

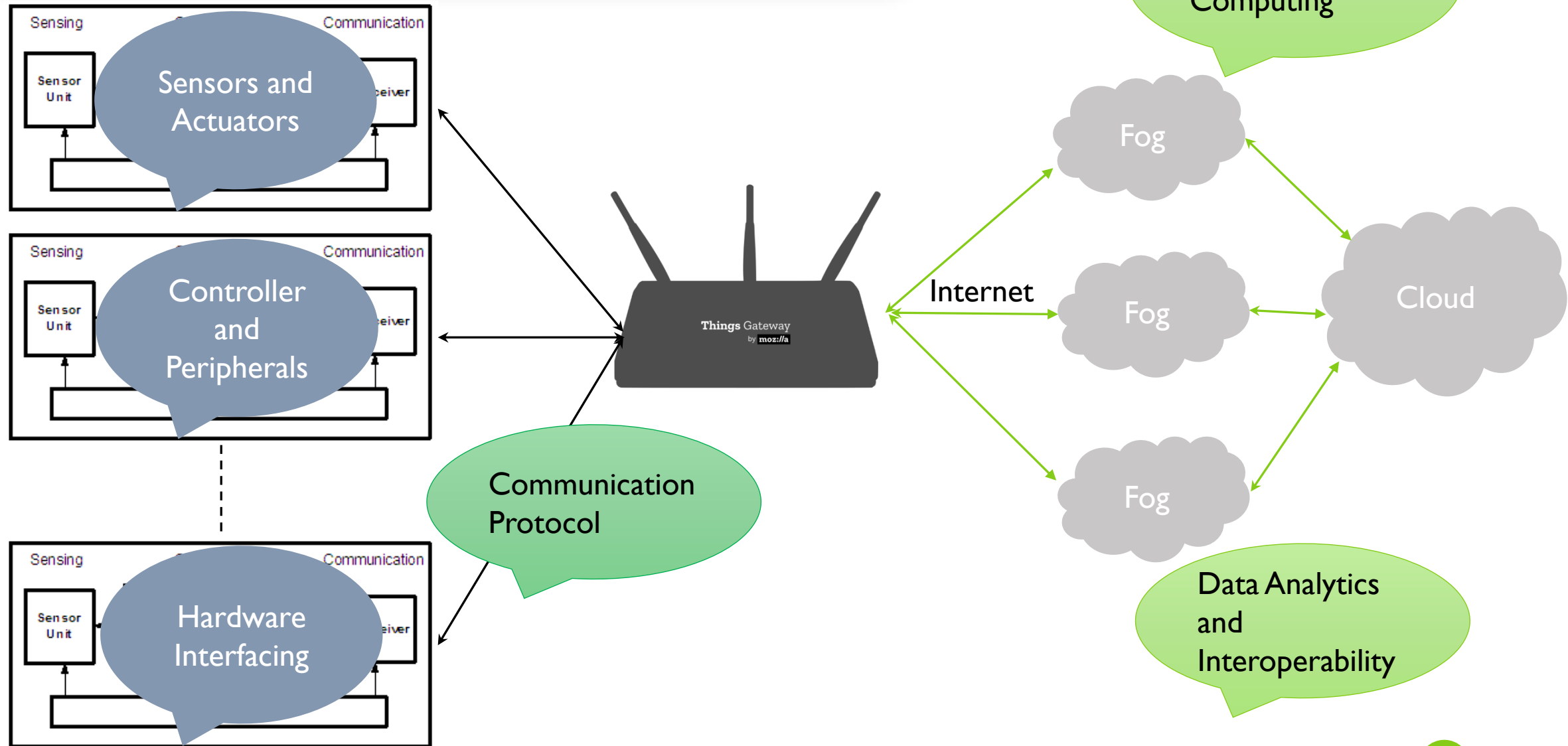
# Arduino Programming

Instructor: Deepak Gangadharan

# Outline

- Arduino IDE
- Arduino Programming

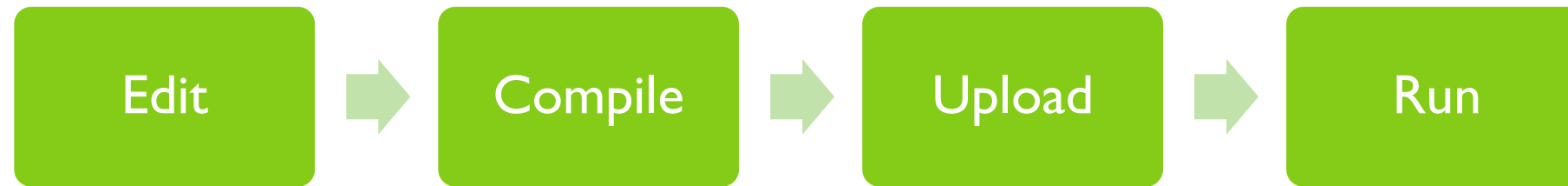
# IoT Architecture



# Arduino Programming

# Preliminaries

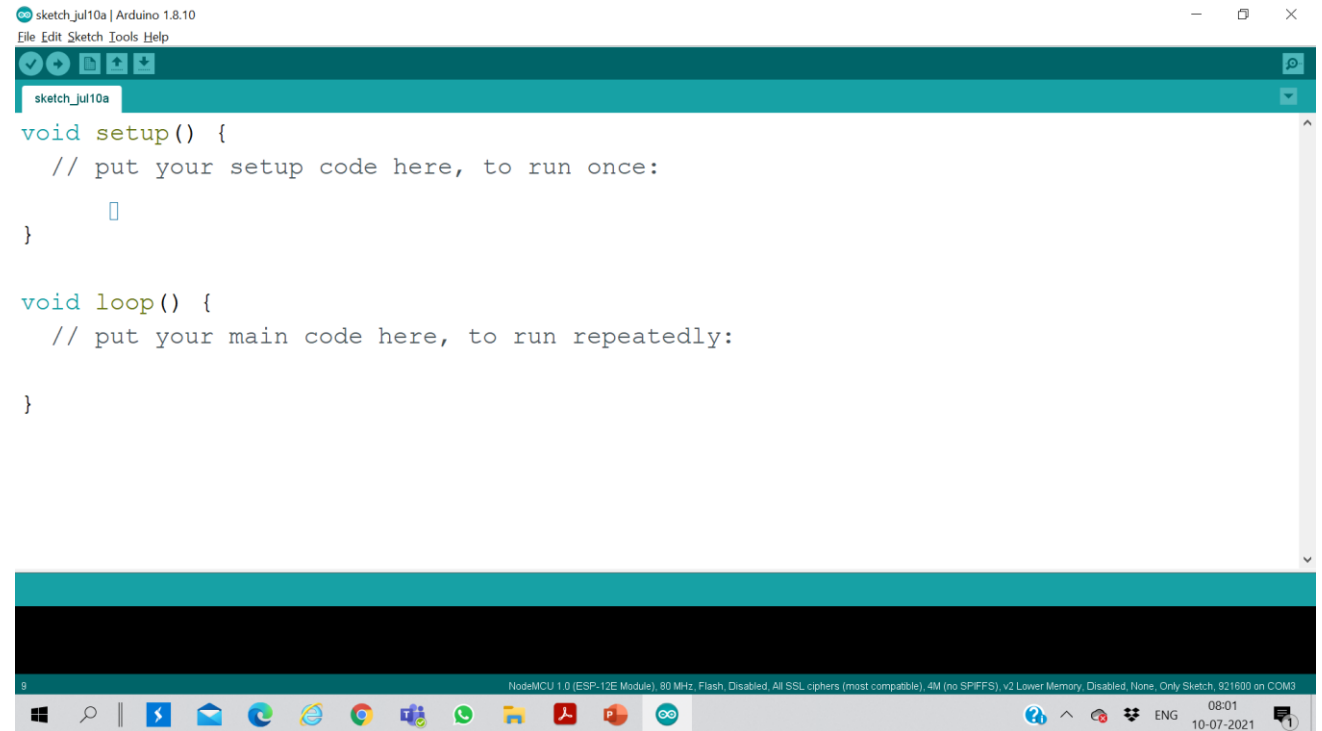
- Written in a programming language similar to C and C++
- Development cycle consists of 4 phases



- Compilation → Translates the sketch to object code
- Run → Sketch is executed as soon as upload finishes

# Basic Structure

- Each sketch has 2 blocks
  - setup – Preparation block
  - loop – Execution block
- Set of statements in setup and loop enclosed within curly braces



The screenshot shows the Arduino IDE interface. The title bar reads "sketch\_jul10a | Arduino 1.8.10". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for opening, saving, and running. The main text area shows the following code:

```
sketch_jul10a
void setup() {
  // put your setup code here, to run once:
  []
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

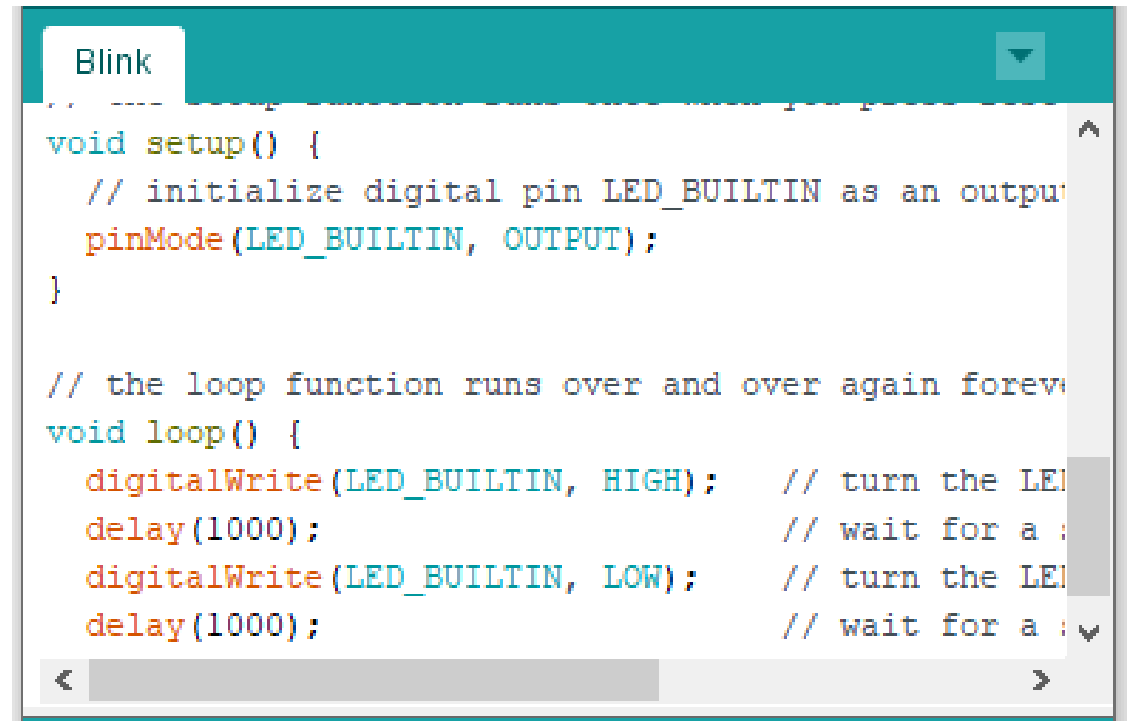
At the bottom, a status bar displays hardware information: "NodeMCU 1.0 (ESP-12E Module), 80 MHz, Flash, Disabled, All SSL ciphers (most compatible), 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 921600 on COM3". The Windows taskbar at the very bottom shows the time as 08:01 on 10-07-2021.

# setup and loop block

- setup
  - Initial segment of the code executed
  - Executed only once during upload/ after reset or power up
  - Initializes pin modes, libraries, variables, etc.
- loop
  - section of code executed repeatedly

# Example 1

- pinMode()
  - Syntax is pinMode(pin, mode)
  - pin refers to the pin number on the board
  - mode set as INPUT or OUTPUT
- Example:
  - pinMode(10, OUTPUT) → Setting pin number 10 as output on the board
  - OUTPUT mode provides enough current to other circuits for e.g., to light a LED brightly
  - Can also use INPUT mode for e.g., in the case of a button press input

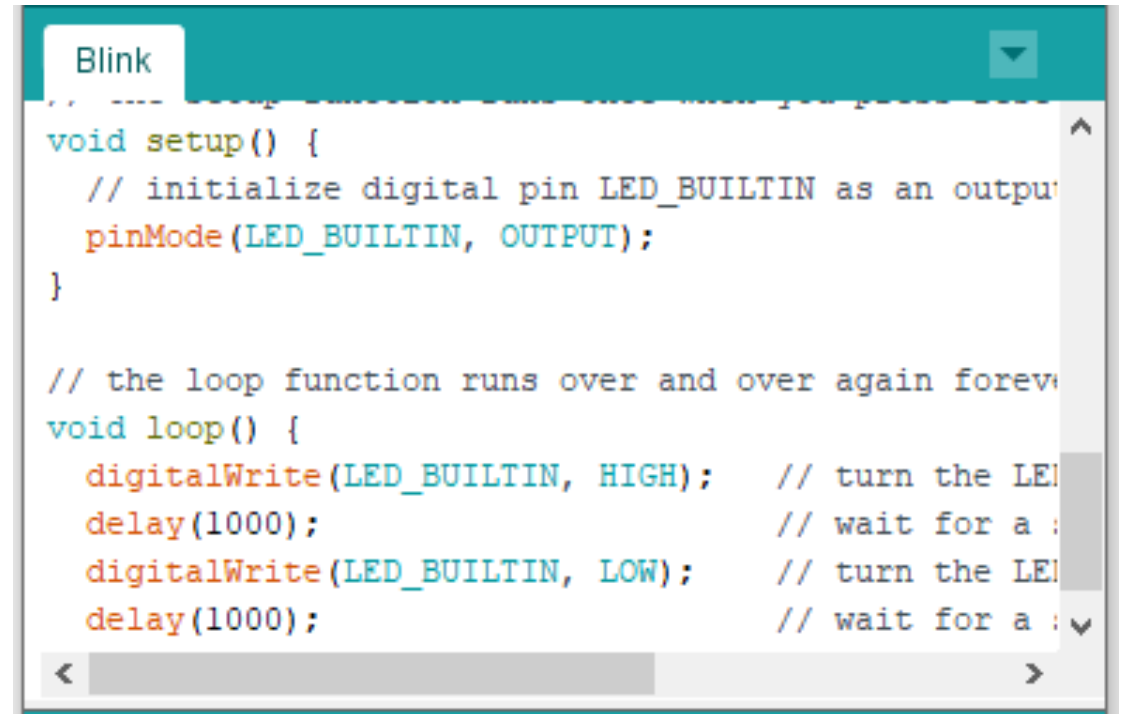
A screenshot of a code editor window titled 'Blink'. The code is written in C++ and is used to make an LED blink. It includes a setup function that initializes the built-in LED pin as an output, and a loop function that turns the LED on and off with a 1000ms delay between each state change.

```
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 positive voltage)  
  delay(1000);                       // wait for a second  
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the pin LOW (no voltage)  
  delay(1000);                       // wait for a second  
}
```



## Example 1 (contd...)

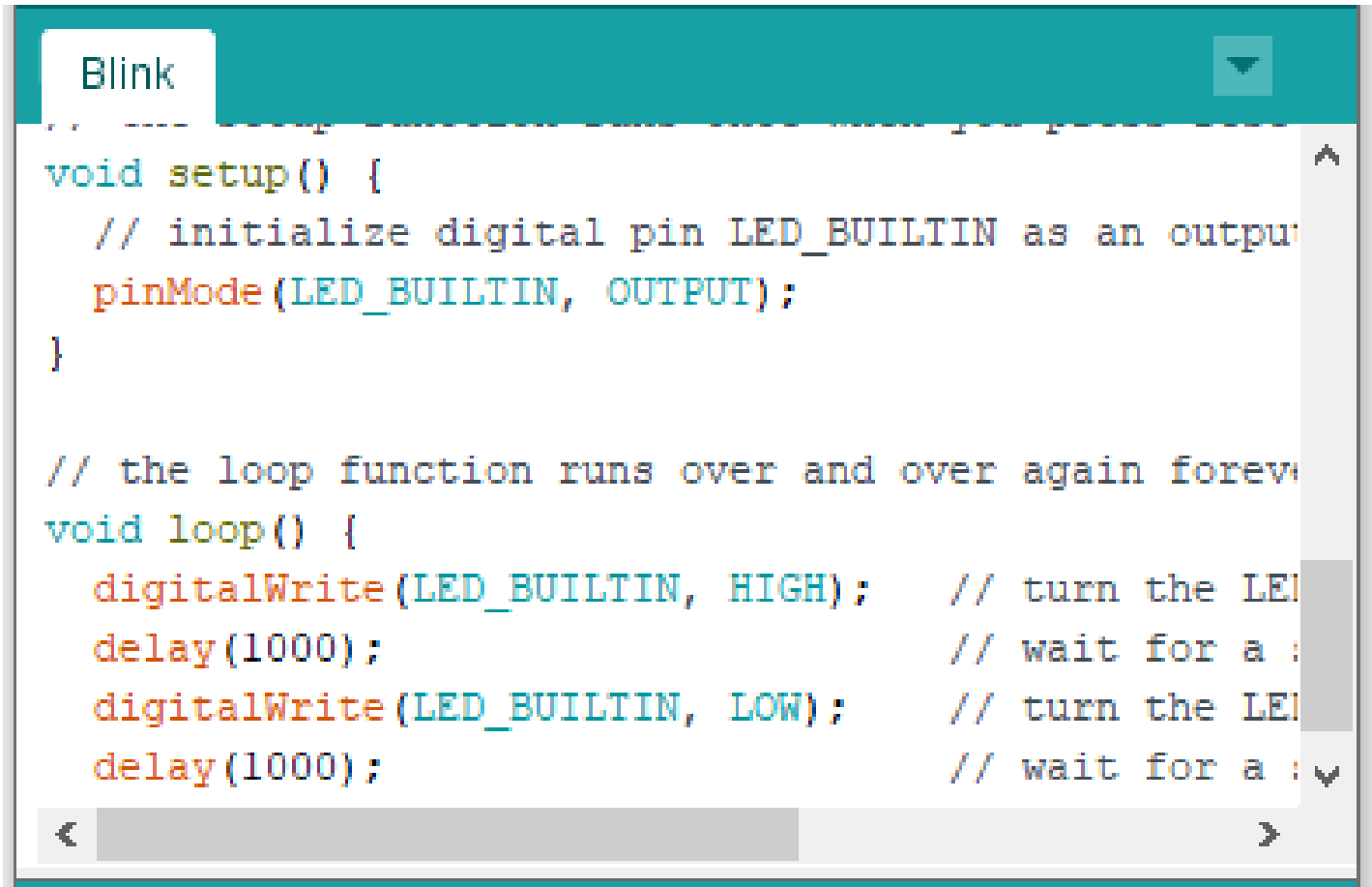
- `digitalWrite()`
  - Syntax is `digitalWrite(pin,value)`
  - `pin` → pin number or declared variable
  - Used to set a HIGH or LOW value at the pin
  - HIGH sets the value of the voltage
  - LOW sets the value to 0 or GND
- Example
  - `digitalWrite(12, HIGH)`
  - LED connected to pin 12 will turn ON.
- `digitalRead(pin)`
  - Used to read the value from a digital pin. For e.g., a sensor value

A screenshot of an IDE window titled "Blink". The code is as follows:

```
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  
}
```

## Example 1 (contd...)

- `delay()`
  - Syntax is `delay(<time>)`
  - A blocking function
  - `<time>` in milliseconds
- What is the code doing?

A screenshot of an IDE window titled "Blink". The code is written in C++ and is a standard Arduino sketch for blinking an LED. It includes a setup function to initialize the LED pin as an output and a loop function that turns the LED on and off with 1000ms delays. The code is color-coded: keywords in blue, comments in green, and function names in red. The IDE has a teal header bar and a scrollbar on the right.

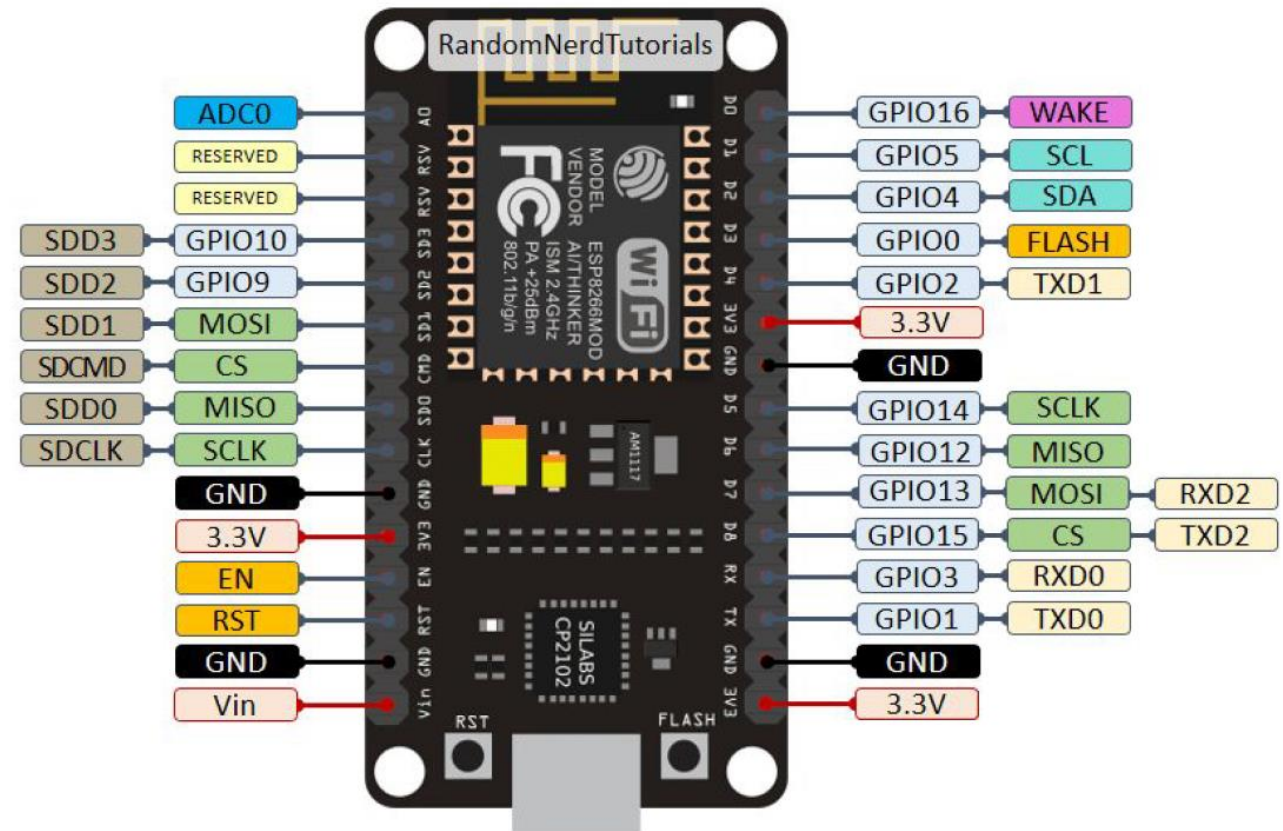
```
void setup() {  
    // initialize digital pin LED_BUILTIN as an output  
    pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
    digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)  
    delay(1000);   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

# Syntax and Program Flow

- setup and loop do not return any value
- Compiler
  - ignores spaces and tabs before code statements, in the parantheses
  - ignores commas, blank lines
- Semicolon ';' used as statement termination → Compiler error if a statement found without ';'
- Program Flow: Entry point is setup() and loop() runs repeatedly

# Serial Communication

- Simple communication scheme that uses UART
- Normally uses 5V for logic 1 and 0V for logic 0 → For 3.3V board uses 3V and 0V
- Messages sent to computer from GPIO I called Tx or Transmitter
- Messages sent to Arduino from computer received on GPIO3 called Rx or Receiver
- `Serial.begin()` → Performs initialization to send and receive data on Rx and Tx pins
- Example: `Serial.begin(14400)` → 14400 is the baud rate or bps.



# analogRead()

- Reads value from a specified analog pin
- ADC used to convert to discrete values
- Time required to read an analog signal on boards like UNO, Nano, etc, is 100 microseconds
- Syntax is `analogRead(pin)`

```
int a_pin = A1;
int value = 0;
void setup()
{
  Serial.begin(4800);
}
void loop()
{
  value = analogRead(a_pin);
  Serial.println(value);
}
```

# Other Program Constructs

- Functions, Data types, Operators, Arrays are very similar as in C/C++ programming languages
- If conditional statement and loops such as for, while are also same as in C/C++
- Switch case statement is also similar as in C/C++

# Arduino Interrupt

- interrupts() allow some essential tasks to run in the background
- It is enabled by default
- Disabling interrupts may impact some functions
- It may be necessary to disable interrupt in some parts of the code which are critical sections

# Arduino Interrupt (contd...)

- Features
  - Monitors a user input
  - Allows processor to do something else
  - Allows quick and efficient reaction to events
- External Interrupts
  - Functions used → `attachInterrupt()` and `detachInterrupt()`



# Arduino Interrupt (contd...)

- `attachInterrupt()`
  - Function to set the interrupt
  - Syntax is `attachInterrupt(digitalPinToInterrupt(pin),ISR,mode)`
  - `digitalPinToInterrupt(pin)` translates digital pin to interrupt number → First parameter passed to `attachInterrupt()`
  - To connect to pin number 4, we can use `digitalPinToInterrupt(4)`
  - mode → when interrupt is triggered in Arduino (LOW, FALLING, CHANGE, RISING)
  - ISR → Interrupt Service Routine, which is called when interrupts arise

# Arduino Interrupt (contd...)

- `detachInterrupt()`
  - Function to turn off the interrupt
  - Syntax is `detachInterrupt(digitalPinToInterrupt(pin))`
  - `digitalPinToInterrupt(pin)` translates digital pin to interrupt number → First parameter passed to `attachInterrupt()`
  - To connect to pin number 4, we can use `digitalPinToInterrupt(4)`

# Thank you