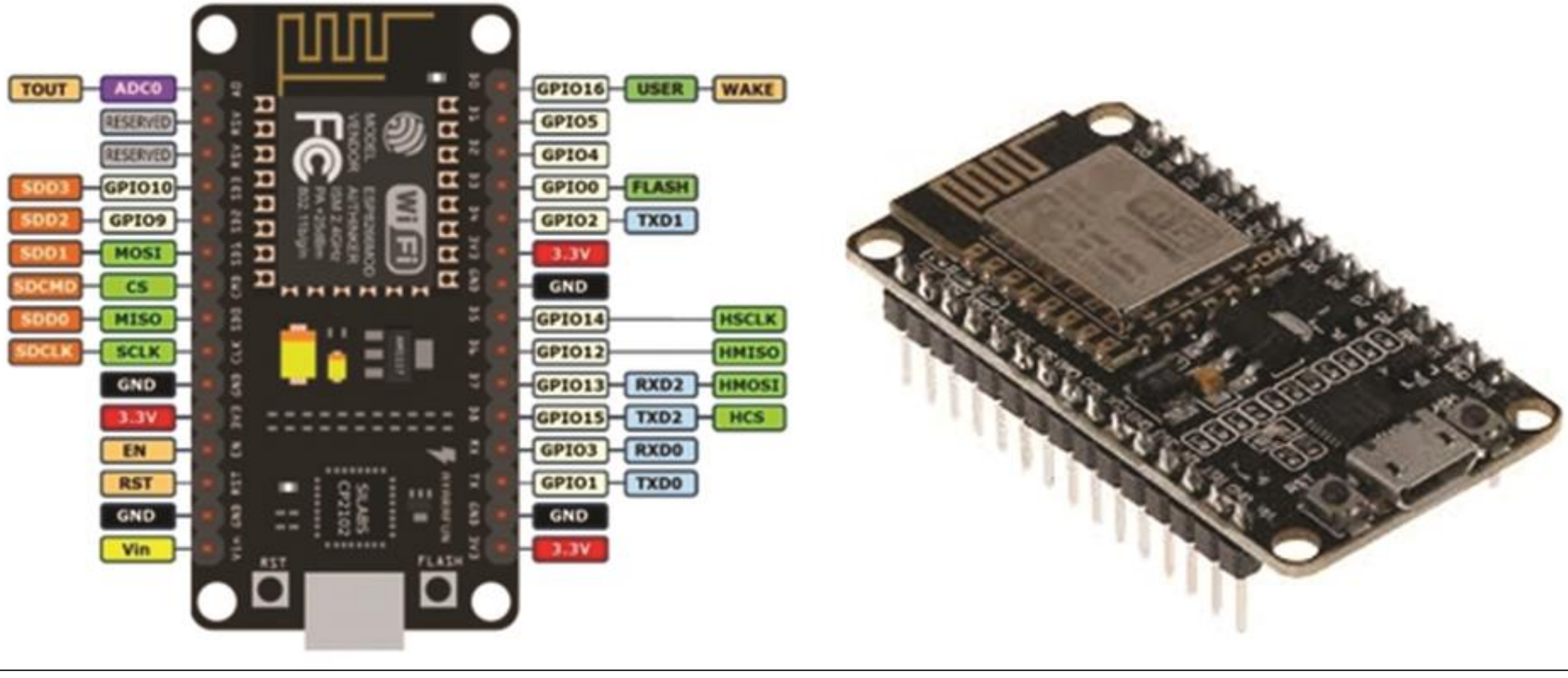
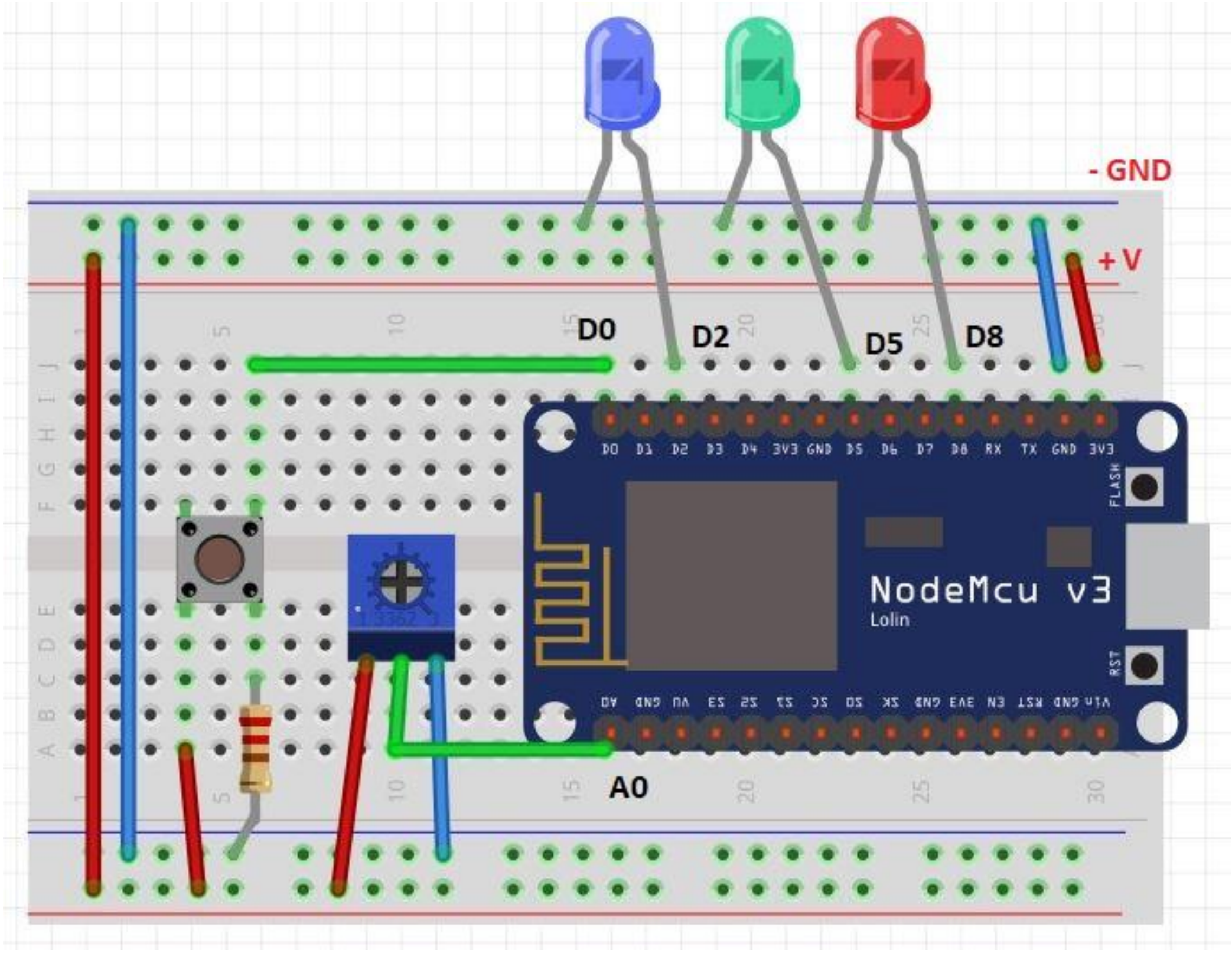
NODEMCU OVERVIEW




- GND
+V
D0
D2
D5
D8
A0
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
FLASH
RST
NodeMcu v3
Lolin
A0 GND VU S3 S2 S1 SC SD SK GND 3V3 EN RST GND Vin
A B C D E F G H I J
1 5 10 15 20 25 30

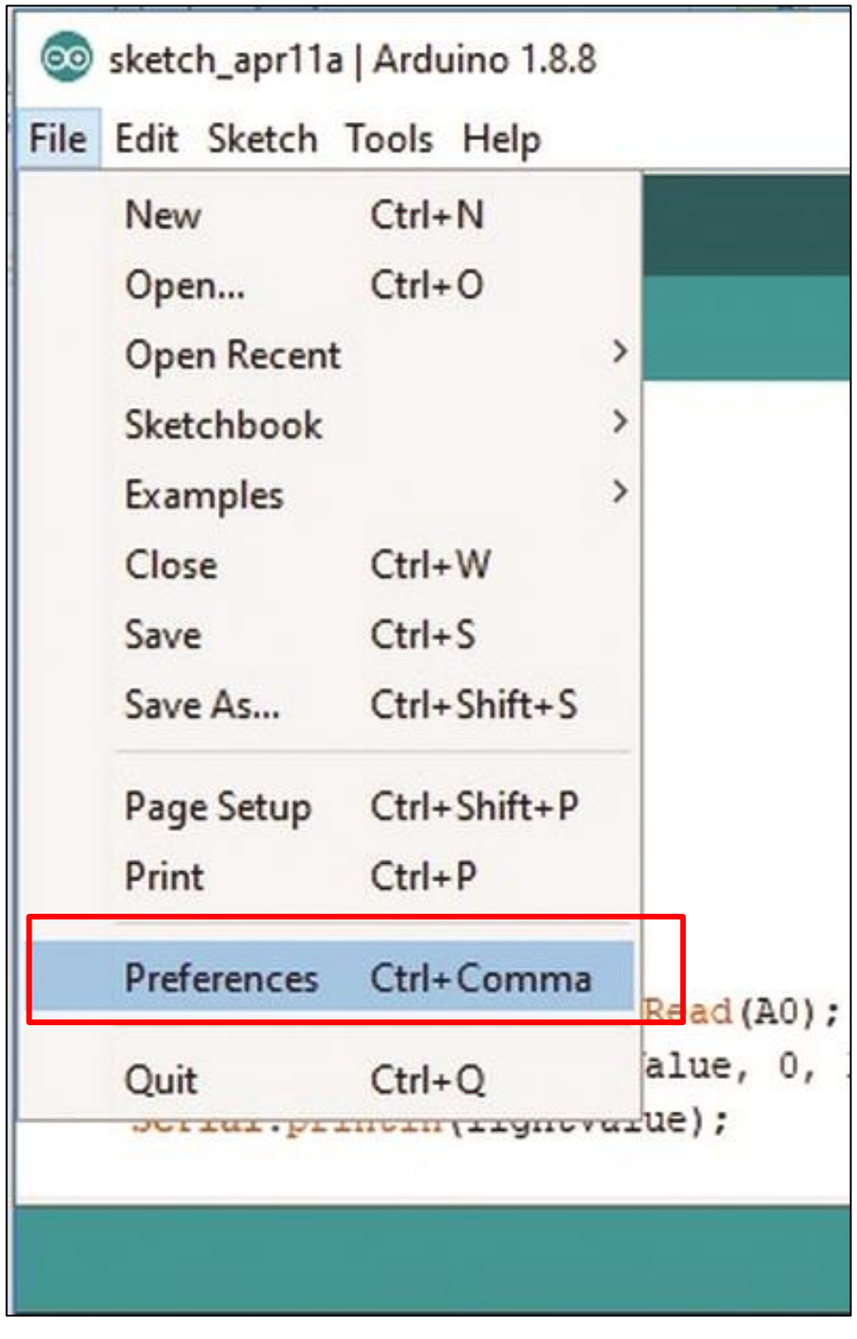
ARDUINO IDE INSTALLATION

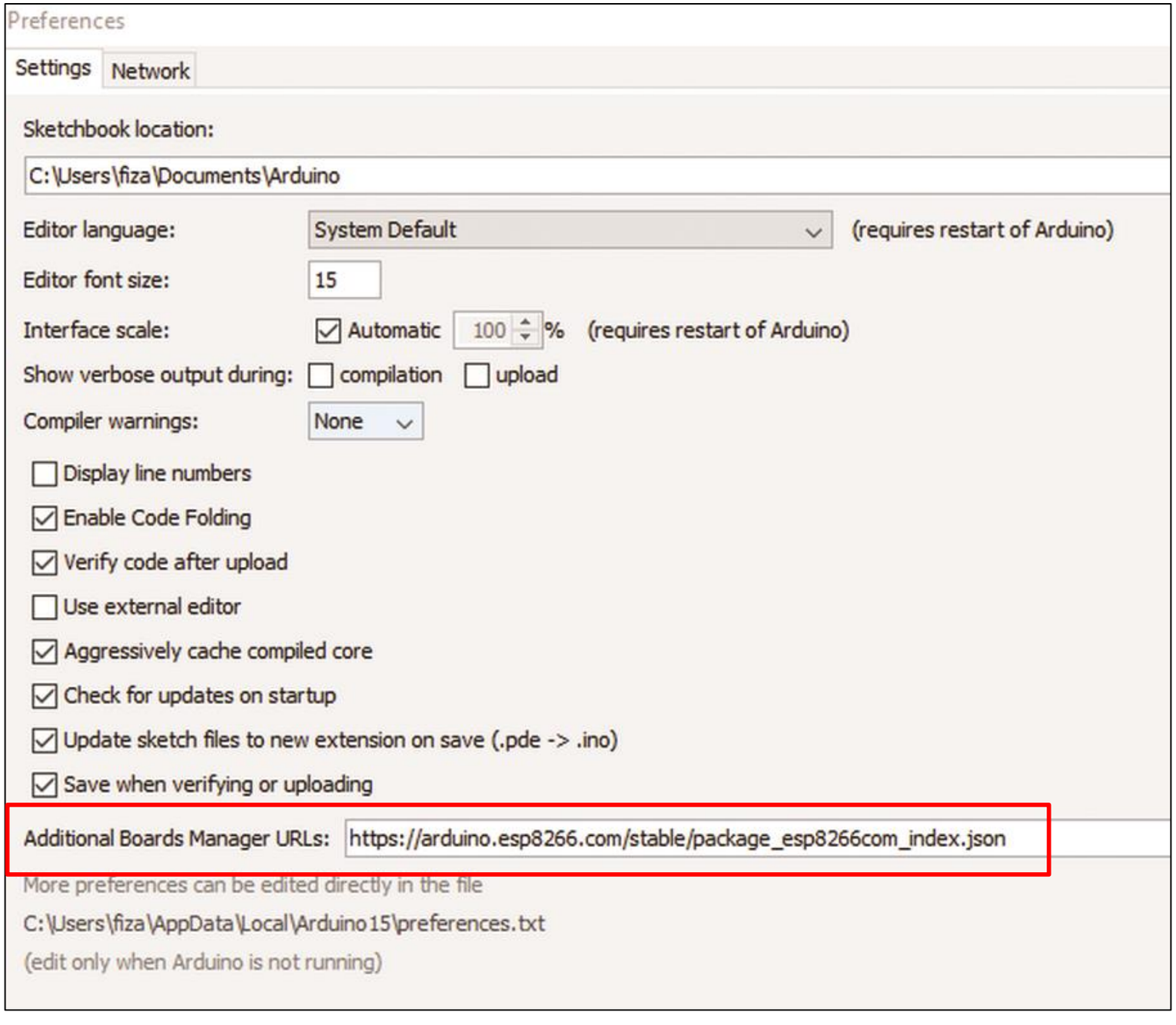
https://www.arduino.cc/en/Main/OldSoftwareReleases

ARDUINO IDE INSTALLATION

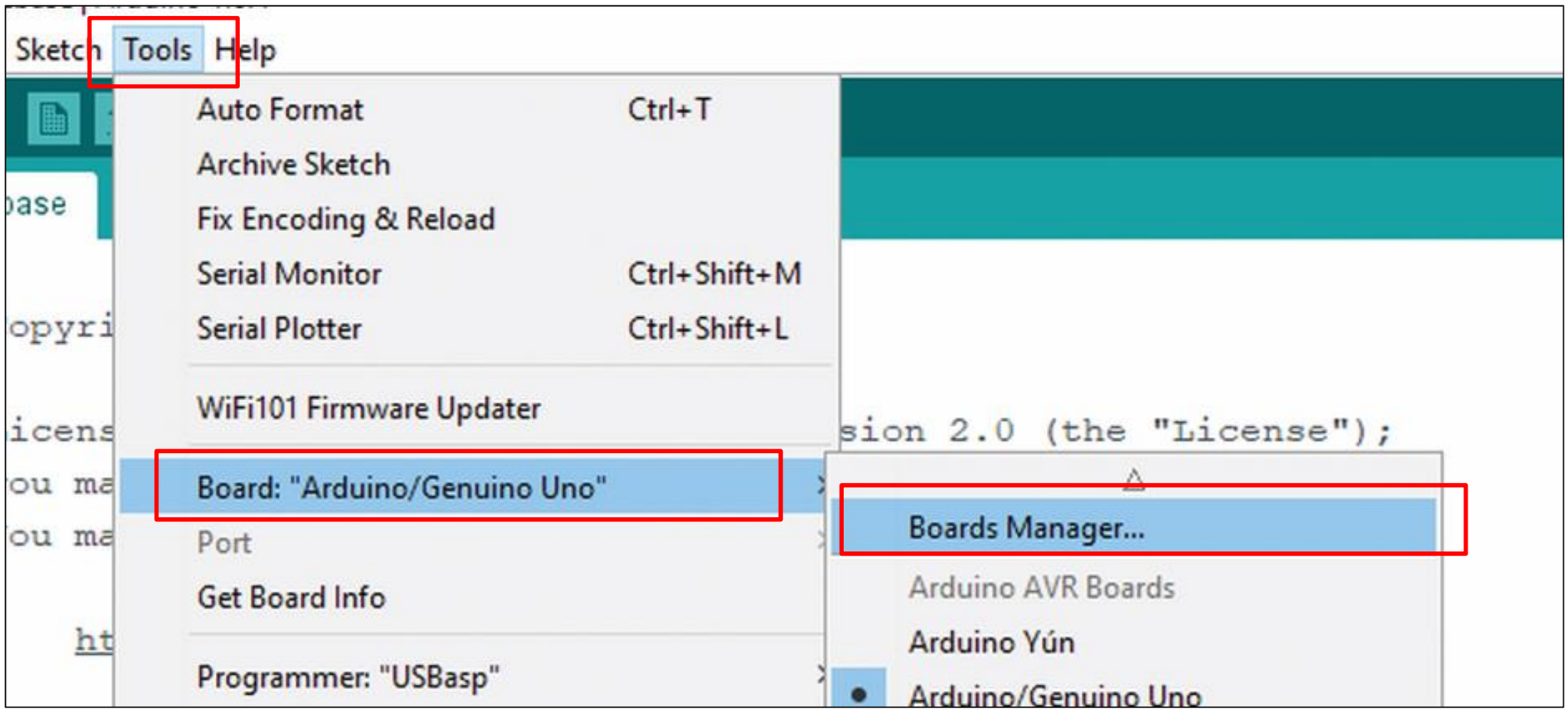
https://www.arduino.cc/en/Main/OldSoftwareReleases

NODEMCU ESP8266 CORE INSTALLATION

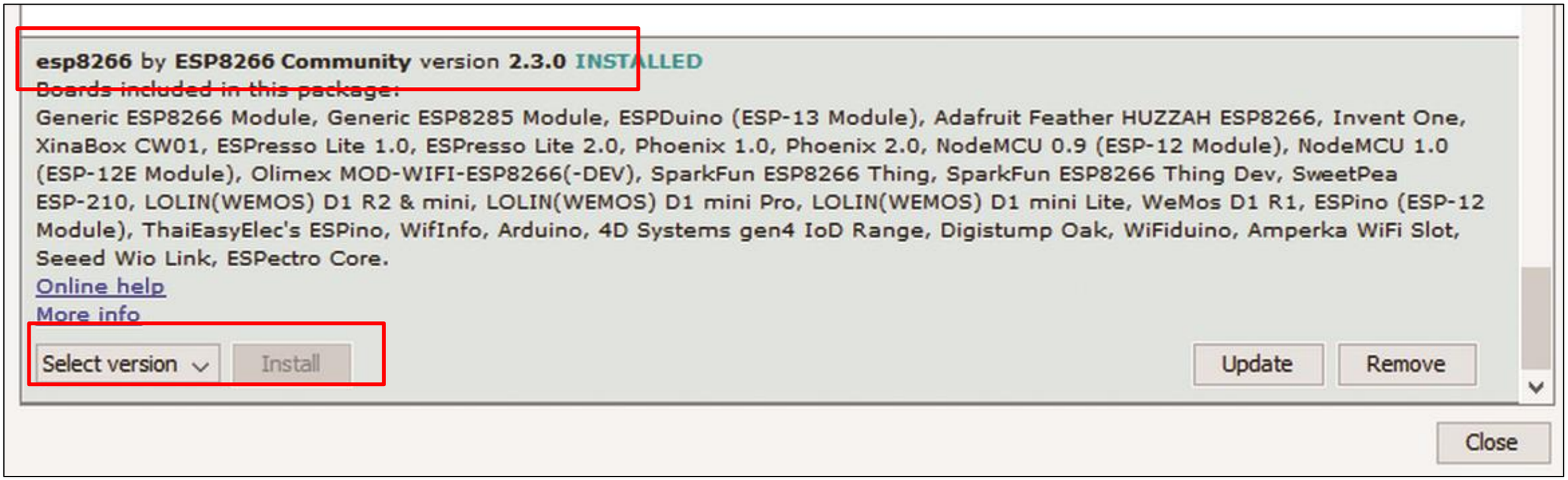




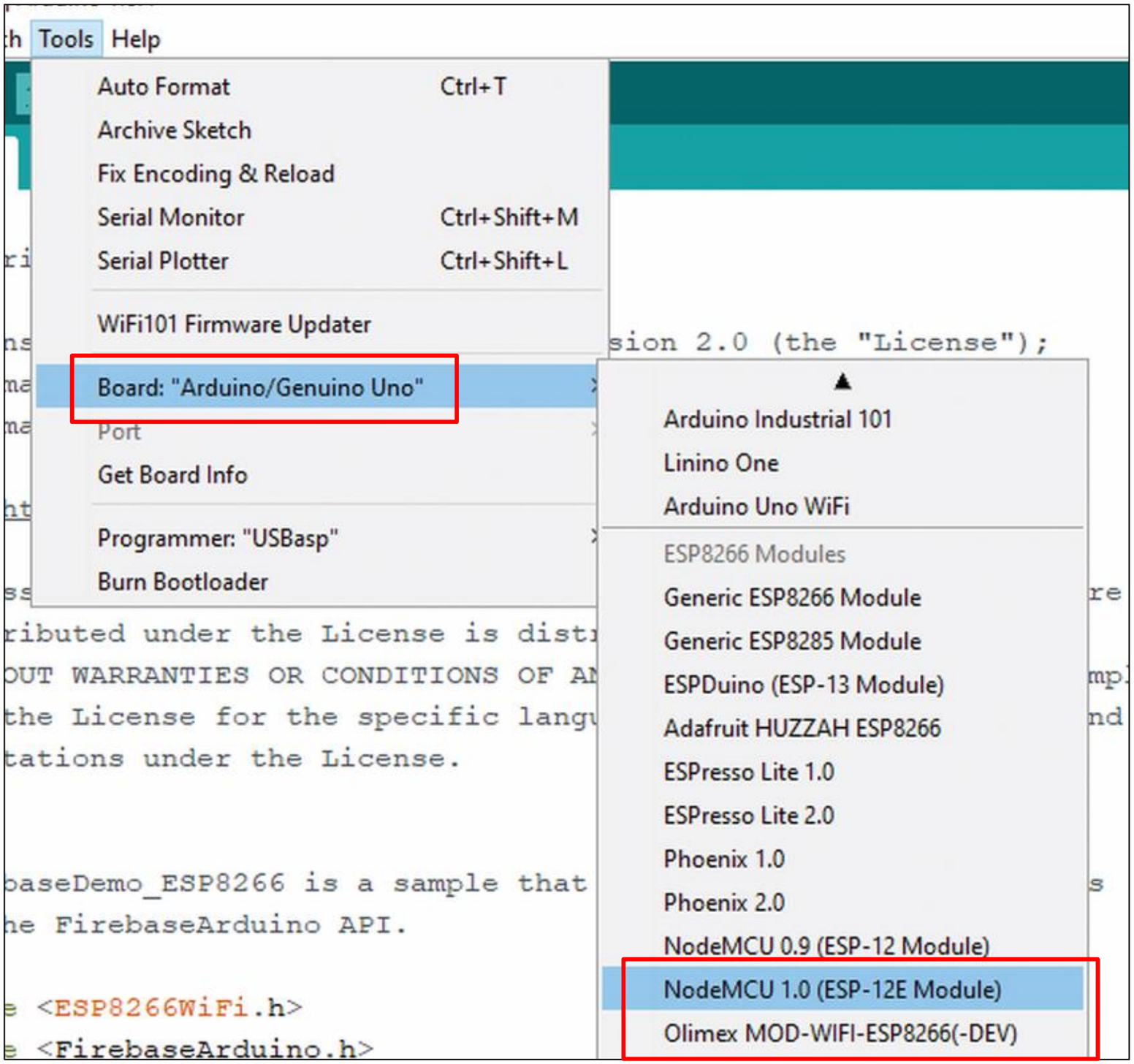
NODEMCU ESP8266 CORE INSTALLATION



NODEMCU ESP8266 CORE INSTALLATION



NODEMCU ESP8266 CORE INSTALLATION



UPLOADING SKETCH

Blink | Arduino 1.8.8

File Edit Sketch Tools Help

Blink §

```
/*
 ESP8266 Blink by Simon Peter
 Blink the blue LED on the ESP-01 module
 This example code is in the public domain
 The blue LED on the ESP-01 module is connected to GPIO1
 (which is also the TXD pin; so we cannot use Serial.print() at the same time)
*/

int LED = D2;                     //Connect an LED to pin D2 or GPIO4

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // Initialize the LED_BUILTIN pin as an output
  pinMode(LED, OUTPUT);             // led
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);   // Turn the built-in LED ON (Note that LOW is the voltage level
                                    // but actually the LED is on; this is because
                                    // it is active low on the ESP-01)
  delay(1000);                      // Wait for a second (1000=1second)
  digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off by making the voltage HIGH
  delay(2000);                      // Wait for two seconds (to demonstrate the active low built in LED)

  digitalWrite(LED, HIGH);          // Turn the LED connected to D2 ON
```

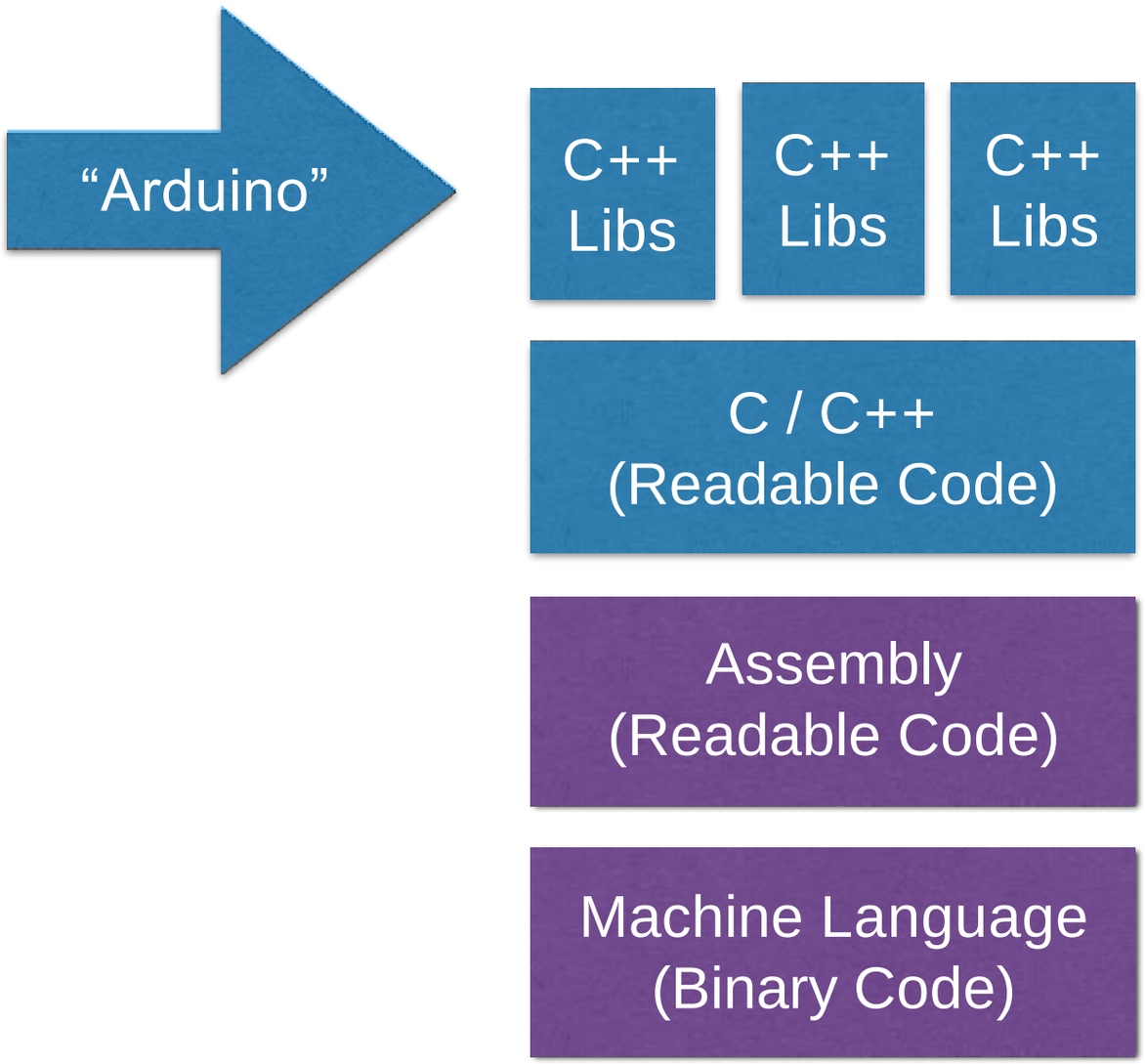
Compiling sketch...

12

NodeMCU 1.0 (ESP-12E Module), 80 MHz, 115200, 4M (3M SPIFFS) on COM8

Arduino Programming

Arduino Langauge



Arduino Langauge

"Arduino"

C++ Libs | C++ Libs | C++ Libs

C / C++
(Readable Code)

```
// the loop routine runs ove
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Assembly Code Example

```
in r16, SREG     ; store SREG value
cli    ; disable interrupts during timed seq
sbi EECR, EEMPE  ; start EEPROM write
sbi EECR, EEPE
out SREG, r16    ; restore SREG value (I-bit
```

Assembly
(Readable Code)

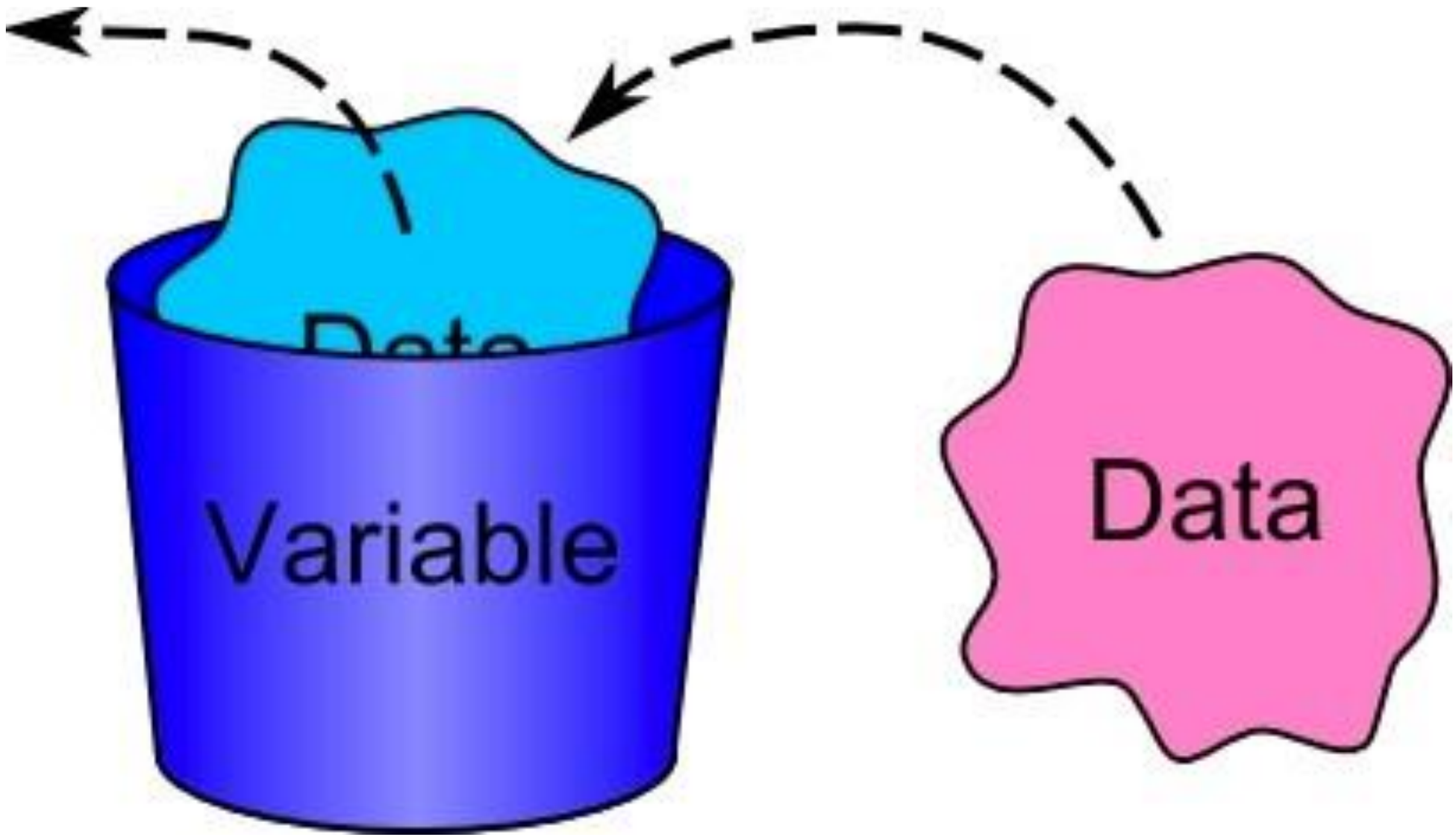
Machine Language
(Binary Code)

```
:100000000C9461000C947E000C947E000C947E0095
:100010000C947E000C947E000C947E000C947E0068
:100020000C947E000C947E000C947E000C947E0058
```

Arduino Syntax

BLINK

```
/*
 ESP8266 Blink by Simon Peter
 Blink the blue LED on the ESP-01 module
 This example code is in the public domain
 The blue LED on the ESP-01 module is connected to GPIO1
 (which is also the TXD pin; so we cannot use Serial.print() at the same time)
*/

int LED = D2;                         //Connect an LED to pin D2 or GPIO4

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);       // Initialize the LED_BUILTIN pin as an output
  pinMode(LED, OUTPUT);               // led
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);     // Turn the built-in LED ON (Note that LOW is the voltage level
                                      // but actually the LED is on; this is because
                                      // it is active low on the ESP-01)
  delay(1000);                        // Wait for a second (1000=1second)
  digitalWrite(LED_BUILTIN, HIGH);    // Turn the LED off by making the voltage HIGH
  delay(2000);                        // Wait for two seconds (to demonstrate the active low built in LED)

  digitalWrite(LED, HIGH);            // Turn the LED connected to D2 ON
  delay(1000);                        // Wait for a second (1000=1second)
  digitalWrite(LED, LOW);             // Turn the LED OFF by making the voltage LOW
  delay(2000);                        // Wait for two seconds (to demonstrate the active high LED)
}
```



Variables

BLINK

```
/*
 ESP8266 Blink by Simon Peter
 Blink the blue LED on the ESP-01 module
 This example code is in the public domain
 The blue LED on the ESP-01 module is connected to GPIO1
 (which is also the TX pin; so we cannot use Serial.print() at the same time)
*/

int LED = D2;                     //Connect an LED to pin D2 or GPIO4

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // Initialize the LED_BUILTIN pin as an output
  pinMode(LED, OUTPUT);             // led
}

// t
void
  di                                                              evel

  delay(1000);                      // Wait for a second (1000=1second)
  digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off by making the voltage HIGH
  delay(2000);                      // Wait for two seconds (to demonstrate the active low built in LED)

  digitalWrite(LED, HIGH);          // Turn the LED connected to D2 ON
  delay(1000);                      // Wait for a second (1000=1second)
  digitalWrite(LED, LOW);           // Turn the LED OFF by making the voltage LOW
  delay(2000);                      // Wait for two seconds (to demonstrate the active high LED)
}
```

Variable

A variable is like a bucket. You choose what types of stuff you want in the bucket and can change the contents of the bucket as often as you like.

When you declare a variable you are telling the program two things:

1. *the types of things you plan to put in the variable, and*
2. *the name of the bucket so you can refer to it later.*

BLINK

```
/*
 ESP8266 Blink by Simon Peter
 Blink the blue LED on the ESP-01 module
 This example code is in the public domain
 The blue LED on the ESP-01 mo
 (which is also the TXD pin; so we cannot use Serial.print() at the same time)
*/

int LED = D2;                     //Connect an LED to pin D2 or GPIO4

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);   // Initialize the LED_BUILTIN pin as an output
  pinMode(LED, OUTPUT);           // led
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);  // Turn the built-in LED O   ote that LOW is the voltage level
                                   // but actually the LED is on  this is because
                                   // it is active low on the ESP-01)
  delay(1000);                     // Wait for a second (1000=1second)
  digitalWrite(LED_BUILTIN, HIGH); // Turn the LED off by making the voltage HIGH
  delay(2000);                     // Wait for two seconds (to demonstrate the active low built in LED)

  digitalWrite(LED, HIGH);
  delay(1000);                     // Wait for a second (1000=1second)
  digitalWrite(LED, LOW);          // Turn the LED OFF by making the voltage LOW
  delay(2000);                     // Wait for two seconds (to demonstrate the active high LED)
}
```

Block comment

Single line comment

Comments

Comments are used to annotate code.

BLINK

```
/*
 ESP8266 Blink by Simon Peter
 Blink the blue LED on the ESP-01 module
 This example code is in the public domain
 The blue LED on the ESP-01 module is connected to GPIO1
 (which is also the TXD pin; so we cannot use Serial.print() at the same time)
*/

int LED = D2;                         //Connect an LED to pin D2 or GPIO4

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);       // Initialize the LED_BUILTIN pin as an output
  pinMode(LED, OUTPUT);               // led
}

// the loop function runs over and over ag
void loop() {
  digitalWrite(LED_BUILTIN, LOW);     // Turn the built-in LED ON (Note that LOW is the voltage level
                                      // but
                                      // it
  delay(1000);                        // Wai
  digitalWrite(LED_BUILTIN, HIGH);    // Tur
  delay(2000);                        // Wai                                  e low built in LED)

  digitalWrite(LED, HIGH);            // Turn the LED connected to D2 ON
  delay(1000);                        // Wait for a second (1000=1second)
  digitalWrite(LED, LOW);             // Turn the LED OFF by making the voltage LOW
  delay(2000);                        // Wait for two seconds (to demonstrate the active high LED)
}
```

Comments

Comments will be ignored by the compiler

Good Comment must be meaningful:
// Initialize the LED_BUILTIN pin as an output

Bad Comment:
// led

BLINK

```
int LED = D2;                                   //Connect an LED to pin D2 or GPI

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);                 // Initialize the LED_BUILTIN pin
  pinMode(LED, OUTPUT);                         // led
}

// the loop function runs over and over agai    forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);               //                ED ON (Note
                                                // b              ually the LED is on; th
                                                // it is          ctive low on the ESP-01
  delay(1000);                                  // Wait for a second (1000=1secon
  digitalWr                                                                  king the
  delay(200                                                                  to demon

  digitalWrite(LED, HIGH);                      // Turn the LED connected to D2 C
  delay(1000);                                  // Wait for a second (1000=1secon
  digitalWrite(LED, LOW);                       // Turn the LED OFF by making the
  delay(2000);                                  // Wait for two seconds (to demon
}
```

Semicolon

A semicolon needs to follow every statement written in the Arduino programming language. It signifies complete statement.

BLINK

```
/*
 ESP8266 Blink by Simon Peter
 Blink the blue LED on the ESP-01 module
 This example c
 The blue LED c
 (which is also
*/

int LED = D2;                    //Connect an LED to pin D2 of ESP8
```

The function, setup(), as the name implies, is used to set up the Arduino board.
Things you should know about setup() :

1. *setup() only runs once.*
2. *setup() needs to be the first function in your Arduino sketch.*
3. *setup() must have opening and closing curly braces.*

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // Initialize the LED_BUILTIN pin as an output
  pinMode(LED, OUTPUT);             // led
}
```

Functions

```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);   // Turn the built-in LED ON (Note that LOW is the voltage level
                                    // but actually the LED is on; this is because
                                    //                                  01)
  delay(1000);                      // Wait for a second (1000=1second)
  digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off by making the voltage HIGH
  delay(2000);                      // Wait for two seconds (to demonstrate the active low built in LED)

  digitalWrite(LED, HIGH);          // Turn the LED connected to D2 ON
  delay(1000);                      // Wait for a second (1000=1second)
  digitalWrite(LED, LOW);           // Turn the LED OFF by making the voltage LOW
  delay(2000);                      // Wait for two seconds (to demonstrate the active high LED)
}
```

The loop() function is where the body of your program will resides.

BLINK

```
/*
 ESP8266 Blink by Simon Peter
 Blink the blue LED on the ESP-01 module
 This example code is in the public domain
 The blue LED on the ESP-01 module is connected to GPIO1
 (which is also the TXD pin; so we cannot use Serial.print() at the same time)
*/

int LED = D2;                                          n D2 or GPIO4

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);      // Initialize the LED_BUILTIN pin as an output
  pinMode(LED, OUTPUT);              // led
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);    // Turn the built-in LED ON (Note that LOW is the voltage level
                                     // but actually the LED is on; this is because
                                     // it is active low on the ESP-01)
  delay(1000);                       // Wait for a second (1000=1second)
  digitalWrite(LED_BUILTIN, HIGH);   // Turn the LED off by making the voltage HIGH
  delay(2000);                       // Wait for two seconds (to demonstrate the active low built in LED)

  digitalWrite(LED, HIGH);           // Turn the LED connected to D2 ON
  delay(1000);                       // Wait for a second (1000=1second)
  digitalWrite(LED, LOW);            // Turn the LED OFF by making the voltage LOW
  delay(2000);                       // Wait for two seconds (to demonstrate the active high LED)
}
```

Curly braces are used to enclose further instructions carried out by a function.

Functions

BLINK

```
/*
 ESP8266 Blink by Simon Peter
 Blink the blue LED on the ESP-01 module
 This example code is in the public domain
 The blue LED on the ESP-01 module is connected to GPIO1
 (which is also the TXD pin; so we cannot use Serial.print() at the same time)
*/

int LED = D2;                           //Connect an LED to pin D2 or GPIO4

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);         // Initialize the LED_BUILTIN pin as an output
  pinMode(LED, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);    // Turn the built-in LED ON (Note that LOW is the voltage level
                                     // but actually the LED is on; this is because
                                     // it is active low on the ESP-01)
  delay(1000);                       // Wait for a second (1000=1second)
  digitalWrite(LED_BUILTIN, HIGH);   // Turn the LED off by making the voltage HIGH
  delay(2000);                       // Wait for two seconds (to demonstrate the active low built in LED)

  digitalWrite(LED, HIGH);           // Turn the LED connected to D2 ON
  delay(1000);                       // Wait for a second (1000=1second)
  digitalWrite(LED, LOW);            // Turn the LED OFF by making the voltage LOW
  delay(2000);                       // Wait for two seconds (to demonstrate the active high LED)
}
```

Instructions

BLINK

```
/*
 ESP8266 Blink by Simon Peter
 Blink the blue LED on the ESP-01 module
 This example code is in the public domain
 The blue LED on the ESP-01 module is connected to GPIO1
 (which is also the TXD pin; so we cannot use Serial.print() at the same time)
*/

int LED = D2;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // Initialize the LED_BUILTIN pin as an output
  pinMode(LED, OUTPUT);             // led
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, LOW);   // Turn the built-in LED ON (Note that LOW is the voltage level
                                    // but actually the LED is on; this is because
                                    // it is active low on the ESP-01)
  delay(1000);                      // Wait for a second (1000=1second)
  digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off by making the voltage HIGH
  delay(2000);                      // Wait for two seconds (to demonstrate the active low built in LED)

  digitalWrite(LED, HIGH);          // Turn the LED connected to D2 ON
  delay(1000);                      // Wait for a second (1000=1second)
  digitalWrite(LED, LOW);           // Turn the LED OFF by making the voltage LOW
  delay(2000);                      // Wait for two seconds (to demonstrate the active high LED)
}
```

Function name pinMode() appear orange as it is a built-in function of Arduino IDE

Function Call

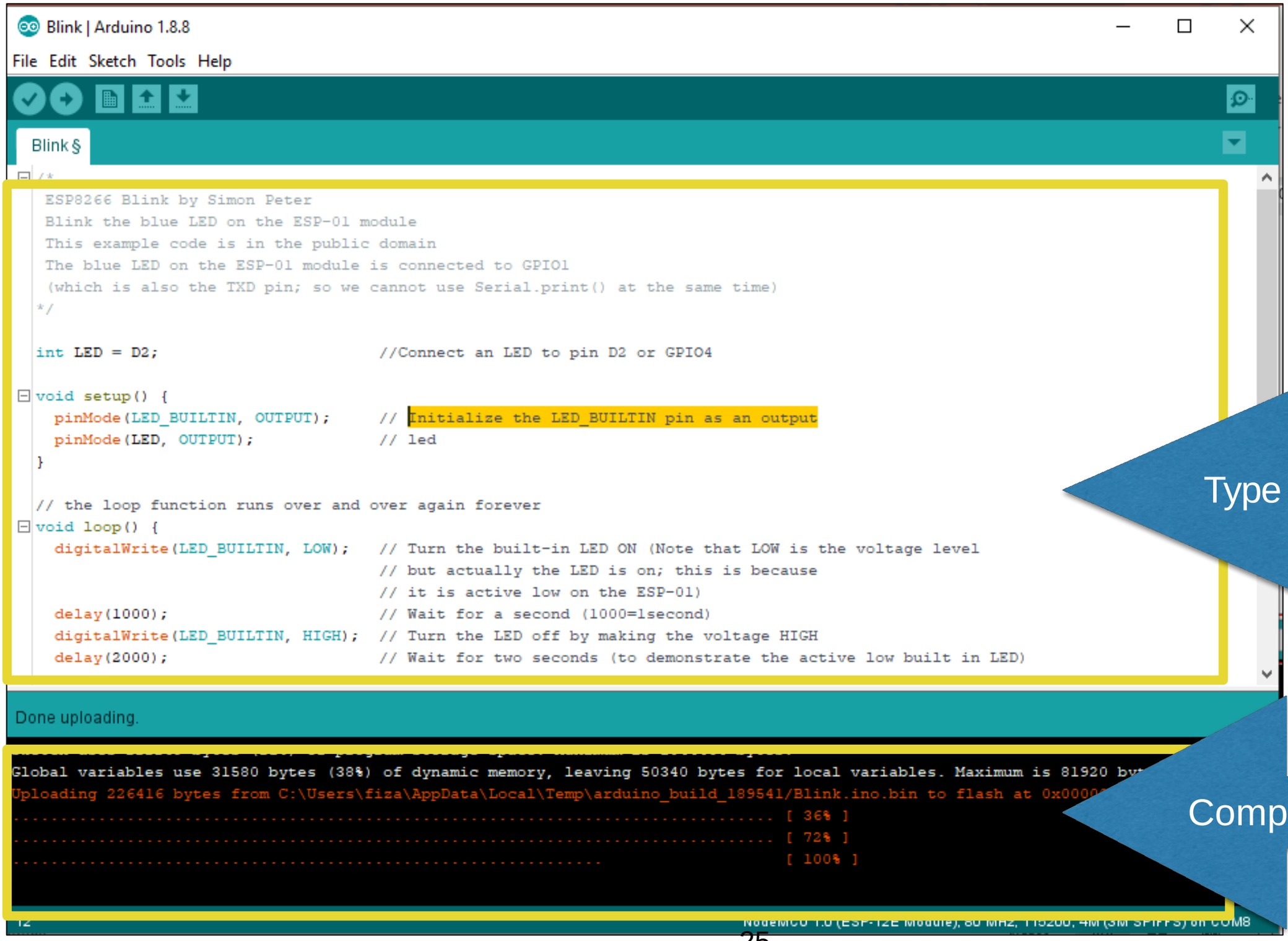
BLINK

```
/*
 ESP8266 Blink by Simon Peter
 Blink the blue LED on the ESP-01 module
 This example code is in the public domain
 The blue LED on the ESP-01 module is connected to GPIO1
 (which is also the TXD pin; s
*/

int LED = D2;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // Initialize the LED_BUILTIN pin as an output
  pinMode(LED, OUTPUT);             // led
}

// the loop functi      over and over again forever
void loop() {
  digitalWrite      OW);    // Turn the built-in LED ON (Note that LOW is the voltage level
                            // but actually the LED is on; this is because
                            // it is active low on the ESP-01)
  delay(1000);              // Wait for a second (1000=1second)
  digitalWrite(LED_     IN, HIGH);  // Turn the LED off by making the voltage HIGH
  delay(2000);              // Wait for two seconds (to demonstrate the active low built in LED)

  digitalWrite(LED, HIGH);          // Turn the LED connected to D2 ON
  delay(1000);              // Wait for a second (1000=1second)
  digitalWrite(LED, LOW);           // Turn the LED OFF by making the voltage LOW
  delay(2000);              // Wait for two seconds (to demonstrate the active high LED)
}
```

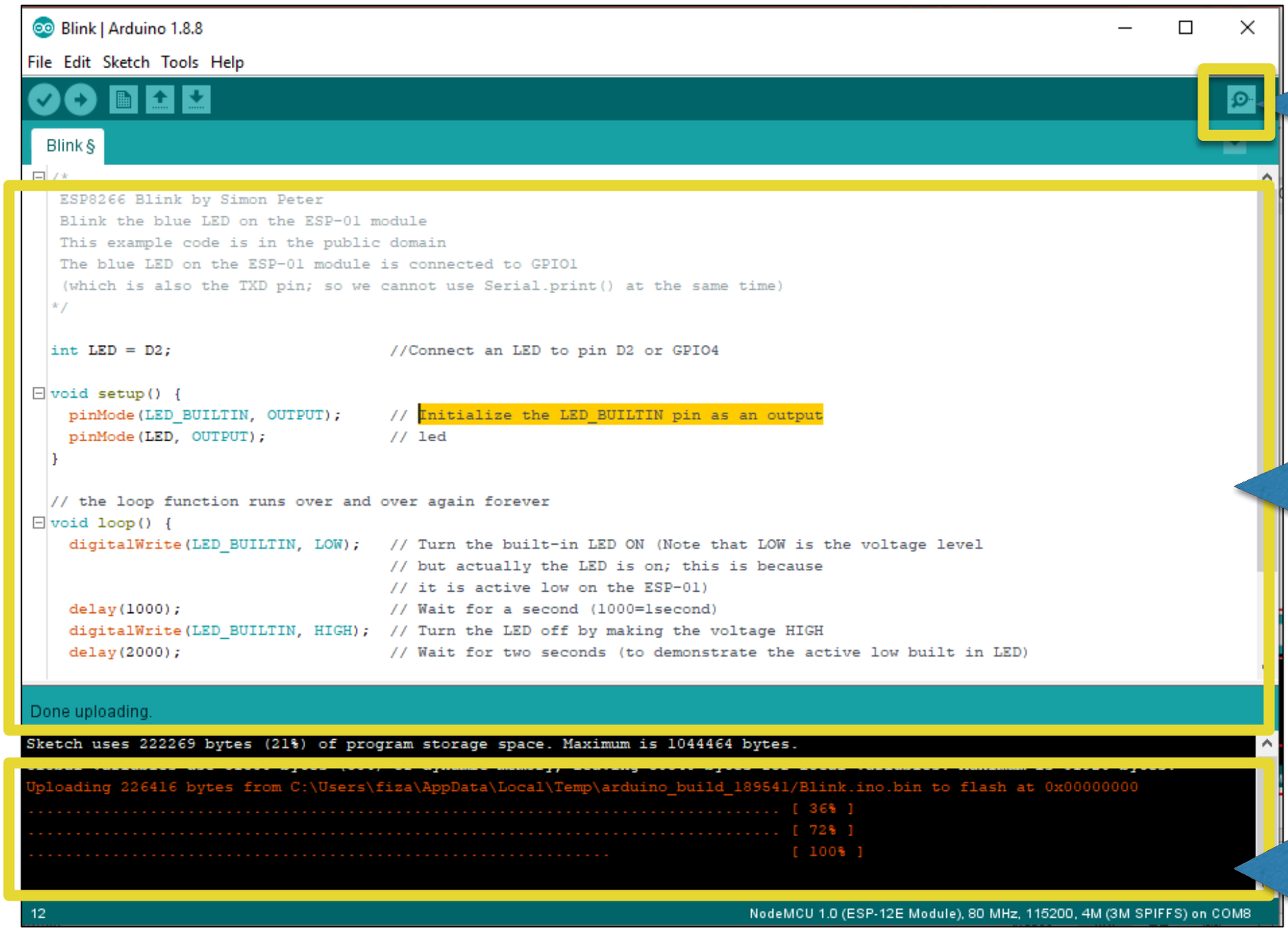
Notice that the word OUTPUT is green-blue. There are certain keywords in Arduino that are used frequently and the color helps identify them. The IDE turns them green-blue automatically.

Arguments

Arduino IDE


Blink | Arduino 1.8.8
File Edit Sketch Tools Help
Blink §
/*
ESP8266 Blink by Simon Peter
Blink the blue LED on the ESP-01 module
This example code is in the public domain
The blue LED on the ESP-01 module is connected to GPIO1
(which is also the TXD pin; so we cannot use Serial.print() at the same time)
*/
int LED = D2; //Connect an LED to pin D2 or GPIO4
void setup() {
pinMode(LED_BUILTIN, OUTPUT); // Initialize the LED_BUILTIN pin as an output
pinMode(LED, OUTPUT); // led
}
// the loop function runs over and over again forever
void loop() {
digitalWrite(LED_BUILTIN, LOW); // Turn the built-in LED ON (Note that LOW is the voltage level
// but actually the LED is on; this is because
// it is active low on the ESP-01)
delay(1000); // Wait for a second (1000=1second)
digitalWrite(LED_BUILTIN, HIGH); // Turn the LED off by making the voltage HIGH
delay(2000); // Wait for two seconds (to demonstrate the active low built in LED)
Type Stuff Here
Done uploading.
Global variables use 31580 bytes (38%) of dynamic memory, leaving 50340 bytes for local variables. Maximum is 81920 by
Uploading 226416 bytes from C:\Users\fiza\AppData\Local\Temp\arduino_build_189541/Blink.ino.bin to flash at 0x0000
[36%]
[72%]
[100%]
Compiler Output

Arduino IDE

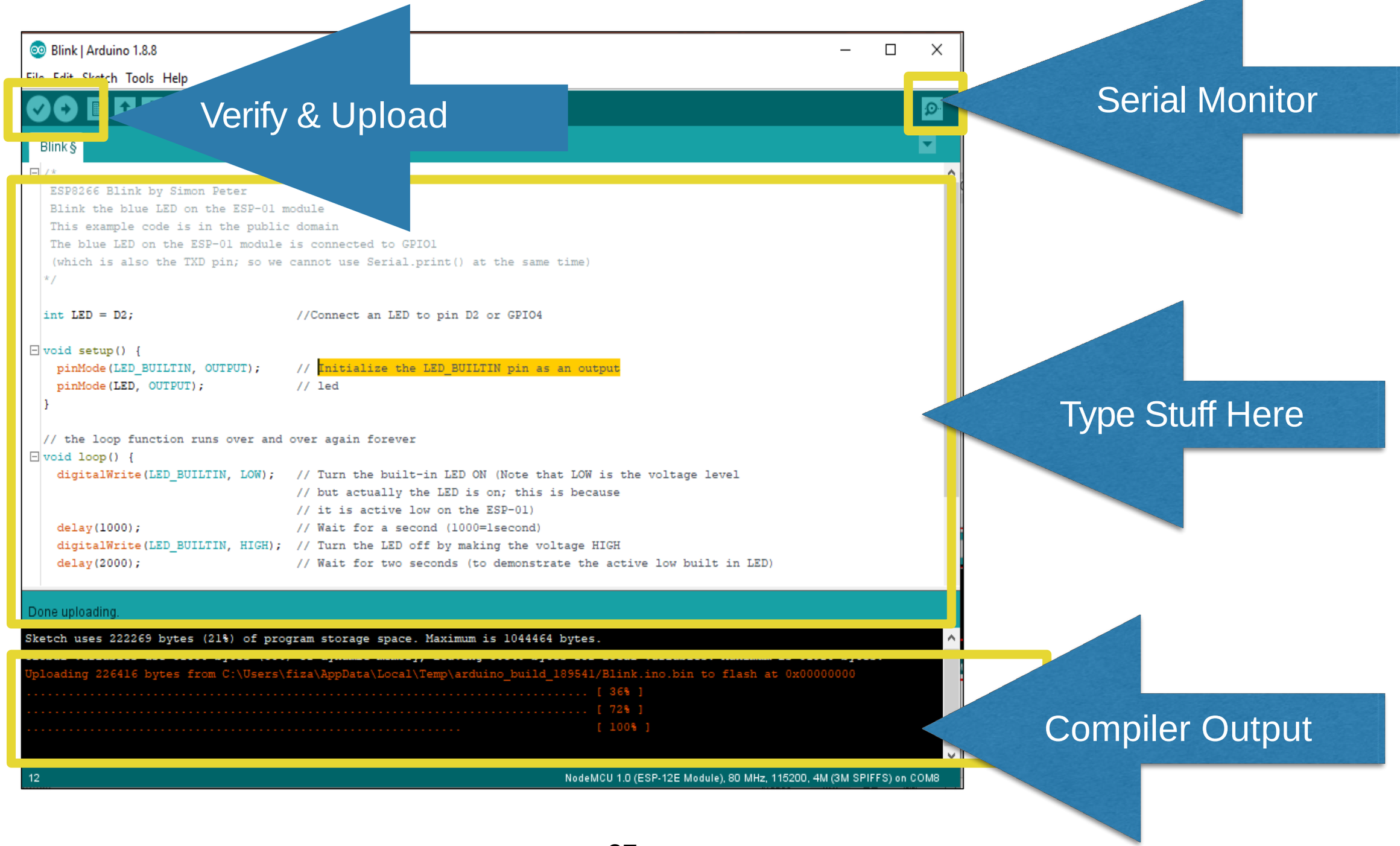


Serial Monitor

Type Stuff Here

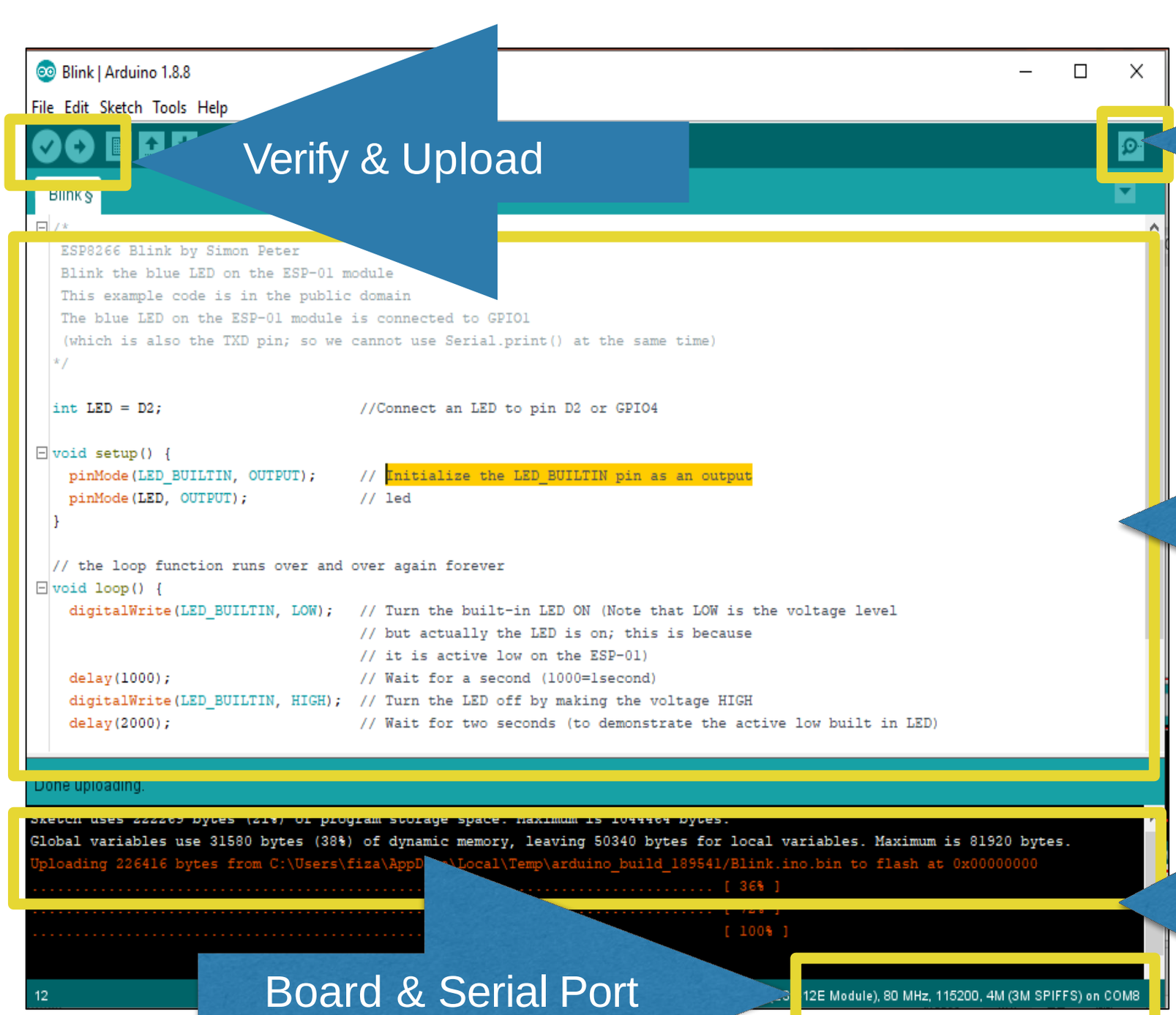
Compiler Output

Arduino IDE



27

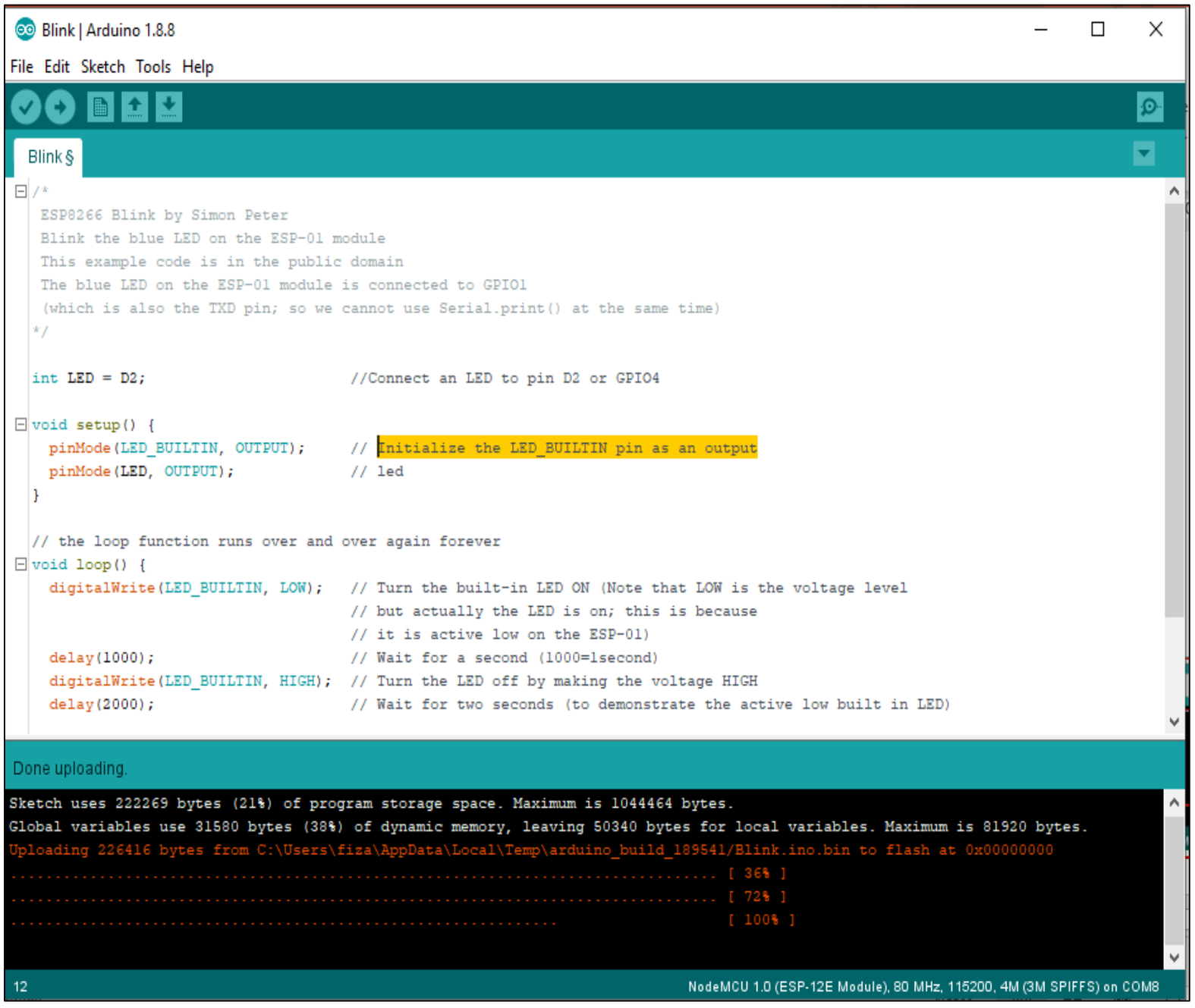
Arduino IDE


Blink | Arduino 1.8.8
File Edit Sketch Tools Help
Blink §
/*
ESP8266 Blink by Simon Peter
Blink the blue LED on the ESP-01 module
This example code is in the public domain
The blue LED on the ESP-01 module is connected to GPIO1
(which is also the TXD pin; so we cannot use Serial.print() at the same time)
*/
int LED = D2; //Connect an LED to pin D2 or GPIO4
void setup() {
pinMode(LED_BUILTIN, OUTPUT); // Initialize the LED_BUILTIN pin as an output
pinMode(LED, OUTPUT); // led
}
// the loop function runs over and over again forever
void loop() {
digitalWrite(LED_BUILTIN, LOW); // Turn the built-in LED ON (Note that LOW is the voltage level
// but actually the LED is on; this is because
// it is active low on the ESP-01)
delay(1000); // Wait for a second (1000=1second)
digitalWrite(LED_BUILTIN, HIGH); // Turn the LED off by making the voltage HIGH
delay(2000); // Wait for two seconds (to demonstrate the active low built in LED)
Done uploading.
Global variables use 31580 bytes (38%) of dynamic memory, leaving 50340 bytes for local variables. Maximum is 81920 bytes.
Uploading 226416 bytes from C:\Users\fiza\AppD \Local\Temp\arduino_build_189541/Blink.ino.bin to flash at 0x00000000
[36%]
[100%]
12
12E Module), 80 MHz, 115200, 4M (3M SPIFFS) on COM8
Verify & Upload
Serial Monitor
Type Stuff Here
Compiler Output
Board & Serial Port

Blink Exercise

Load the Blink Example and program it to your board

Change the values of delay() to see how it affects the behavior



Check the correct board and serial port are selected in the tools menu!

Hello World

Serial Example

Serial objects

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

```
void loop() {
  Serial.println("Hello World!");
  delay(2000);
}
```

Serial objects

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

void loop()
  Serial.println("Hello World!");
  delay(2000);
}
```

Enables Serial

Baud Rate

Serial objects

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

void loop() {
  Serial.println("Hello World!");
  delay(2000);
}
```

Print
and Println

Variables
Strings
Control Characters

Serial objects

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

void loop() {
  Serial.println("Hello World!");
  delay(2000);
}
```

Print
and Println

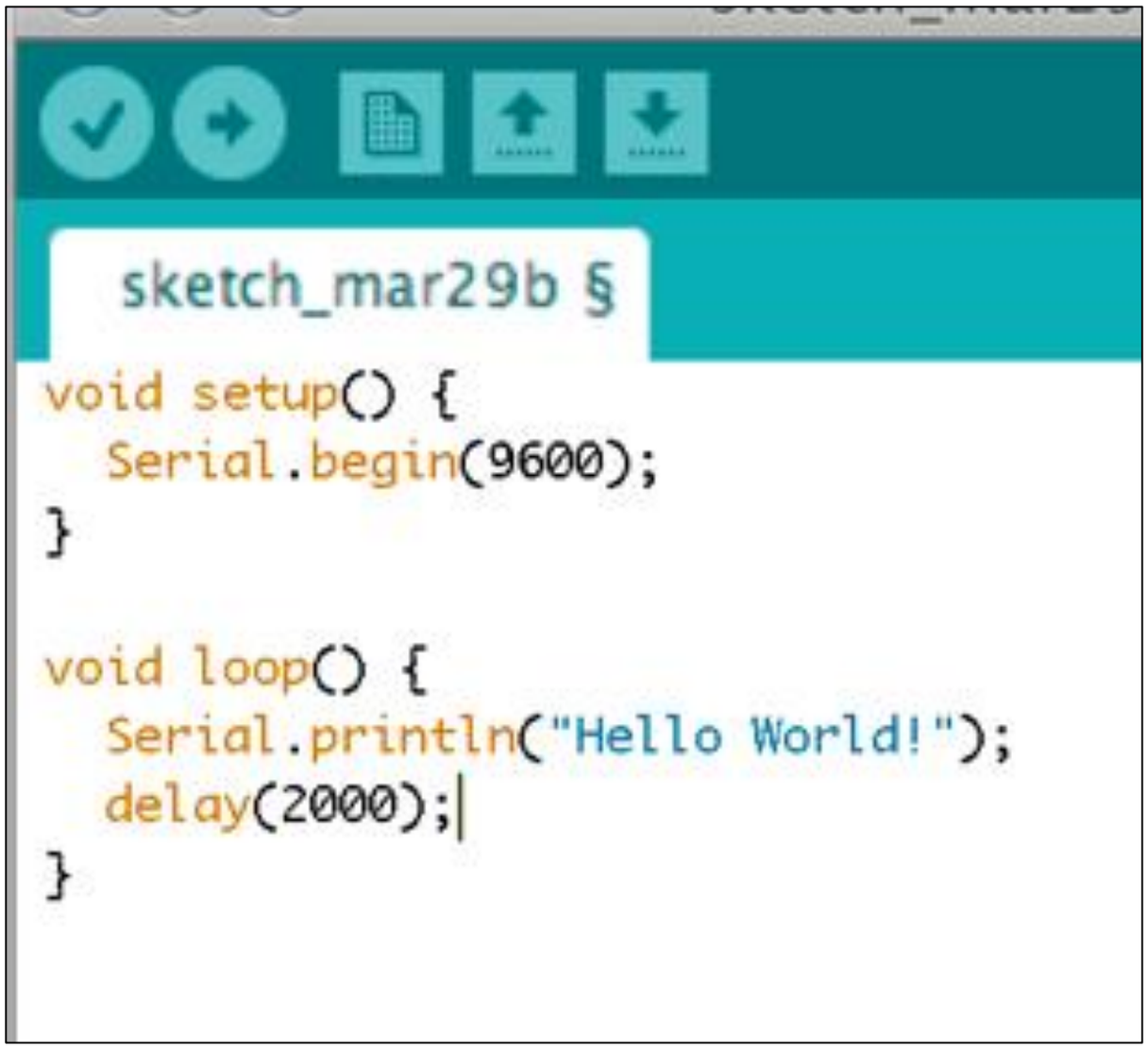
Variables
Strings
Control Characters

NOTE: Strings and Variables Can't be used on the same line

Hello World (Serial)

- Load up the serial code to the right

- Exercise:
  - Change the 2000 in delay into a variable.
  - Print value of variable on same line as “Hello World”



How Much **Memory** is in your Arduino?

Variable Types

	Bits	Unsigned Range	Signed Range
byte	8	0 to 255	N/A
char	8	0 to 255 ‘A’..’b’..’X’	N/A
int	16	0 to 65535	-32,767 to 32,766
long	32	0 to 4,294,967,295	-2,147,483,648 to 2,147,483,647
float	32	±3.4028235E+38	n/a
double	32	n/a	n/a

Variable Do and Don’t

- DO Use Descriptive Names
  - “BlueLED”, “ActivityIndicator”
- DON’T Use Bad Names
  - “Integer”, “Pin13”
- DO Stick to a naming convention
  - Variables are Case Sensitive!
- DON’T use same name for Local and Global Variables

Variable Scope

```
int LEDpin = 13;
int ButtonPin = 2;

void setup() {
  pinMode(LEDpin, OUTPUT);
  pinMode(ButtonPin, INPUT);
}

void loop() {
  int buttonValue = digitalRead(ButtonPin);
  digitalWrite(LEDpin, buttonValue);
}
```

Variable Scope

```
int LEDpin = 13;
int ButtonPin = 2;

void setup() {
  pinMode(LEDpin, OUTPUT);
  pinMode(ButtonPin, INPUT);
}

void loop() {
  int buttonValue = digitalRead(ButtonPin);
  digitalWrite(LEDpin, buttonValue);
}
```

Variable Scope

```
int LEDpin = 13;
int ButtonPin = 2;

void setup() {
  pinMode(LEDpin, OUTPUT);
  pinMode(ButtonPin, INPUT);
}

void loop() {
  int buttonValue = digitalRead(ButtonPin);
  digitalWrite(LEDpin, buttonValue);
}
```

Global

Local to loop()

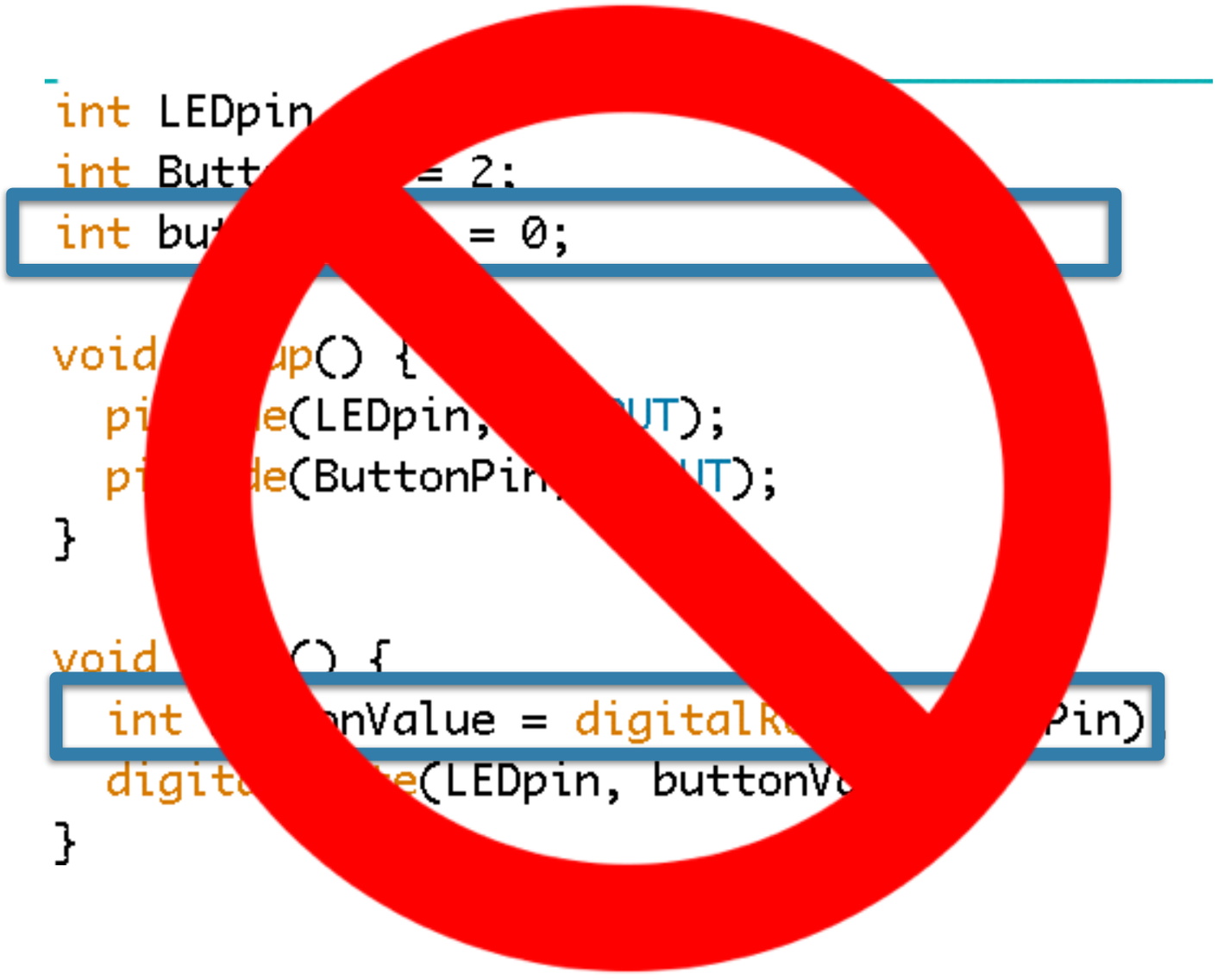
Variable Don’t!

```
int LEDpin = 13;
int ButtonPin = 2;
int buttonValue = 0;

void setup() {
  pinMode(LEDpin, OUTPUT);
  pinMode(ButtonPin, INPUT);
}

void loop() {
  int buttonValue = digitalRead(ButtonPin);
  digitalWrite(LEDpin, buttonValue);
}
```

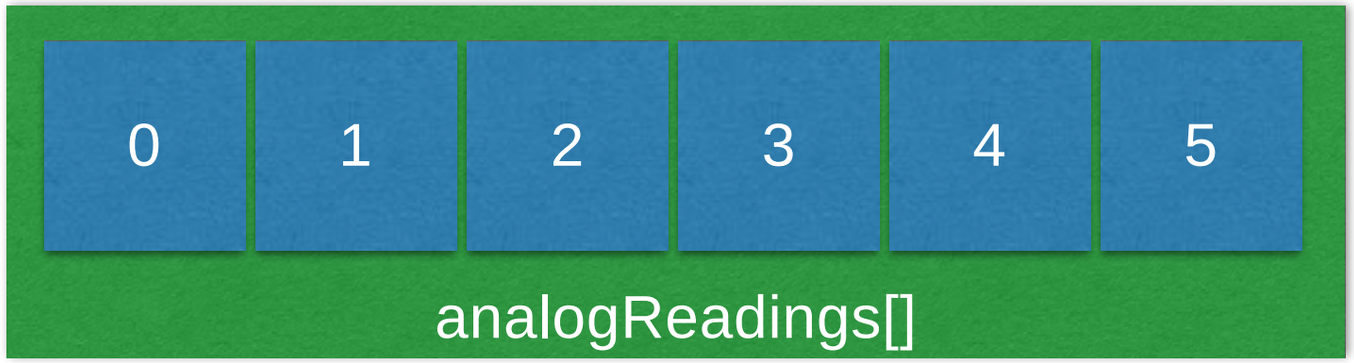
Variable Don’t!

```
int LEDpin = 13;
int ButtonPin = 2;
int buttonValue = 0;

void setup() {
  pinMode(LEDpin, OUTPUT);
  pinMode(ButtonPin, INPUT);
}

void loop() {
  int buttonValue = digitalRead(ButtonPin);
  digitalWrite(LEDpin, buttonValue);
}
```

Variable Don’t!

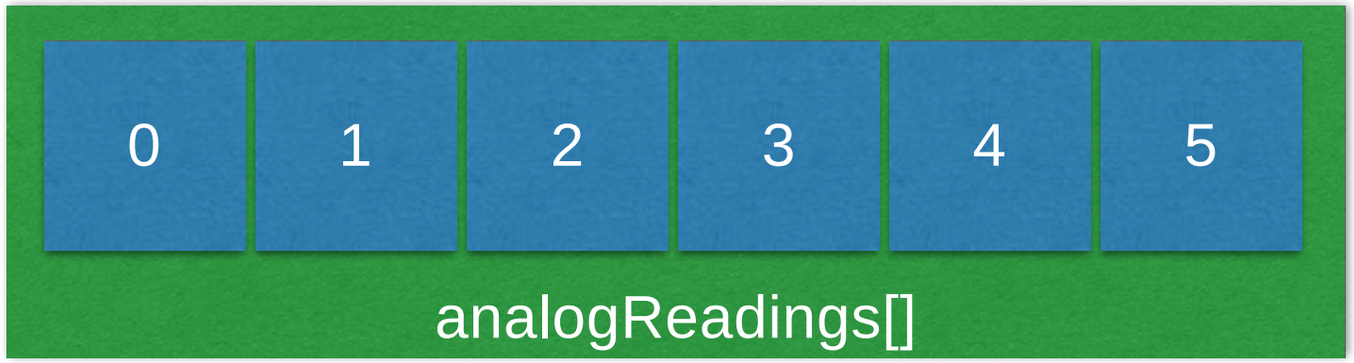


Arrays




```
int analogReadings[6];
anlaogReadings[0] = anlaogRead(A0);
anlaogReadings[1] = anlaogRead(A1);
anlaogReadings[2] = anlaogRead(A2);
anlaogReadings[3] = anlaogRead(A3);
anlaogReadings[4] = anlaogRead(A4);
anlaogReadings[5] = anlaogRead(A5);
```

Arrays

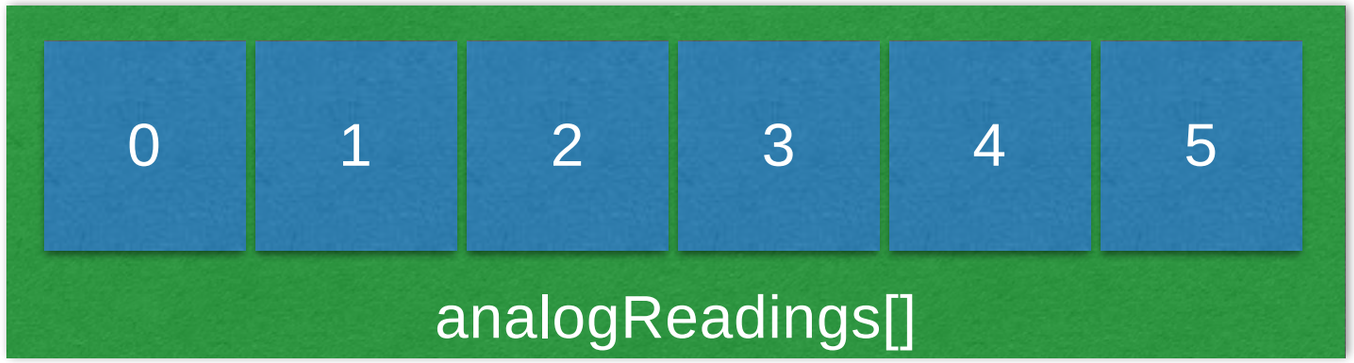



```
int analogReadings[6];
anlaogReadings[0] = anlaogRead(A0);
anlaogReadings[1] = anlaogRead(A1);
anlaogReadings[2] = anlaogRead(A2);
anlaogReadings[3] = anlaogRead(A3);
anlaogReadings[4] = anlaogRead(A4);
anlaogReadings[5] = anlaogRead(A5);
```

Size

Elements

Arrays



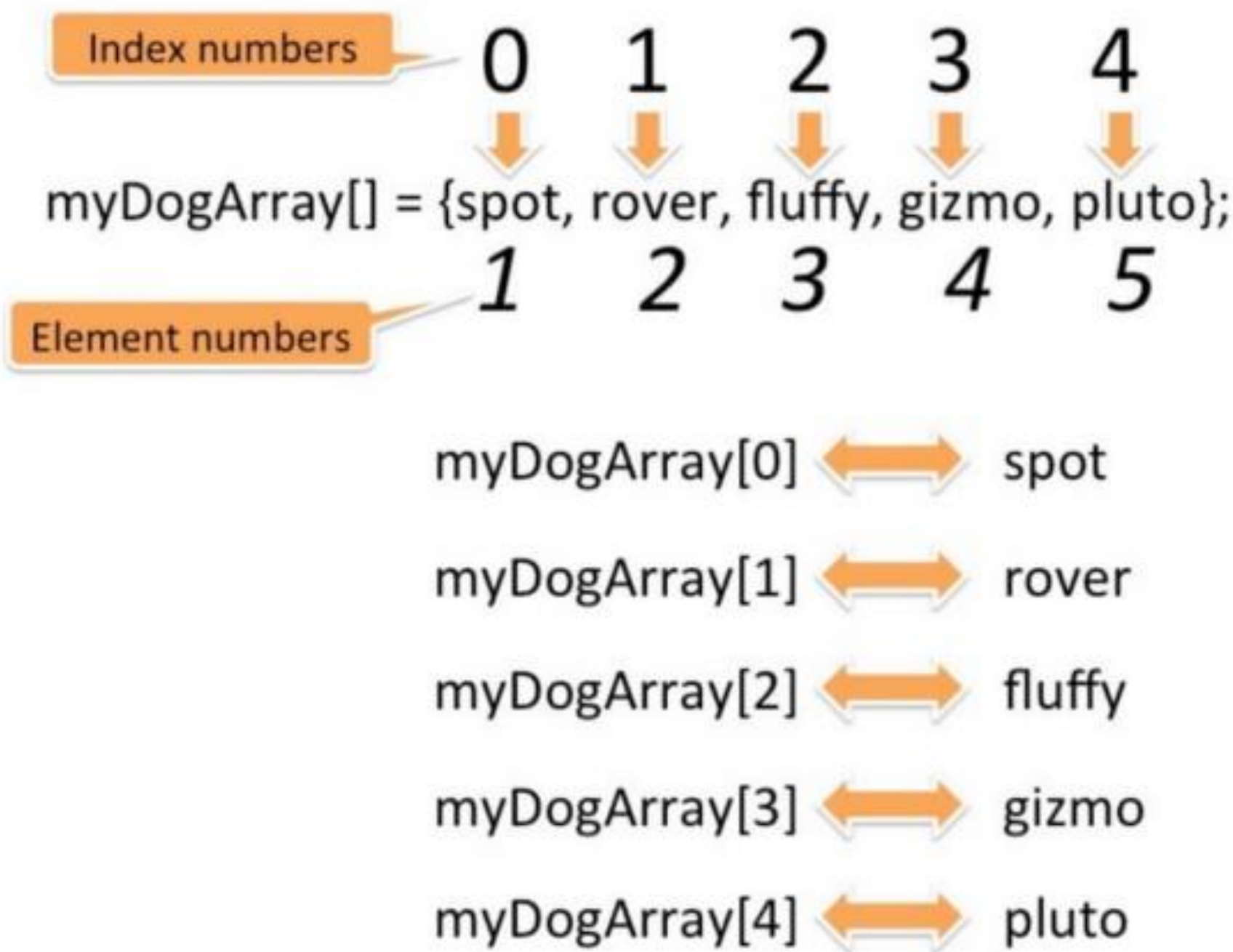

```
int analogReadings[6];
anlaogReadings[0] = anlaogRead(A0);
anlaogReadings[1] = anlaogRead(A1);
anlaogReadings[2] = anlaogRead(A2);
anlaogReadings[3] = anlaogRead(A3);
anlaogReadings[4] = anlaogRead(A4);
anlaogReadings[5] = anlaogRead(A5);
```

Size

Elements

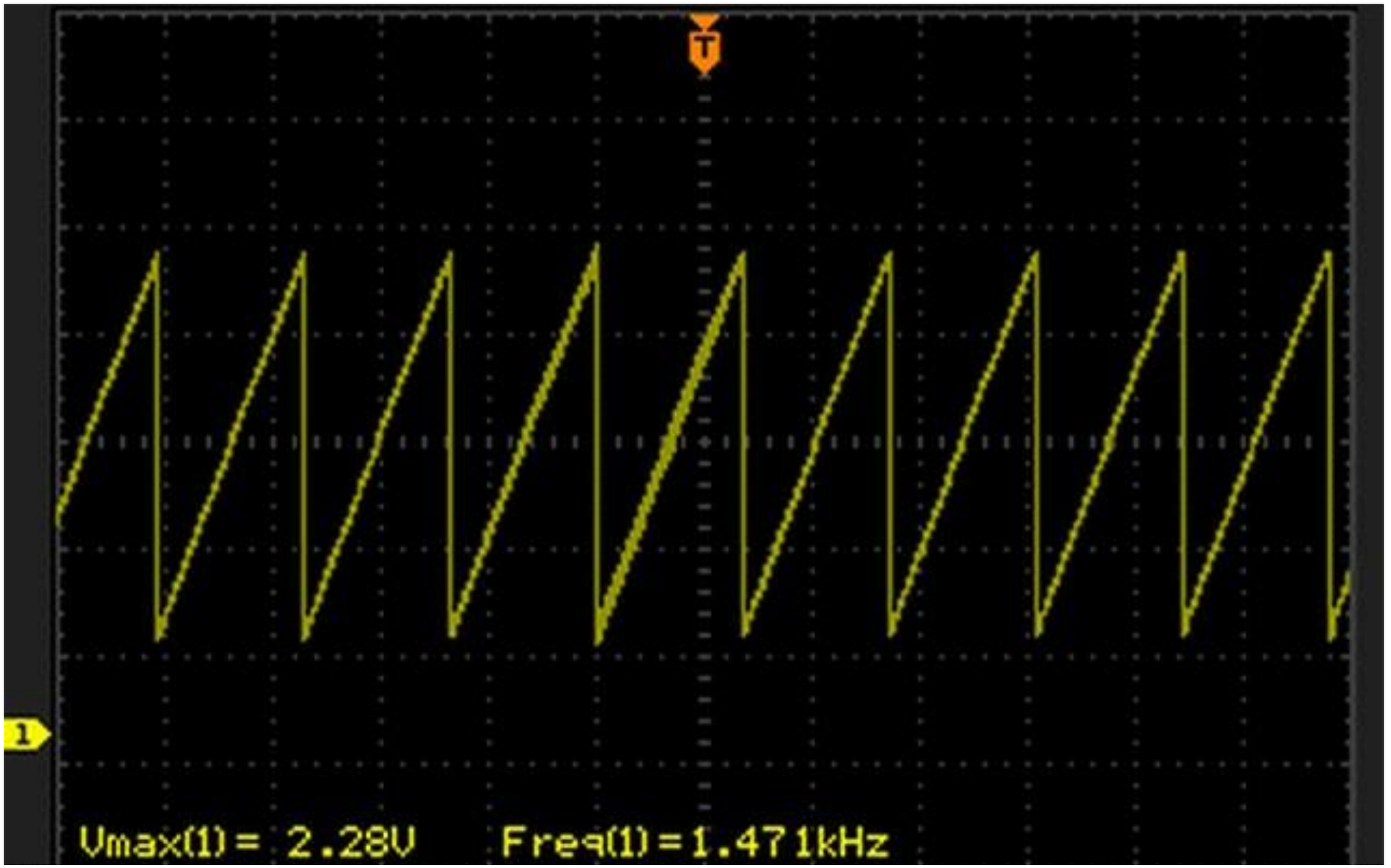
arrays are
0-index based.
So last element
is always
“1 less”
than the size!

Arrays


Index numbers
0 1 2 3 4
myDogArray[] = {spot, rover, fluffy, gizmo, pluto};
1 2 3 4 5
Element numbers
myDogArray[0] spot
myDogArray[1] rover
myDogArray[2] fluffy
myDogArray[3] gizmo
myDogArray[4] pluto

Array Exercise

```
int timer = 100; // The higher the number, the slower the timing.
int ledPins[] = {
 D2, D5, D8 }; // an array of pin numbers to which LEDs are attached
int pinCount = 3; // the number of pins (i.e. the length of the array)
void setup() {
 // the array elements are numbered from 0 to (pinCount - 1).
 // use a for loop to initialize each pin as an output:
 for (int thisPin = 0; thisPin < pinCount; thisPin++) {
 pinMode(ledPins[thisPin], OUTPUT);
 }
}
void loop() {
 // loop from the lowest pin to the highest:
 for (int thisPin = 0; thisPin < pinCount; thisPin++) {
 // turn the pin on:
 digitalWrite(ledPins[thisPin], HIGH);
 delay(timer);
 // turn the pin off:
 digitalWrite(ledPins[thisPin], LOW);
 }
 // loop from the highest pin to the lowest:
 for (int thisPin = pinCount - 1; thisPin >= 0; thisPin--) {
 // turn the pin on:
 digitalWrite(ledPins[thisPin], HIGH);
 delay(timer);
 // turn the pin off:
 digitalWrite(ledPins[thisPin], LOW);
 }
}
```



Pin Functions

pinMode()

	Analog (A0..A5)	Digital (0..13)
INPUT	Digital Input, Pull-Up **Off**	Digital Input, Pull-Up **Off**
INPUT_PULLUP	Digital Input, Pull-Up **On**	Digital Input, Pull-Up **On**
OUTPUT	Digital Output	Digital Output

Analog Pins can be used as Digital Pins
pinMode(INPUT, Ax) isn’t necessary for analogRead()

digitalRead() & digitalWrite()

```
int ButtonPin = 4;
int ButtonValue;
ButtonValue = digitalRead(ButtonPin);
```

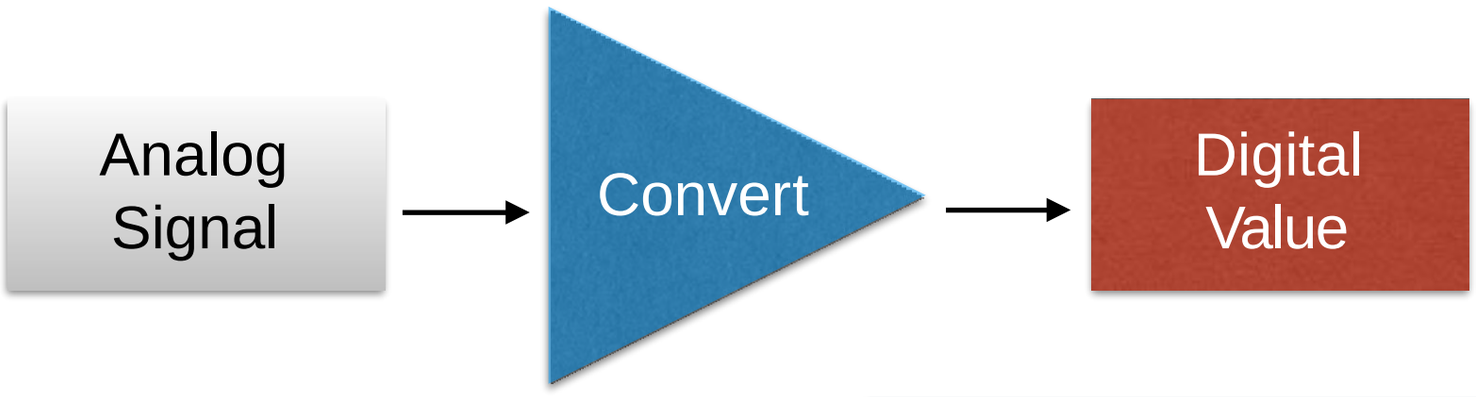
```
int LEDPin = 7;
digitalWrite(LEDPin, HIGH);
digitalWrite(LEDPin, LOW);
```

Input / Output

IO_Exercise

```
int LED = D2;
int ButtonPin = D0;


void setup() {
  // put your setup code here, to run once:
  pinMode(LED, OUTPUT);
  pinMode(ButtonPin,INPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
  int buttonValue = digitalRead(ButtonPin);
  digitalWrite(LED, buttonValue);

}
```

analogRead()



Voltage

0v to 5v

10-bits, stored as 8-bits

Values 0 to 1023

$$\frac{5 \text{ volts}}{1023 \text{ Steps}} = 4.887\text{mV per Step}$$

Calling analogRead() on an Analog Pin, automatically converts to Input

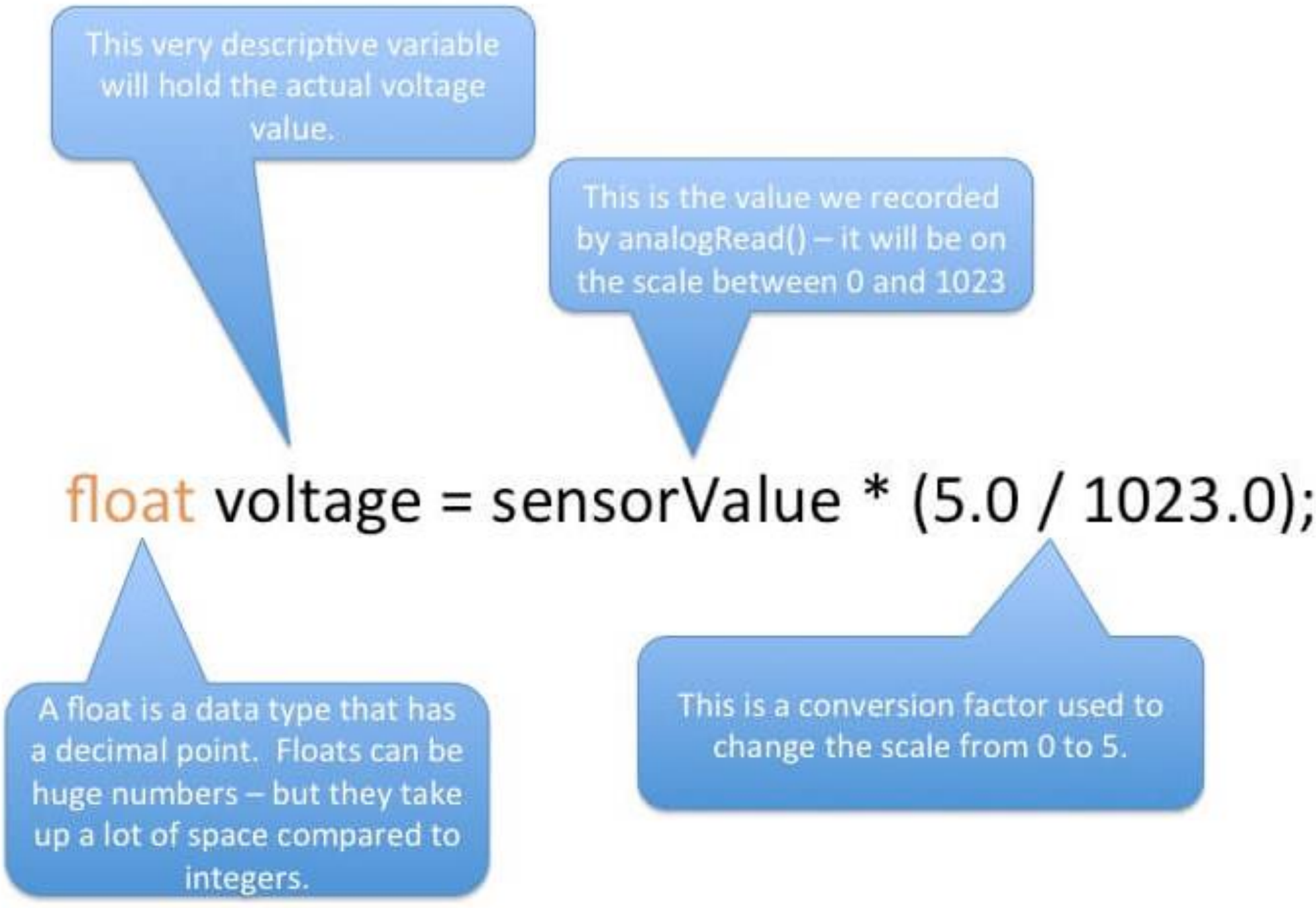
AnalogRead Exercise

```
int LED = D2;

void setup() {
  Serial.begin(9600);
  // put your setup code here, to run once:
  pinMode(LED, OUTPUT);

}
```

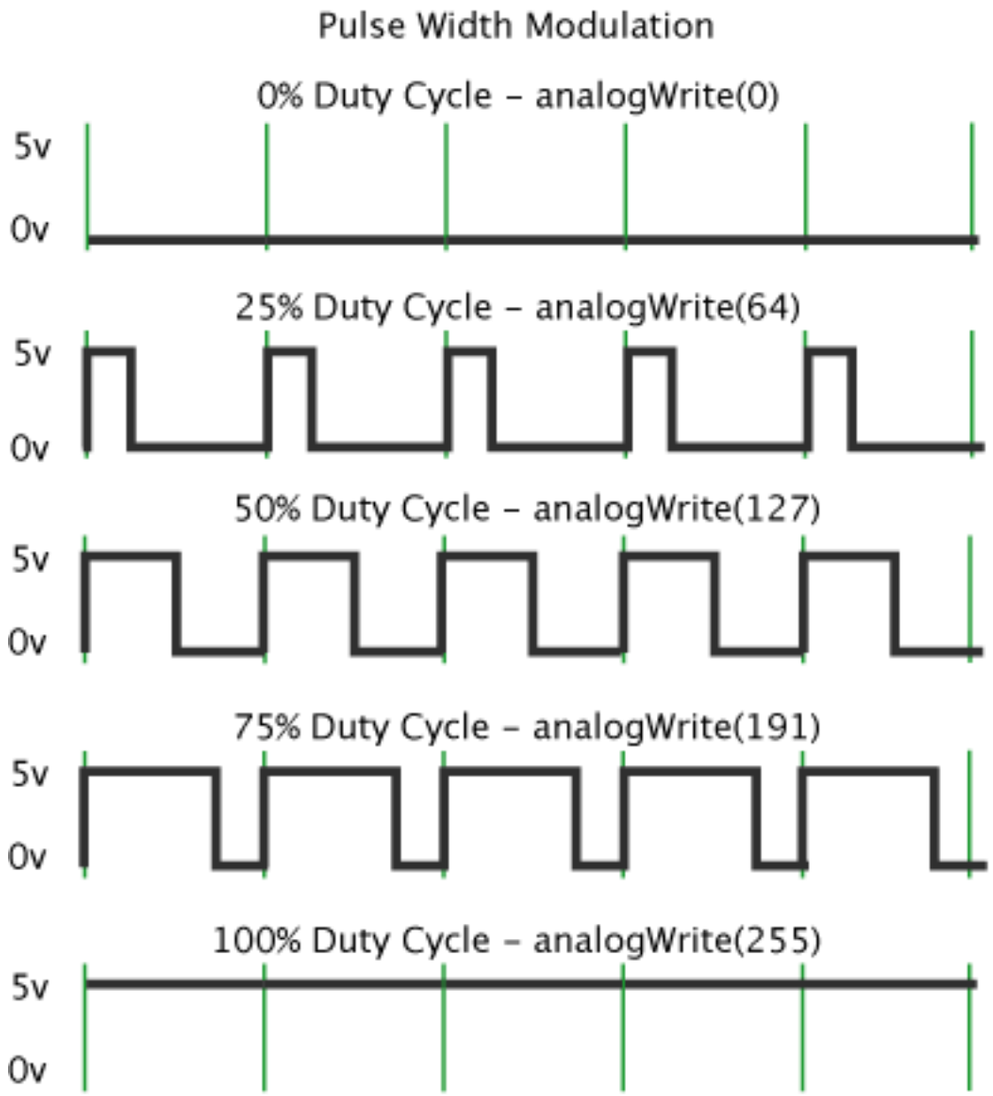
Remember to open the Serial Monitor!

```
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  Serial.println("Sensor value: ");
  Serial.println(sensorValue);
  delay(1000);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage
  float voltage = sensorValue * (5.0 / 1024.0);
  // print out the value you read:
  Serial.println("voltage: ");
  Serial.println(voltage);
  delay(1000);
}
```

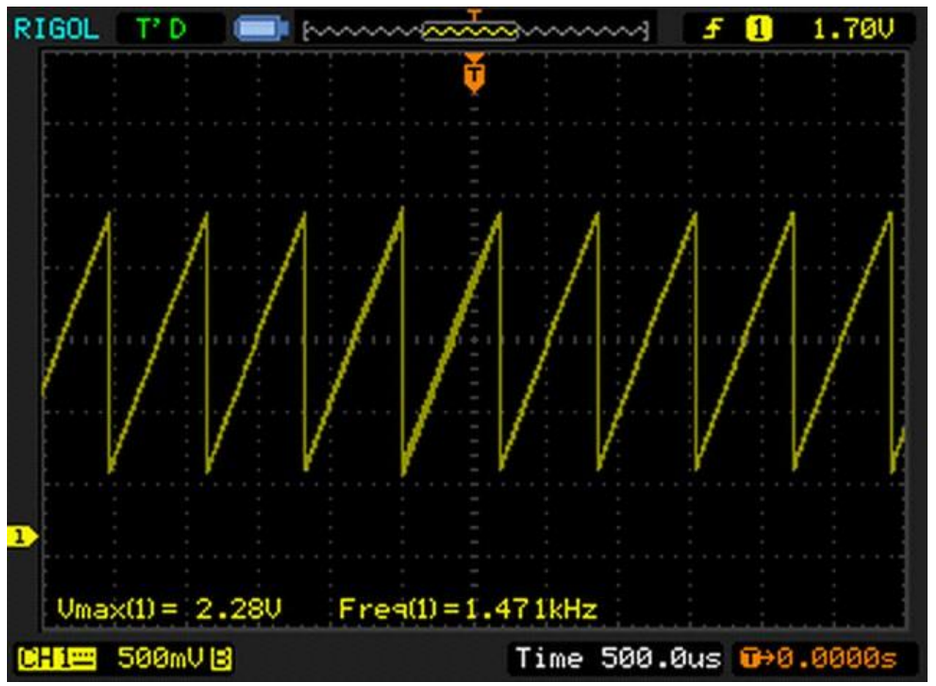
AnalogRead Exercise



analogWrite()



Pulse Width Modulation (PWM)



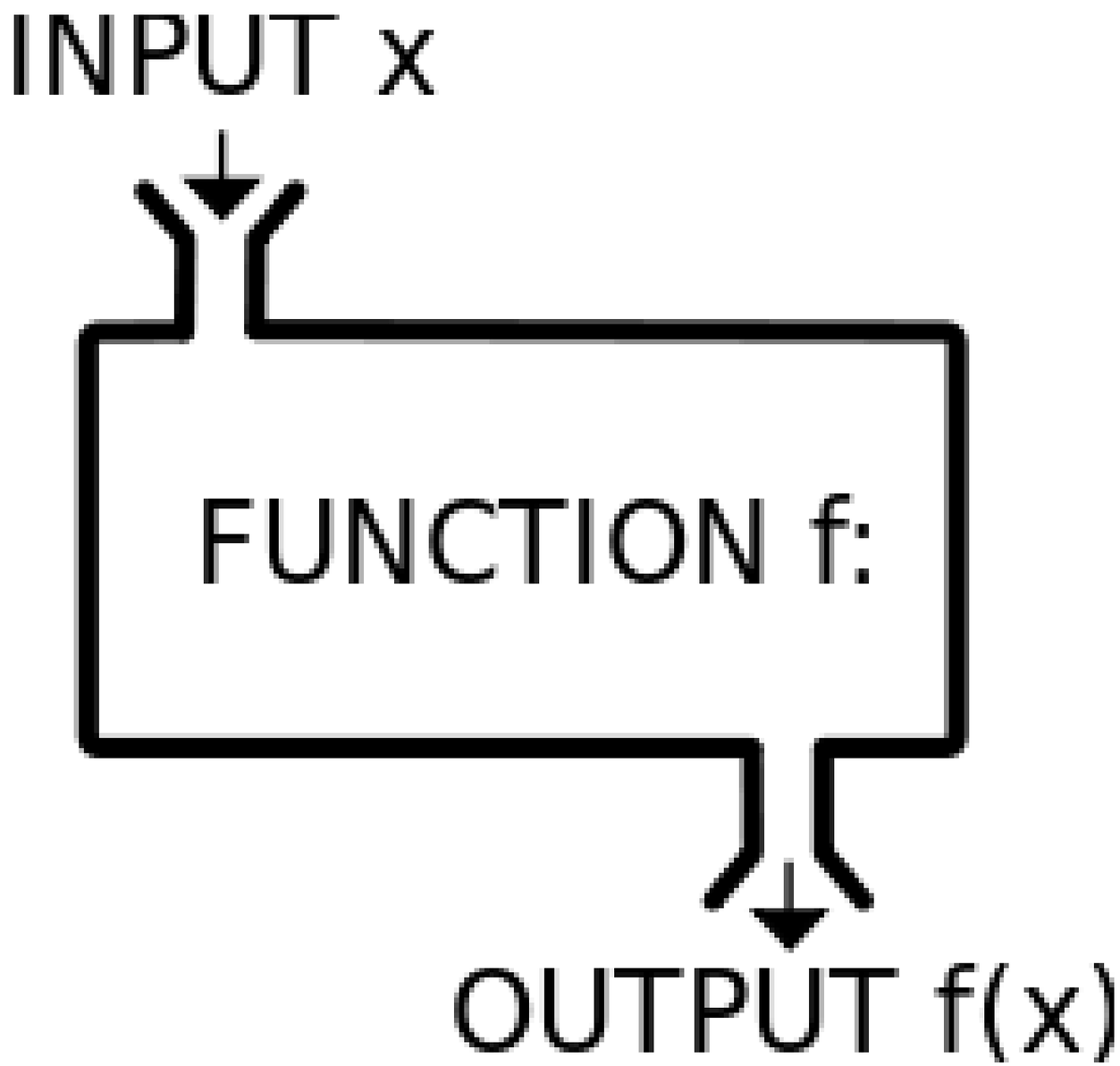
Actual Analog

analogWrite() isn’t Analog (Except on the Due)
Uno Pins: 3, 5, 6, 9, 10, 11

AnalogWrite Exercise

```
int LEDpin = D2;

/* By default PWM frequency is 1000Hz and we are using same
   for this application hence no need to set */

void setup(){
  Serial.begin(9600);
  analogWrite(LEDpin, 512);  /* set initial 50% duty cycle */
}

void loop(){
  /* read continuous POT and set PWM duty cycle according */
  uint16_t dutycycle =  analogRead(A0);
  /* limit dutycycle to 1023 if POT read cross it */
  if(dutycycle > 1023) dutycycle = 1023;
  Serial.print("Duty Cycle: ");  Serial.println(dutycycle);
  analogWrite(LEDpin, dutycycle);
  delay(100);
}
```

Remember to open the Serial Monitor!



functions()

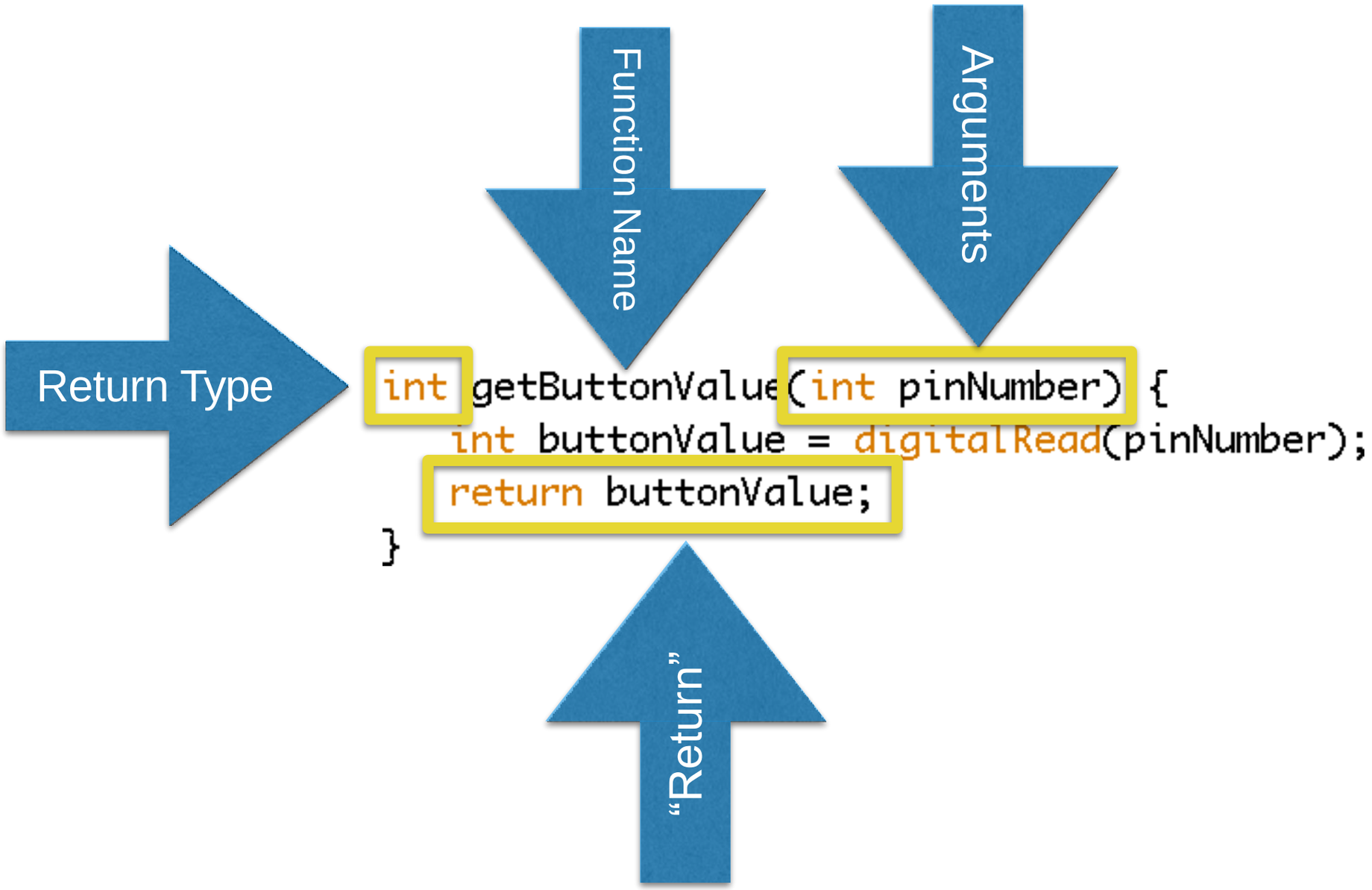
Functions

Getting Data Back

```
int getButtonValue(int pinNumber) {
    int buttonValue = digitalRead(pinNumber);
    return buttonValue;
}
```

Functions

Getting Data Back



Functions

Returning Nothing

Return Type

```
void flashLED(int pinNumber, int delayTime) {
    digitalWrite(pinNumber, HIGH);
    delay(delayTime);
    digitalWrite(pinNumber, LOW);
    delay(delayTime);
}
```

If the function doesn’t return anything, declare it as void

Function Exercise

- “Re-Write” the built-in Blink Example to use a Function

```
int LED = D2;


void setup() {
  // put your setup code here, to run once:
  pinMode(LED, OUTPUT);

}

void flashLED(){
  digitalWrite(LED,HIGH);
  delay(1000);
  digitalWrite(LED,LOW);
  delay(1000);
  }
void loop() {
  // put your main code here, to run repeatedly:
  flashLED();
}
```

Control Structures

control operators

==	Equal to
> >=	Greater than (or equal)
< <=	Less Than (or equal)
!=	Not Equal to

||	OR
&&	AND
|	Bitwise OR
&	Bitwise AND

IF

```
// These constants won't change:
const int ledPin = D2; // pin that the LED is attached to
void setup() {
 // initialize the LED pin as an output:
 pinMode(ledPin, OUTPUT);
 // initialize serial communications:
 Serial.begin(9600);
}
void loop() {
 // declare local variable
   char user_input;

 if (Serial.available() > 0) {     // is a character available?
   user_input = Serial.read();
    if (user_input == 'a') {
      digitalWrite(ledPin, HIGH);
    }
    if (user_input == 'b') {
      digitalWrite(ledPin, LOW);
    }
 }

}
```

*As soon as a character is sent from the serial monitor window, **Serial.available()** returns 1. When the if statement is evaluated again, it evaluates to true (because 1 > 0) and the code in the body of the if statement is run.*

The character that was sent from the serial monitor window is the stored in the user_input variable using the following line of code.

The sketch will switch the LED on if 'a' is sent to the Arduino and switch the LED off if 'b' is sent to the Arduino (note that these are case sensitive, so 'A' and 'B' won't work).

#1 if-statement mistake

```
int buttonValue = digitalRead(2);
if (buttonValue = HIGH) {
   digitalWrite(13, HIGH);
} else {
  digitalWrite(13, LOW);
}
```

```
int buttonValue = digitalRead(2);
if (buttonValue == HIGH) {
   digitalWrite(13, HIGH);
} else {
  digitalWrite(13, LOW);
}
```

#1 if-statement mistake

```
int buttonValue = digitalRead(2);
if (buttonValue = HIGH) {
   digitalWrite(13, HIGH);
} else {
   digitalWrite(13, LOW);
}
```

```
int buttonValue = digitalRead(2);
if (buttonValue == HIGH) {
   digitalWrite(13, HIGH);
} else {
   digitalWrite(13, LOW);
}
```

= != ==

IF-ELSE

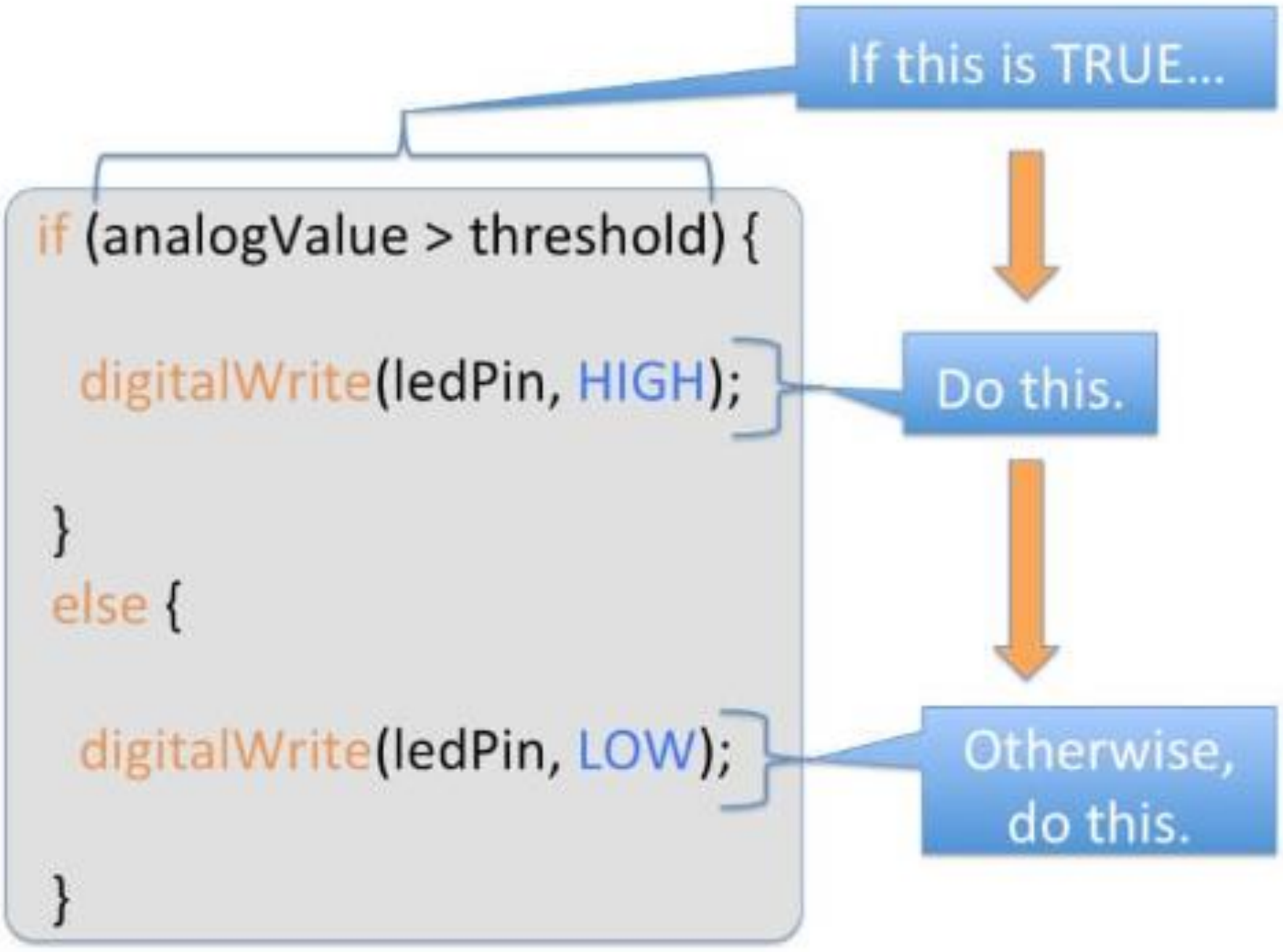
When the conditional expression evaluates to **true**:
1.Code in the body of the **if** statement is run.
2.Code in the body of the **else** statement is not run

When the conditional expression evaluates to **false**:
1.Code in the body of the **if** statement is not run.
2.Code in the body of the **else** statement is run.

IF-ELSE

```
// These constants won't change:
const int analogPin = A0;    // pin that the sensor is attached to
const int ledPin = D2;       // pin that the LED is attached to
const int threshold = 400;   // an arbitrary threshold level that's
                             //  in the range of the analog input

void setup() {
 // initialize the LED pin as an output:
 pinMode(ledPin, OUTPUT);
 // initialize serial communications:
 Serial.begin(9600);
}
void loop() {
 // read the value of the potentiometer:
 int analogValue = analogRead(analogPin);
 // if the analog value is high enough, turn on the LED:
 if (analogValue > threshold) {
 digitalWrite(ledPin, HIGH);
 }
 else {
 digitalWrite(ledPin,LOW);
 }
 // print the analog value:
 Serial.println(analogValue);
 delay(1); // delay in between reads for stability
}
```

IF-ELSE


If this is TRUE...
if (analogValue > threshold) {
digitalWrite(ledPin, HIGH);
}
else {
digitalWrite(ledPin, LOW);
}
Do this.
Otherwise, do this.

NESTED IF-ELSE

```
// These constants won't change:
const int ledPin = D2; // pin that the LED is attached to
void setup() {
 // initialize the LED pin as an output:
 pinMode(ledPin, OUTPUT);
 // initialize serial communications:
 Serial.begin(9600);
}
void loop() {
 // declare local variable
  char user_input;

if (Serial.available() > 0) {     // is a character available?
  user_input = Serial.read();       //
  if (user_input == 'a') {
     // switch the LED on if the character 'a' is received
    digitalWrite(ledPin, HIGH);
    delay(10000);
  }else{
    // switch the LED off if any character except 'a' is received
    digitalWrite(ledPin, LOW);
    delay(10000);
  }
}

}
```

The code resides nested inside the if(Serial.available()>0) statement

IF-ELSE IF

```
// These constants won't change:
const int ledPin1 = D2; // pin that the LED is attached to
const int ledPin2 = D5; // pin that the LED is attached to
const int ledPin3 = D8; // pin that the LED is attached to

void setup() {
 // initialize the LED pin as an output:
 pinMode(ledPin1, OUTPUT);
 pinMode(ledPin2, OUTPUT);
 pinMode(ledPin3, OUTPUT);
 // initialize serial communications:
 Serial.begin(9600);
}

char user_input = 0;

void loop(){
if (Serial.available() > 0) {    // is a character available?
    char user_input = Serial.read();
```

```
void loop(){
if (Serial.available() > 0) {    // is a character available?
    char user_input = Serial.read();

  if (user_input == 'a') {
    // the character 'a' was received, blink the LED once per second
    digitalWrite(ledPin1, HIGH);  // switch the LED on
    delay(500);                   // leave the LED on for 500ms
    digitalWrite(ledPin1, LOW);   // switch the LED off
    delay(500);                   // leave the LED off for 500ms
  }
  else if (user_input == 'b') {
    // the character 'b' was received, blink the LED every 400ms
    digitalWrite(ledPin2, HIGH);  // switch the LED on
    delay(200);                   // leave the LED on for 200ms
    digitalWrite(ledPin2, LOW);   // switch the LED off
    delay(200);                   // leave the LED off for 200ms
  }
  else {
    // any character except 'a' or 'b' was received
    digitalWrite(ledPin3, HIGH);  // switch the LED on
    delay(100);                   // leave the LED on for 100ms
    digitalWrite(ledPin3, LOW);   // switch the LED off
    delay(100);                   // leave the LED off for 100ms
    }
  }
}
```

```
const int min = 0;        // Lowest reading at analog pin
const int max = 1023;     // Highest reading at analog pin
void setup() {
 // initialise serial communication:
 Serial.begin(9600);
}
void loop() {

   // read the sensor:
   int sensorReading = analogRead(A0);

   // map the sensor range to a range of four options:
   int range = map(sensorReading, min, max, 0, 3);

   // do something different depend
   // range value:
   switch (range) {
   case 0: //Potentiometer turned between 0-25%
   Serial.println("low");
   break;
   case 1: //Potentiometer turned upto 26-50%
   Serial.println("medium");
   break;
   case 2:   //Potentiometer turned upto 51-75%
   Serial.println("high");
   break;
   case 3:   //Potentiometer turned
   Serial.println("ridiculous high"
   break;
   }
   delay(1);          // delay in between reads for stability
}
```

Switch Statement

switch case statements are useful when you want a list of options executed based on a variable value. 2. The *break* keyword is used in every case to exit from the switch case statement.

The switch statement is similar to using if with multiple else if constructs. Switch is used in conjunction with break

Variable you test against the cases.

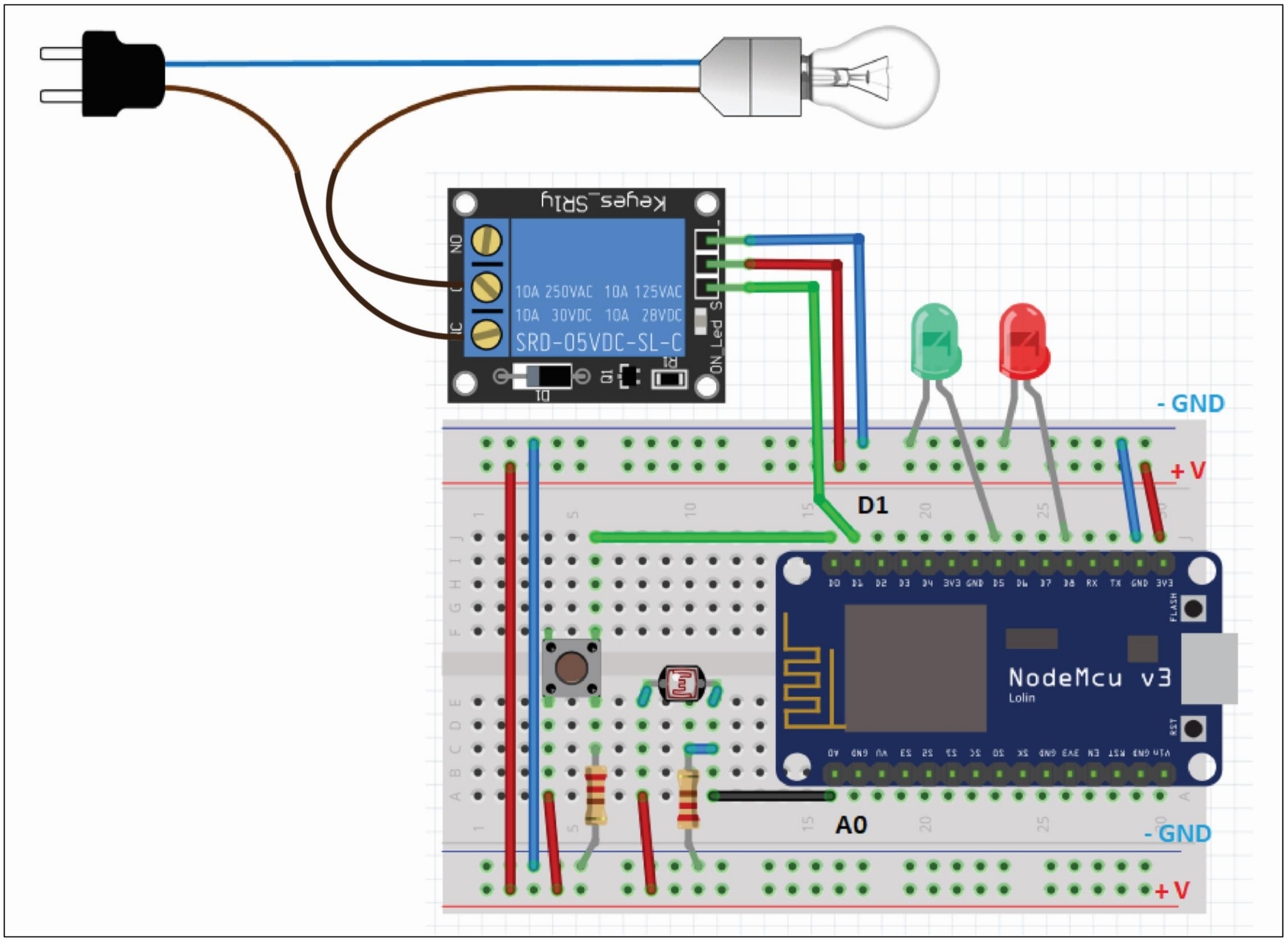
```
switch (range) {
 case 0:
    Serial.println("low");
    break;
 case 1:
    Serial.println("medium");
    break;
  case 2:
    Serial.println("high");
    break;
}
```

The case either matches the variable or not.

If a case matches, the code for that case gets executed.

The break keyword lets the program know the case is complete

CHALLENGE


Keyes_SR1y
NO
C
NC
10A 250VAC 10A 125VAC
10A 30VDC 10A 28VDC
SRD-05VDC-SL-C
D1
R1
S
Led
ON
NO
- GND
+ V
D1
NodeMcu v3
Lolin
FLASH
RST
A0
- GND
+ V

1. Make the LED switch ON every time the photoresistor exceed the threshold OR when the button is pressed.
2. Using a single button, create multiple options based on how long the button is pressed