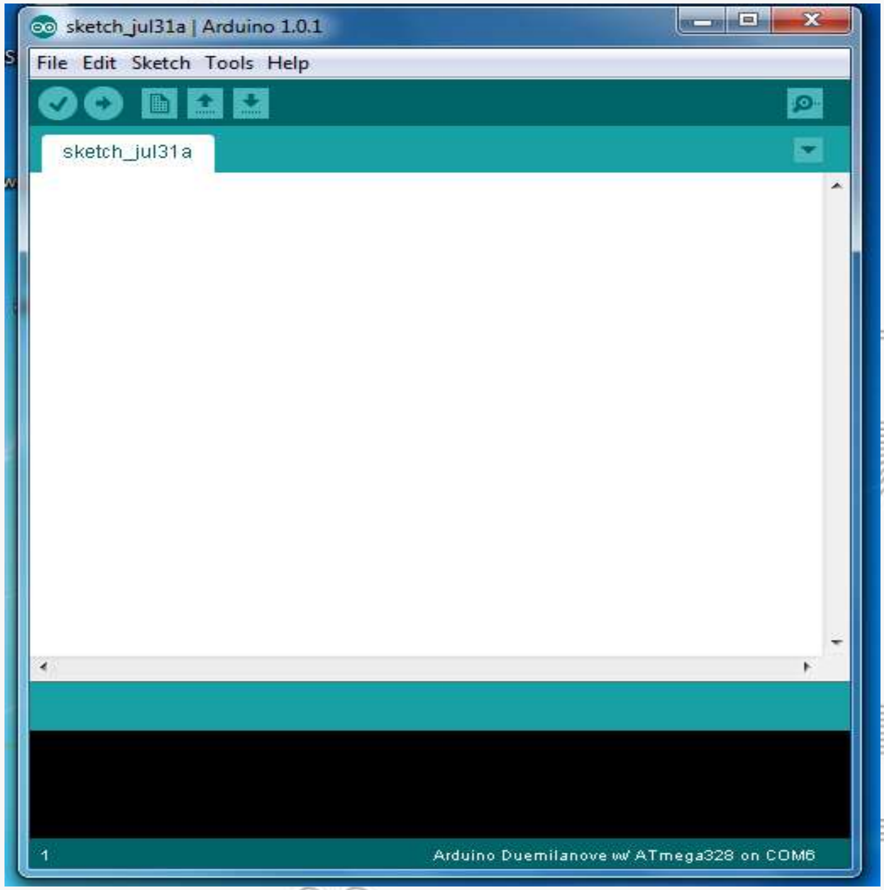
Introduction to Arduino IDE

ABLEDUCATION™
ACTIVITY BASED LEARNING

Arduino Software

The Arduino programming platform was designed in JAVA to help newcomers become familiar with programming. The language used to write programs (codes)in C/C++ and only uses two functions to make a functional program.


sketch_jul31a | Arduino 1.0.1
File Edit Sketch Tools Help
sketch_jul31a
1
Arduino Duemilanove w/ ATmega328 on COM6

Structure of Arduino

The basic structure of the arduino programming language is fairly simple & runs in two parts. These two parts are :

- Functions
- Enclose blocks of statement .

```
void setup ( )
{
   statements;
}
 void loop ( )
 {
    statements;
 }
```

- Setup() is the preparation , loop() is an execution.

- The setup function should follow the declaration of any variable at the beginning of the program.

- It is the first function to run in the program , is run only once, and is used to set pinMode or initialize serial communication.

- The loop function follow next and includes the code to be executed continuously reading inputs , triggering outputs etc.

Setup()

- The setup() function is called once when your program starts. It is used to initialize pin modes, or begin serial monitor.
- It must be included in a program even if there are no statements to run.

```
 void setup()
{
   pinMode(pin, OUTPUT);      //  sets the 'pin ' as  output
}
```

Loop()

After calling the setup() function, the loop() function does precisely what its name suggests, and loops consecutively allowing to change, respond, and control the arduino board.

```
void loop()
{
  digitalWrite(pin, HIGH);      // turns ‘pin ’ on
  delay(1000);                  // pauses  for  one second
   digitalWrite(pin, LOW);      // turns  ‘pin’  off
   delay(1000);                 //  pauses for one second
}
```

Programming Syntax

Similar to other programming, the formatting requirement is the same.

// - Single line comment
/* */ - Multiline comment
{ } – used to define a block of code that starts and ends.
; - used to define the end of a line of programs(codes).

Variables

- A variable is a way of naming and storing a numerical value for later use by the program.
- A variable needs to be declared and optionally assigned to the value to be stored.
- A variable can be named any word that is not already one of the keywords in the arduino language.
- Declaring a variable means defining its value type, as int, long, float etc.

Variables

- The following program declares a variable called input Variable and then assigns it the value obtained on analog input pin2.

  int inputVariable = 0; //declare a variable and assign value 0

  inputVariable = analogRead(2); //set variable to value of analog pin2

- 'inputVariable' is the variable itself. The first line declares that it will contain an int(short for integer). The second line sets the variable to the value at analog pin2.

Variable Scope

- A variable can be declared at the beginning of the program before void setup(), locally inside the functions, and sometimes within a statement block such as for loops.
- When the variable is declared, determines the variable scope, or the ability of certain parts of a program to make use of the variable.

Types of variable :-

- A global variable is one that can be seen and used by every function and statement in a program, before the setup() function.
- A local variable is one that is defines inside a function or as a part of for loop. It is only visible and can only be used inside the function in which it was declared.

Example

The following example shows how to declare a few different types of variables and demonstrates each variable’s visibility.

```
 int value ;             // ‘value ’  is visible to any function
  void setup( )
  {
     //no setup needed
   }
  void loop( )
 {
  for (int i=0; i<20; )     // ‘i’ is visible only inside loop
 {
     i++;
  }
    float f ;                    // f is visible only inside loop
 }
```

Different types of variable datatypes

- **byte - ** byte stores an 8 bit numerical value without decimal points. They have a range of 0-255 .

  byte someVariable = 180; //declares 'someVariable' as a byte type

- **int - ** Integers are the primary datatype for storage of numbers without decimal points and store a 16 -bit value with a range of 32,767 to

  -32,768.

  int someVariable = 1500; //declares 'someVariable' as an integer type

Different types of variable datatype

- **long -** Extended size datatype for long integers, without decimal points, stored in a 32- bit value with a range of 2,147,483,647 to

  -2,147,483,648 .

  long someVariable = 90000; //declares 'somevariable 'as a long type
- **float –** A datatype for floating point numbers, or numbers that have a decimal point are stored 32bit value a range of 3.4028235E+38 to

  -3.4028235E+38.

  float someVariable =3.14; //declares 'someVariable' as a floating type

Constants

Arduino programming has a few predefined values, which are called constants. They are used to make the programs easier to read. Constants are classified in groups.

- **TRUE/FALSE -** These are boolean constants that define logic levels.

FALSE is easily defined as zero while TRUE is often defined as one ,but can also be anything else except zero . So in a boolean sense , -1,2,and -200 are also defined as TRUE.

```
if(b== TRUE)
 {
   do Something;
  }
```

Constants

- **HIGH/LOW** - These constant define pin levels as HIGH or LOW and are used when reading or writing digital pins . HIGH is defined as logic 1,ON,or 5 volts while LOW is defined as logic 0,OFF,or 0 volts.

  digitalwrite (13, HIGH) ;

- **Input/output** - Constant used with the pinMode() function to define the mode of a digital pin as either INPUT or OUTPUT.

  pinMode(13, OUTPUT); //define pin no 13 as OUTPUT

Conditional Statements

- **if –** if statement test whether a certain condition has been reached, such as an analog value being above a certain number, and executes any statements inside the brackets if the statement is true, if false the program skips over the statement.

```
if(someVariable ?? value)
{
   doSomething;
}
```

- **if…else –** This is decisions making conditional statement . If you wanted to test a digital input , and do one thing if the input goes HIGH or instead do another thing, if the input was LOW, do another operation as follows :-

```
 if (inputPin == HIGH)
 {
   doThingA;
  }
else
{
   doThingB;
}
```

- **for –** The for loop is used to repeat a block of statements enclosed in curly braces a specified number of times. There are three parts of for loop and they are separated by semicolons(;).

```
 for(initialization; condition ; expression)
{
   doSomething;
 }
```

Example : Starts the integer ‘i’ at 0, test to see if ‘i’ is still less than 20 and if true ,increments ‘i’ by 1 and executes the enclosed statements.

```
for (int i=0; i<20 ; i++)  //declare I,tests if less than 20 ,increments i by 1
{
   digitalWrite (13, HIGH);  // turn pin 13 on
    delay(250);        // pauses for ¼ second
    digitalWrite(13,LOW);   // turns pin 13  off
}
```

- **while** - while loop will loop continuously , and infinitely , until the expression inside the parenthesis becomes false.

```
while (someVariable  <  200)     // test if less than 200
{
  doSomething;                         //executes enclosed statements
  someVariable++;                      // increments variable by 1
}
```

- **do…while** - The do while loop is same as the while loop but in this loop the condition is tested at the end of the loop , so the do while loop will always run at least once.

```
  do
 {
    x=readSensors();    //assign the value of readSensors() to x
    delay(50);              //pauses 50 millisecond
   } while(x<100);      // loops if x is less than 100
```

Commands

- **pinMode -** pinMode used in void setup() to configure a specified pin to behave either as an INPUT or an OUTPUT .

  pinMode(pin, OUTPUT); //set 'pin' to output

- **digitalRead(pin) -** Reads the value from a specified digital pin with the result either HIGH or LOW .

  value = digitalRead(pin); //set the value equal to the input pin

- **digitalWrite(pin ,value) -** Outputs either logic level HIGH or LOW

  The pin can be specified as either a variable or constant(0-13) .

  digitalWrite(pin , HIGH); //set 'pin' to high

Example

The following example reads a pushbutton connected to a digital input and turns on an LED connected to a digital output when the button has been pressed.

```
int led  = 13;  // connected led to pin 13
 int pin  = 7;    // connected  pushbutton  to pin 7
 int value = 0;  // variable  to store the read value
void setup( )
{
   pinMode(led, OUTPUT) ;        //set pin 13 as Output
   pinMode(pin, INPUT) ;           // set pin 7 as input
 }
void loop( )
{
   value = digitalRead(pin) ;     // set 'value' equal to the input pin
    digitalWrite(led, value) ;     // set 'led' to the button's value
}
```

Commands

- **analogRead(pin)**

Reads the value from a specified analog pin with a 10 bit resolution. This function only works on the analog pins (0-5). The resulting integers values range from 0 to 1023.

value = analogRead(pin); // set ‘value’ equal to ‘pin’

Note : Analog pins do not need to be declared as INPUT nor OUTPUT.

- **analogWrite(pin, value)**

Write the value from a specified analog pin .

analogWrite(pin , value) ; //write ‘value’ to analog ‘pin’

Example

The following example reads an analog value from an analog input pin, converts the value by dividing by 4, and output a PWM pin.

```
int led =10;    // led  with 220 ohm resistor on pin 10
 int pin = 0;     // potentiometer on analog pin 0
 int value ;      // value for reading
 void setup ( ) {}   // no setup needed
 void loop( )
 {
     value = analogRead(pin);   //set 'value' equal to 'pin'
      value = 4;                          // convert 0-1023 to 0-255
     analogWrite(led , value);    // output PWM signal to led
 }
```

Commands

- **Serial.begin(rate) -** Open serial port and set the baud rate for serial data transmission . The typical baud rate for communicating with the computer is 9600 although other speeds are supported .

```
void setup( )
 {
   serial.begin(9600);  // open serial port set data rate to 9600 bps
 }
```

Note : When using serial communication ,digital pin 0(RX) and 1(TX) can not be used at the same time .

- **Serial.println(data) -** Print data to the serial port.

  Serial.println(analogValue); //send the value of '**analogValue**'.

Example

The following simple example takes a reading from analog pin0 and send this data to the computer every 1 second.

```
void setup( )
 {
    Serial.begin(9600);   // set serial communication to 9600 bps
 }
void loop( )
{
  Serial.println(analogRead(0));     // send analog value
  delay(1000);                       // pause for 1 second
}
```

Programming Math Operators

These are used for manipulating numbers.

= (assignment) makes something equal to something else. For example, x = 10*2, thus x = 20.

% (modulo) – this gives the remainder when one number is divided by another. For example 12 % 10 gives 2.

+ (addition)
- (subtraction)
*** (multiplication)**
/ (division)

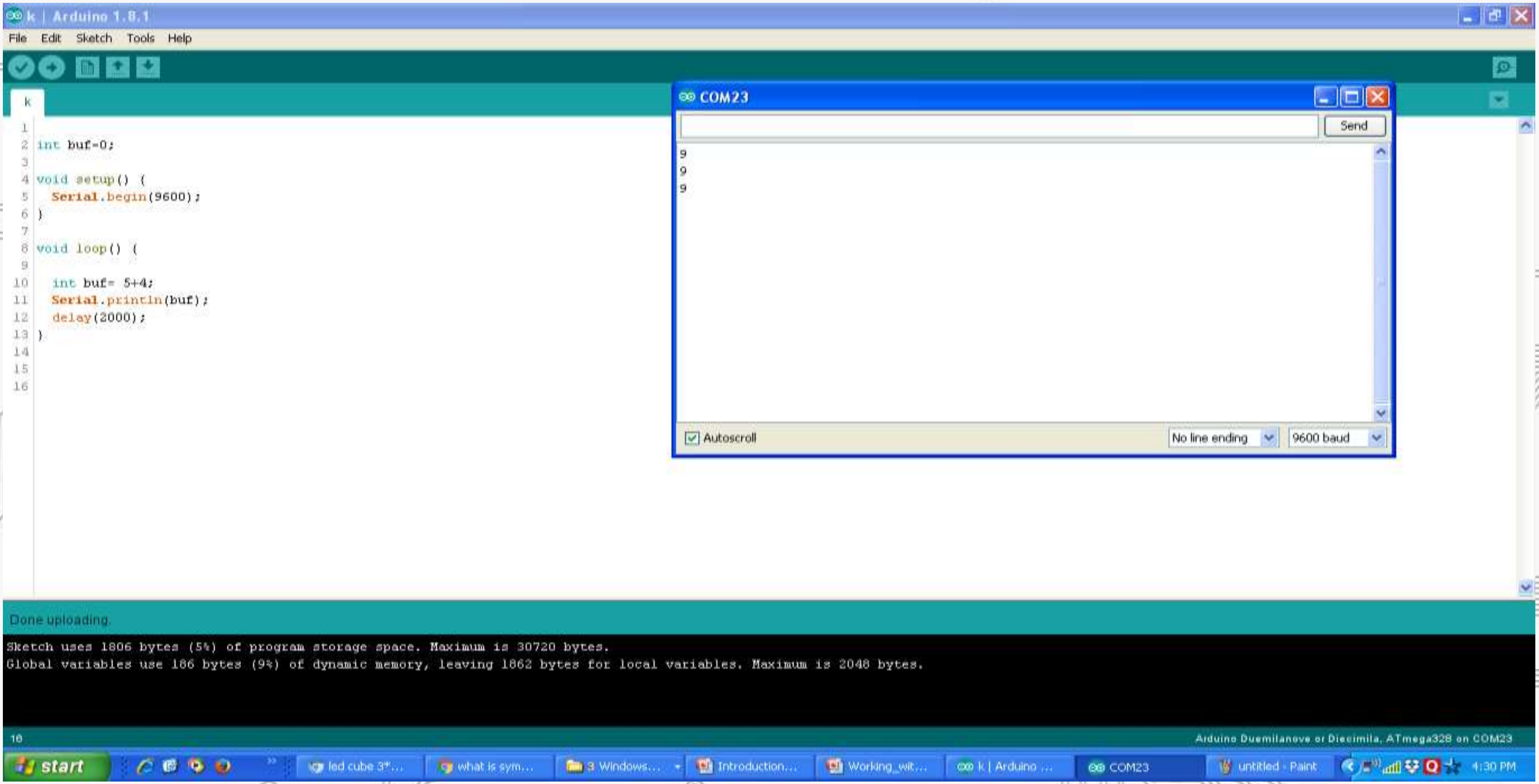
Comparison Operators

- **== (equal to**) - For example 12==10 is FALSE and 12 ==12 is TRUE.

- **!= (not equal to)** - For example 12!=10 is TRUE and 12!=12 is FALSE.

- **< (less than)**
- **> (greater than)**

Sample program: Addition of two Numbers



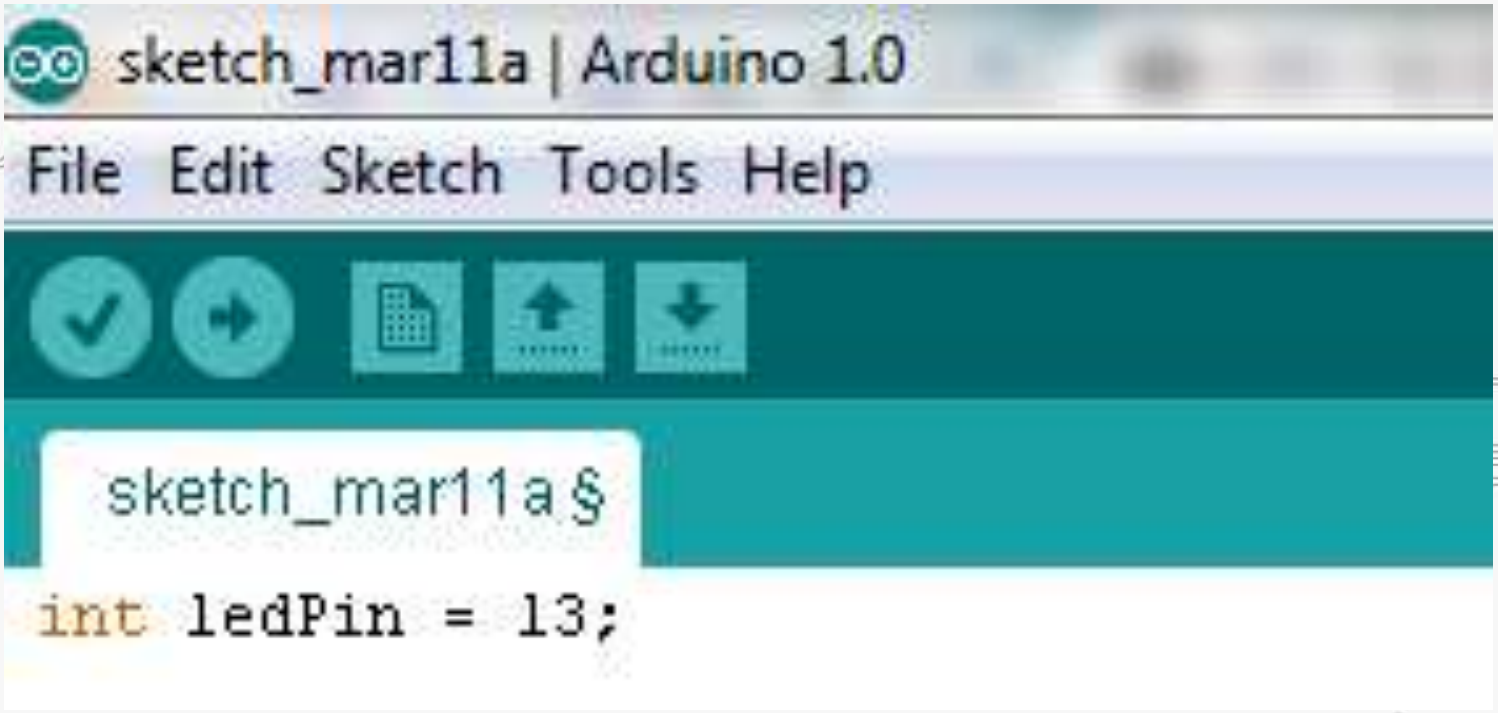
k | Arduino 1.8.1
File Edit Sketch Tools Help
COM23
Send
Autoscroll
No line ending
9600 baud
Done uploading.
Arduino Duemilanove or Diecimila, ATmega328 on COM23

```

int buf=0;

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

void loop() {

  int buf= 5+4;
  Serial.println(buf);
  delay(2000);
}
```

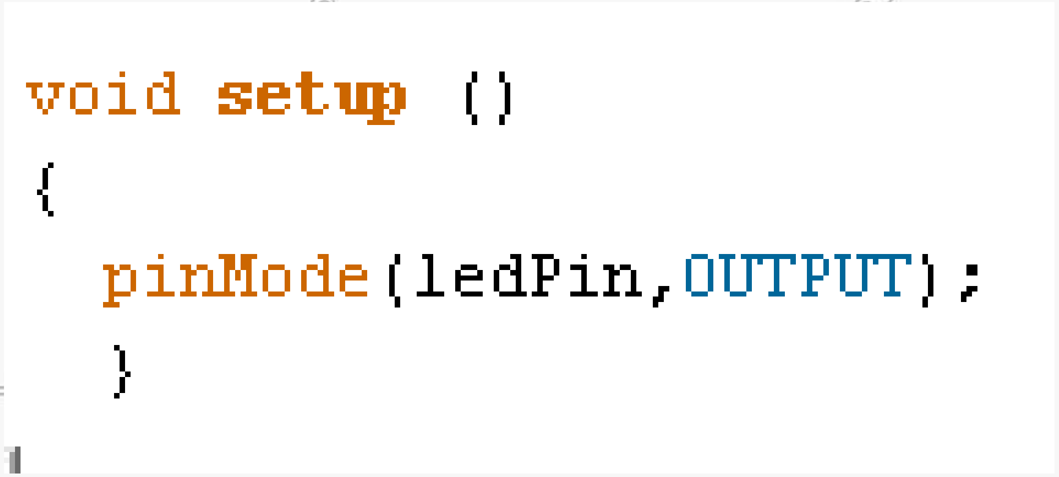
```
9
9
9
```

```
Sketch uses 1806 bytes (5%) of program storage space. Maximum is 30720 bytes.
Global variables use 186 bytes (9%) of dynamic memory, leaving 1862 bytes for local variables. Maximum is 2048 bytes.
```

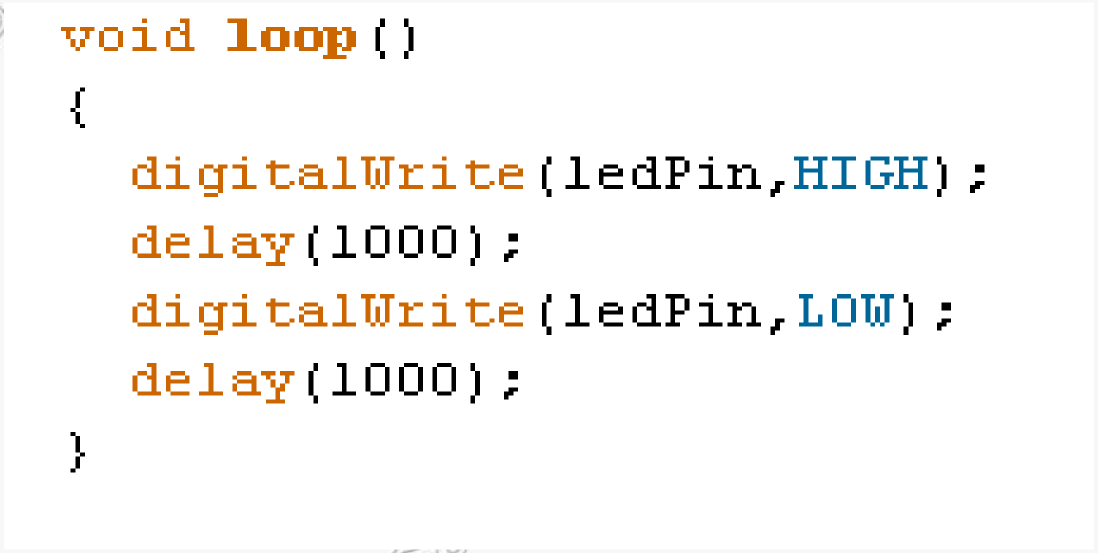
Sample Program: LED Blinking



```
int ledPin = 13;
```



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



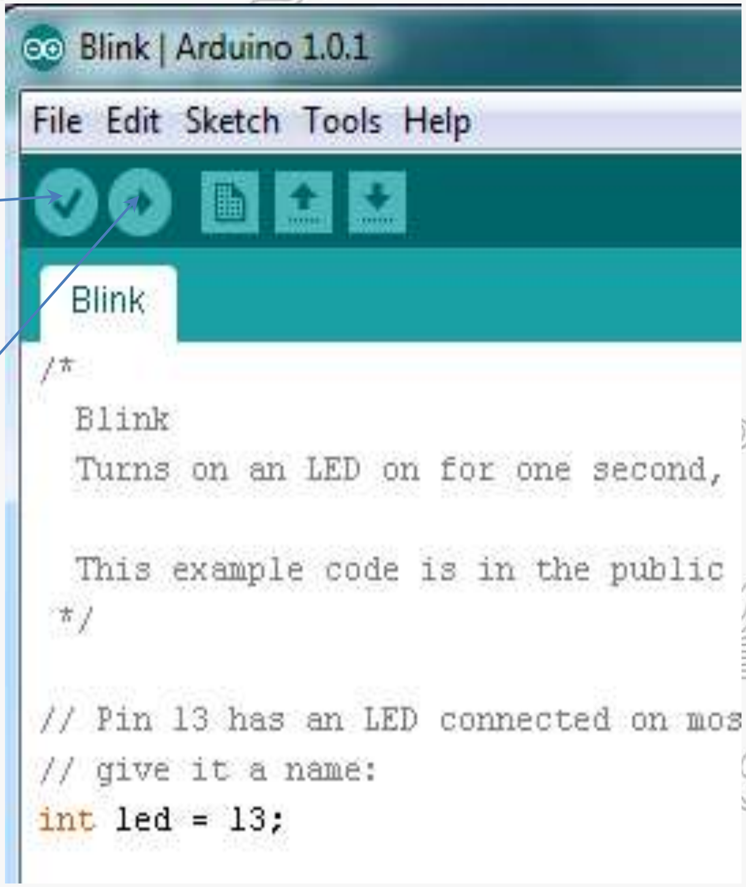
```
void loop()
{
   digitalWrite(ledPin,HIGH);
   delay(1000);
   digitalWrite(ledPin,LOW);
   delay(1000);
}
```

Compile & Upload

To compile your sketch, click the checkmark.

Make sure your Arduino is plugged into an available USB port.

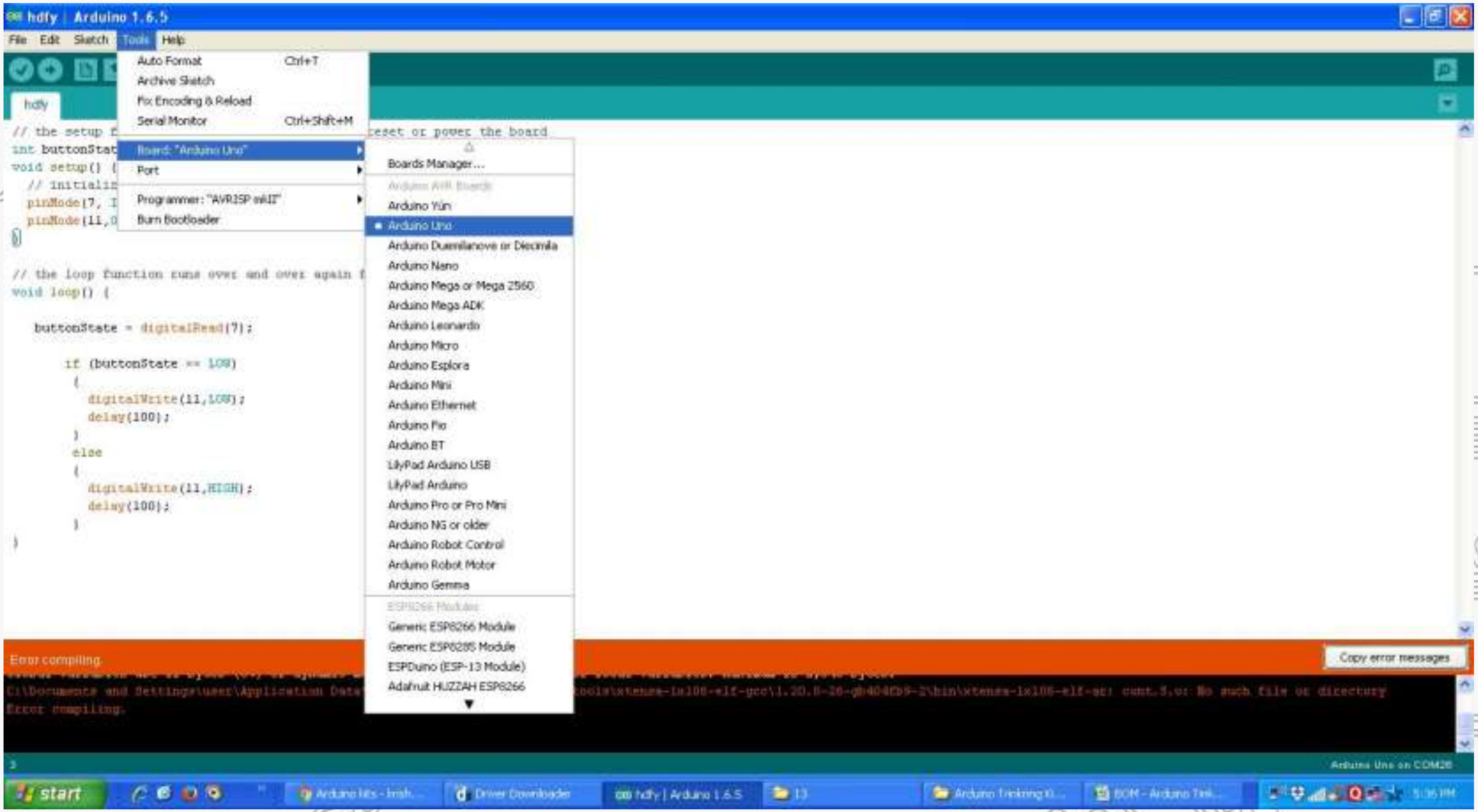
Click the arrow to upload the program to Arduino. If everything is attached correctly. The LED should blink.



```
/*
  Blink
  Turns on an LED on for one second,

  This example code is in the public
 */

// Pin 13 has an LED connected on mos
// give it a name:
int led = 13;
```

ABLEDUCATION™
ACTIVITY BASED LEARNING

Select board to upload program



```
// the setup f                              reset or power the board
int buttonStat
void setup() {
  // initializ
  pinMode(7, I
  pinMode(11,O
}

// the loop function runs over and over again f
void loop() {

  buttonState = digitalRead(7);

      if (buttonState == LOW)
        {
          digitalWrite(11,LOW);
          delay(100);
        }
        else
        {
          digitalWrite(11,HIGH);
          delay(100);
        }
}
```

Error compiling.

```
C:\Documents and Settings\user\Application Data              tools\xtensa-lx106-elf-gcc\1.20.0-26-gb404fb9-2\bin\xtensa-lx106-elf-ar: cont.S.o: No such file or directory
Error compiling.
```

Arduino Uno on COM26

www.abledućation.com

www.ablkart.com