

# Take precautions on campus

- ★ **Wear a face covering** in crowded places and when sitting close to others
- ♥ **Get vaccinated if you haven't already**
- ▲ **Wash your hands regularly** - hand sanitiser is available across campus
- **Follow one-way systems** - make space for each other when moving around
- **Take two Covid-19 tests each week**
- ◆ **Stay at home if you feel unwell**, even if it's just a cold

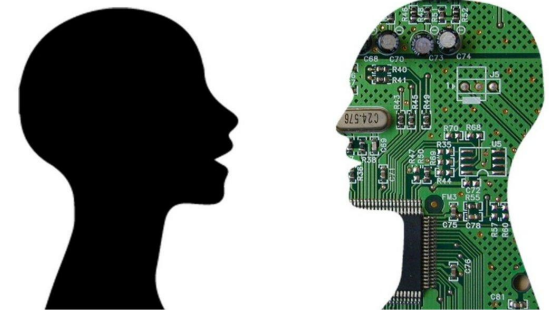
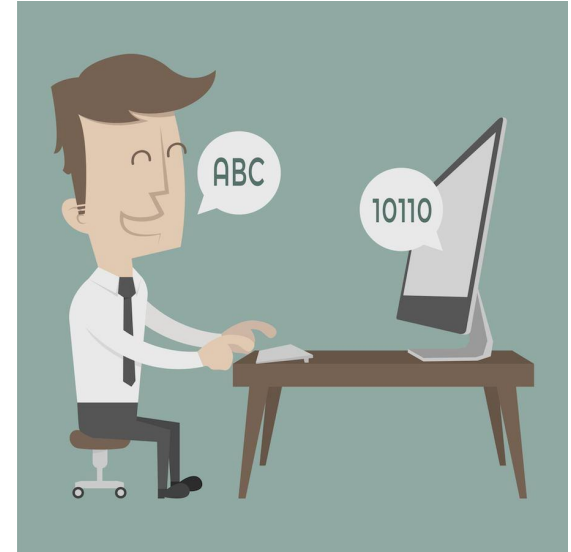


Enjoy **campus** safely

# **Introduction to C** **with reference to Arduino**

# Programming Languages

- Interface between human language and binary instructions
- User friendly and fast to code
- Can be high level (python), low level (C) or even very low level (Assembly)
- Languages are usually distinguished in compiled and interpreted languages
- Very different syntax (like different spoken languages)



<https://www.vikingcodeschool.com/web-design-basics/components-of-good-ux>

<https://twitter.com/organicinc/status/799682205219950592>

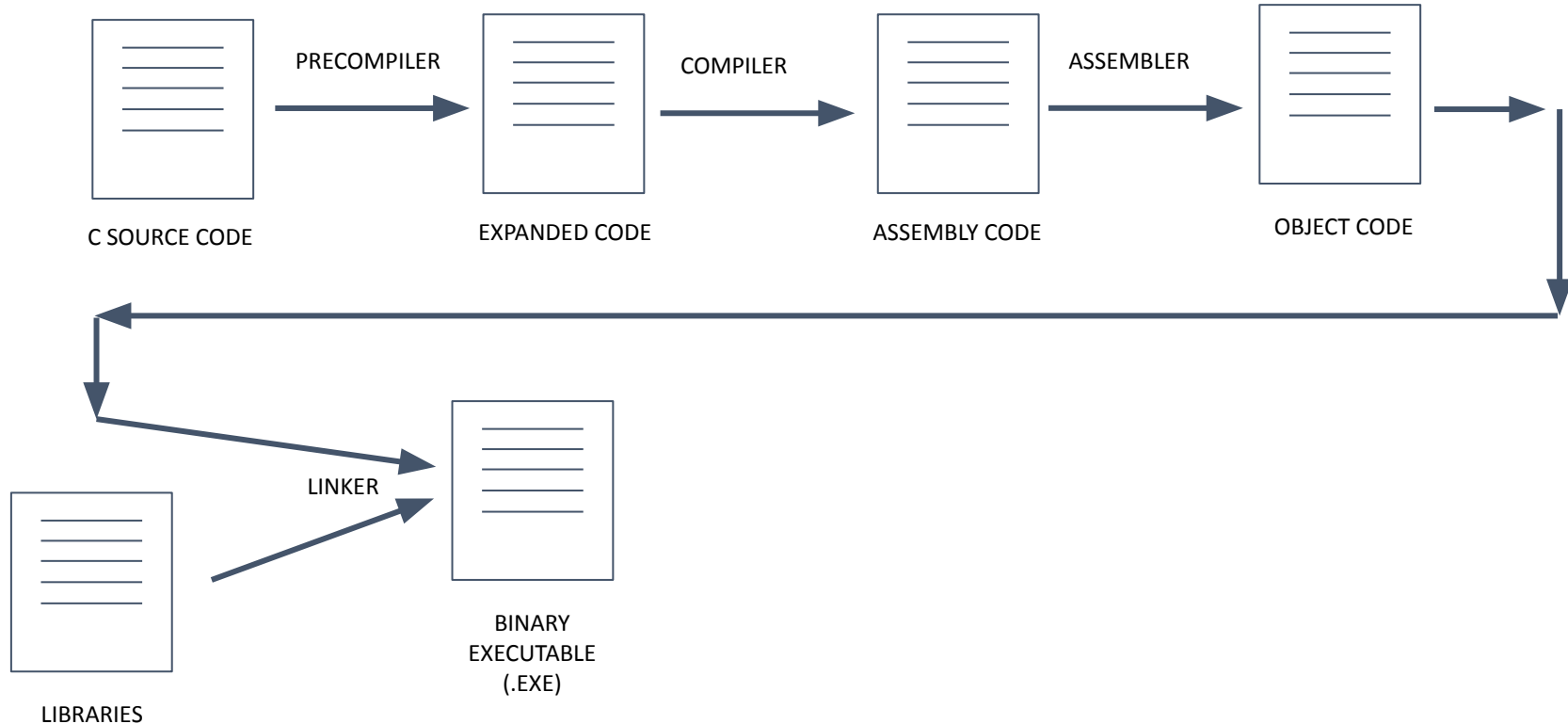
# What is C?

- C is a general-purpose programming language used in many applications
- It is also a compiled, procedural programming language
  - Before execution, the program must be converted by the compiler into machine language that the processor can understand
  - The compiled program or "script" can then be executed or "run"
- Rigid syntax
- Parentheses are used to group blocks of code
  
- Arduino is a variant of C, shares a lot of the same structure, but varies slightly in syntax

# C basics:

- Variables:
  - Sequence of bits stored in memory for later use.
  - Variables always have names and, **in C**, types
  - Names must be unique
- Functions:
  - A sequence of instructions
  - Functions have names and parameters and, **in C**, a return type
  - Function names must be unique in C (overloading not possible)
- Libraries:
  - Collections of useful functionalities coded by the creators of the language/other people (github is an example)
  - One can `#include` (import) a library and use its contents

# The C workflow



# The essence of any computer

Bit:

- Smallest piece of information that a processor can work with
- Each bit can only be 0 or 1
- A sequence of bits can represent a number in binary format or instructions for the processor
- Any program has to be converted at some point to a binary stream for the processor to interpret it
- 8 bits form one byte. Bits are stored in memory in bytes

Binary representation of an integer:

sign: + = 0   - = 1

	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	
→	1	0	0	0	0	1	0	1

= -5

# Variable types

Declaration:            `variable_type variable_name;`

Common types:

- **int** = integer variable (e.g. -1, 100, -54). Usually occupies 16 or 32 bits of memory
- **char** = character variable (e.g. 'a', 'z', ';'). Occupies 8 bits of memory or 1 byte
- **long** = integer variable, but occupies double the size of a normal integer. Can store bigger numbers
- **short** = integer type, occupies less bits than an int but more than a char (byte)
- **unsigned int/char/long** = can only store positive values. Use for bytes or very big positive numbers
- **float** = single precision floating point number (e.g. 1.56, -4653.235, etc...)
- **double** = double precision floating point number. Usually occupies double the space of a float
- **arrays**: `int[], char[], float[], etc...` = sequence of **ints**, **chars**, etc... The length must be specified in the square brackets upon declaring the variable
- **Multidimensional arrays** (matrix) `int[][]`, `char[][]`, etc...

**Important:** a sequence of characters is a string only if it ends with the character **'/0'**, known as terminator.

Arduino only variable types:

- **String** = similar to an array of char in C, but more flexible. Can be slow to process
- **Byte** = same as unsigned char
- **Bool** = boolean variable, can only be True or False



# First Program - 'Hello World'

Launch an online C compiler: [Programiz](#) or other

```
/*  
    Multi-line comments  
*/  
#include <stdio.h>  
int main() {  
    printf("Hello, World!\n"); // single line comment  
    return 0;  
}
```

# #define

- Used to define constants
- Precompiler substitutes constant name for its value
- Very common in Arduino programming
- **#defines** come **after** **#includes**

Example:

```
#include <stdio.h>
```

```
#define N 4
```

```
int i=N;
```

```
void main(){
```

```
    printf("%d\n", i);
```

```
}
```

# Interacting with the user: scanf

```
#include <stdio.h>

char name[40];

int main() {
    printf("What's your name?"); // single line comment
    scanf("%s", name);
    printf("Hi %s, nice to meet you!\n", name);
    return 0;
}
```

# if ... else

```
if (condition){  
    ...  
}  
else if(condition){  
    ...  
}  
else{  
    ...  
}
```

Equalities: $A==b$ , $A!=b$ , $A<>b$	AND: $(a<b) \ \&\& \ (c<d)$
Inequalities: $A < b$ , $A \leq b$ , $A \geq b$	OR: $(a<b) \    \ (c<d)$

When using complex conditions, use parentheses to group them correctly

# Functions

Return type

Function name

Parameters and parameter types



```
void name(int number1, char param2){
```

```
.....    Body of the  
          function
```

```
}
```

Notes on function names:

- must be unique and without spaces or special characters/symbols
- must start with a letter
- cannot be keywords (example "int")

Notes on return types:

- If a function has return type 'void', it does not return any value
- If a function has a return type (e.g. int, float, char), then it needs to have a return (e.g. return 0;, return x;)

In general:

- Functions can call other functions
- A function can call itself (recursion)

# Functions (continued)

```
void name(int a);
```

```
int main(){  
    \\ main body here  
    name(3); \\ calling the function  
}
```

```
void name (int a){  
    \\ function body  
}
```

## *Exercise:*

Try coding a script that uses scanf to acquire 2 numbers "x" and "y" and returns the greater of the two

- Use functions
- Use if else statements

When you are finished, try acquiring 4 numbers and using more complex conditions (e.g. input 1 greater than input 3 AND input 2 greater than 4 OR the sum of inputs 1 and 3 is greater than input 2 and 4)

# Solutions:

```
#include <stdio.h>

int max(int x, int y);

void main(){
    int input1, input2, max_val;

    printf("Input first value:\n");
    scanf("%d", &input1);
    printf("Input second value:\n");
    scanf("%d", &input2);
    max_val = max(input1, input2);
    printf("%d", max_val);
}

int max(int x, int y){
    if(x>=y){
        return x;
    }
    else{
        return y;
    }
}
```

```
#include <stdio.h>

char fun1(float x, float y, float q, float z);

void main(){
    float input1, input2, input3, input4;
    char result;

    printf("Input first value:\n");
    scanf("%f", &input1);
    printf("Input second value:\n");
    scanf("%f", &input2);
    printf("Input second value:\n");
    ...
    result = fun1(input1, input2, input3, input4);
    printf("%c", result);
}

char fun1(float x, float y, float q, float z){
    if((x>=y && q>=z) || (x+q>=y+z)){
        return 't';
    }
    else{
        return 'f';
    }
}
```



# Loops:

- Two main types of loops:
  - For
  - While (do{...} while(...))

```
for(int i=0; i<5; i++){
```

```
    ...
```

```
}
```

```
while(x>q){
```

```
    ...
```

```
    x--;
```

```
}
```

## *Exercise:*

Write a program that sums up all of the odd numbers from 0 to 16.

- Use a for loop or a while loop

## *Solutions:*

```
#include <stdio.h>

void main(void){
    int res;
    for(int i=1; i<16; i+=2){
        res+=i;
    }
    printf("The total is: %d\n", res);
}
```

```
#include <stdio.h>

void main(void){
    int res;
    for(int i=0; i<16; i++){
        if(i%2!=0){
            res+=i;
        }
    }
    printf("The total is: %d\n", res);
}
```

# Arrays

Declaration:

```
int myArray[10]; //Array initialised at 0 by default
```

Initialisation:

```
int myArray[]={0,1,2,3,4,5,6,7,8,9};
```

or

```
myArray[0]=0;
```

```
myArray[1]=1;
```

```
...
```

```
myArray[9]=9;
```

You can sometimes use for  
or while loops to achieve  
the same

**NOTE:**

Arrays in C start at index **0**

## *Exercise:*

Write a program to find the min and max values in an array

- Define and initialize the array
- Use auxiliary variables
- Use a for or while loop to accomplish the task
- Use a for loop to print the array
- Also print the min and max values in the form:
  - "Min val: ..."
  - "Max val: ..."

## *Solution:*

```
#include <stdio.h>
#define N 11

float v[]={3.4, 5.5, -3.5, 9.6, 29.8, -40.3, 12.1, 0.0, 1.2, -2.0, 15.4};
float min_val, max_val;

void main(void){
    min_val=v[0];
    max_val=v[0];
    for(int i=1; i<N; i++){
        if(v[i]>max_val){
            max_val=v[i];
        }
        if(v[i]<min_val){
            min_val=v[i];
        }
    }
    for(int i=0; i<N; i++){
        printf("%.2f ", v[i]);
    }
    printf("\nMin val: %.2f\n", min_val);
    printf("Max val: %.2f\n", max_val);
}
```

## *Exercise:*

Write a program that sorts an array from smallest to largest

- Use a while loop to store the numbers a user is inputting in an array. Stop either when you got 10 numbers or when the user inputs a letter
- Use a for or while loop to sort the array in ascending order.
- Print the unsorted and sorted array at the end using for loops

When you're finished, have a look for more efficient sorting algorithms.

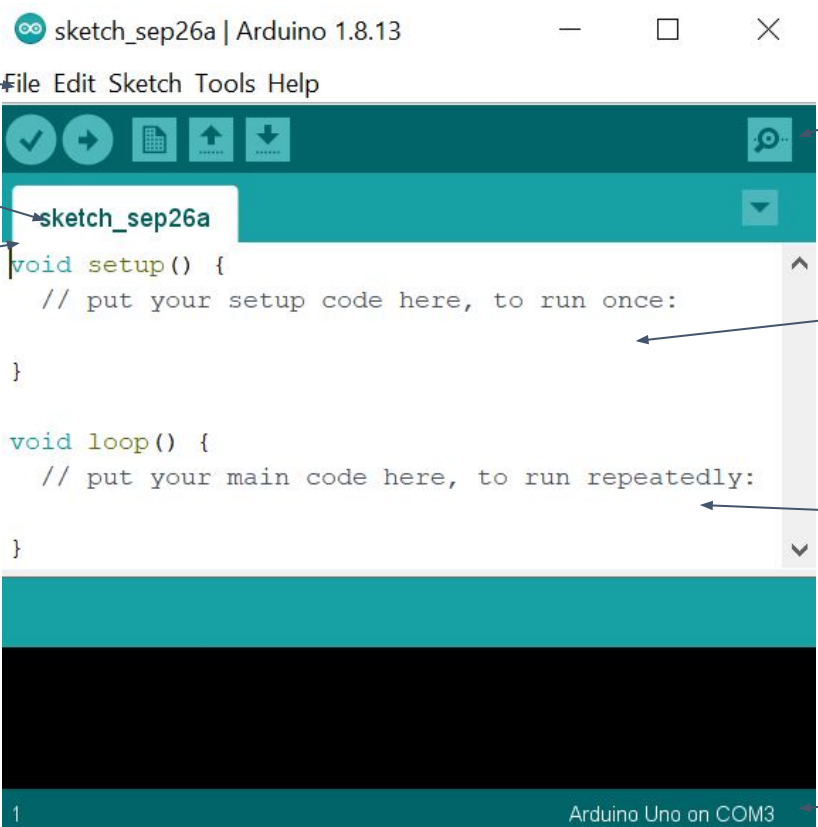
# What is Arduino?

- Open-source electronics platform based on easy-to-use hardware and software
- Powerful tool for students, hobbyists, enthusiasts and makers to begin tinkering and make low-cost models
- Has the following key advantages
  - Inexpensive
  - Cross-platform (Windows, OS, Linux)
  - Simple, clear programming environment
  - Open source and extensible software
- Download Arduino IDE from [here](#)

Ref: <https://docs.arduino.cc/foundations/basics/whats-arduino>



# Arduino IDE overview



The screenshot shows the Arduino IDE interface with the following components and annotations:

- Menu bar:** File Edit Sketch Tools Help
- Sketch name:** sketch\_sep26a
- Code editor:** Contains the following code:

```
void setup() {  
    // put your setup code here, to run once:  
  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
  
}
```
- Serial Monitor:** Indicated by a button in the top right corner.
- Board and Port:** Indicates board type and port (Arduino Uno on COM3)

Annotations and their locations:

- Menu bar
- Sketch name
- Add any #include and #define here, and long comments to explain code
- Shortcuts (including to compile and for serial monitor)
- Add code to be run once here
- Add code to be looped through continuously here
- Indicates board type and port

# void setup() vs void loop()?

## setup() function

- called when a sketch starts
- Used to initialise variables, pin modes, start using libraries, etc
- Runs only once after each power-up or reset

e.g.

```
void setup(){  
    Serial.begin(9600);  
    pinMode(buttonPin, INPUT);  
}
```

## loop() function

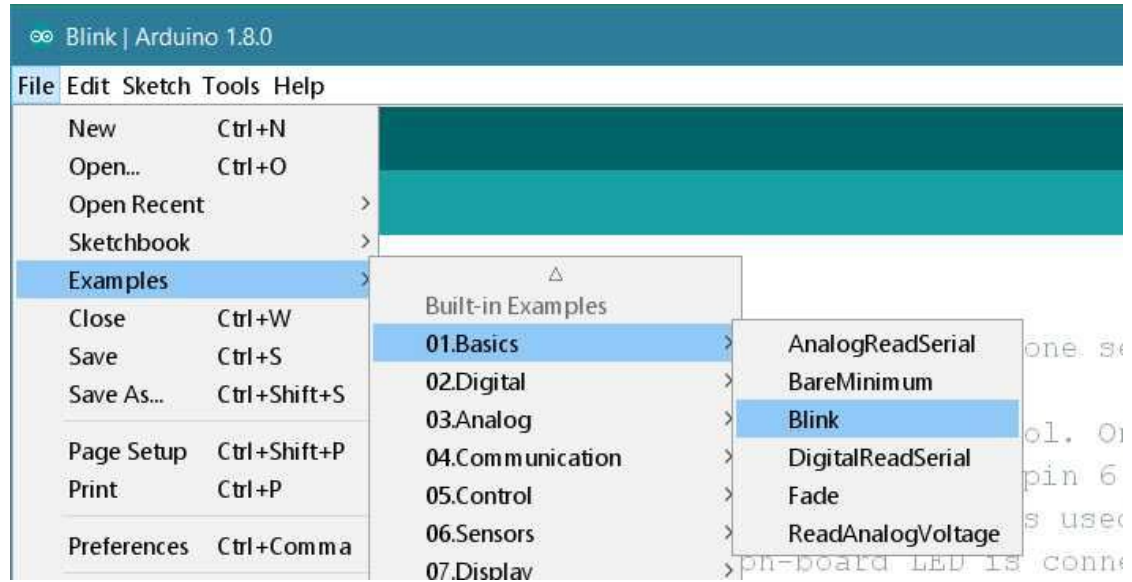
- Comes after the setup() function
- Loops consecutively
- Use it to actively control the Arduino board (change and respond)

e.g.

```
void loop(){  
    if (digitalRead(buttonPin)== HIGH){  
        Serial.write('H');  
    }  
    else {  
        Serial.write('L');  
    }  
}
```

# Example codes

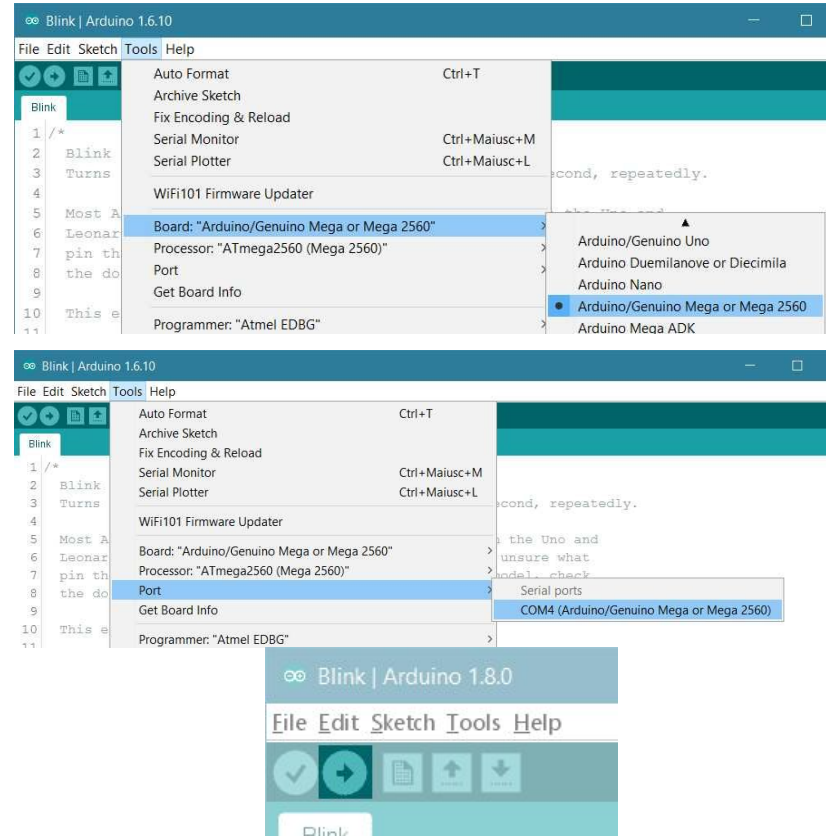
- Arduino has several example sketches that you can use
- Useful starting point to understand how to use various, sensors (e.g. buttons, piezo sensors, etc) and actuator (e.g. servos, steppers, etc)



# Compiling and running code

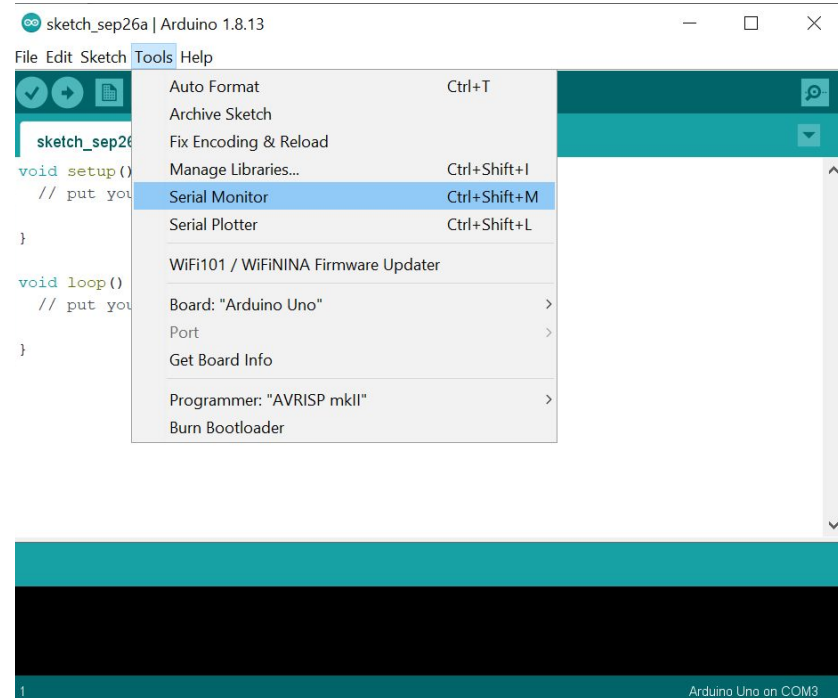
Key steps to upload code successfully

1. Select board type
2. Select port (e.g. COM5).  
Note: you can disconnect and reconnect board to find out the serial port
3. Upload the program  
Note: when uploaded, the LED on the board should blink



# Serial monitor and serial plotter

- Serial plotter and monitor can be accessed through 'Tools' menu
- Plotter useful for monitoring sensor data or actuator commands real time
- Monitor useful for sending and displaying messages or sensor readings/actuator commands to be saved



# More Information

- For more information about C and Arduino please refer to the module's supplementary document (link [here](#))
- Contains additional links to programming tutorials and useful links to Arduino cheat sheets

Thank you for listening!!

Any questions?







# Programming with C

- Libraries <https://www.arduino.cc/en/guide/libraries>
- Symbols
- Commands <https://www.arduino.cc/reference/en/#structure>
- Variables <https://www.arduino.cc/reference/en/#variables>
- Functions <https://www.arduino.cc/reference/en/#functions>
- Decision making
- Loops
- Data structures
- Etc

# Variables and Data Types for Arduino

## Constants

HIGH | LOW

INPUT | OUTPUT | INPUT\_PULLUP

LED\_BUILTIN

true | false

Floating Point Constants

Integer Constants

## Conversion

(unsigned int)

(unsigned long)

byte()

char()

float()

int()

long()

word()

## 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

## Utilities

PROGMEM

sizeof()

# Example – Goodbye, world!

```
#include <stdio.h>
```

The second part of the code is the actual code which we are going to write. The first code which will run will always reside in the `main` function.

```
int main() {  
    ... our code goes here  
}
```

The `int` keyword indicates that the function `main` will return an integer - a simple number. The number which will be returned by the function indicates whether the program that we wrote worked correctly. If we want to say that our code was run successfully, we will return the number 0. A number greater than 0 will mean that the program that we wrote failed.

For this tutorial, we will return 0 to indicate that our program was successful:

```
return 0;
```

Notice that every line in C must end with a semicolon, so that the compiler knows that a new line has started.

Last but not least, we will need to call the function `printf` to print our sentence.

Code

[Run](#) [Reset](#) [Solution](#) [Help](#)

```
1 #include <stdio.h>  
2  
3 int main() {  
4     printf("Goodbye, World!");  
5     return 0;  
6 }
```

Output

Expected Output

Goodbye, World!

Powered by Sphere Engine™

[Binary trees](#)  
[Unions](#)  
[Pointer Arithmetics](#)  
[Function Pointers](#)  
[Bitmasks](#)  
[Contributing Tutorials](#)

Sponsors

 DigitalOcean

 Sphere online judge  
Let's code together!

[https://www.learn-c.org/en/Hello%2C\\_World%21](https://www.learn-c.org/en/Hello%2C_World%21)

# setup()

[Sketch]

## Description

The `setup()` function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The `setup()` function will only run once, after each powerup or reset of the Arduino board.

## Example Code

```
int buttonPin = 3;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

void loop() {
  // ...
}
```

<https://www.arduino.cc/reference/en/language/structure/sketch/setup/>

# loop()

[Sketch]

## Description

After creating a `setup()` function, which initializes and sets the initial values, the `loop()` function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

## Example Code

```
int buttonPin = 3;

// setup initializes serial and the button pin
void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

// loop checks the button pin each time,
// and will send serial if it is pressed
void loop() {
  if (digitalRead(buttonPin) == HIGH) {
    Serial.write('H');
  }
  else {
    Serial.write('L');
  }

  delay(1000);
}
```

<https://www.arduino.cc/reference/en/language/structure/sketch/loop/>

## Example Code

The main uses of curly braces are listed in the examples below.

### Functions

```
void myfunction(datatype argument) {  
  // any statement(s)  
}
```

Function type

Function name and arguments

### Loops

```
while (boolean expression) {  
  // any statement(s)  
}  
  
do {  
  // any statement(s)  
} while (boolean expression);  
  
for (initialisation; termination condition; incrementing expr) {  
  // any statement(s)  
}
```

### Conditional Statements

```
if (boolean expression) {  
  // any statement(s)  
}  
  
else if (boolean expression) {  
  // any statement(s)  
}  
else {  
  // any statement(s)  
}
```

<https://www.arduino.cc/reference/en/language/structure/further-syntax/curlybraces/>

## 'If/else' conditional statement

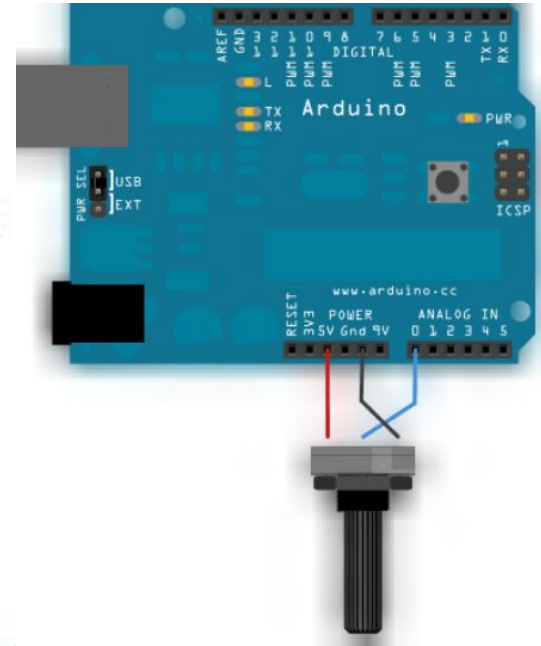
```
if (someCondition) {  
  // do stuff if the condition is true  
}
```

```
if (someCondition) {  
  // do stuff if the condition is true  
} else {  
  // do stuff if the condition is false  
}
```

```
if (someCondition) {  
  // do stuff if the condition is true  
} else if (anotherCondition) {  
  // do stuff only if the first condition is false  
  // and the second condition is true  
}
```

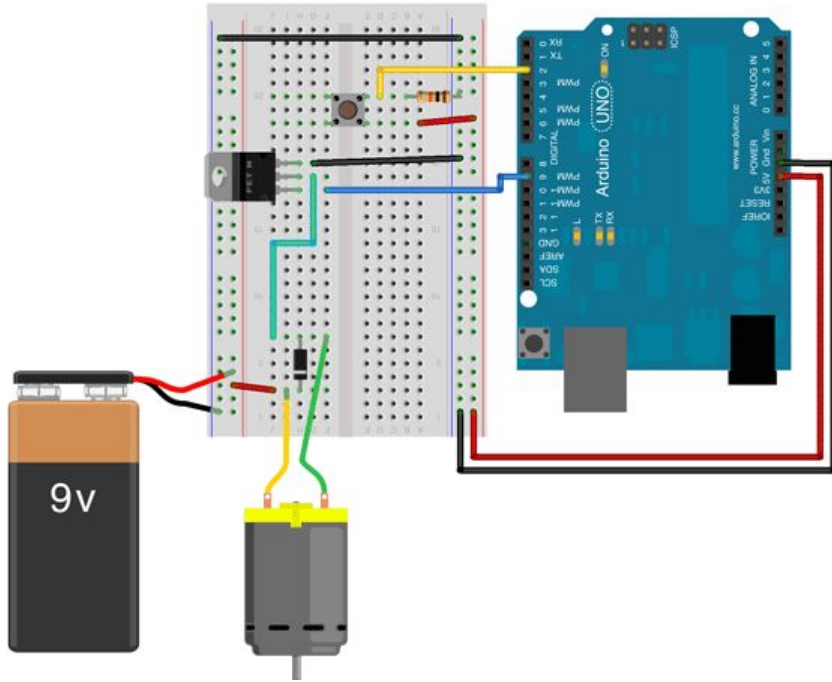
## Example of conditional statements

```
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  
}
```





## Example – Motor Driver



MotorDriver1

```
const int switchPin = 2;
const int motorPin = 9;
int switchState = 0;

void setup() {
  pinMode(switchPin, INPUT);
  pinMode(motorPin, OUTPUT);
}

void loop() {
  switchState = digitalRead(switchPin);
  if(switchState == HIGH)
  {
    digitalWrite(motorPin, HIGH);
  }
  else
  {
    digitalWrite(motorPin, LOW);
  }
}
```

## Example – 'For' Loop Structure

### Syntax

```
for (initialization; condition; increment) {  
    // statement(s);  
}
```

### Example Code

```
// Dim an LED using a PWM pin  
int PWMpin = 10; // LED in series with 470 ohm resistor on pin 10  
  
void setup() {  
    // no setup needed  
}  
  
void loop() {  
    for (int i = 0; i <= 255; i++) {  
        analogWrite(PWMpin, i);  
        delay(10);  
    }  
}
```

<https://www.arduino.cc/reference/en/language/structure/control-structure/for/>