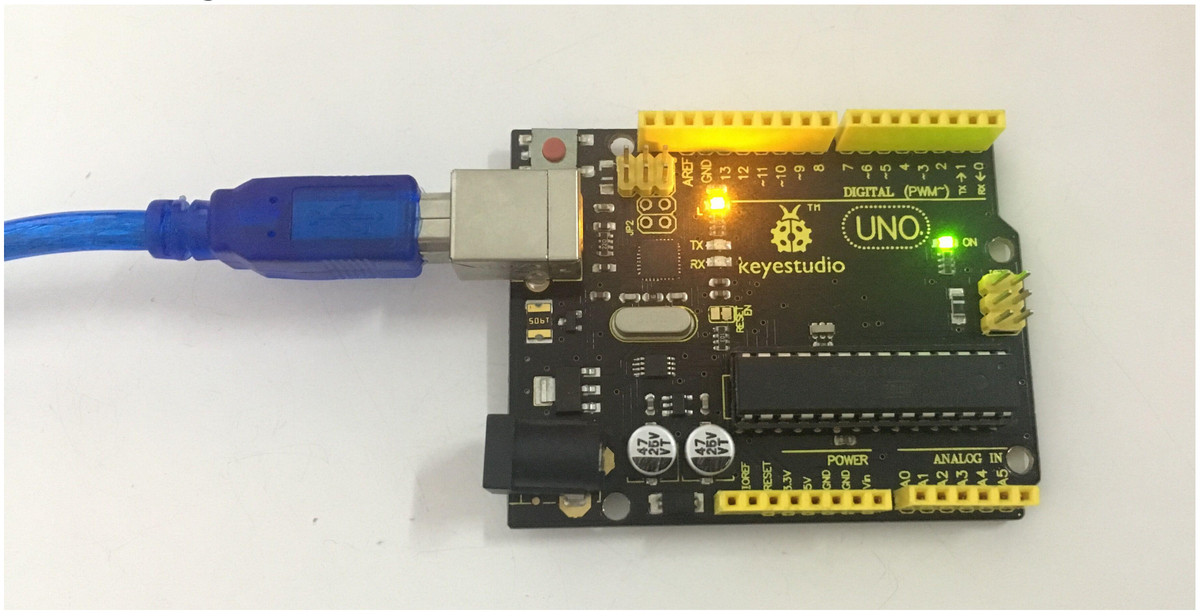
keyestudio UNO R3 Control Board

Keyestudio UNO R3 development board is a microcontroller board based on the ATmega328P (datasheet), fully compatible with ARDUINO UNO REV3. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, 2 ICSP headers and a reset button.

It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Connect It Up

Connect the UNO board to your computer using the USB cable. The green power LED should go on.



Upload the Code

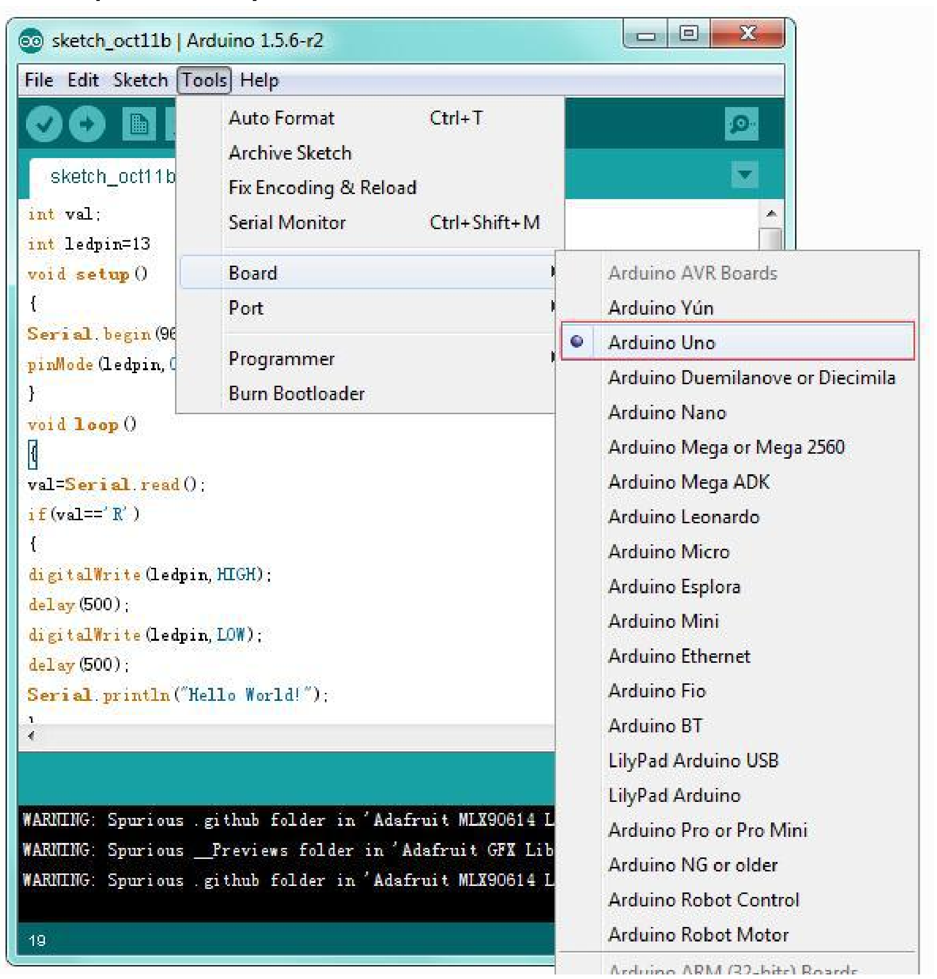
Below is an example code for displaying the Hello World!

```
//////////////////////////////////////////////////////////////////
int val;
int ledpin=13;
void setup()
{
Serial.begin(9600);
pinMode(ledpin,OUTPUT);
}
void loop()
{
val=Serial.read();
if(val=='R')
{
```

```
digitalWrite(ledpin,HIGH);
delay(500);
digitalWrite(ledpin,LOW);
delay(500);
Serial.println("Hello World!");
}
}
////////////////////////////////////////////////////////////////////////////////
```

Select the Arduino Board

Open the Arduino IDE, you'll need to click the "Tools", then select the Board that corresponds to your Arduino.

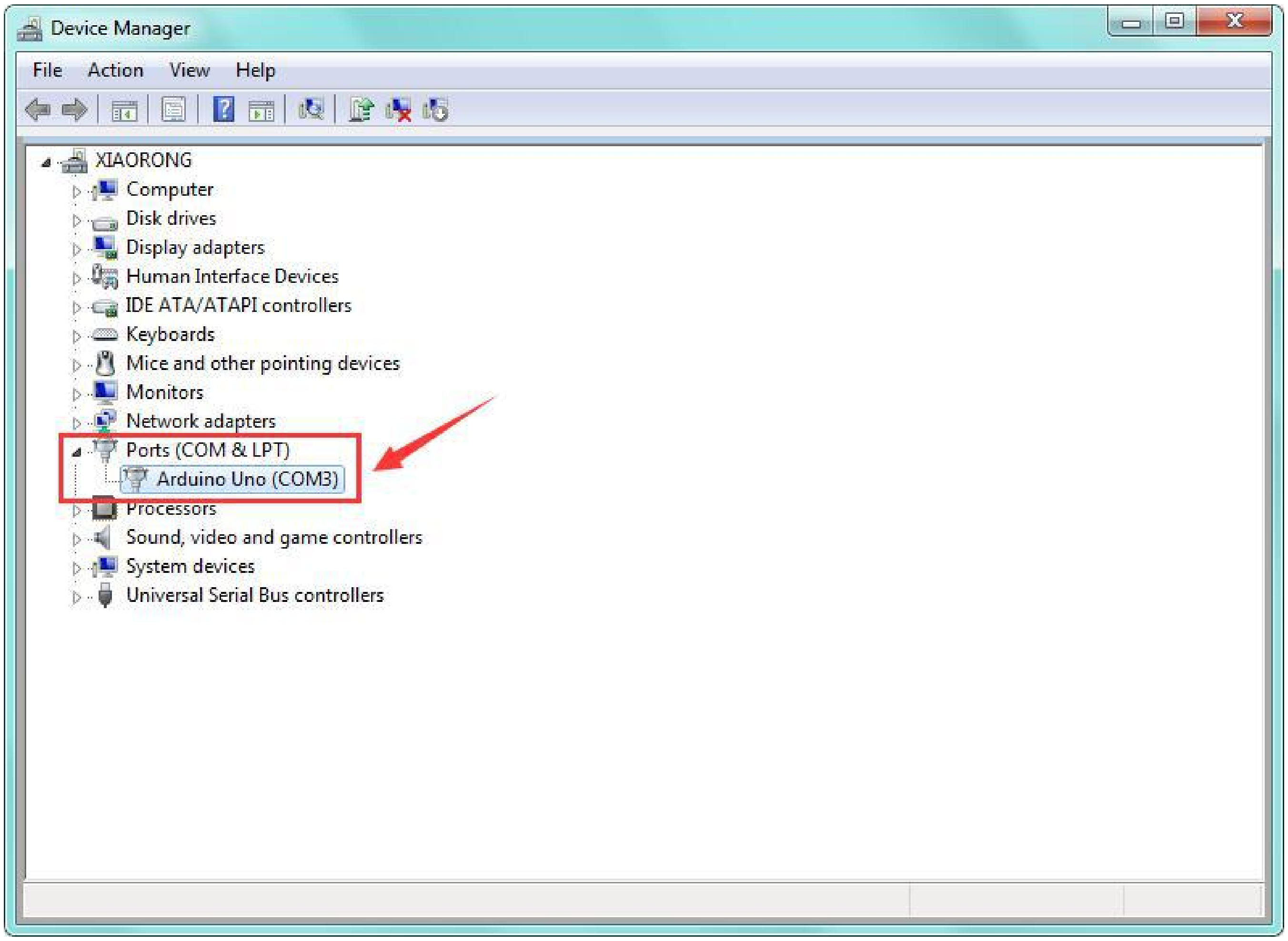


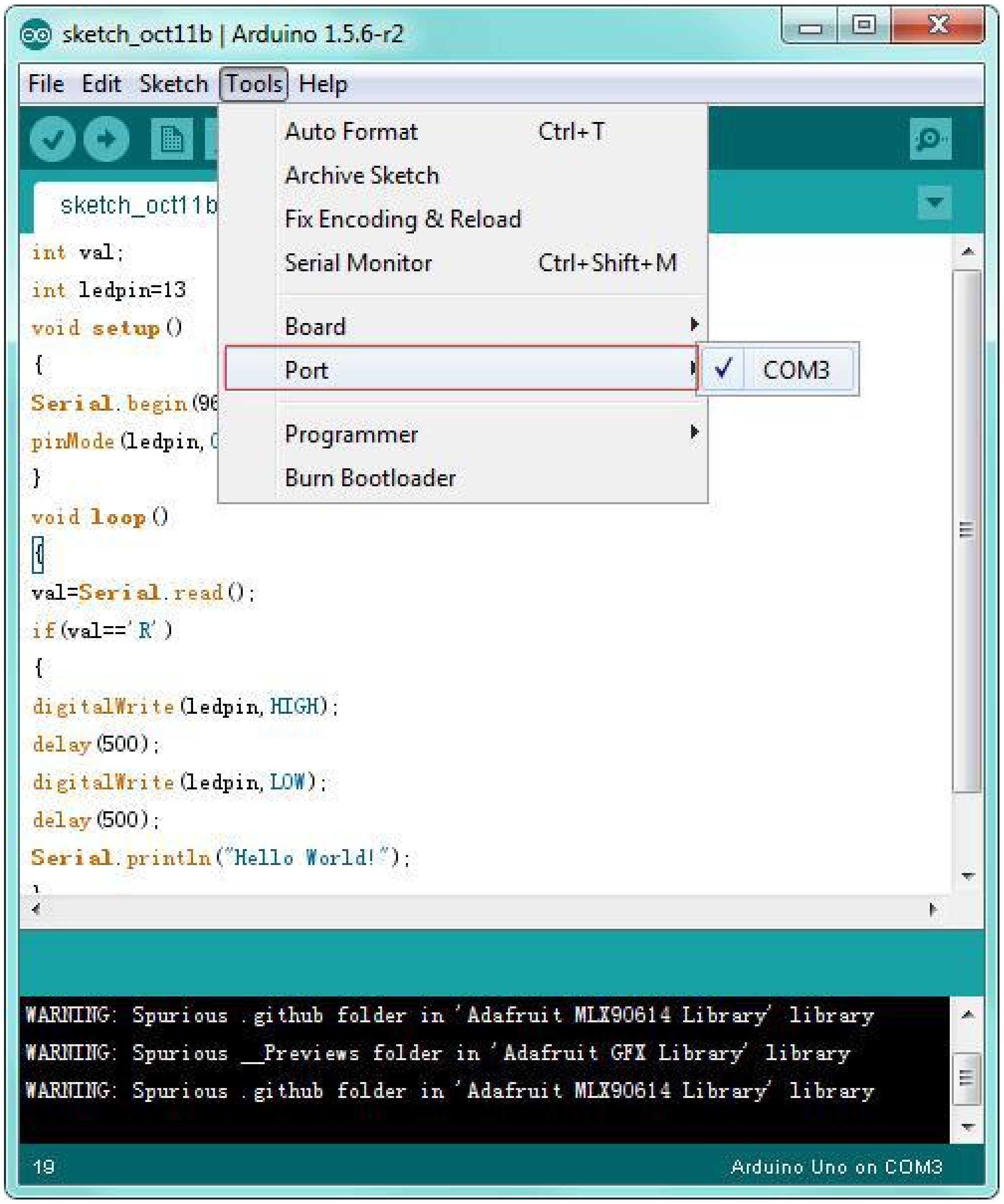
```
int val;
int ledpin=13
void setup()
{
Serial.begin(96
pinMode(ledpin,(
}
void loop()
{
val=Serial.read();
if(val=='R')
{
digitalWrite(ledpin,HIGH);
delay(500);
digitalWrite(ledpin,LOW);
delay(500);
Serial.println("Hello World!");
```

```
WARNING: Spurious .github folder in 'Adafruit MLX90614 L
WARNING: Spurious __Previews folder in 'Adafruit GFX Lib
WARNING: Spurious .github folder in 'Adafruit MLX90614 L
```

Select your serial port

Select the serial device of the Arduino board from the Tools | Serial Port menu. **Note:** to avoid errors, the COM Port should keep the same as the Ports shown on Device Manager.



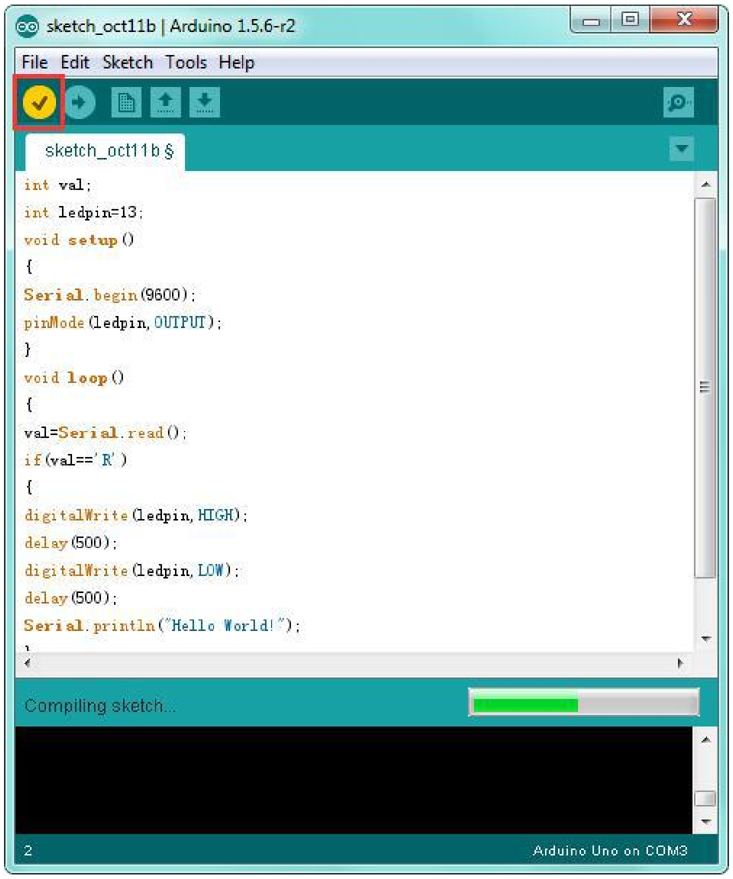


```
int val;
int ledpin=13
void setup()
{
Serial.begin(96
pinMode(ledpin,O
}
void loop()
{
val=Serial.read();
if(val=='R')
{
digitalWrite(ledpin,HIGH);
delay(500);
digitalWrite(ledpin,LOW);
delay(500);
Serial.println("Hello World!");
}
```

```
WARNING: Spurious .github folder in 'Adafruit MLX90614 Library' library
WARNING: Spurious __Previews folder in 'Adafruit GFX Library' library
WARNING: Spurious .github folder in 'Adafruit MLX90614 Library' library
```

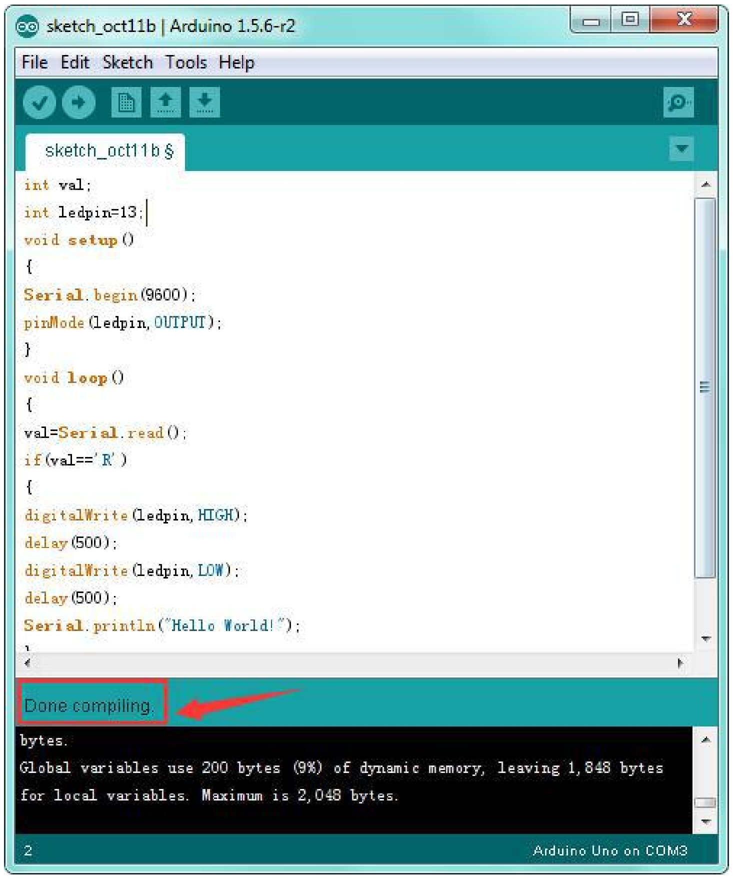
19 Arduino Uno on COM3

Then click verify button to check the errors. If compiling successfully, the message "Done compiling" will appear in the status bar.



```
int val;
int ledpin=13;
void setup()
{
Serial.begin(9600);
pinMode(ledpin,OUTPUT);
}
void loop()
{
val=Serial.read();
if(val=='R')
{
digitalWrite(ledpin,HIGH);
delay(500);
digitalWrite(ledpin,LOW);
delay(500);
Serial.println("Hello World!");
```

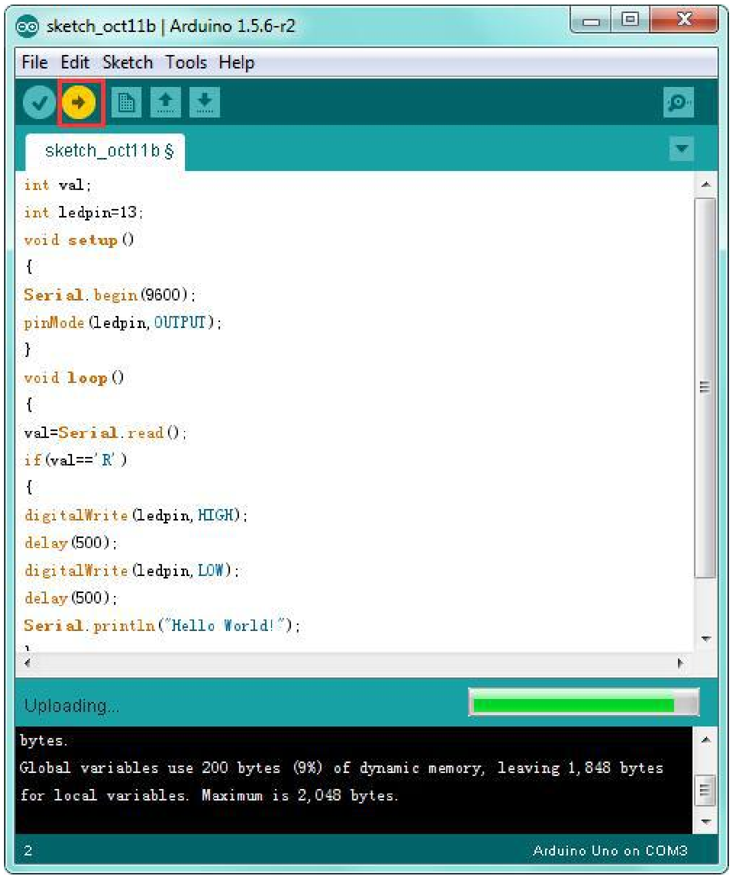
```
Compiling sketch...
```



```
int val;
int ledpin=13;
void setup()
{
Serial.begin(9600);
pinMode(ledpin,OUTPUT);
}
void loop()
{
val=Serial.read();
if(val=='R')
{
digitalWrite(ledpin,HIGH);
delay(500);
digitalWrite(ledpin,LOW);
delay(500);
Serial.println("Hello World!");
```

```
Done compiling.
bytes.
Global variables use 200 bytes (9%) of dynamic memory, leaving 1,848 bytes
for local variables. Maximum is 2,048 bytes.
```

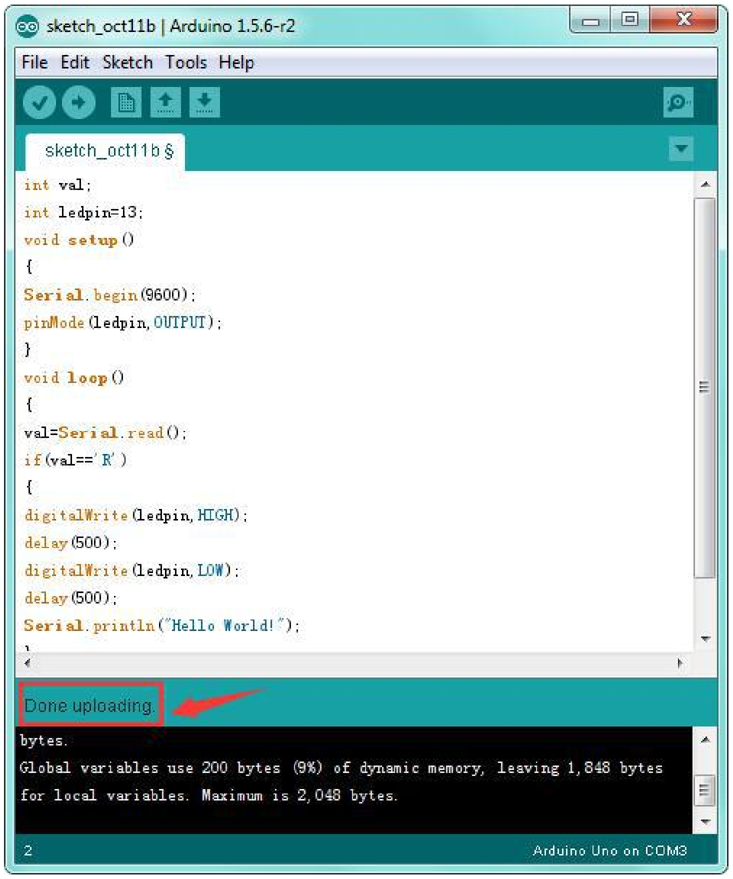
After that, click the “Upload” button to upload the code. If the upload is successful, the message "Done uploading" will appear in the status bar.



```
int val;
int ledpin=13;
void setup()
{
Serial.begin(9600);
pinMode(ledpin,OUTPUT);
}
void loop()
{
val=Serial.read();
if(val=='R')
{
digitalWrite(ledpin,HIGH);
delay(500);
digitalWrite(ledpin,LOW);
delay(500);
Serial.println("Hello World!");
```

Uploading...

```
bytes.
Global variables use 200 bytes (9%) of dynamic memory, leaving 1,848 bytes
for local variables. Maximum is 2,048 bytes.
```



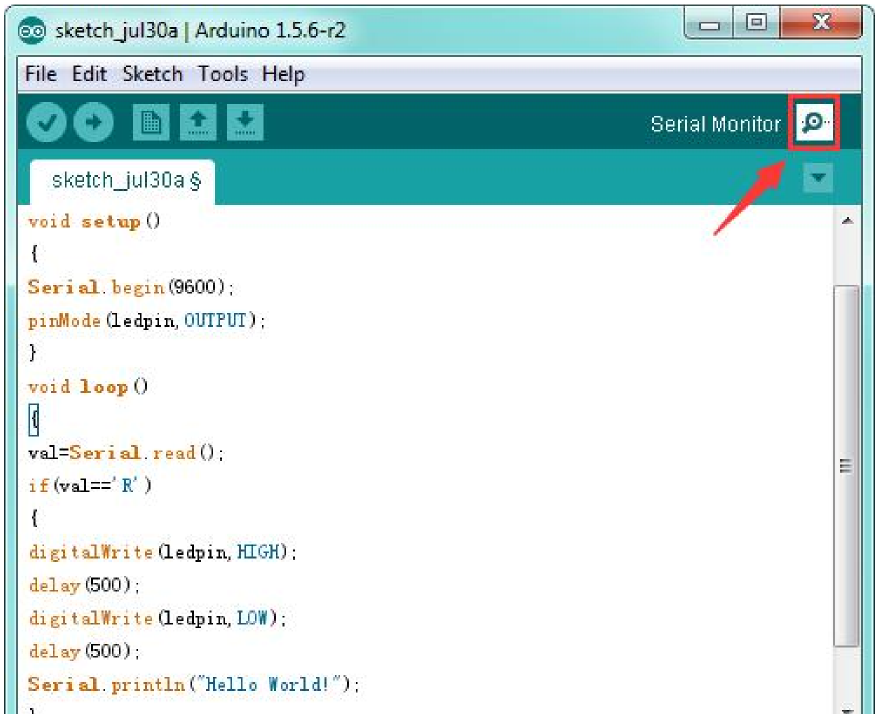
```
int val;
int ledpin=13;
void setup()
{
Serial.begin(9600);
pinMode(ledpin,OUTPUT);
}
void loop()
{
val=Serial.read();
if(val=='R')
{
digitalWrite(ledpin,HIGH);
delay(500);
digitalWrite(ledpin,LOW);
delay(500);
Serial.println("Hello World!");
```

Done uploading.

```
bytes.
Global variables use 200 bytes (9%) of dynamic memory, leaving 1,848 bytes
for local variables. Maximum is 2,048 bytes.
```

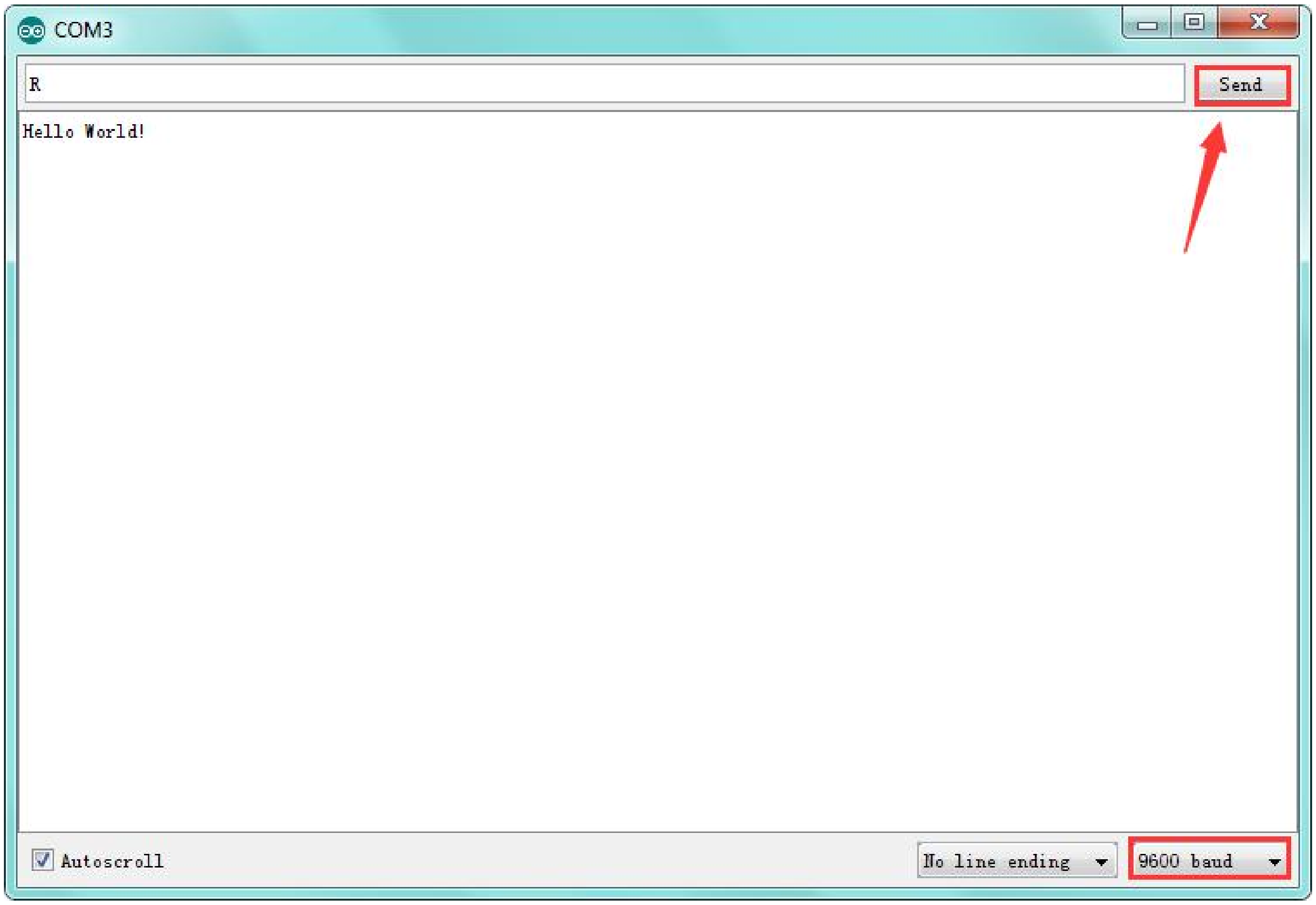
Open the Serial Monitor

After that, click the monitor button to open the serial monitor.



```
void setup()
{
Serial.begin(9600);
pinMode(ledpin,OUTPUT);
}
void loop()
{
val=Serial.read();
if(val=='R')
{
digitalWrite(ledpin,HIGH);
delay(500);
digitalWrite(ledpin,LOW);
delay(500);
Serial.println("Hello World!");
```

Then set the baud rate as 9600, enter an “R” and click Send, you should see the RX led on the board blink once, and then D13 led blink once, finally "Hello World!" is showed on the monitor, and TX led blink once.
Congrats! Your first simple program is complete.



```
Hello World!
```

9600 baud

Other Links:

You might also want to look at:

the examples for using various sensors and actuators;

the reference for the Arduino language;

You can download the UNO datasheet from the link:

https://drive.google.com/open?id=1PQnMRVBaPwLdfzw_7T3OlLhOyEP0rY_S

Software Download:

https://www.arduino.cc/en/Main/OldSoftwareReleases#1.5.x

Troubleshooting:

If you have problems, please see the troubleshooting suggestions.