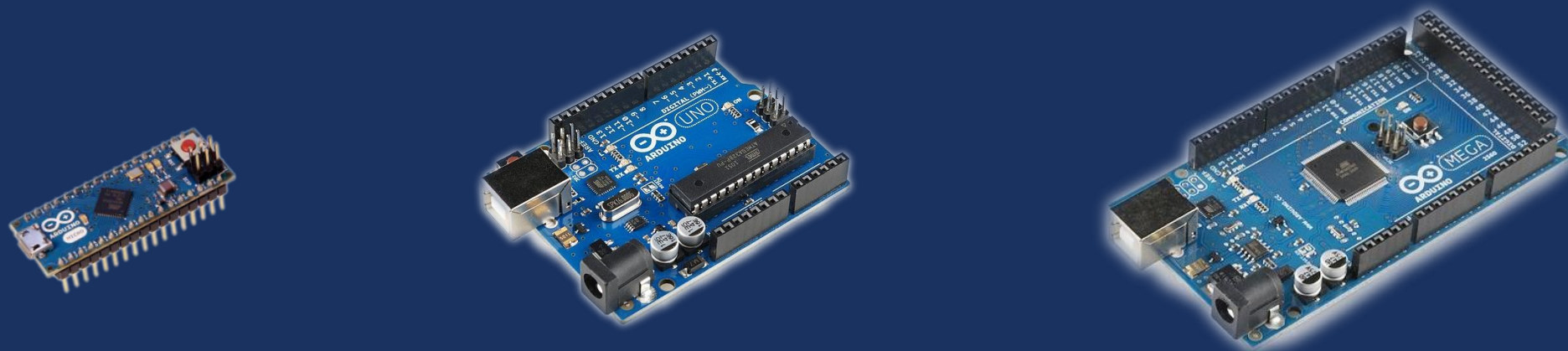


مبانی برنامه‌نویسی آردوینو با زبان C++

احمد رضا حیدری، دانشگاه کاشان - بهار ۱۴۰۴



سر فصل های دوره و زمان بندی جلسات

- آشنایی مقدماتی با میکروکنترلرها و برد آردینو (جلسه ۱ پنج شنبه ۱۴۰۴/۲/۱۸ ساعت ۱۵ به صورت مجازی)
- کد نویسی آردینو با زبان C++ (جلسه ۲ جمعه ۱۴۰۴/۲/۱۹ ساعت ۱۵ به صورت مجازی)
- مبانی کار با سخت افزار آردوینو (جلسه ۳ یکشنبه ۱۴۰۴/۲/۲۱ ساعت ۲۰ به صورت حضوری)
- راه اندازی سنسورها و ماژولها (جلسه ۴ دوشنبه ۱۴۰۴/۲/۲۲ ساعت ۲۰ به صورت حضوری)
- راه اندازی و کنترل سرعت موتور DC (جلسه ۵ یکشنبه ۱۴۰۴/۲/۲۸ ساعت ۲۰ به صورت حضوری)
- راه اندازی سروو موتور و کنترل زاویه ای (جلسه ۵ یکشنبه ۱۴۰۴/۲/۲۸ ساعت ۲۰ به صورت حضوری)
- راه اندازی LCD کاراکتری برای نمایش اطلاعات (جلسه ۶ دوشنبه ۱۴۰۴/۲/۲۹ ساعت ۲۰ به صورت حضوری)
- راه اندازی و کنترل بازوی رباتیک با آردوینو (جلسه ۶ دوشنبه ۱۴۰۴/۲/۲۹ ساعت ۲۰ به صورت حضوری)

شروع کار با آردوینو

آردوینو یک شرکت، پروژه و جامعه کاربری متن باز در زمینه سخت افزار و نرم افزار کامپیوتری است که کیت هایی مبتنی بر میکروکنترلر طراحی و تولید می کند. این کیت ها برای ساخت دستگاه های دیجیتال و اشیای تعاملی استفاده می شوند که توانایی حس کردن و کنترل اشیای موجود در دنیای فیزیکی را دارند. محصولات این پروژه به صورت متن باز هم در بخش سخت افزار و هم در بخش نرم افزار عرضه می شوند. ([ویکی پدیا](#))

منابع آردوینو در وبسایت رسمی آردوینو [:arduino.cc](http://arduino.cc)

■ [Getting started guide](#)

■ [Arduino introduction](#)

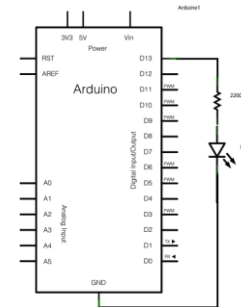
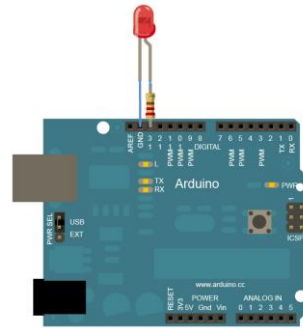
■ [Arduino tutorials](#)

■ [How it works: Foundations](#)

■ [Language reference](#)

■ [Arduino Hardware](#)

■ [Arduino IDE](#)



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```

آموزش نصب ARDUINO IDE

Downloads

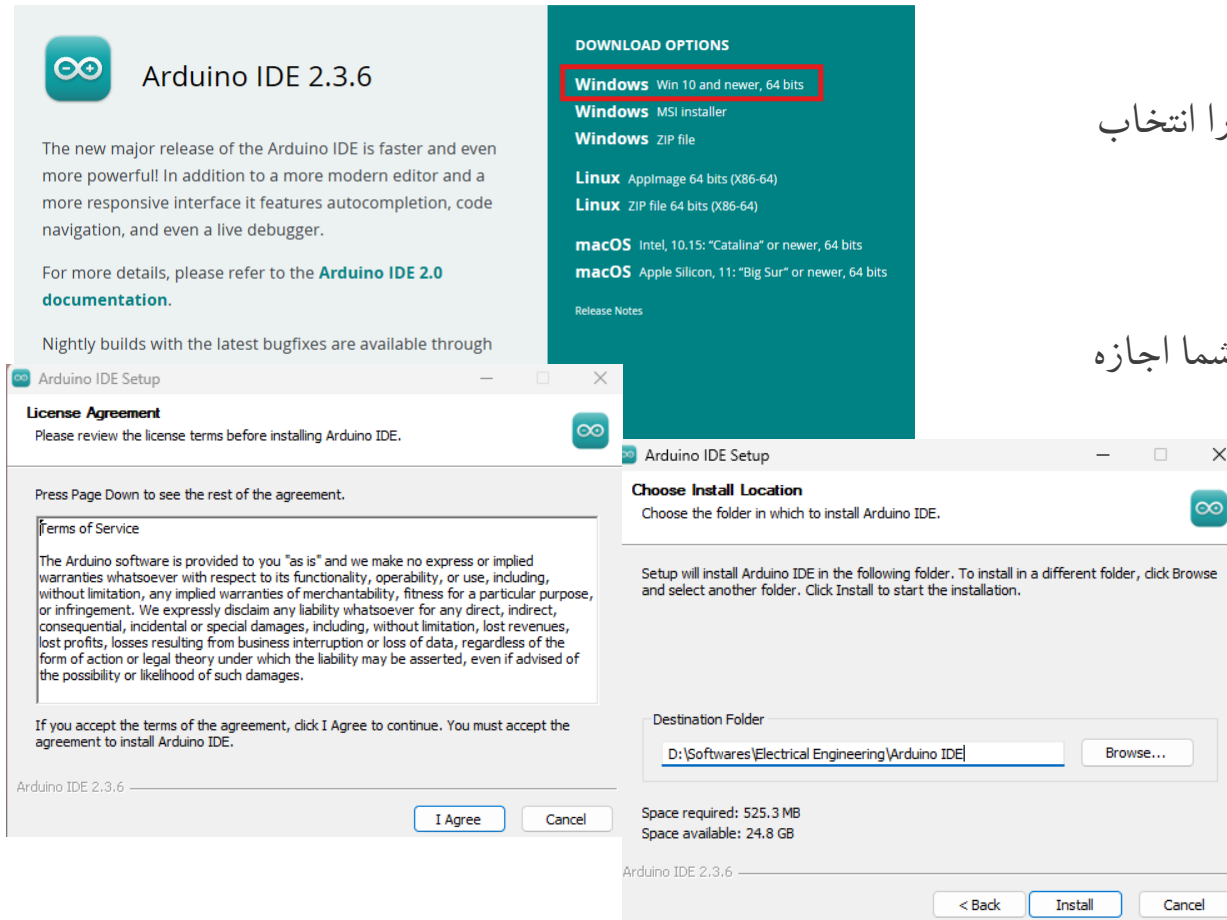
وارد سایت رسمی آردوینو شوید:

<https://www.arduino.cc/en/software>

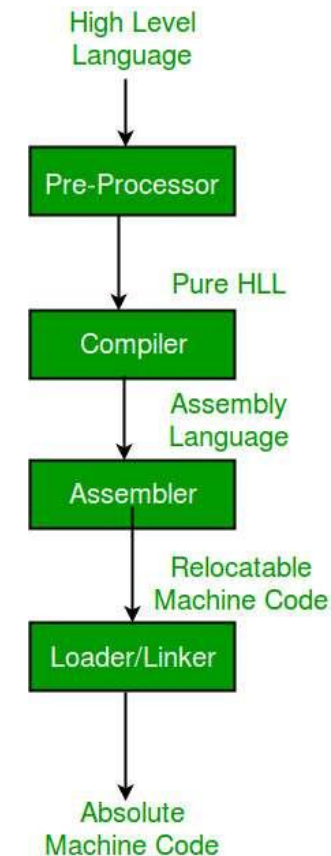
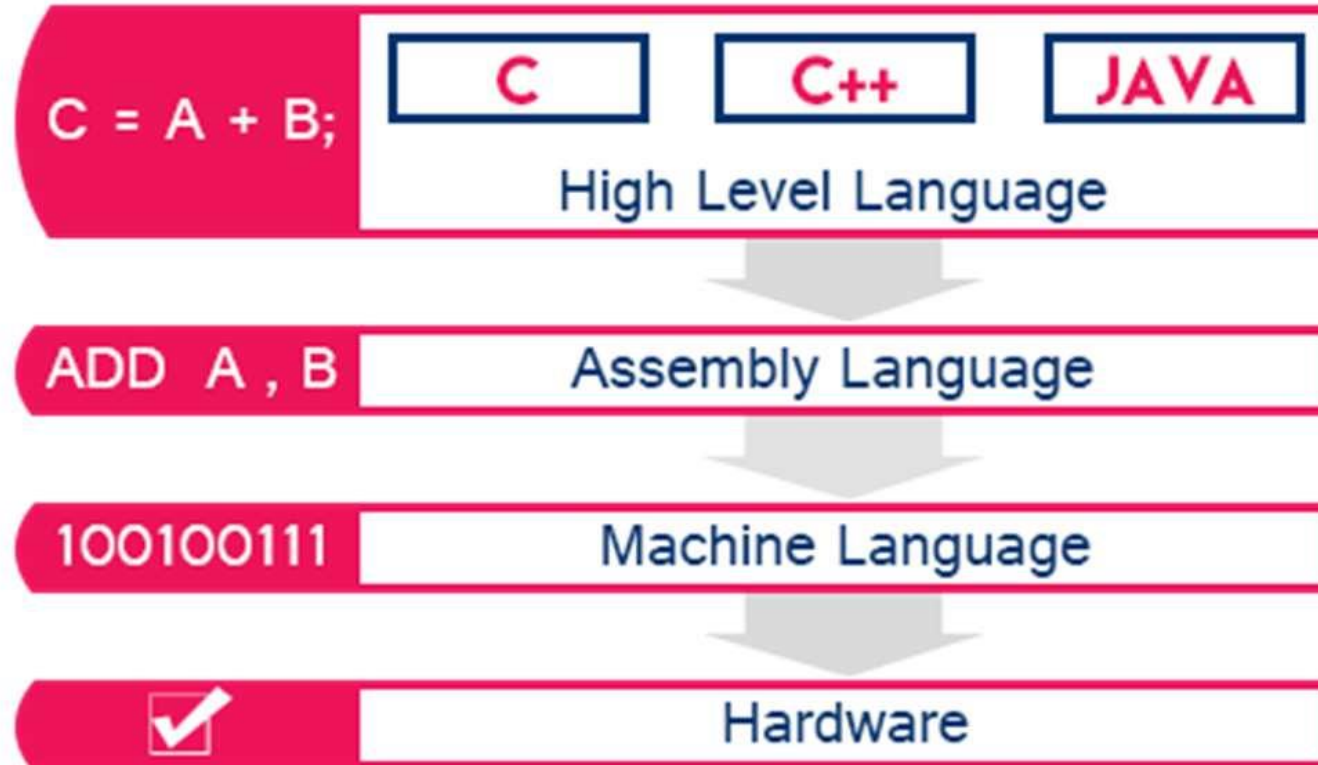
بسته به سیستم عامل خود یکی از گزینه ها را انتخاب کنید در اینجا ما ویندوز را انتخاب میکنیم.

نصب در ویندوز فایل exe را اجرا کنید.

گزینه های پیش فرض را بپذیرید (Next → Install) در حین نصب، اگر از شما اجازه نصب درایورها خواسته شد، Yes بزنید.

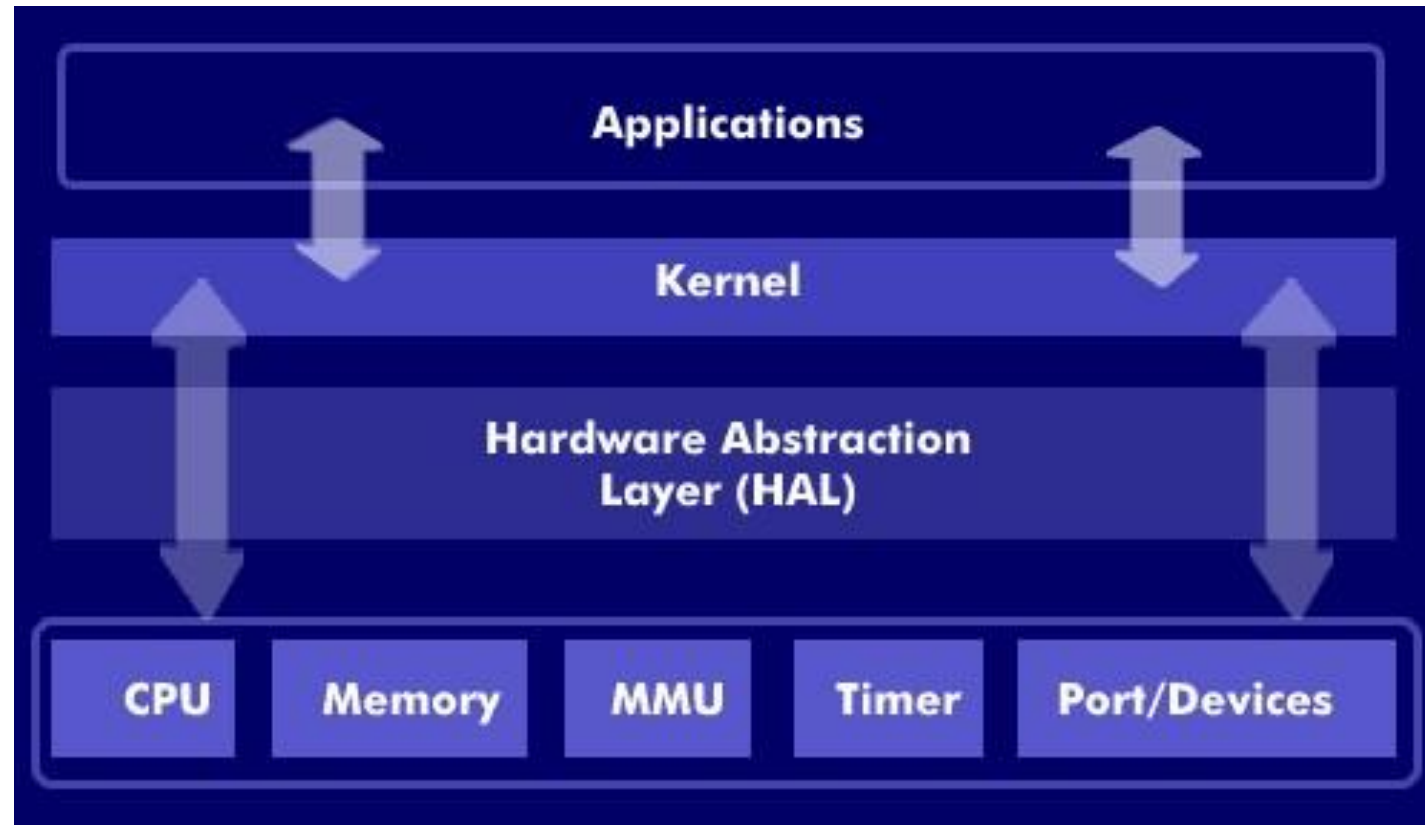


برنامه نویسی میکرو: زبان سطح بالا C

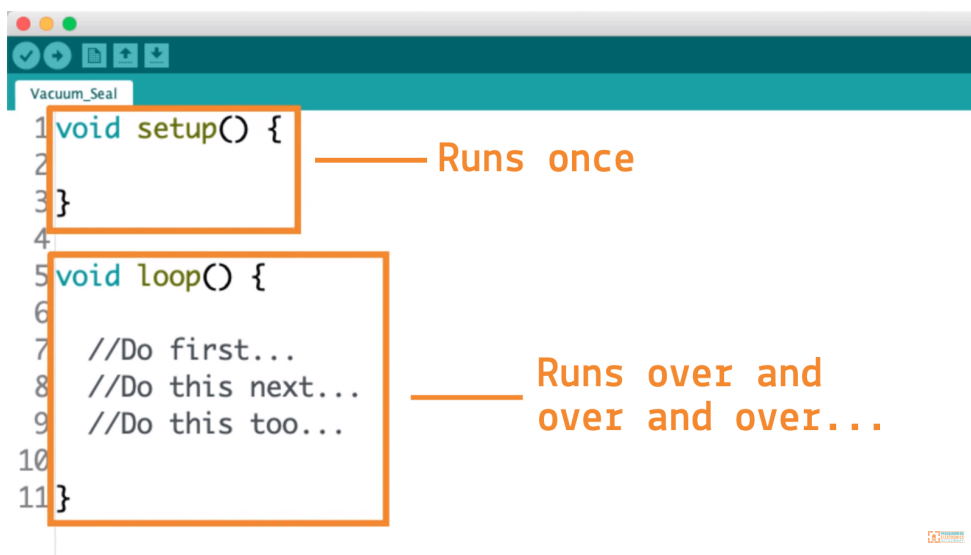


ساده سازی دسترسی به لایه های سخت افزار

توابع HAL (Hardware Abstraction Layer) کدهای آماده ای هستند که در صورت فراخوانی کنترل رجیسترها و تنظیمات لازم را انجام می دهند و نیازی به بازنویسی آنها نیست



آشنایی با توابع SETUP و LOOP در برنامه‌نویسی آردوینو



```
// the setup function runs once when you press reset or power the board  
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000); // wait for a second  
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW  
  delay(1000); // wait for a second  
}
```

بیشتر بردهای آردوینو به گونه‌ای طراحی شده‌اند که تنها یک برنامه روی میکروکنترلر آن‌ها اجرا شود. این برنامه می‌تواند برای انجام یک عمل ساده مانند چشمک زدن یک LED نوشته شده باشد، یا می‌تواند صدها عمل را به صورت چرخه‌ای اجرا کند. دامنه عملکرد برنامه‌ها بسته به هدف آن‌ها متفاوت است.

برنامه‌ای که روی میکروکنترلر بارگذاری می‌شود، به محض روشن شدن، شروع به اجرا خواهد کرد. هر برنامه دارای تابعی به نام `void setup` و `void loop` است.

تابع `setup` تنها یک بار، در ابتدای اجرای برنامه و پس از روشن شدن یا ریست شدن میکروکنترلر اجرا می‌شود. این تابع برای انجام تنظیمات اولیه استفاده می‌شود.

پس از اجرای کامل تابع `setup`، تابع `loop` به طور مداوم و پی‌درپی اجرا می‌شود و منطق اصلی برنامه در آن قرار دارد.

آشنایی با توابع SETUP و LOOP در برنامه‌نویسی آردوینو

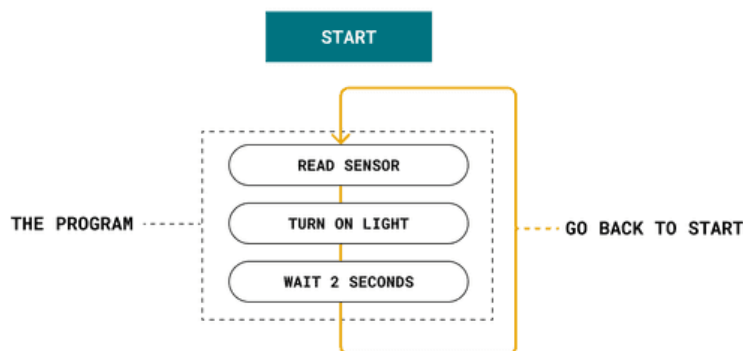
درون تابع setup ، می‌توانید برای مثال:

- تعیین پین‌ها به عنوان ورودی یا خروجی
- مقداردهی اولیه به متغیرها یا کتابخانه‌ها
- راه‌اندازی ارتباط سریال یا سنسورها

درون تابع loop، می‌توانید برای مثال:

- یک سنسور را بخوانید.
- یک چراغ را روشن کنید.
- بررسی کنید که آیا یک شرط برقرار است یا نه.
- یا همه موارد بالا را انجام دهید.

سرعت اجرای برنامه بسیار بالاست، مگر اینکه به آن بگوییم که آهسته‌تر اجرا شود. سرعت اجرا به اندازه برنامه و زمانی که اجرای آن توسط میکروکنترلر طول می‌کشد بستگی دارد، اما معمولاً در حد میکروثانیه (یک میلیونم ثانیه) است.



ورودی/خروجی دیجیتال DIGITAL I/O چیست؟

پایه‌های دیجیتال در آردوینو فقط دو حالت دارند:

روشن (HIGH) یا خاموش (LOW)

کاربردها:

- روشن/خاموش کردن LED

- خواندن وضعیت یک دکمه

دستورات رایج:

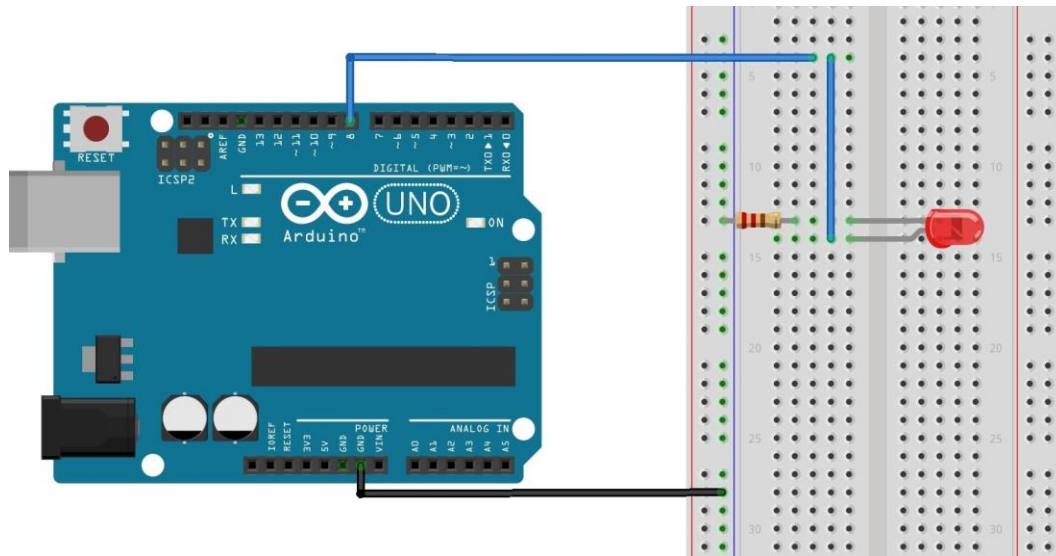
- تنظیم پین به صورت خروجی ← `pinMode(pin, OUTPUT);`

- تنظیم پین به صورت ورودی ← `pinMode(pin, INPUT);`

- ارسال سیگنال ۵ ولت یا روشن کردن ← `digitalWrite(pin, HIGH);`

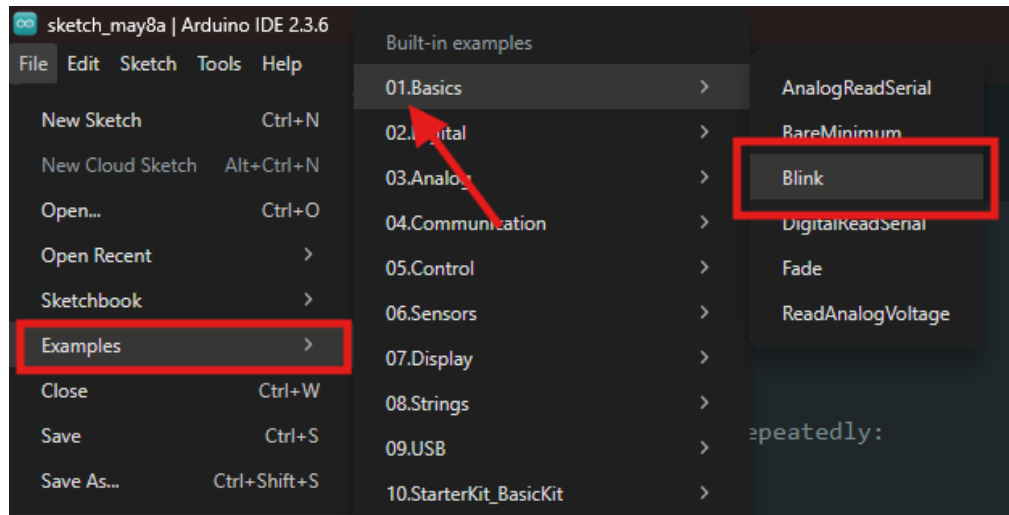
- ارسال سیگنال ۰ ولت یا خاموش کردن ← `digitalWrite(pin, LOW);`

- خواندن وضعیت از ورودی دیجیتال ← `digitalRead(pin);`



مثال ساده DIGITAL I/O (اولین مثال آردوینو، مثال چشمک زن)

نرم افزار آردوینو در خود مثال های متعددی دارد که در اینجا مثال Blink یا چشمک زن که یکی از ساده ترین مثال های آردوینو هست را مشاهده میکنید.

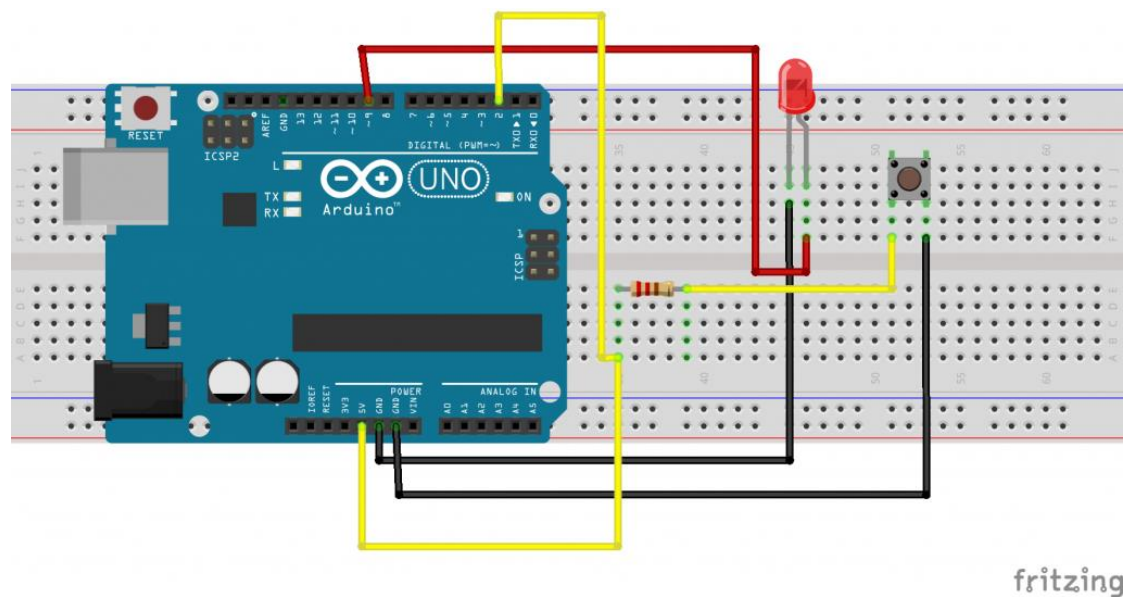


```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                     // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                     // wait for a second
}
```

مثال ساده DIGITAL I/O (روشن کردن چراغ با استفاده از دکمه)

هدف: روشن کردن یک LED با فشار دادن یک دکمه



```
void setup() {  
  pinMode(2, INPUT);  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  if (digitalRead(2) == HIGH) {  
    digitalWrite(13, HIGH);  
  } else {  
    digitalWrite(13, LOW);  
  }  
}
```

ورودی آنالوگ ANALOG INPUT

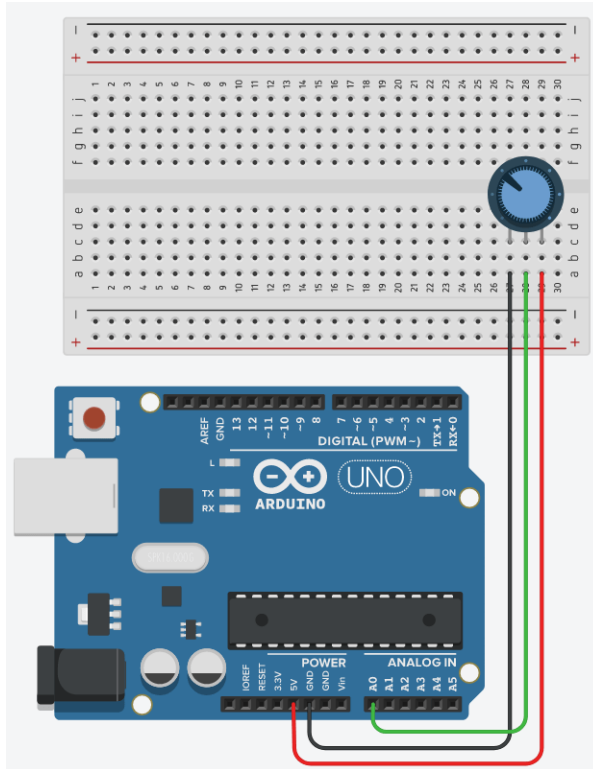
بعضی حسگرها مثل پتانسیومتر یا سنسور دما، خروجی پیوسته دارند

بین‌های A0 تا A5 در آردوینو برای ورودی آنالوگ هستند.

مقدار ورودی بین ۰ تا ۵ ولت خوانده می‌شود و به عددی بین ۰ تا ۱۰۲۳ تبدیل می‌شود (توسط ADC - مبدل آنالوگ به دیجیتال)

دستور لازم برای ورودی آنالوگ:

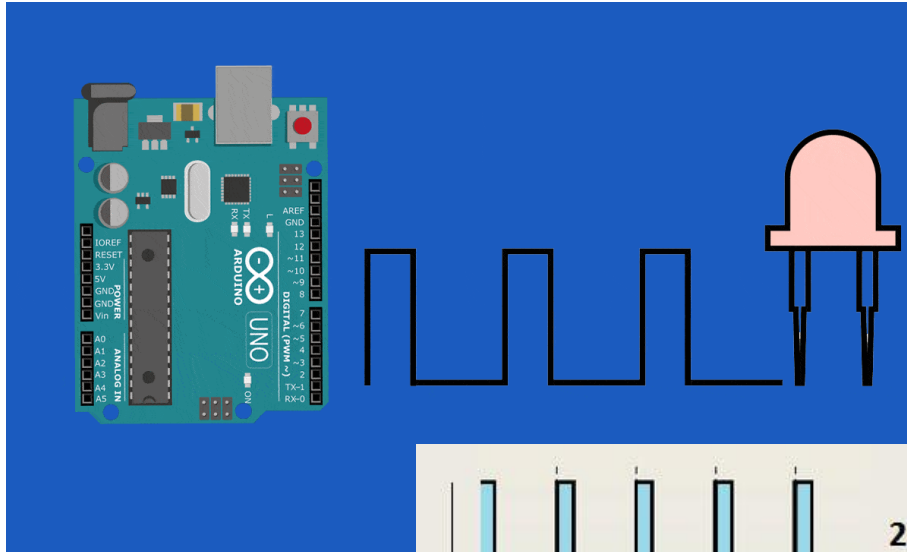
■ خواندن بین A0 آردوینو ← `analogRead(A0);`



```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  int sensorValue = analogRead(A0);  
  Serial.println(sensorValue);  
  delay(200);  
}
```

مثال خواندن یک ورودی آنالوگ (توسط پتانسیومتر)

خروجی شبه آنالوگ (PWM) ANALOG OUTPUT



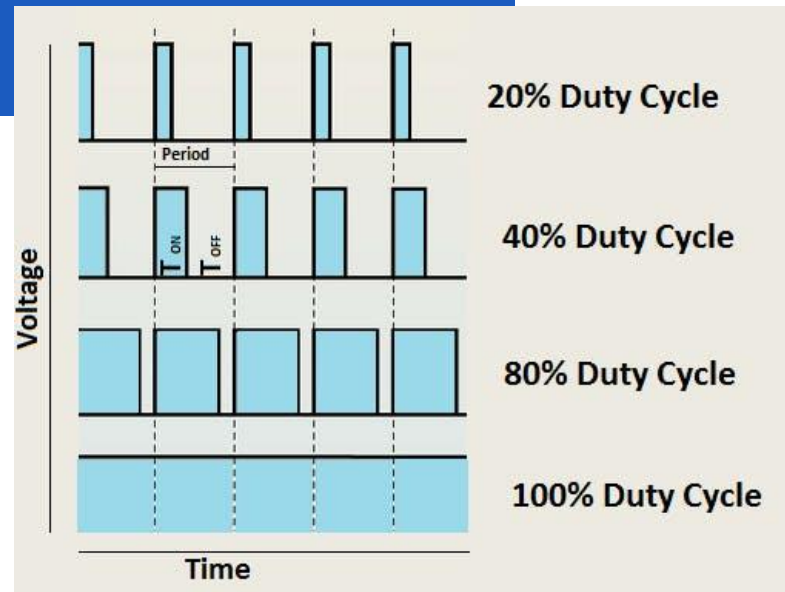
آردوینو خروجی آنالوگ واقعی ندارد، ولی می‌تواند با PWM خروجی شبه آنالوگ تولید کند کاربرد: کنترل روشنایی LED، سرعت موتور

توجه شود در آردوینو UNO فقط پین‌های ۳، ۵، ۶، ۹، ۱۰ و ۱۱ (پین‌های دارای علامت ~ در کنار شماره) دیجیتال چنین قابلیت‌هایی دارند

دستورات لازم برای خروجی آنالوگ:

■ تنظیم پین به صورت خروجی ← `pinMode(9, OUTPUT);`

■ نوشتن مقدار بر روی پین ← `analogWrite(9, pwmValue);`



مثال کنترل روشنایی LED با PWM (دایمر)

در این مثال، روشنایی یک LED با چرخاندن پتانسیومتر تغییر می‌کند. مقدار خوانده‌شده از A0 به مقدار PWM بین ۰ تا ۲۵۵ تبدیل می‌شود و به پین خروجی داده می‌شود.

```
void setup() {  
  pinMode(9, OUTPUT);  
}  
  
void loop() {  
  int sensorValue = analogRead(A0);  
  int pwmValue = map(sensorValue, 0, 1023, 0, 255);  
  analogWrite(9, pwmValue);  
  delay(10);  
}
```

تابع map تبدیل یک عدد از یک بازه (range) به عددی متناظر در بازه‌ای دیگر

به فرض در این مثال: تبدیل مقدار خوانده‌شده از سنسور (۰ تا ۱۰۲۳) به مقدار PWM (۰ تا ۲۵۵)

ARDUINO PROGRAMMING CHEAT SHEET

Arduino Programming Cheat Sheet

Primary source: Arduino Language Reference
<https://www.arduino.cc/reference/en/>

Structure & Flow

```
Basic Program Structure
void setup() {
  // Runs once when sketch starts
}
void loop() {
  // Runs repeatedly
}

Control Structures
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
for (int i = 0; i < 10; i++) { ... }
break; // Exit a loop immediately
continue; // Go to next iteration
switch (var) {
  case 1:
    ...
    break;
  case 2:
    ...
    break;
  default:
    ...
}
return x; // x must match return type
return; // For void return type

Function Definitions
<ret. type> <name>(<params>) { ... }
e.g. int double(int x) {return x*2;}
```

Operators

General Operators

- = assignment
- + add - subtract
- * multiply / divide
- % modulo
- == equal to != not equal to
- < less than > greater than
- <= less than or equal to
- >= greater than or equal to
- && and || or
- ! not

Compound Operators

- ++ increment -- decrement
- += compound addition -= compound subtraction
- *= compound multiplication /= compound division
- &= compound bitwise and |= compound bitwise or

Bitwise Operators

- & bitwise and | bitwise or
- ^ bitwise xor ~ bitwise not
- << shift left >> shift right

Pointer Access

- & reference: get a pointer
- * dereference: follow a pointer

Built-in Functions

Pin Input/Output

Digital I/O - pins 0-13 A0-A5

```
pinMode(pin, INPUT|OUTPUT|INPUT_PULLUP);
int digitalRead(pin);
digitalWrite(pin, HIGH|LOW);
```

Analog In - pins A0-A5

```
int analogRead(pin);
analogReference(DEFAULT|INTERNAL|EXTERNAL);
```

PWM Out - pins 3 5 6 9 10 11

```
analogWrite(pin, value) // 0-255
```

Advanced I/O

```
tone(pin, freq_Hz, [duration_msec])
noTone(pin)
shiftOut(dataPin, clockPin, (MSBFIRST|LSBFIRST), value)
shiftIn(dataPin, clockPin, (MSBFIRST|LSBFIRST))
unsigned long pulseIn(pin, (HIGH|LOW), [timeout_usec])
```

Time

```
unsigned long millis() // Overflows at 50 days
unsigned long micros() // Overflows at 70 minutes
delay(msec)
delayMicroseconds(usec)
```

Math

```
min(x, y) max(x, y) abs(x)
sin(rad) cos(rad) tan(rad)
sqrt(x) pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

Random Numbers

```
randomSeed(seed) // long or int
long random(max) // 0 to max-1
long random(min, max)
```

Bits and Bytes

```
lowByte(x) highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn) // bitn: 0=LSB 7=MSB
```

Type Conversions

```
char(val) byte(val)
int(val) word(val)
long(val) float(val)
```

External Interrupts

```
attachInterrupt(interrupt, func, (LOW|CHANGE|RISING|FALLING))
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```

Libraries

Serial - comm. with PC or via RX/TX

```
begin(long speed) // Up to 115200
end()
int available() // #bytes available
int read() // -1 if none available
int peek() // Read w/o removing
flush()
print(data) println(data)
write(byte) write(char * string)
write(byte * data, size)
SerialEvent() // Called if data rdy
```

SoftwareSerial.h - comm. on any pin

```
SoftwareSerial(rxPin, txPin)
begin(long speed) // Up to 115200
listen() // Only 1 can listen
isListening() // at a time.
read, peek, print, println, write
// Equivalent to Serial library
```

EEPROM.h - access non-volatile memory

```
byte read(addr)
write(addr, byte)
EEPROM[index] // Access as array
```

Servo.h - control servo motors

```
attach(pin, [min_usec, max_usec])
write(angle) // 0 to 180
writeMicroseconds(uS) // 1000-2000; 1500 is midpoint
int read() // 0 to 180
bool attached()
detach()
```

Wire.h - I²C communication

```
begin() // Join a master
begin(addr) // Join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(byte) // Step 2
send(char * string)
send(byte * data, size)
endTransmission() // Step 3
int available() // #bytes available
byte receive() // Get next byte
onReceive(handler)
onRequest(handler)
```

Variables, Arrays, and Data

Data Types

bool	true false
char	-128 - 127, 'a' '\$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	currently same as float
void	return type: no return value

Strings

```
char str1[8] = {'A', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
// Includes \0 null termination
char str2[8] = {'A', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
// Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numeric Constants

123	decimal
0b0111011	binary
0173	octal - base 8
0x7B	hexadecimal - base 16
123U	force unsigned
123L	force long
123UL	force unsigned long
123.0	force floating point
1.23e6	1.23*10 ⁶ = 1230000

Qualifiers

static	persists between calls
volatile	in RAM (nice for ISR)
const	read-only
PROGRAM	in flash

Arrays

```
byte myPins[] = {2, 4, 8, 3, 6};
int myInts[6]; // Array of 6 ints
myInts[0] = 42; // Assigning first
// index of myInts
myInts[6] = 12; // ERROR! Indexes
// are 0 through 5
```



CC BY SA by Mark Liffiton
version: 2024-02-14
source: <https://github.com/liffiton/Arduino-Cheat-Sheet/>
Adapted from:
- Original: Gavin Smith
- SVG version: Frederic Dufourg
- Arduino board drawing: Fritzing.org

<https://github.com/liffiton/Arduino-Cheat-Sheet>

BUILT-IN FUNCTIONS

Built-in Functions

Pin Input/Output

Digital I/O - pins 0-13 A0-A5

```
pinMode(pin,
  {INPUT|OUTPUT|INPUT_PULLUP})
int digitalRead(pin)
digitalWrite(pin, {HIGH|LOW})
```

Analog In - pins A0-A5

```
int analogRead(pin)
analogReference(
  {DEFAULT|INTERNAL|EXTERNAL})
```

PWM Out - pins 3 5 6 9 10 11

```
analogWrite(pin, value) // 0-255
```

Advanced I/O

```
tone(pin, freq_Hz, [duration_msec])
noTone(pin)
shiftOut(dataPin, clockPin,
  {MSBFIRST|LSBFIRST}, value)
shiftIn(dataPin, clockPin,
  {MSBFIRST|LSBFIRST})
unsigned long pulseIn(pin,
  {HIGH|LOW}, [timeout_usec])
```

Time

```
unsigned long millis()
  // Overflows at 50 days
unsigned long micros()
  // Overflows at 70 minutes
delay(msec)
delayMicroseconds(usec)
```

Math

```
min(x, y)    max(x, y)    abs(x)
sin(rad)     cos(rad)     tan(rad)
sqrt(x)       pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

Random Numbers

```
randomSeed(seed) // long or int
long random(max) // 0 to max-1
long random(min, max)
```

Bits and Bytes

```
lowByte(x)    highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn) // bitn: 0=LSB 7=MSB
```

Type Conversions

```
char(val)      byte(val)
int(val)        word(val)
long(val)       float(val)
```

External Interrupts

```
attachInterrupt(interrupt, func,
  {LOW|CHANGE|RISING|FALLING})
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```

STRUCTURE & FLOW

Structure & Flow

Basic Program Structure

```
void setup() {  
    // Runs once when sketch starts  
}  
void loop() {  
    // Runs repeatedly  
}
```

Control Structures

```
if (x < 5) { ... } else { ... }  
while (x < 5) { ... }  
for (int i = 0; i < 10; i++) { ... }  
break;    // Exit a loop immediately  
continue; // Go to next iteration  
switch (var) {  
    case 1:  
        ...  
        break;  
    case 2:  
        ...  
        break;  
    default:  
        ...  
}  
return x; // x must match return type  
return;  // For void return type
```

Function Definitions

```
<ret. type> <name>(<params>) { ... }  
e.g. int double(int x) {return x*2;}
```

OPERATORS

Operators

General Operators

```
= assignment
+ add      - subtract
* multiply  / divide
% modulo
== equal to  != not equal to
< less than > greater than
<= less than or equal to
>= greater than or equal to
&& and      || or
! not
```

Compound Operators

```
++ increment
-- decrement
+= compound addition
-= compound subtraction
*= compound multiplication
/= compound division
&= compound bitwise and
|= compound bitwise or
```

Bitwise Operators

```
& bitwise and    | bitwise or
^ bitwise xor    ~ bitwise not
<< shift left    >> shift right
```

Pointer Access

```
& reference: get a pointer
* dereference: follow a pointer
```

VARIABLES, ARRAYS, AND DATA

Variables, Arrays, and Data

Data Types

bool	true false
char	-128 - 127, 'a' '\$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	currently same as float
void	return type: no return value

Strings

```
char str1[8] =  
    {'A','r','d','u','i','n','o','\0'};  
    // Includes \0 null termination  
char str2[8] =  
    {'A','r','d','u','i','n','o'};  
    // Compiler adds null termination  
char str3[] = "Arduino";  
char str4[8] = "Arduino";
```

Numeric Constants

123	decimal
0b01111011	binary
0173	octal - base 8
0x7B	hexadecimal - base 16
123U	force unsigned
123L	force long
123UL	force unsigned long
123.0	force floating point
1.23e6	1.23*10^6 = 1230000

Qualifiers

static	persists between calls
volatile	in RAM (nice for ISR)
const	read-only
PROGMEM	in flash

Arrays

```
byte myPins[] = {2, 4, 8, 3, 6};  
int myInts[6];    // Array of 6 ints  
myInts[0] = 42;   // Assigning first  
                  // index of myInts  
myInts[6] = 12;   // ERROR! Indexes  
                  // are 0 though 5
```

LIBRARIES

Libraries

```
Serial - comm. with PC or via RX/TX
begin(long speed) // Up to 115200
end()
int available() // #bytes available
int read() // -1 if none available
int peek() // Read w/o removing
flush()
print(data) println(data)
write(byte) write(char * string)
write(byte * data, size)
SerialEvent() // Called if data rdy


SoftwareSerial.h - comm. on any pin
SoftwareSerial(rxPin, txPin)
begin(long speed) // Up to 115200
listen() // Only 1 can listen
isListening() // at a time.
read, peek, print, println, write
// Equivalent to Serial library

EEPROM.h - access non-volatile memory
byte read(addr)
write(addr, byte)
EEPROM[index] // Access as array

Servo.h - control servo motors
attach(pin, [min_usec, max_usec])
write(angle) // 0 to 180
writeMicroseconds(uS) // 1000-2000; 1500 is midpoint
int read() // 0 to 180
bool attached()
detach()

Wire.h - I2C communication
begin() // Join a master
begin(addr) // Join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(byte) // Step 2
send(char * string)
send(byte * data, size)
endTransmission() // Step 3
int available() // #bytes available
byte receive() // Get next byte
onReceive(handler)
onRequest(handler)
```

ARDUINO LANGUAGE REFERENCE

 **DOCS**

Search on Docs

ARDUINO.CC

← All Docs

En

Language Reference

Functions

Variables

Structure

Home / Programming / Language Reference

Language Reference

Arduino programming language can be divided in three main parts: functions, values (variables and constants), and structure.

Functions

Variables

Structure

For controlling the Arduino board and performing computations.

Digital I/O digitalRead() digitalWrite() pinMode()	Math abs() constrain() map() max() min() pow() sq() sqrt()	Bits and Bytes bit() bitClear() bitRead() bitSet() bitWrite() highByte() lowByte()
Analog I/O analogRead() analogReadResolution() analogReference() analogWrite() analogWriteResolution()	Trigonometry cos() sin() tan()	External Interrupts attachInterrupt() detachInterrupt() digitalPinToInterrupt()
Advanced I/O noTone()	Characters isAlpha()	Interrupts interrupts()

Help

<https://docs.arduino.cc/language-reference>

FUNCTIONS

Digital I/O

`digitalRead()`
`digitalWrite()`
`pinMode()`

Math

`abs()`
`constrain()`
`map()`
`max()`
`min()`
`pow()`
`sq()`
`sqrt()`

Bits and Bytes

`bit()`
`bitClear()`
`bitRead()`
`bitSet()`
`bitWrite()`
`highByte()`
`lowByte()`

Analog I/O

`analogRead()`
`analogReadResolution()`
`analogReference()`
`analogWrite()`
`analogWriteResolution()`

Trigonometry

`cos()`
`sin()`
`tan()`

External Interrupts

`attachInterrupt()`
`detachInterrupt()`
`digitalPinToInterrupt()`

<https://docs.arduino.cc/language-reference>

FUNCTIONS

Advanced I/O

noTone()
pulseIn()
pulseInLong()
shiftIn()
shiftOut()
tone()

Characters

isAlpha()
isAlphaNumeric()
isAscii()
isControl()
isDigit()
isGraph()
isHexadecimalDigit()
isLowerCase()
isPrintable()
isPunct()
isSpace()
isUpperCase()
isWhitespace()

Interrupts

interrupts()
noInterrupts()

<https://docs.arduino.cc/language-reference>

FUNCTIONS

Time

delay()
delayMicroseconds()
micros()
millis()

Random Numbers

random()
randomSeed()

Communication

SPI
Print
Serial
Stream
Wire

USB

Keyboard
Mouse

Wi-Fi

Wi-Fi Overview
WiFi Network
IPAddress
WiFiClient
WiFiServer
WiFiUDP

<https://docs.arduino.cc/language-reference>

VARIABLES

Constants

Floating Point Constants

HIGH | LOW

INPUT | INPUT_PULLUP | OUTPUT

Integer Constants

LED_BUILTIN

true | false

Data Types

array

bool

boolean

byte

char

double

float

int

long

short

size_t

string

String()

unsigned char

unsigned int

unsigned long

void

word

Variable Scope & Qualifiers

const

scope

static

volatile

<https://docs.arduino.cc/language-reference>

VARIABLES

Conversion

byte()

char()

float()

int()

long()

(unsigned int)

(unsigned long)

word()

Utilities

PROGMEM

sizeof()

<https://docs.arduino.cc/language-reference>

STRUCTURE

Sketch

loop()
setup()

Arithmetic Operators

+ (addition)
= (assignment operator)
/ (division)
* (multiplication)
% (remainder)
- (subtraction)

Pointer Access Operators

* (dereference operator)
& (reference operator)

<https://docs.arduino.cc/language-reference>

STRUCTURE

Control Structure

break
continue
do...while
else
for
goto
if
return
switch...case
while

Comparison Operators

== (equal to)
> (greater than)
>= (greater than or equal to)
< (less than)
<= (less than or equal to)
!= (not equal to)

Bitwise Operators

<< (bitshift left)
>> (bitshift right)
& (bitwise AND)
~ (bitwise NOT)
| (bitwise OR)
^ (bitwise XOR)

STRUCTURE

Further Syntax

`/* */` (block comment)
`{ }` (curly braces)
`#define`
`#include`
`;` (semicolon)
`//` (single line comment)

Boolean Operators

`&&` (logical AND)
`!` (logical NOT)
`||` (logical OR)

Compound Operators

`+=` (compound addition)
`&=` (compound bitwise AND)
`|=` (compound bitwise OR)
`^=` (compound bitwise XOR)
`/=` (compound division)
`*=` (compound multiplication)
`%=` (compound remainder)
`-=` (compound subtraction)
`--` (decrement)
`++` (increment)

راه های ارتباطی و لینک های مربوط به دوره

در صورت وجود هرگونه ابهام و مشکل در حین دوره میتوانید از راه های زیر با بنده در ارتباط باشید:

theheidari@gmail.com

در تلگرام و ایتا [@xheidari](https://t.me/xheidari)

<https://www.linkedin.com/in/xheidari/>

در ضمن تمامی محتوای ارائه شده دوره به تدریج در گروه تلگرامی و لینک گیت هاب دوره آپلود میشود:

<https://github.com/xheidari/ArduinoCourse>

شبیه سازی ها در سایت Tinkercad انجام خواهد شد و در لینک زیر شبیه سازی های انجام شده در کلاس قرار میگیرد:

<https://www.tinkercad.com/joinclass/KD54P7ADW>

با تشکر از توجه شما!