ENG20009
Engineering Technology Inquiry Project

Week 1 Workshop – Getting started – basic programming

Swinburne University of Technology

Contents

Arduino

In order to program Arduino three main things are needed.

1. Text editor

   To write the human readable code

2. Compiler

   Compiler to convert the human readable code to the binary code. In other words Arduino understandable code

3. Programmer

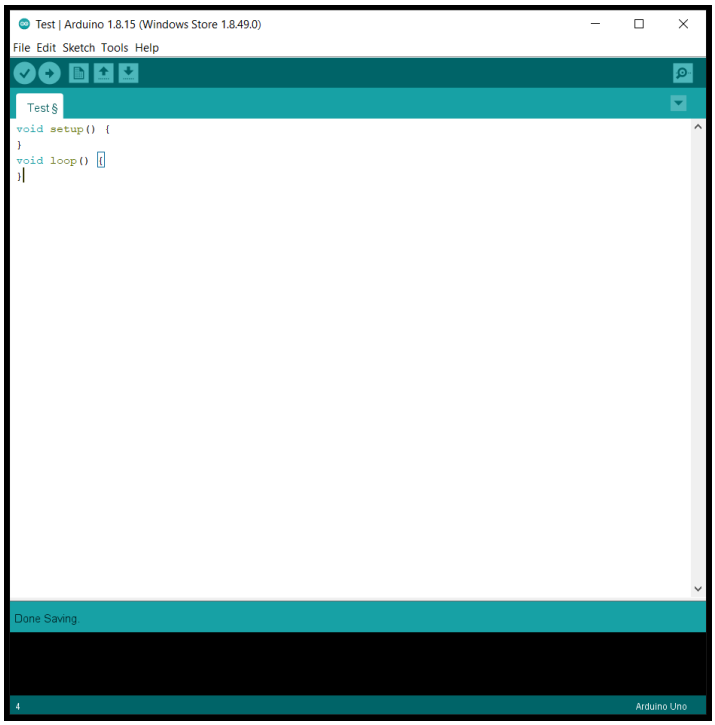
   Programmer is used to upload the binary code to Arduino

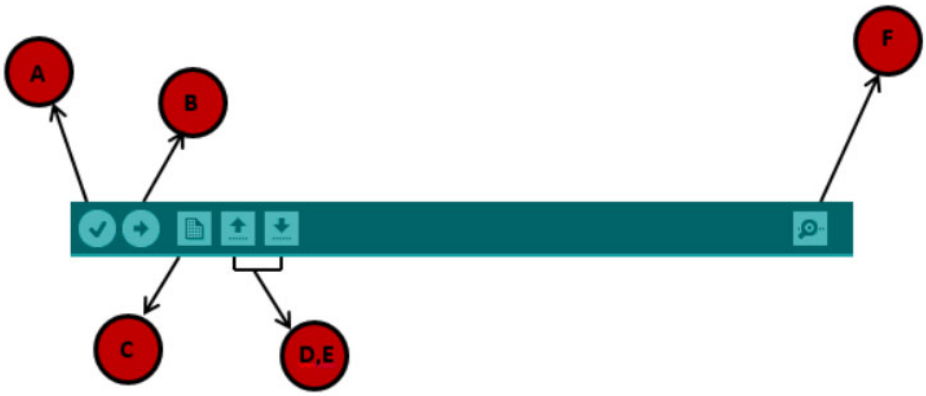
Text editor , compiler and the programmer is integrated in the Arduino software which is called Arduino IDE. Software will convert he human readable code to the binary code and upload it to the Arduino

Arduino IDE

Download the Arduino IDE via below link

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





A: Used to compile the program

B: Used to upload a program to Arduino board

C: Shortcut used to create a new sketch

D: Used to open one of the example sketch

E: Used to save your sketch

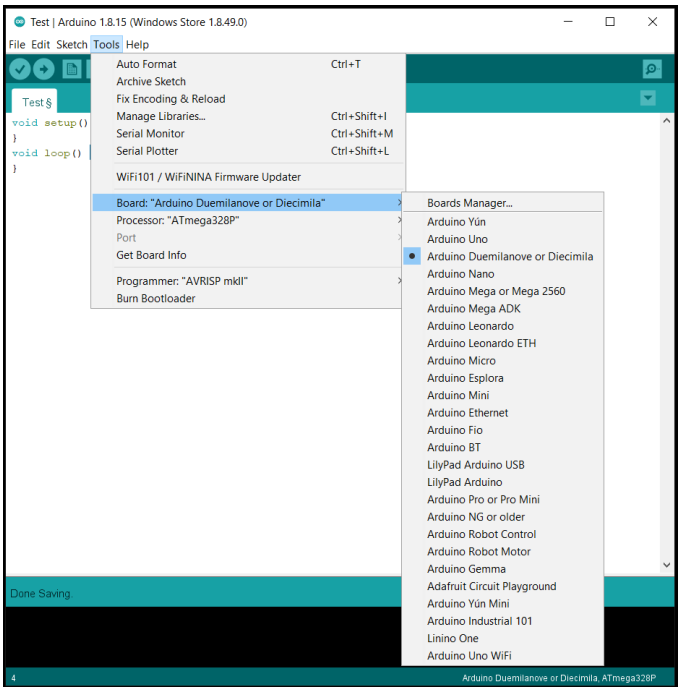
F: Serial monitor used to receive serial data from the board and send the serial data to the board

Connecting the Arduino board

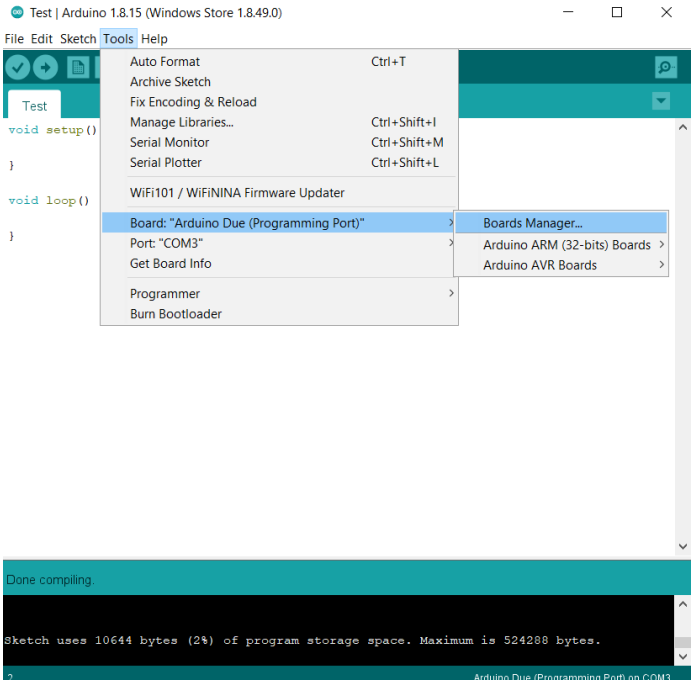
Step 1 : connect the Arduino to PC via USB cable

Step 2: Open Arduino IDE

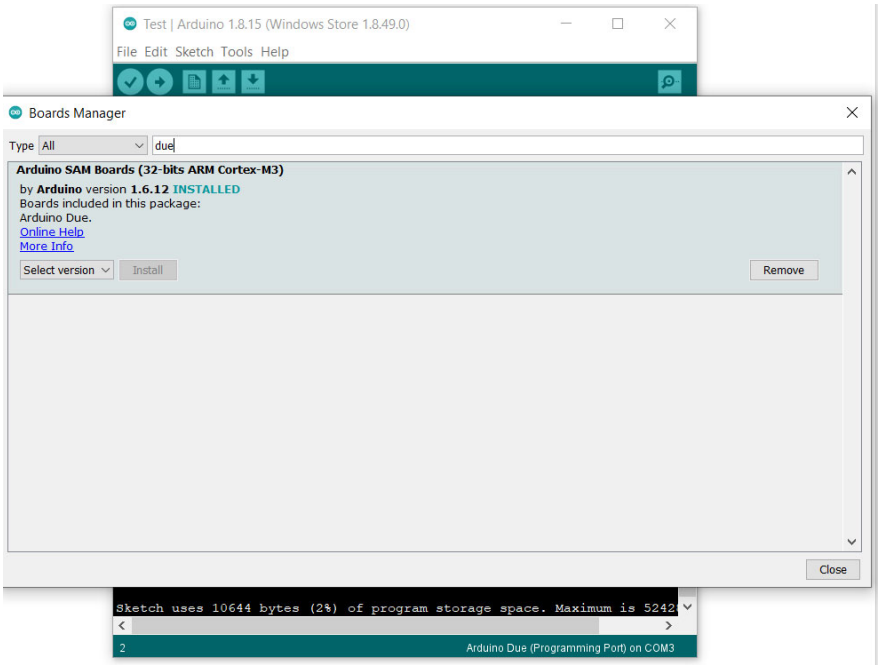
Step 3 : Select the Arduino board



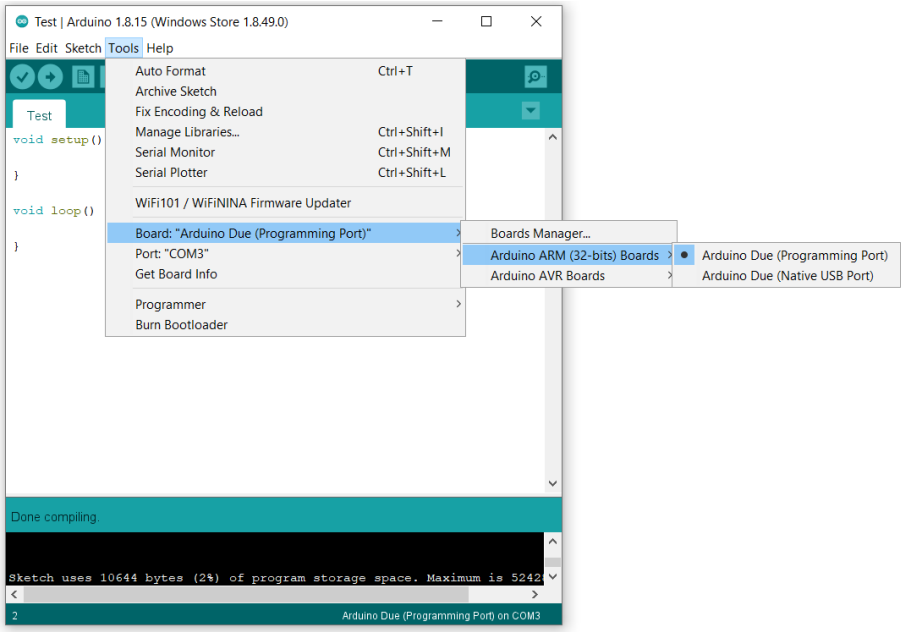
Step 3.1 If you cannot find the board go to board manager



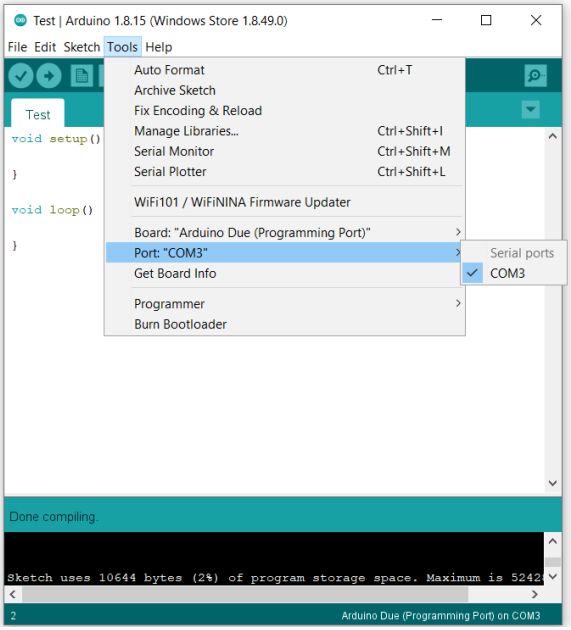
Step 3.2 Type due and search for the board. After that install the board.



Step 3.3 Select the board and the relevant port.



Step 4 : Select Arduino serial port (Depending on the PC number changes)

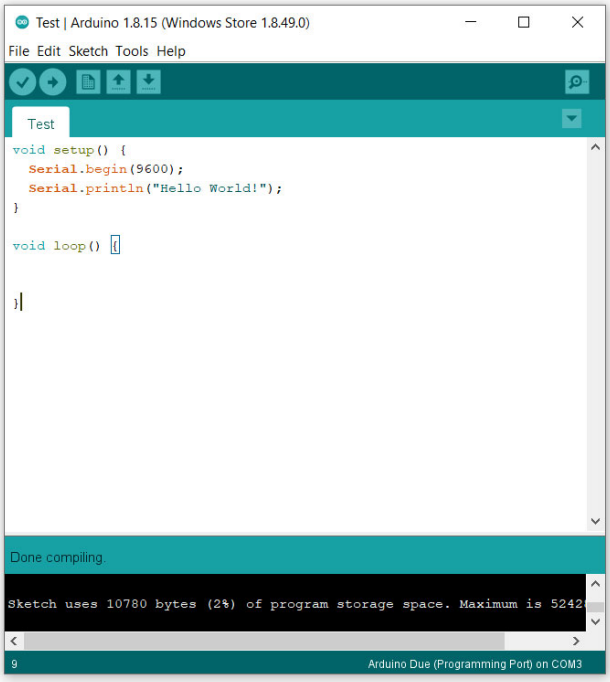


Step 5: Write the code and upload it

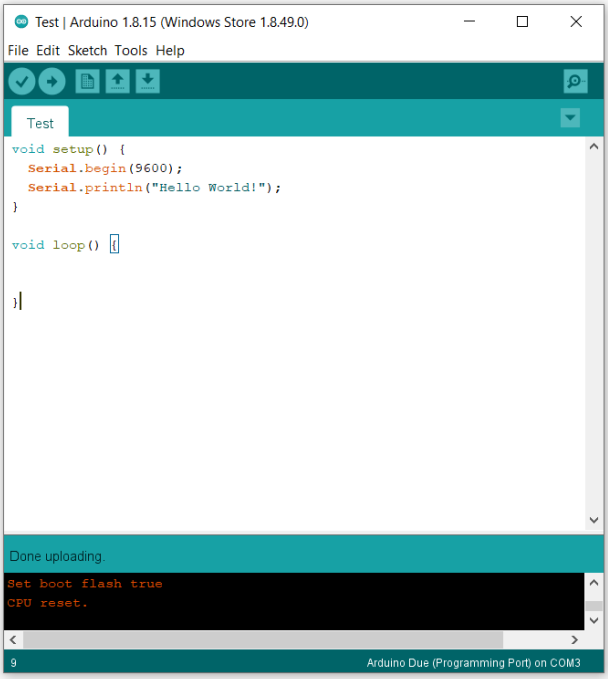
Example 1

Write a sample program to display Hello world in the serial monitor

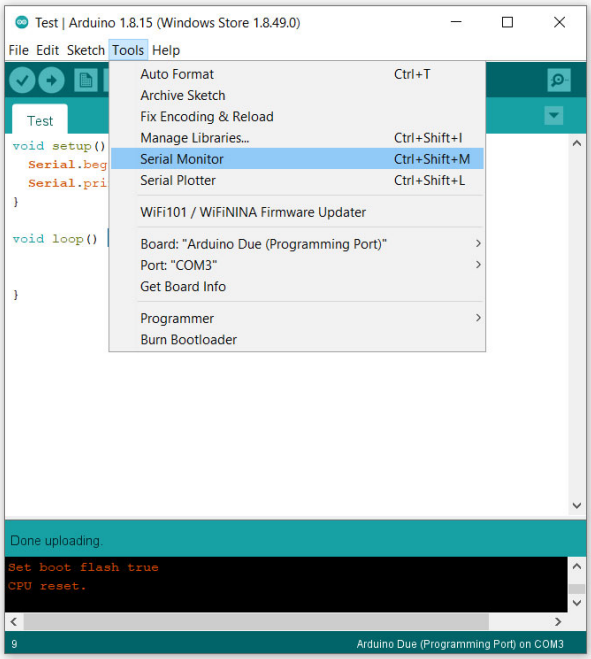
Type the fallowing code and Click compile button to compile the program

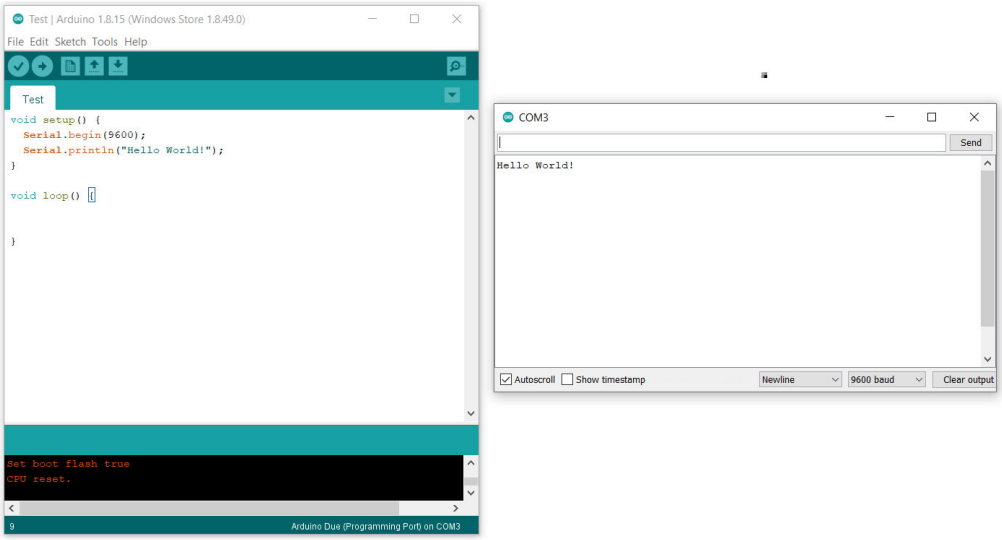


Click **upload** button after writing the code to upload the code to Arduino



Open serial monitor by clicking on the serial monitor icon





Arduino Programming

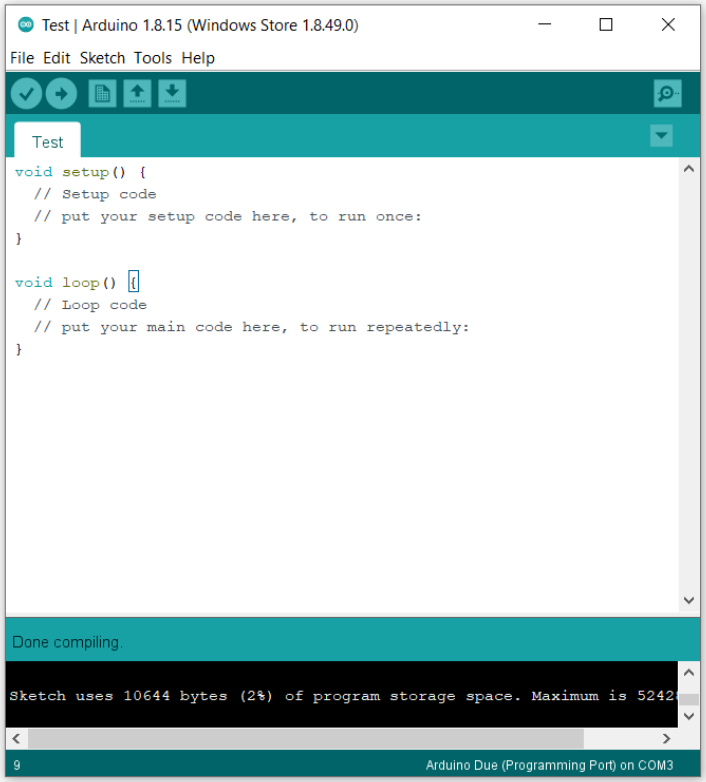
Arduino program can be divide in to three main parts

1. Structure
2. Values (constants and variables)
3. Functions

Structure

Arduino code structure includes two main parts

1. Setup code
2. Loop code



Setup code

- This part of the code is executed right after a power up or reset.
- Setup code will only execute one time
- Main purpose of the setup code is to initialize variables, pin modes, start using libraries

Loop code

- This part of the code is in a loop function
- This part is executed right after setup code
- Code will be executed repeatedly (infinitely)
- Main purpose of Loop code is to do the main task of application

Data types

Type of the variable determine how much space that the variable occupies in the storage and how the bit pattern is stored.

Void

Key word void is only used in function declaration. It indicates that the function is expected to return no information to the function from which it was called.

```
void setup()
{
}
```

Boolean

Boolean holds two values which is either true or false. One byte of memory is occupied by each Boolean variable.

```
boolean value = false ; // declared Boolean variable is initialized as false
boolean state = true ; // declared Boolean variable is initialized as true
```

Char

Data type that uses one byte of storage memory to store a character. For a character single quotes are used (eg: 'A'). where as for a strings or multiple characters double quotes are used (eg: "ENG") These characters are saved as numbers in the storage. For the encoding information refer to ASCII charts.

```
Char chr_a = 'a' ;//declared varaible type char  is initialized as a
Char chr_b = 55 ;//declared varaible type char  is initialized as 55
```

Unsigned Char

This unsigned data type occupies one byte of storage. Unsigned char data type encodes numbers from 0-255.

```
Unsigned Char chr_y = 150 ; //declared unsigned variable type char, is
initialized as 150
```

Byte

8 bit unsigned number is used to store byte data.

```
byte a = 50 ;// declared variable type byte, is initialized as 50
```

Int

In order to save the numbers mostly Integer data type is used. When a variable is declared as int, it uses 16 bits (2 bytes) of data for the storage. This yields the range of -32768 to 32767. Size of the int varies from the type of the board. For example in Arduino Due int uses 4 bytes for the storage.

```
int count = 32 ; //declared variable type int, is initialized as 32
```

Unsigned Int

Data storage of int and unsigned int is similar but they only stores the positive values. Which yields a range of 0 -65535. In Arduino Due unsigned int is stored in the 4 bytes of storage.

```
Unsigned int count = 500 ; // declared unsigned variable type int is
initialized as 500
```

Word

In Arduino Uno and ATMEGA based bords stores word as 16 bit unsigned number. On Due and Zero it is stored as 32 bit unsinged number.

```
word a = 700 ;// declared variable type word, is initialized as 700
```

Long

Long data type is an extended size variables for number storage. It stores in 32 bits of storage from -2147483648 to 2147483647.

```
Long a = 123467 ;// declared variable type long, is initialized as 1234567
```

Unsigned long

Unsigned long data are the similar to the long data type. However unsigned long does not save negative numbers making the range from 0 to 2^32-1.

```
Unsigned Long a = 111111 ;// declared variable type unsigned long, is initialized as 111111
```

Short

Short is a 16 bit data type which leads the range from 2^15 to 2^15 -1.

```
short a = 20 ;// declared variable type short, is initialized as 20
```

Float

For the numbers with decimal points Float data type is used. Analog and continuous values are stored as Float variables as they have better resolution than the integers. Float numbers are saved in 32 bits of memory in the storage and the range varies from -3.4028235E+38 to 3.4028235E+38.

```
float a = 1.234;// declared variable type Float, is initialized as 1.234
```

Double

On the Uno and ATMEGA based boards double variable uses 32 bits(4 bytes) of memory for storage. Which makes it similar to the Float data type. In Arduino Due Float variable uses 64 bits (8 byte) for the data storage.

```
double a =98.765;//declared variable type double is initialized as 98.765
```

Variables and Constants

In Arduino there are three places where the variables can be declared. They are

- Local variables
- Global variables
- Formal variables

Local variables :

Variables which are declared inside the function or a block is called as local variables. These variables can only used inside the declared function or the block. Local variables are not known to the outside of the function

```
void setup()
{
}
void loop()
{
int a; // local variable declaration
a = 100; // initialization.
}
```

Global variables :

variables which are declared outside of all the function is called global variables. Usually global variables are declared at the top of the program. These variables will hold the values through out the life time of the program. Furthermore these variables can be accessed by the all the functions and the blocks in the program after declaration.

```
float b = 0 ; //Global variable declaration and initialization
void setup()
{
}
void loop()
{
int a; // local variable declaration
a = 100; // initialization
c=a+b;

}
```

Formal variables :

variables which are in function parameters are called the formal variables

Functions

Functions contain a block of code that can accomplish a specific task. Functions normally take in data and process it and return the results. Functions can be called over and over again in the program. Functions can be called inside a function. Typical case of using function is when the same action is needed to perform in multiple times in the program. There are two required functions in Arduino, Setup () and loop () function. Other functions should create outside these two functions. Syntax for functions in Arduino is as follows.

```
return type function name (argument1,argument2,...)
{
  Statements
```

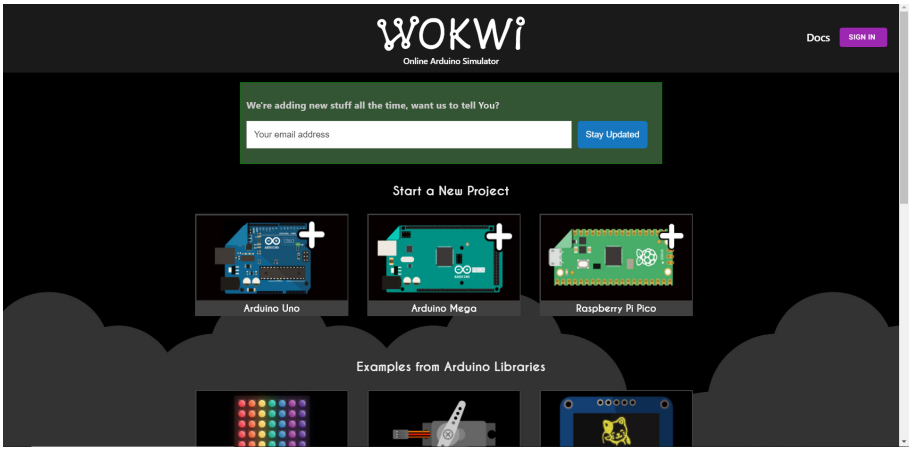
}

- Return type : Type of the value returned by the function can be any c data type
- Function Name : This is the identifier which the function can be called
- Arguments: parameters passed to the function. Can be any c data type
- Statements: statements or the function body

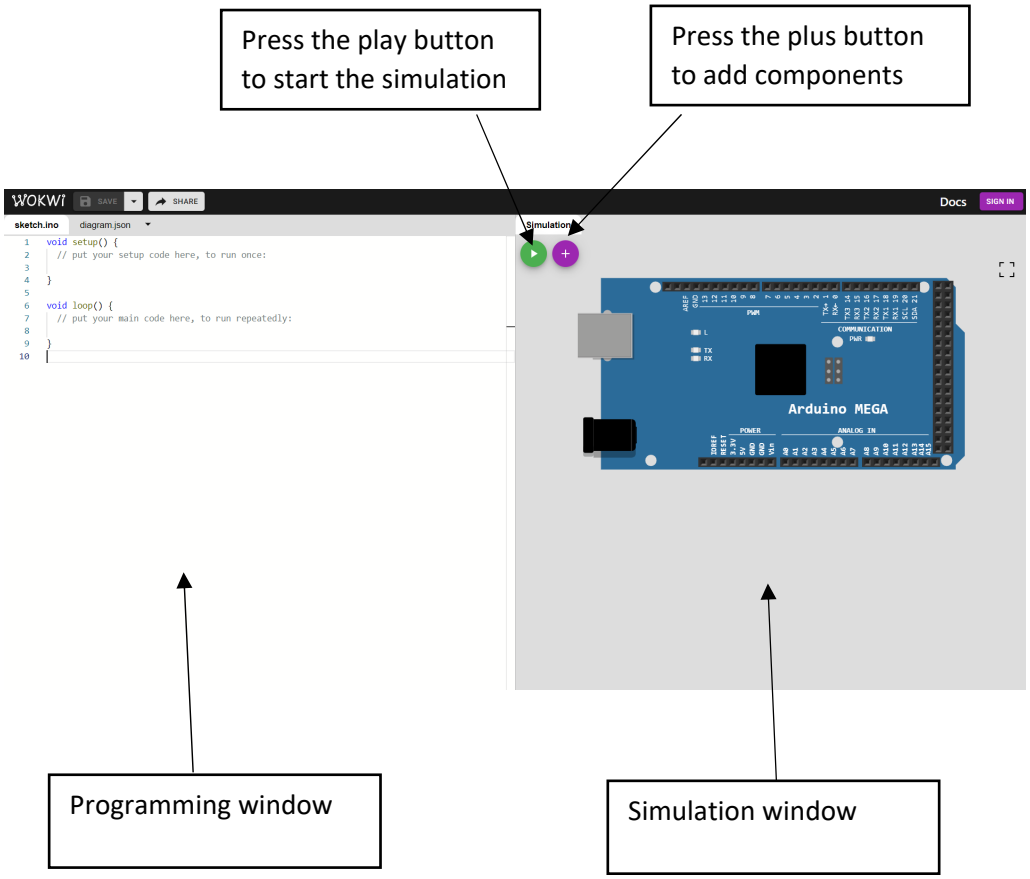
Introduction to Arduino Simulator

In order to access the Arduino simulator use the fallowing link.
https://wokwi.com/

Above link should lead to the fallowing webpage



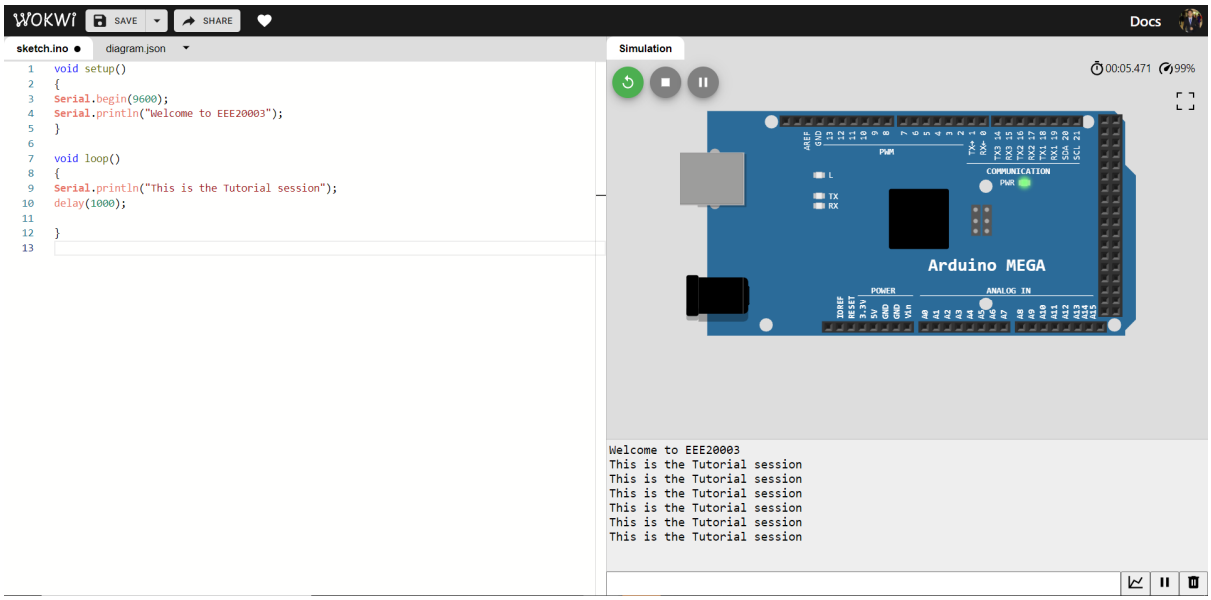
Select Arduino Mega tab in order to use the Arduino Mega for the simulation. After clicking the tab it should lead to the following webpage.



Example 2

Write a simple program to display ENG 2003 at the beginning of the application and display This is the Tutorial session every time the loop is executed

Type the fallowing code in the figure.

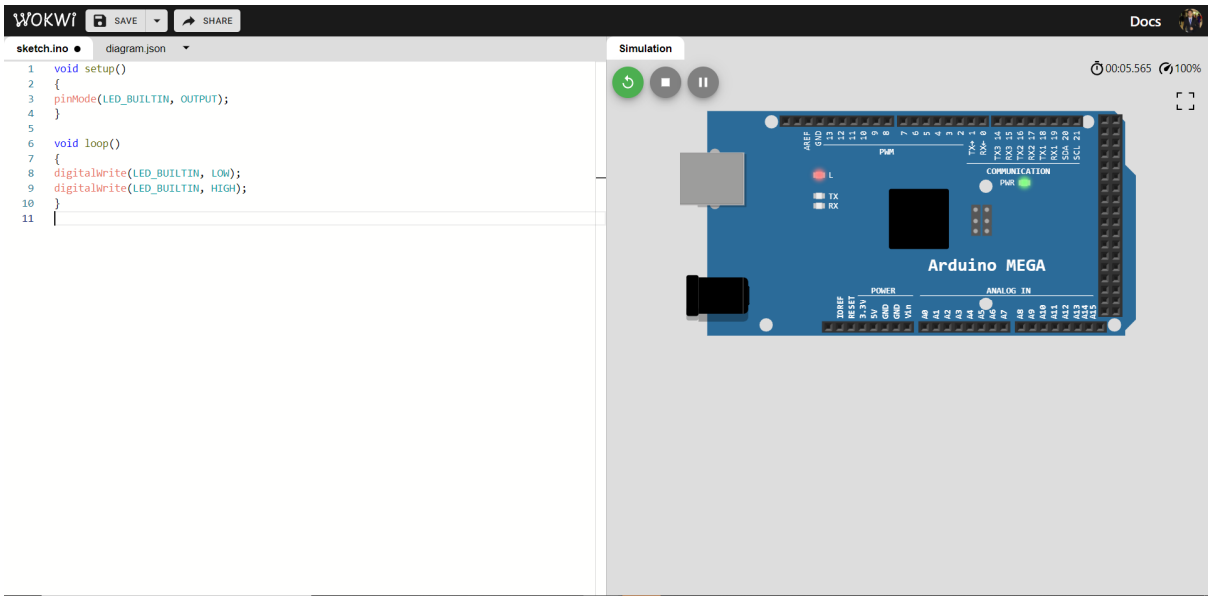


Other than the setup code and the loop code following parts can be included in the Arduino sketch

Comments : usually used to explain the parts of the codes Arduino will ignore this part.
Block comments : usually used to write the information about the author license writing **instructions**. Arduino will ignore this part.
Libraries inclusion: This is used to include the necessary libraries into the sketch. #include component can be used for the library inclusion.
Constant definition: This is used to define the constants
Global variable declaration: This is used to define the global variables
Curly braces: Open curly braces(brackets) must always followed by a closed curly braces. In other words braces should be balanced.
Semi Colon: used to end a statement
#define: #define is a useful C++ component that allows the programmer to give a name to a

Example 3

Write a simple program to switch on the inbuilt LED

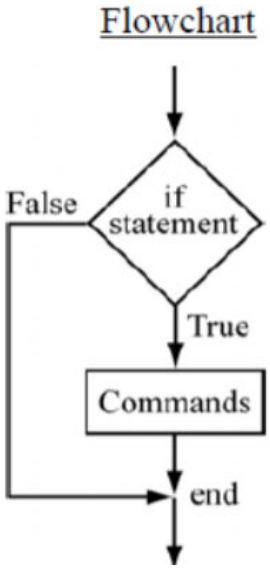


Quick overview on control structures and Loops

Control Structures

Decision making structures are required when the programmer need to run part or parts of the program depending on the inputs. If else statements and switch case statements are the most common control structures in Arduino programming.

If statements



If statement check the statement in the parentheses and according to the statement output relevant statement or the block statement will be executed. Syntax for the if statement as follows.

```
if (expression)
{
   Block of statements;
}
```

If it is a one statement braces {} are not needed.

If else statement

If statement followed by an optional else statement which will execute when the statement is false. Syntax for the if else statement as follows.

```
if (expression)
{
   Block of statements;
}
else {
```

```
    Block of statements;
}
```

If else if else statement

If statement can be followed by elseif and else statement. This is useful when multiple conditions are needed to check. Syntax for the if else if else statement as follows.

```
if (expression_1) {
   Block of statements;
}

else if(expression_2) {
   Block of statements;
}
.
.
.
else {
   Block of statements;
}
```

switch case statement

switch case statements are similar to the if statements and it controls the flow of the program by allowing programmers to execute different parts of the program depending on various conditions. Syntax for the switch case statement as follows.

```
switch (variable) {
   case label :
   // statements
   break;

case label:
   // statements
   break;


default:
   // statements
```

```
            break;

        }
```

Loops

Loop statements allows the programmer to execute a statement or a block statement multiple times. while loops , do while loops, for loops, nested loops and infinite loops are the most common loops used in Arduino programming

while loop

while loop will loop continuously until the statement inside the parenthesis becomes false. Syntax for the while loop as follows.

```
while(expression) {
   Block of statements;
}
```

do while loop

do while loop is similar to the while loop but here the loop continuation condition is tested at the beginning of the loop before performing the body of the loop. Syntax for the do while loop as follows.

```
do {
   Block of statements;
}
while (expression);
```

for loop

for loop will execute the statements for predetermined number of times. Syntax for the for loop as follows.

```
for ( initialize; control; increment or decrement) {
   // statement block
}
```

Nested loops

Arduino allows the programmer to have loop inside another loop and it is called as nested loops. for loop syntax for a nested loop as follows.

```
for ( initialize ;control; increment or decrement) {
   // statement block
   for ( initialize ;control; increment or decrement) {
      // statement block
   }
}
```

Infinite loops

When there is no termination condition in the loop, loop becomes an infinite loop. Syntax for the infinite loop as follows.

```
for (;;) {
   // statement block
}

while(1) {
   // statement block
}
```