INTERNET OF THINGS TECHNOLOGY
15CS81

Mr. Anand S. Hiremath,
Dept. of CSE, BLDEA's CET, Vijayapur
http://ashiremath.wordpress.com
ashiremath@bldeacet.ac.in

Punishable

- **PLEASE SWITCH OFF THE PHONE/MOBILE.**

References:

- David Hanes, Gonzalo Salgueiro, Patrick Grossetete, Robert Barton, Jerome Henry,"IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things", 1stEdition, Pearson Education (Cisco Press Indian Reprint). (ISBN: 9789386873743)
- Srinivasa K G, "Internet of Things", CENGAGE Leaning India, 2017.
- https://store.arduino.cc/usa/arduino-uno-rev3
- https://www.arduino.cc/reference/en/
- https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
- https://www.raspberrypi.org/documentation/configuration/
- https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up
- https://www.raspberrypi.org/documentation/usage/python/

MODULE-5

IoT Physical Devices and Endpoints – Arduino UNO and Raspberry Pi

and

Smart and Connected Cities

Topics Covered

- IoT Physical Devices and Endpoints – Arduino UNO:
  - Introduction to Arduino,
    - Why Arduino?
    - Which Arduino?
  - Exploring Arduino UNO learning Board
    - Things that Arduino do
  - Installing the Software (Arduino IDE),
    - Connecting Arduino UNO learning Board
  - Fundamentals of Arduino Programming.
    - Difference between Analog, Digital and PWM Pins

Arduino UNO
Introduction to Arduino

- Arduino is a basic single board microcontroller designed to make applications, interactive controls, or environments easily adaptive.
  - The hardware consists of a board designed around an 8-bit microcontroller, or a 32-bit ARM.
  - Current models feature things like a USB interface, analog inputs, and GPIO pins which allows the user to attach additional boards.
- Introduced in 2005, the Arduino platform was designed to provide a cheaper way for students and professionals to create applications that play in the human interface world using sensors, actuators, motors, and other rudimentary products.
- It offers a simple integrated **IDE (integrated development environment)** that runs on regular personal computers and allows users to write programs for Arduino using C or C++.

Arduino UNO
Introduction to Arduino

- **Why Arduino?**
  - **Inexpensive:**
    - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand.
  - **Cross-platform:**
    - The Arduino software runs on Windows, Macintosh OS and Linux operating systems.
  - **Simple, clear programming environment:**
    - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well.
  - **Open source and extensible software:**
    - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries.
  - **Open source and extensible hardware:**
    - The Arduino is based on Atmel's ATMEGA microcontrollers. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Arduino UNO
Introduction to Arduino

- **Which Arduino?**
  - Entry Level
    - Easy to use and ready to first creative projects. These boards and modules are the best to start learning and tinkering with electronics and coding.
  - Enhanced Features
    - Experience the excitement of more complex projects, with advanced functionalities, or faster performances.
  - Internet of Things
    - Make connected devices easily with IoT and the world wide web.
  - Wearable
    - Add smartness to projects and sewing the power of electronics directly to textiles.

Arduino UNO
Introduction to Arduino

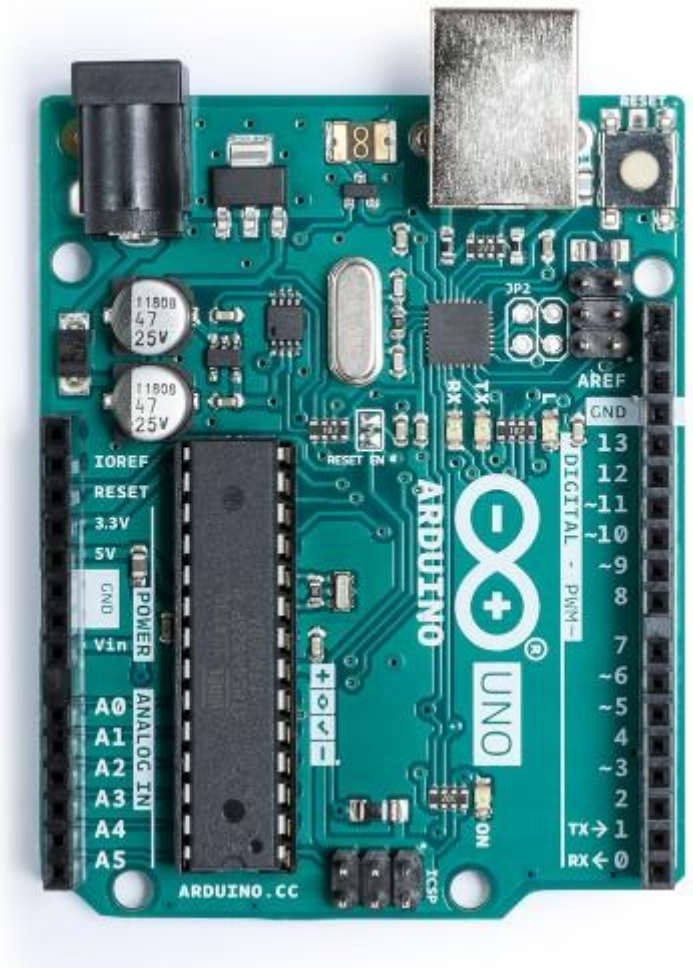
- ARDUINO UNO
  - A microcontroller board based on the ATmega328P.
  - It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.
  - Connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
- ARDUINO MEGA 2560
  - A microcontroller board based on the ATmega2560.
  - It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.
  - It is the recommended board for 3D printers and robotics projects.
- ARDUINO MICRO
  - A microcontroller board based on the ATmega32U4, featuring a built-in USB which makes the Micro recognisable as a mouse or keyboard.
  - It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, an ICSP header, and a reset button.

Arduino UNO
Introduction to Arduino

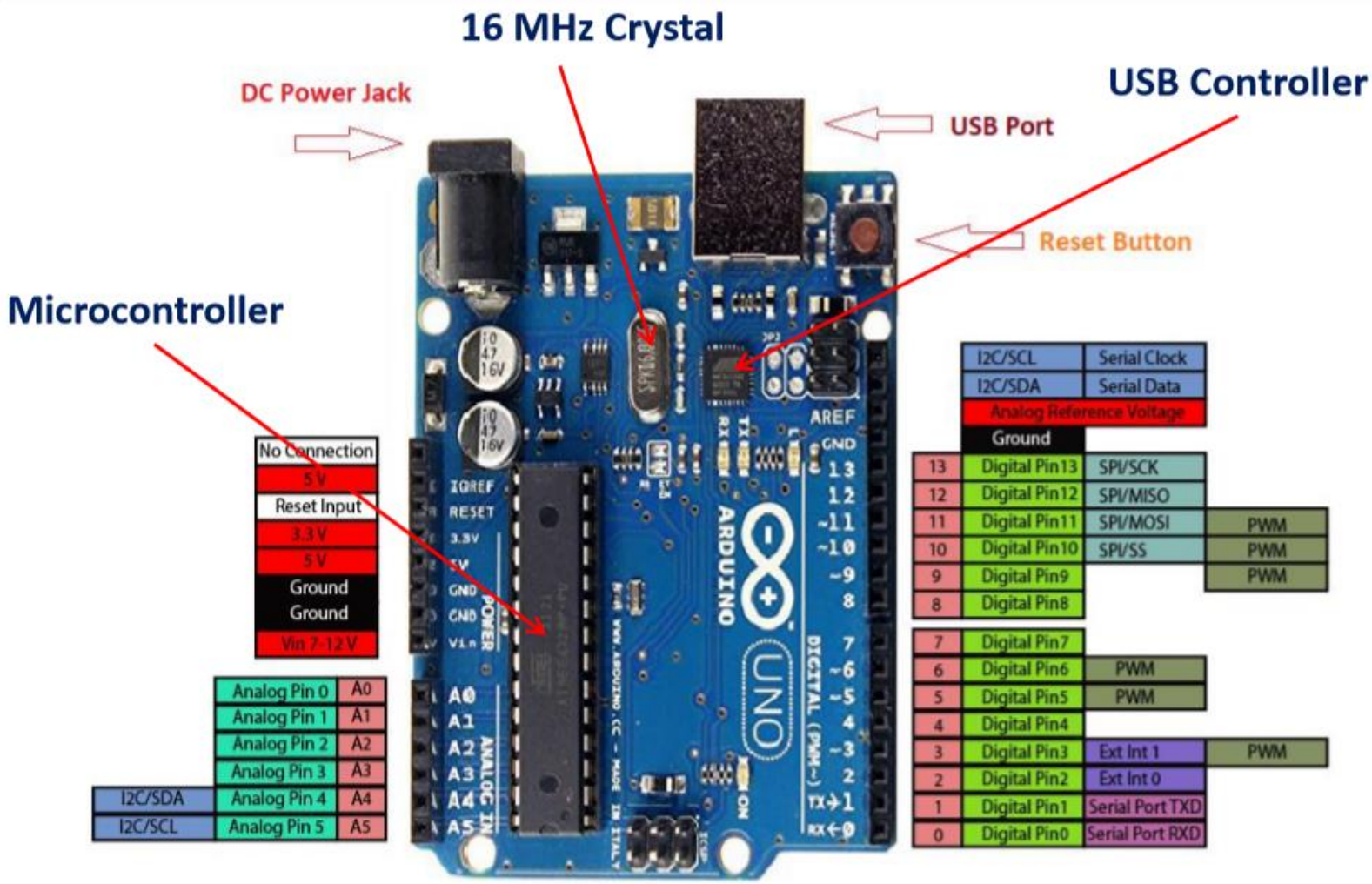
- ARDUINO MKR1000
  - It is based on the Atmel ATSAMW25 ARM SoC (System on Chip), that is part of the Smart Connect family of Atmel Wireless devices, specifically designed for IoT projects and devices.
  - The ATSAMW25 is composed of three main blocks:
    - SAMD21 Cortex-M0+ 32bit low power ARM MCU
    - WINC1500 low power 2.4GHz IEEE® 802.11 b/g/n Wi-Fi
    - ECC508 Crypto Authentication
    - PCB Antenna.

Arduino UNO

Exploring Arduino UNO learning Board



Arduino UNO
Exploring Arduino UNO learning Board



Arduino UNO
Exploring Arduino UNO learning Board

- 14 digital pins on the Uno can be used as an input or output,
- 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values).
- Serial:
  - Pin 0 (RX) and Pin 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts:
  - Pin 2 and Pin 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- PWM:
  - Pin 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output.
- SPI (Serial Peripheral Interface):
  - Pin 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED:
  - 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- TWI (Two Wire Interface):
  - A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.
- Reset:
  - Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.
- AREF (Analog REFerence):
  - Reference voltage for the analog inputs.

Arduino UNO
Exploring Arduino UNO learning Board

- Things that Arduino Can Do

Arduino UNO
Installing the Software (Arduino IDE)

- The Arduino Software (IDE) allows you to write programs and upload them to board. In the Arduino Software page you will find two options:
  - 1. Online IDE (Arduino Web Editor). It will allow to save sketches in the cloud, having them available from any device and backed up.
  - 2. Offline, should use the latest version of the desktop IDE.

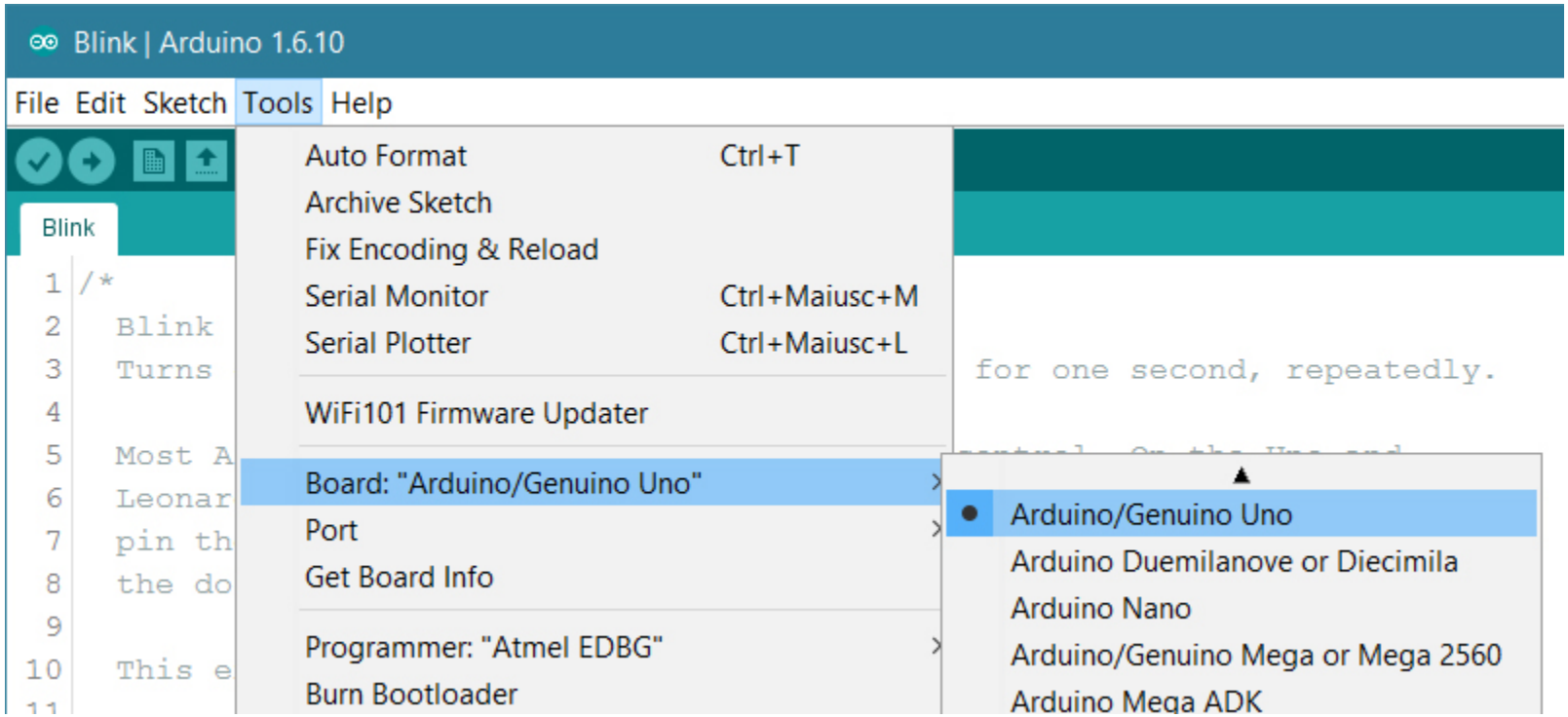
- Install the Arduino Desktop IDE accordingly to operating system.
  - Windows
  - Mac OS X
  - Linux
  - Portable IDE (Windows and Linux)
  - Choose board in the list here on the right to learn how to get started with it and how to use it on the Desktop IDE.

Arduino UNO
Installing the Software (Arduino IDE)

- **Connecting Arduino UNO Learning Board:**
  - If you want to program your Arduino Uno while offline you need to install the Arduino Desktop IDE.
  - Connect your Uno board with an A B USB cable; sometimes this cable is called a USB printer cable.
  - If you used the Installer, Windows – from XP up to 10 – will install drivers automatically as soon as you connect your board.

Arduino UNO
Installing the Software (Arduino IDE)

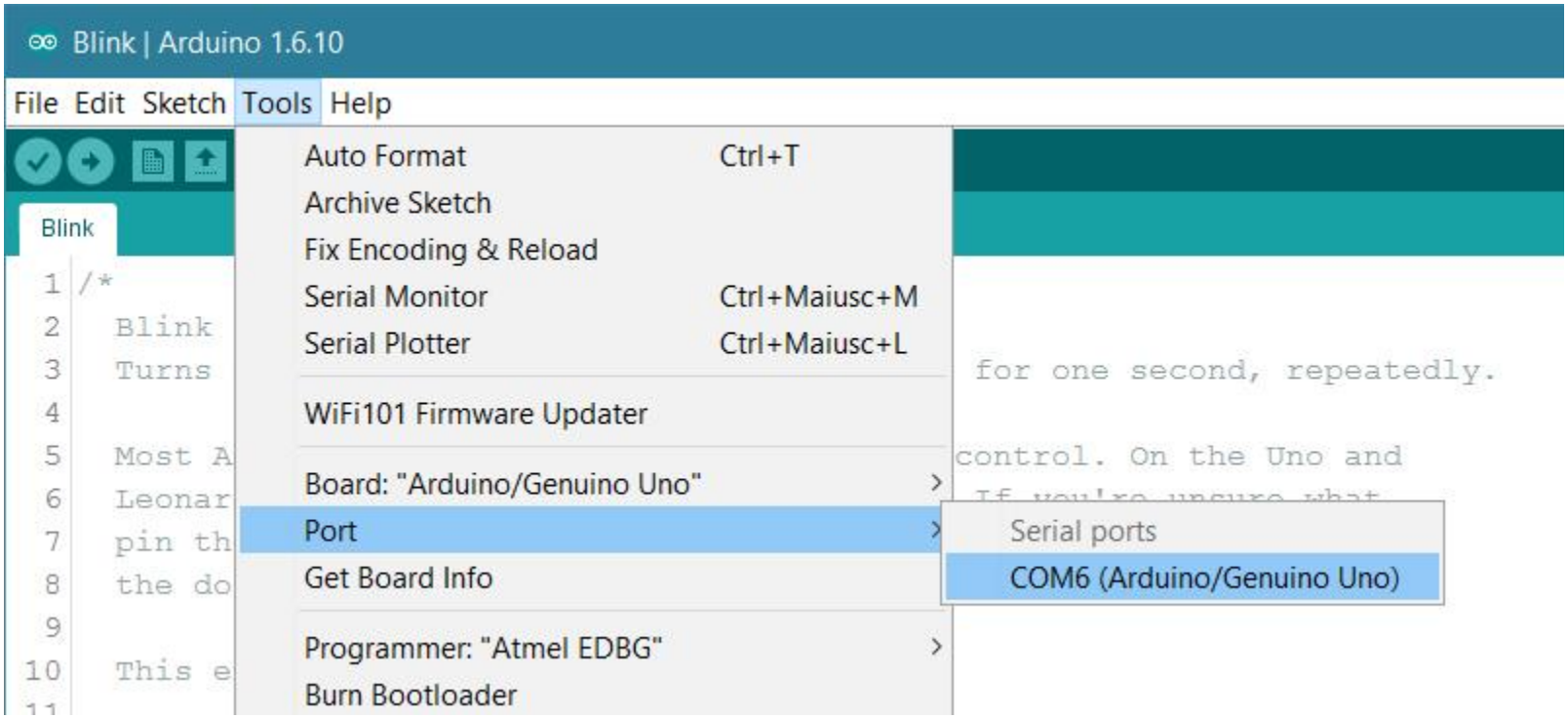
- You'll need to select the entry in the Tools > Board menu that corresponds to your Arduino or Genuino board.



Arduino UNO
Installing the Software (Arduino IDE)

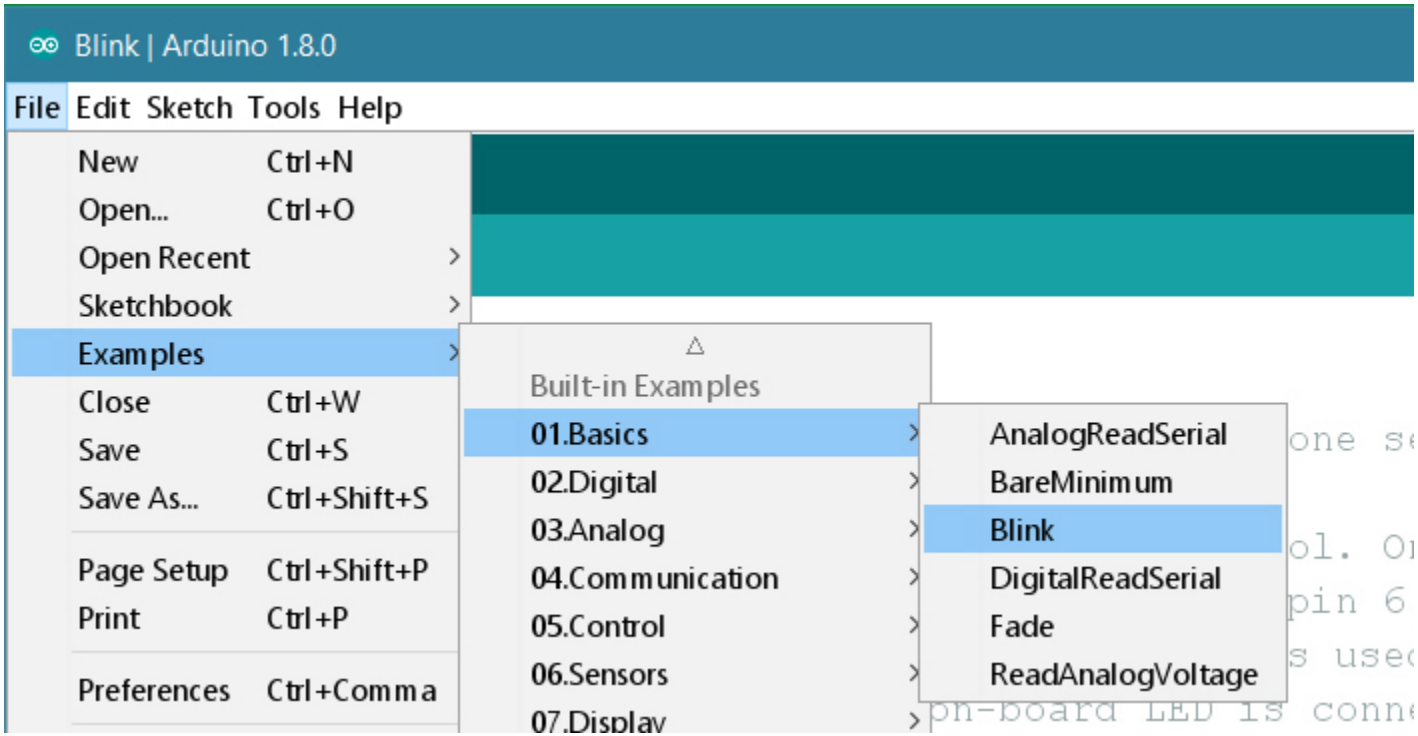
- Select the serial device of the board from the Tools | Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your board and re-open the menu; the entry that disappears should be the Arduino or Genuino board. Reconnect the board and select that serial port.

Arduino UNO
Installing the Software (Arduino IDE)



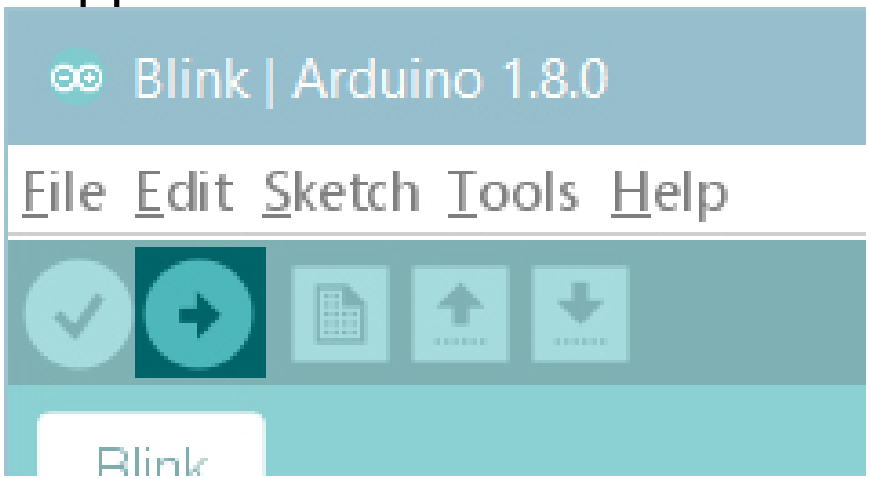
Arduino UNO
Installing the Software (Arduino IDE)

- Open your first sketch
- Open the LED blink example sketch: File > Examples >01.Basics > Blink.



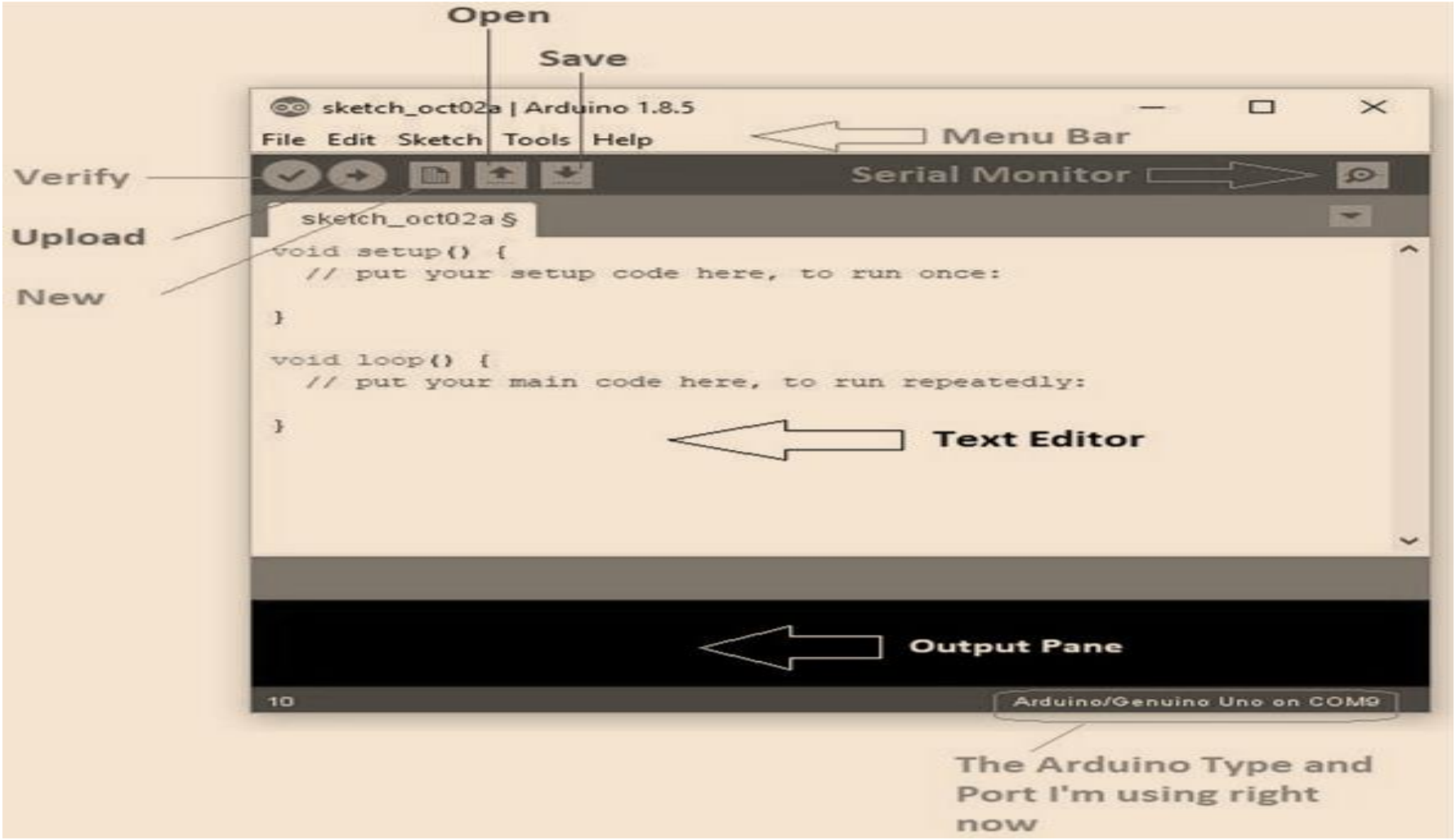
Arduino UNO
Installing the Software (Arduino IDE)

- Upload the program
- Now, simply click the "Upload" button in the environment. Wait a few seconds – you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.



- A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in orange). If it

Arduino UNO
Installing the Software (Arduino IDE)



Arduino UNO
Installing the Software (Arduino IDE)

File	
New	This is used to open new text editor window to write your code
Open	Used for opening the existing written code
Open Recent	The option reserved for opening recently closed program
Sketchbook	It stores the list of codes you have written for your project
Examples	Default examples already stored in the IDE software
Close	Used for closing the main screen window of recent tab. If two tabs are open, it will ask you again as you aim to close the second tab
Save	It is used for saving the recent program
Save as	It will allow you to save the recent program in your desired folder
Page setup	Page setup is used for modifying the page with portrait and landscape options. Some default page options are already given from which you can select the page you intend to work on
Print	It is used for printing purpose and will send the command to the printer
Preferences	It is page with number of preferences you aim to setup for your text editor page
Quit	It will quit the whole software all at once

Arduino UNO
Installing the Software (Arduino IDE)

- Technical Specification

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Arduino UNO
Fundamentals of Arduino Programming

- The Arduino IDE supports the languages C and C++ using special rules of code structuring.
- The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures.
- User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution.

Arduino UNO
Fundamentals of Arduino Programming

- Sketch
  - A sketch is a program written with the Arduino IDE.[57] Sketches are saved on the development computer as text files with the file extension .ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension .pde.
  - A minimal Arduino C/C++ program consist of only two functions:
    - **setup()**: This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.
    - **loop()**: After setup() function exits (ends), the loop() function is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.

Arduino UNO
Fundamentals of Arduino Programming

- Blink example:
  - Most Arduino boards contain a light-emitting diode (LED) and a current limiting resistor connected between pin 13 and ground, which is a convenient feature for many tests and program functions.
  - A typical program used by beginners, akin to Hello, World!, is "blink", which repeatedly blinks the on-board LED integrated into the Arduino board.
  - This program uses the functions pinMode(), digitalWrite(), and delay(), which are provided by the internal libraries included in the IDE environment.

Arduino UNO
Fundamentals of Arduino Programming

```
#define LED_PIN 13                  // Pin number attached to LED.

void setup() {
    pinMode(LED_PIN, OUTPUT);       // Configure pin 13 to be a digital output.
}

void loop() {
    digitalWrite(LED_PIN, HIGH);    // Turn on the LED.
    delay(1000);                    // Wait 1 second (1000 milliseconds).
    digitalWrite(LED_PIN, LOW);     // Turn off the LED.
    delay(1000);                    // Wait 1 second.
}
```

Arduino UNO
Fundamentals of Arduino Programming

- **Variables and Data Types**

DATA TYPE	CONTENTS
`void`	No data type
`boolean`	True or false
`char`	One character, stored as an ASCII number ('A', 'B', 'C'...)
`unsigned char`	Decimal numbers, from 0 to 255

Arduino UNO
Fundamentals of Arduino Programming

- **Variables and Data Types**

DATA TYPE	CONTENTS
`byte`	Decimal numbers, from 0 to 255
`int`	Decimal numbers, from –32,768 to 32,767 (Arduino Due, from –2,147,483,648 to 2,147,483,647)
`unsigned int`	Decimal numbers, from 0 to 65,535 (Arduino Due, from 0 to 4,294,967,295)
`word`	Decimal numbers, from 0 to 65,535
`long`	Decimal numbers, from –2,147,483,648 to 2,147,483,647
`unsigned long`	Decimal numbers, from 0 to 4,294,967,295
`short`	Decimal numbers, –32,768 to 32,767
`float`	Floating point numbers, from $-3.4028235 \times 10^{38}$ 3.4028235×10^{38}
`double`	Floating point numbers
`string`	An array of char
`String`	Advanced arrays of char
`array`	A collection of variables

Arduino UNO
Fundamentals of Arduino Programming

- **if and if ..else Statement**

```
if (expression)
 {
statement;
}

if (expression)
{
do_this;
}
else
{
do_that;
}
```

Arduino UNO
Fundamentals of Arduino Programming

- **while Loop**

```
while (button == false)
{
button = check_status(pin4);
}
```

Arduino UNO
Fundamentals of Arduino Programming

- **Digital I/O**
  - pinMode()
  - digitalRead()
  - digitalWrite()

- **pinMode()**
- Before using a pin as a digital input or output, must first configure the pin, which is done with pinMode().
- pinMode() uses two parameters: pin and mode.
  - pinMode(pin, mode)
  - The pin parameter is simply the digital pin number want to set.
  - The mode parameter is one of three constants: INPUT or OUTPUT,

Arduino UNO
Fundamentals of Arduino Programming

- **digitalRead()**
- In order to read the state of a digital pin, you must use digitalRead():
  - result = digitalRead(pin);
- The pin parameter is the pin number you want to read from.
- This function returns either HIGH or LOW, depending on the input

Arduino UNO
Fundamentals of Arduino Programming

- **digitalWrite()**
- To write the state of a pin that was declared as an OUTPUT, use the digitalWrite() function:
  - digitalWrite(pin, value);
- The pin parameter is the pin number you want to write to, and the value is the logical level you want to write; HIGH or LOW.

Arduino UNO
Fundamentals of Arduino Programming

- **Analog I/O**
- analogRead()
- To read a value from an analog pin, you call analogRead().
  - int analogRead(pin)
  - analogRead() reads the voltage value on a pin and returns the value as an int.
  - The pin argument denotes the analog pin you want to read from. When referring to an analog pin, call them as A0, A1, A2,…A6.
  - This function takes approximately 100 microseconds to perform.

Arduino UNO
Fundamentals of Arduino Programming

- analogWrite()
  - analogWrite() is used to write an analog output on a digital pin.
  - Arduinos use Pulse-width modulation (PWM).
  - PWM is digital but can be used for some analog devices.
  - It uses a simple technique to “emulate” an analog output.
  - It relies on two things:
    - a pulse width and a duty cycle.
  - It is a way of simulating any value within a range by rapidly switching between 0 volts and 5 volts.

Arduino UNO
Fundamentals of Arduino Programming

- **Time Functions**
- **delay()**
  - tells the microcontroller to wait for a specified number of milliseconds before resuming the sketch.

- **millis()**
  - millis() returns the number of milliseconds that the sketch has been running, returning the number as an unsigned long.

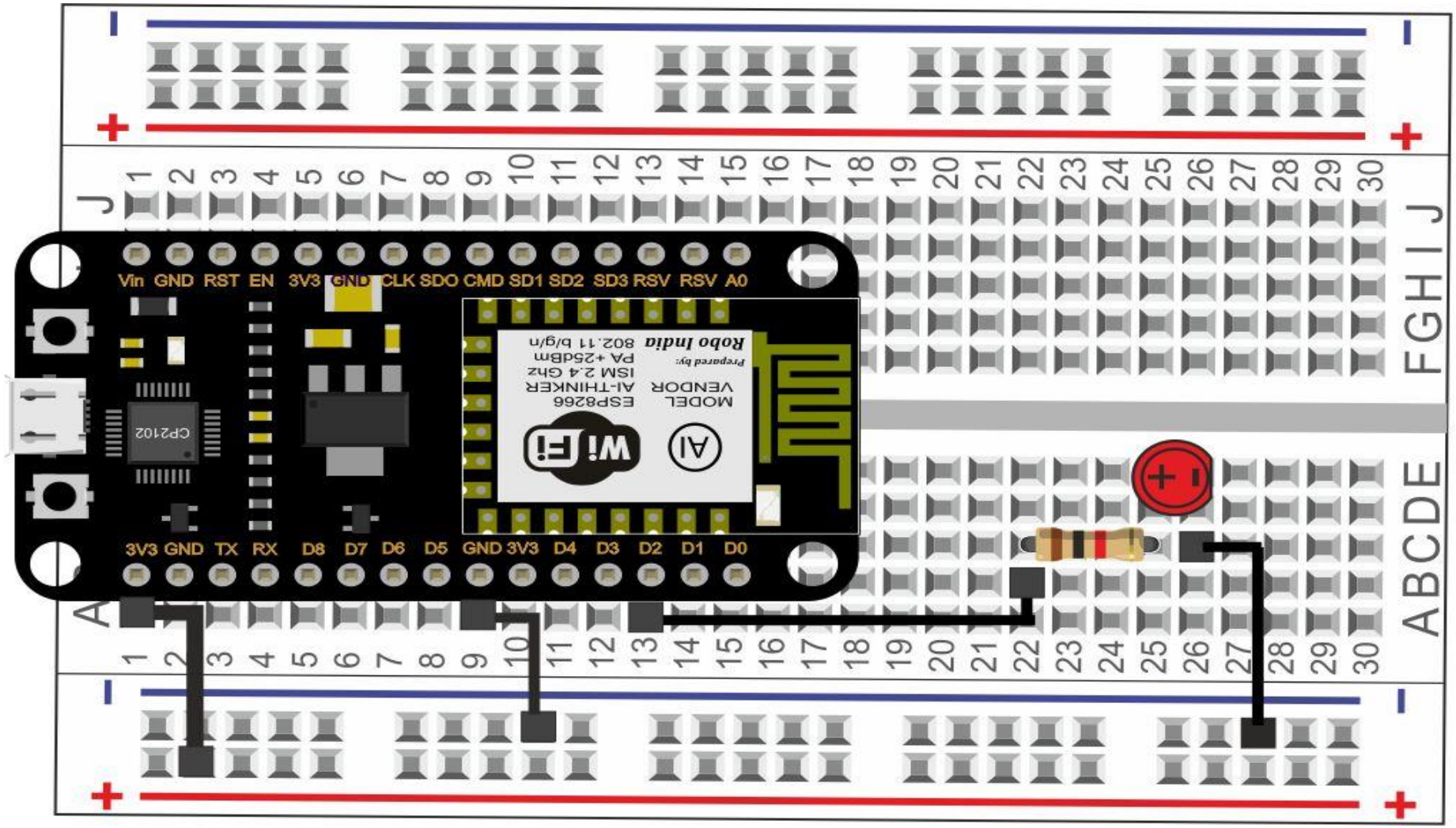
Arduino UNO
Fundamentals of Arduino Programming

- **Mathematical Functions**
- **min()**
  - min() returns the smaller of two numbers.
  - E.g. result = min(x, y)
- **max()**
  - max() returns the higher of two values.
  - result = max(x, y)

Arduino UNO
Fundamentals of Arduino Programming

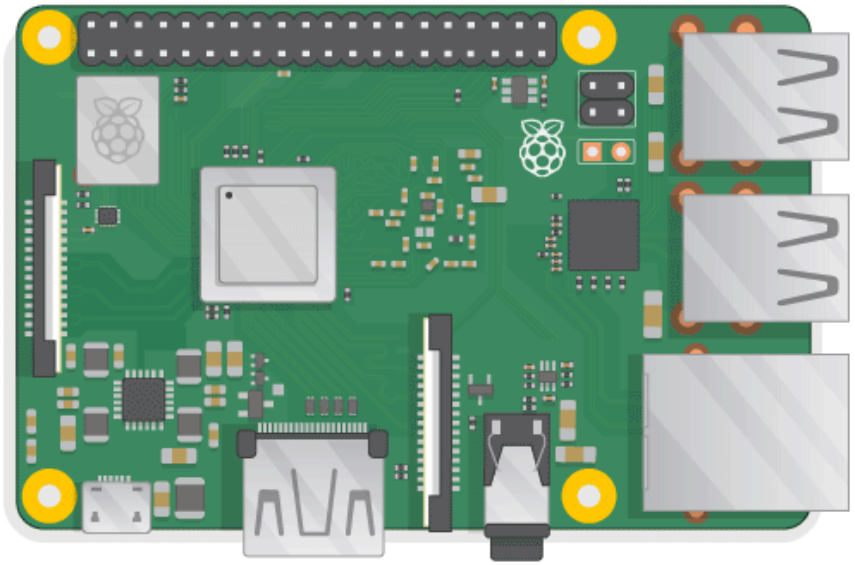
- **random()**
  - Arduinos are capable of generating pseudo-random numbers using the random() function:
  - result = random(max);
  - result = random(min, max);
  - This function takes one or two parameters specifying the range for the random number to be chosen.
  - If the min parameter is omitted, the result will be a number between zero and max, otherwise the number will be between min and max.
    - The result is returned as a long.

Arduino UNO
Fundamentals of Arduino Programming



Topics Covered

- IoT Physical Devices and Endpoints – RaspberryPi:
  - Introduction to Raspberry Pi,
  - Exploring the Raspberry Pi Learning Board,
    - Description of System on Chip (SoC)
    - Raspberry Pi interface
  - Raspberry Pi Operating Systems
    - Operating System (Not Linux Based)
    - Operating System (Linux Based)
    - Media centre Operating System
    - Audio Operating System
    - Recalbox
  - Operating Systems Setup on Raspberry Pi
    - Formatting SD Card
    - OS Instllation
    - First Boot
    - Login Information
  - Raspberry Pi commands
  - Configuring RaspberryPi,
  - Programming RaspberryPi with Python,



Topics Covered

- Smart and Connected Cities,
  - An IoT Strategy for Smarter Cities
    - Vertical IoT Needs for Smarter Cities
    - Global vs. Siloed Strategies
  - Smart City IoT Architecture
    - Street Layer
    - City Layer
    - Data Center Layer
    - Services Layer
    - On-Premises vs. Cloud
  - Smart City Security Architecture
  - Smart City Use-Case Examples
    - Connected Street Lighting
      - Connected Street Lighting Solution
      - Street Lighting Architecture
    - Smart Parking
      - Smart Parking Use Cases
      - Smart Parking Architecture
    - Smart Traffic Control
      - Smart Traffic Control Architecture
      - Smart Traffic Applications
    - Connected Environment
      - The Need for a Connected Environment
      - Connected Environment Architecture