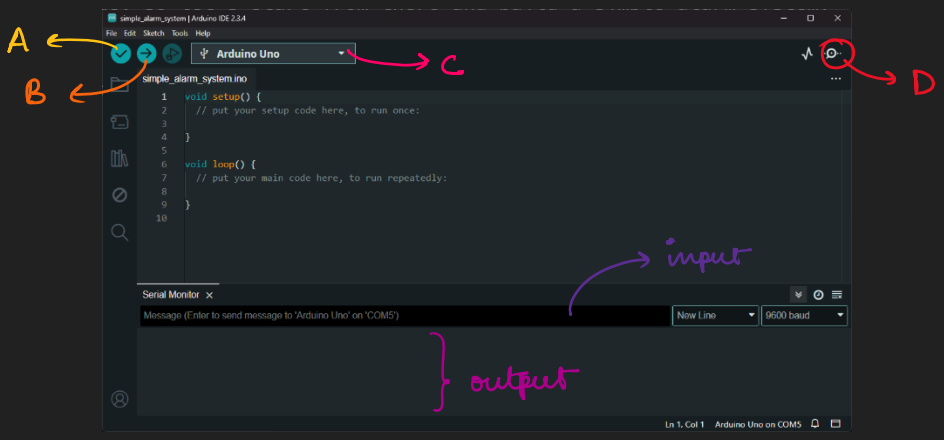
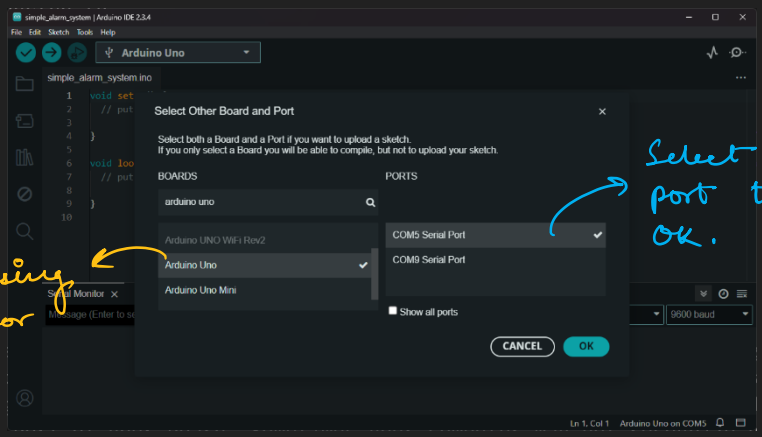
What shall we do today?

1. We discussed what we can do with IoT the previous day with the example of a water pump system. We finished by installing Arduino IDE where we are going to write our code, so let's start from there and light an LED, this is probably the hello world of electronics.

First look at Arduino IDE.



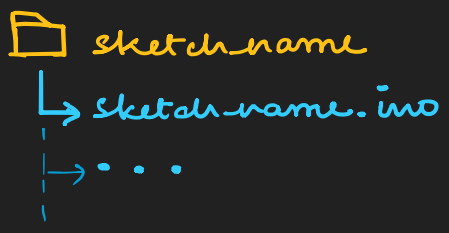
1. Arduino IDE will look like this when you start it. There are a lot of features in Arduino IDE, but for now we will look at some of the easier, useful ones.
   A. Verify - Check if the code is okay.
   B. Upload - We write the code on our computer, but at the end of the day, the microprocessor, Arduino UNO will do the real job, so we use this button to send the code to Arduino UNO.
   C. Select board and port - We connect the microprocessor to our computer with a cable from our computer's USB port. In this drop-down menu, we have to select the name of the board (microprocessor) we are using and the port it is connected to.



   Selecting serial (USB) ports can be tricky sometimes. It is showing 2 serial ports, COM9 and COM5, but I didn't even connect the board yet. In such a case, plug in your board and see which new serial port appears there, and that is the port of your board. Sometimes your computer may not recognize your board at all. It is most likely a driver issue and can be fixed by installing the driver responsible for communication with the board, CH340/CH341 in the case of the Arduino UNOs that we will be using. Visit https://www.wch-ic.com/downloads/CH341SER_ZIP.html and install the drivers to proceed.
   D. Toggle serial monitor - When we write simple programs in C or python, we typically print output and take input to and from the terminal window. But

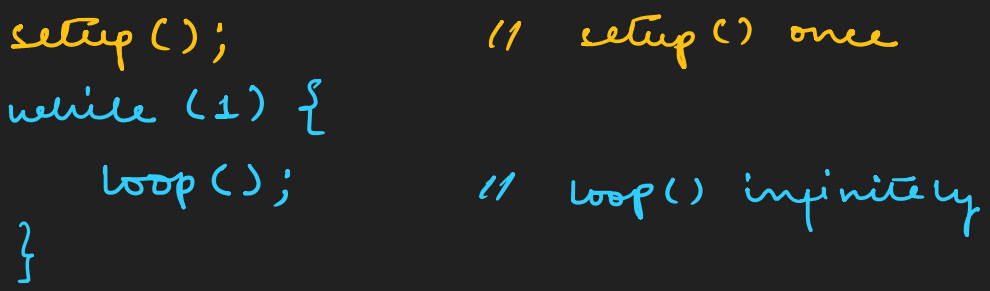
there is no terminal window in microcontrollers. So, we can communicate with our boards using the serial monitor. We can send and receive text to and from our board using the serial monitor just like we do in a terminal window. We will see it in action when we make something.

2. Every project we create in Arduino IDE is called a sketch, and every sketch has a name. If our sketch name is arduino_led, then Arduino IDE will create a folder arduino_led and a file arduino_led.ino inside it.



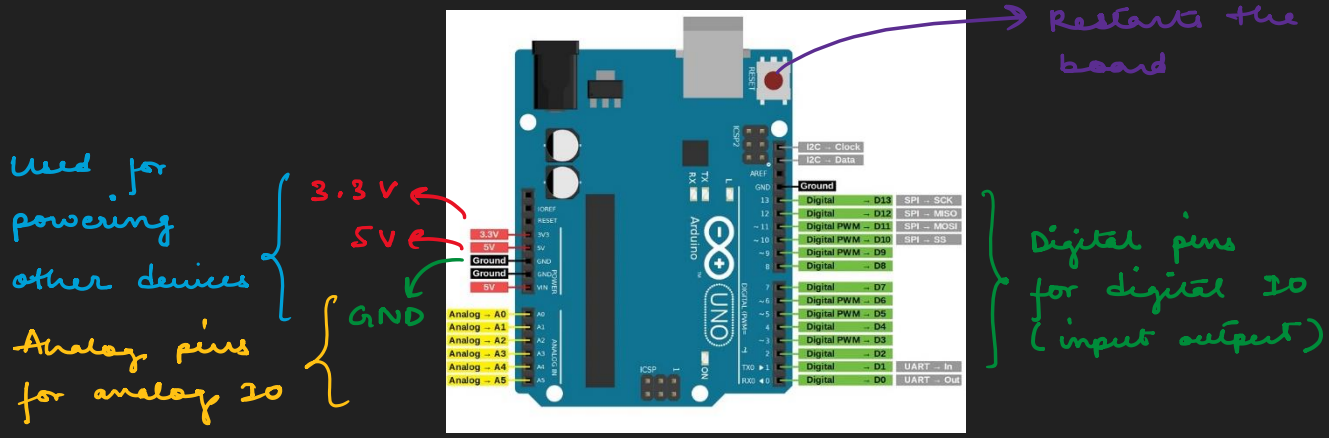
arduino_led.ino is the starting file of the project, ino is the extension of Arduino's own language. It is like C, but easier.

3. The starting file should have 2 functions, setup() and loop(), much like the main() function in C.
4. When we power on our board, the setup() function is called once, then the loop() function is called infinitely. There is also a reset button in the boards to manually restart the board, in which case it will restart from the setup() function and infinitely call loop().



Understanding pins of Arduino UNO.

1. Visit https://robu.in/arduino-pin-configuration/ if you want to read more about Arduino pins.



IO in microcontroller boards.

1. IO means input output.
2. You may recall that the general purpose of any computer is to accept some input, process it and generate some output.
3. In the case of microcontrollers, we use specific pins present on the board to send input or accept output.

4. A sensor may send data from a pin, in which case, the pin is an INPUT pin from the board's perspective.
5. If the board sends data to an actuator, then the pin will be an OUTPUT pin.

What is digital IO?

1. During digital IO, the board sends or accepts binary data. True or false, 0 or 1, HIGH or LOW.
2. We can use the digitalRead() and digitalWrite() functions for that.

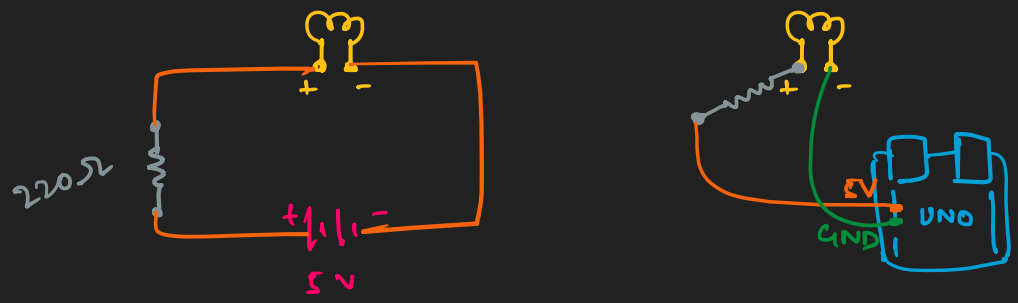
What is analog IO?

1. Analog IO allows the board to send or accept more values that just 0 and 1.
2. Arduino UNO allows a range of 0 to 2^{10} for analog IO.
3. We use the analogRead() and analogWrite() functions for analog IO.

How exactly do digital and analog IO work?

1. It is all about voltage.
2. Arduino UNO operates on 5V, so if we divide the entire voltage range into two parts, say 0-2.5V and 2.5V-5V, we can consider 0-2.5V as 0 and 2.5V-5V as 1; and that is how Arduino UNO understands digital IO.
3. Analog IO also uses varying voltage, but the voltage range is divided into 1024 equal parts and can be used to send and receive values from 0 to 1024 (excluding 1024).
4. When we want to send more complex data like strings, we rely on protocols like Serial, SPI and I2C. We will understand them when we use them in the future.

Now how do we use all these information to light an LED?



1. If we tried to light an LED without an Arduino UNO, we would require an LED, a battery and a resistor, and a bunch of copper wires to connect them.
2. We can use the 5V and GND pins of the UNO to replace a battery. Then if we intertwine the positive (longer) leg of the LED with one end of the resistor, and then plug in the other end of the resistor in the 5V pin and the smaller leg of the LED in the GND pin, the LED will light up.
3. Note: Polarity does not matter for resistor, but it does for LEDs. The longer leg is positive and the smaller leg is negative. Also, LEDs are generally rated less than 5V, so we should always use a resistor when we work with them.

What more things can we do by using UNO and LED?

1. All we did was use UNO as the battery for our LED, the advantage of using UNO is that we can control when and how to light the LED with our program.

2. To do that, we have to insert the positive end (which is the free end of the resistor) into a digital pin of our UNO, say D8.
3. Now we can control the output to the D8 pin to light the LED whenever we want.

Write an Arduino code to blink an LED.

1. When we create a new sketch, we are greeted with a template.

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

}

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

}
```

2. In setup(), we set D8 pin as an output pin. Since all the digital and analog pins can be used for both input and output, if we want to use a pin, we have to tell in setup() whether we want to use it as input or output by using the pinMode() function.

```
pinMode(8, OUTPUT);
```

   8 refers to the D8 pin and OUTPUT is a predefined macro.
3. Now that we have told Arduino to use pin 8 as OUTPUT pin, we have to make the LED blink. To do that, we have to send 5V to D8 for some time interval, say 1 second, then send 0V to D8 for another 1 second, and then repeat.
4. Remember that loop() function is infinitely called, so we can do something like this:

```
void loop() {
  send 5V to D8;
  wait 1 second;
  send 0V to D8;
  wait 1 second;
}
```

   And since loop() is called infinitely, the board will keep on blinking the LED.
5. To send only 5V and 0V (only two states, on and off), we can use:

```
digitalWrite(8, HIGH);        // sends 5V to pin 8
digitalWrite(8, LOW);         // sends 0V to pin 8
```

   8 is the pin where we want to write the value. HIGH is 1 or on, LOW is 0 or off.
6. To wait for 1 second, we use:

```
delay(1000);                  // time is in milliseconds
```

7. So, to sum it up, the final program will look like this:

```
void setup() {
  // put your setup code here, to run once:
  pinMode(8, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(8, HIGH);
  delay(1000);
  digitalWrite(8, LOW);
  delay(1000);
}
```