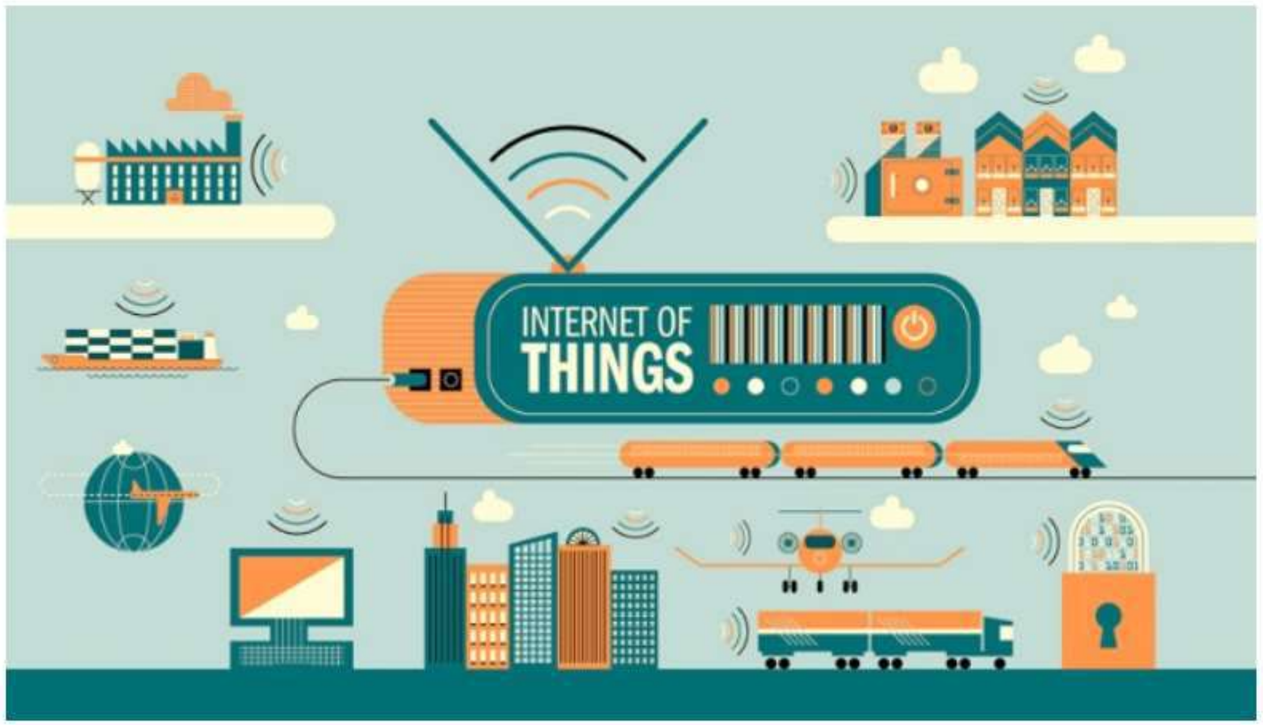
REVA UNIVERSITY



Prof Raghavendran T S
Asst Prof, School of C&IT, Reva University, Bengaluru
Raghavendran.ts@reva.edu.in

www.reva.edu.in

B22EK0601

Unit 4

1) **IoT Physical Devices and Endpoints – Arduino UNO**
-Introduction to Arduino
-Overview of the Arduino UNO board
-Installing the Arduino IDE software
-Fundamentals of Arduino programming

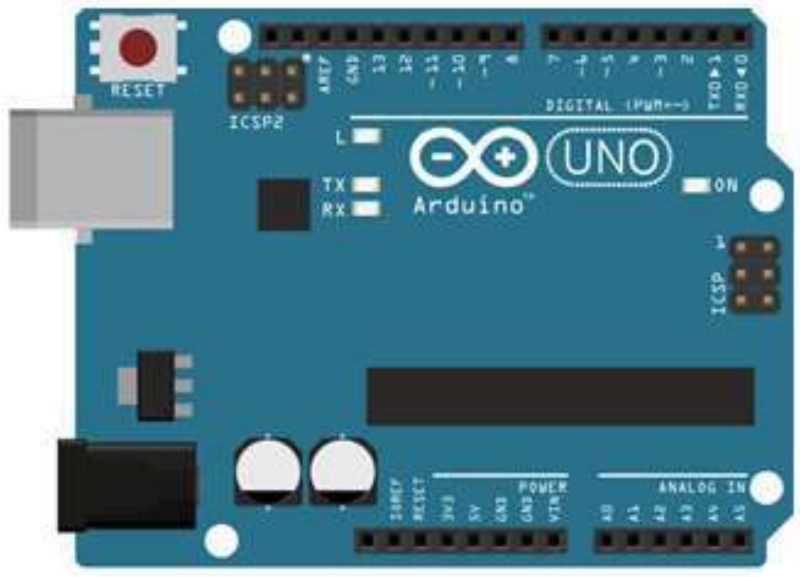
2) **IoT Physical Devices and Endpoints – Raspberry Pi**
Introduction to Raspberry Pi

-Hardware layout of the Raspberry Pi board
-Operating systems supported on Raspberry Pi
-Configuring the Raspberry Pi for use
-Programming Raspberry Pi using Python
-Wireless temperature monitoring system using Raspberry Pi
-Introduction to DS18B20 temperature sensor
-Connecting DS18B20 to Raspberry Pi
-Accessing temperature data from DS18B20 sensors
-Connecting Raspberry Pi via SSH
-Remote access to Raspberry Pi

3)**Smart and Connected Cities**
-Concept and importance of smart and connected cities
-IoT strategies for smarter cities
-Smart city IoT architecture
-Smart city security architecture
-Smart city use-case examples (e.g., smart lighting, traffic management, waste monitoring)

Arduino Basics

- Arduino is an open-source platform that's composed of very simple and easy-to-use hardware and software.
- In a nutshell your Arduino can read sensor data and control components such as lights, motors, thermostats, and garage doors.



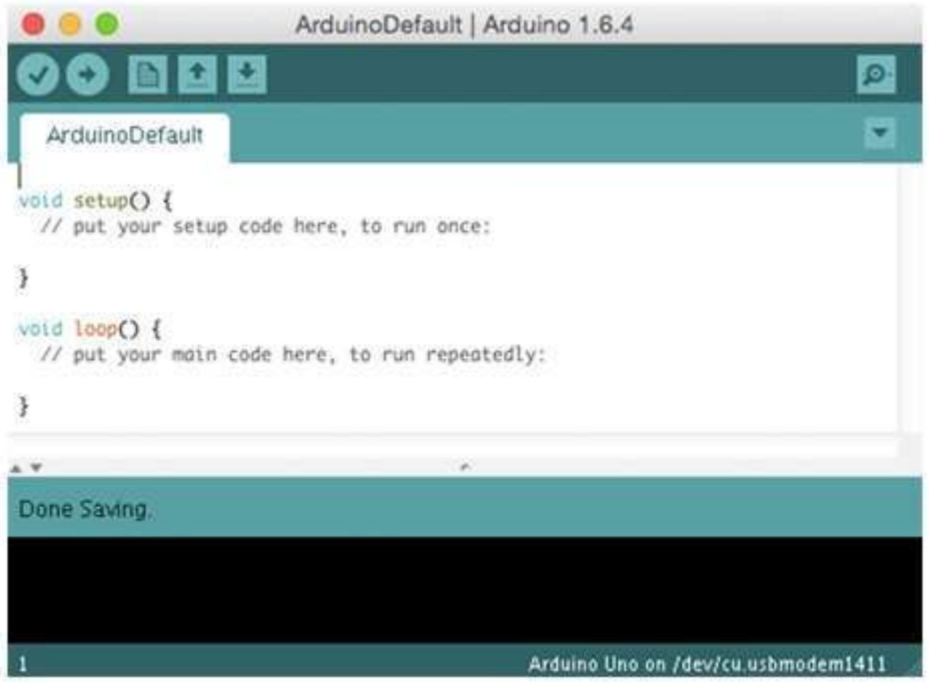
- Digital Pins- 14 Digial Pins, can be both Input or Output
- Analog Pins:- 6 Analog pins from A0 to A5. Analog pin range 0-1023.
- USB Connector:- A USB connector lets you connect Arduino to the computer, power the board, upload code, and receive logs on a serial monitor.
- Battery Power:- IoT applications that need to be placed in remote locations will need their own power source. You can use the battery power connector to power the board.

Software Requirements

- Arduino provides a C-like language for programming Arduino boards. You will be using the Arduino IDE for writing code and uploading it to an Arduino board. You can install the latest version of Arduino IDE from

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

- Once Arduino IDE has been installed on your machine, open it and, as shown below.

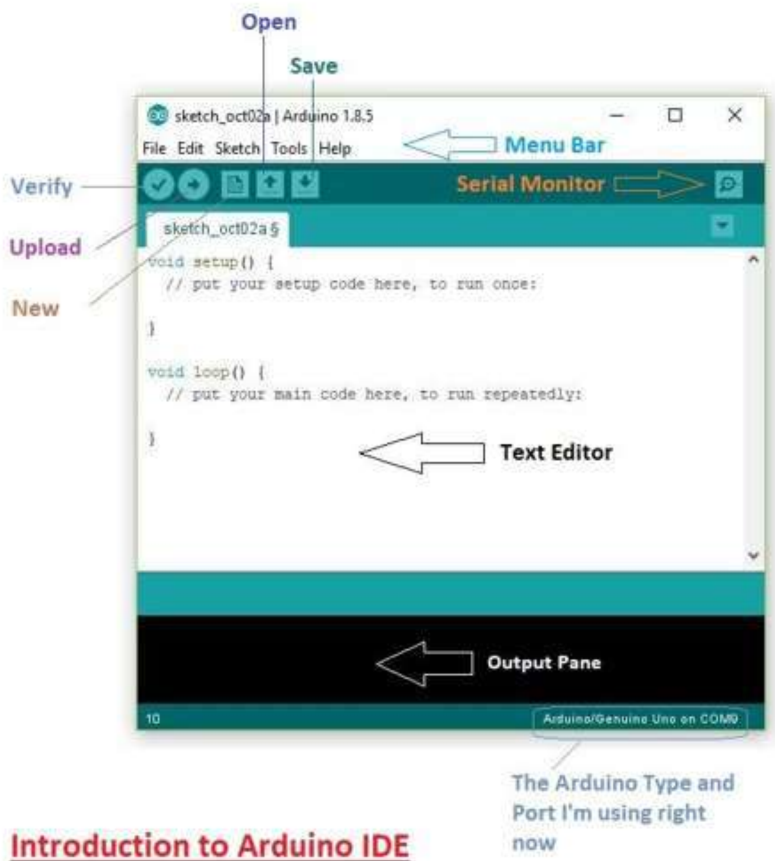


```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

Done Saving.

1 Arduino Uno on /dev/cu.usbmodem1411

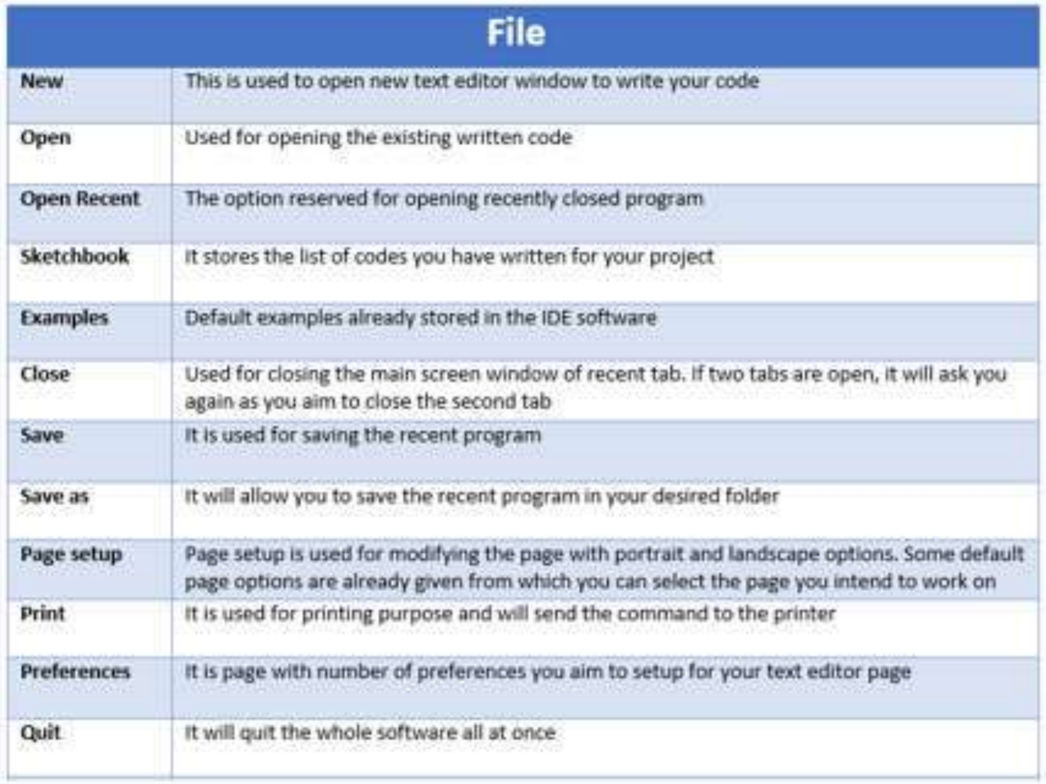
The Arduino IDE



Open
Save
sketch_oct02a | Arduino 1.8.5
File Edit Sketch Tools Help
Menu Bar
Verify
Upload
New
Serial Monitor
sketch_oct02a §
Text Editor
Output Pane
10
Arduino/Genuine Uno on COM9
The Arduino Type and Port I'm using right now

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

}

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

}
```

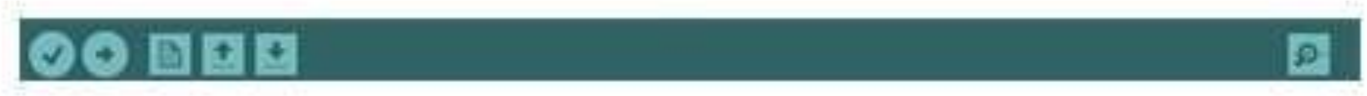
Introduction to Arduino IDE



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

The Arduino IDE

- Toolbar

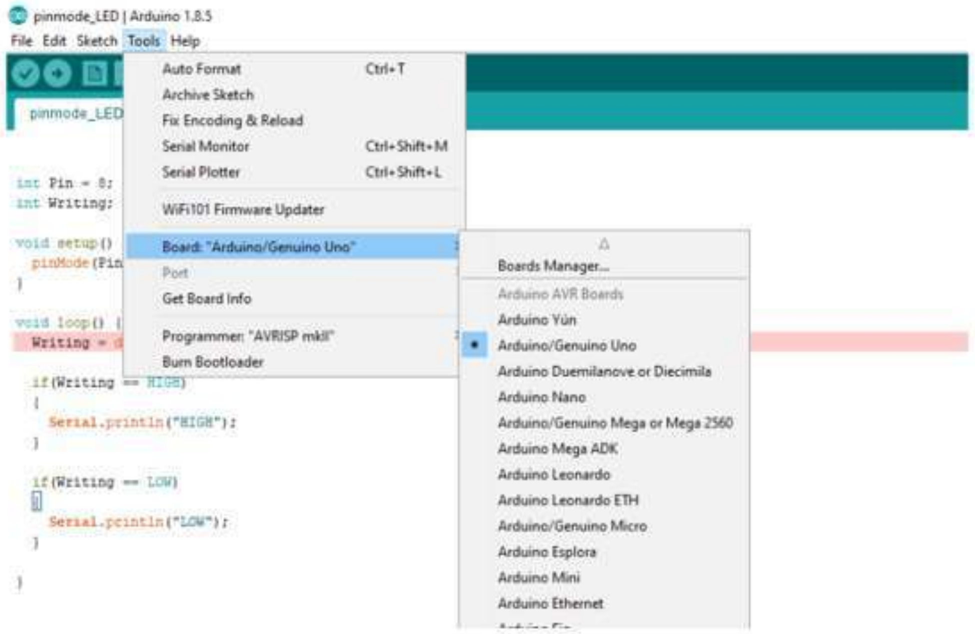


- Verify/Compile: This is the first button from the left (the tick mark). Click this button to verify and compile your code for correctness. You can view the results in the Status window at the bottom .
- Upload: This is the second button from left (right-pointing arrow). If your Arduino board is connected to your machine that is running the Arduino IDE, this will upload the code on the Arduino board. You can view the deployment results in the Status window at the bottom.

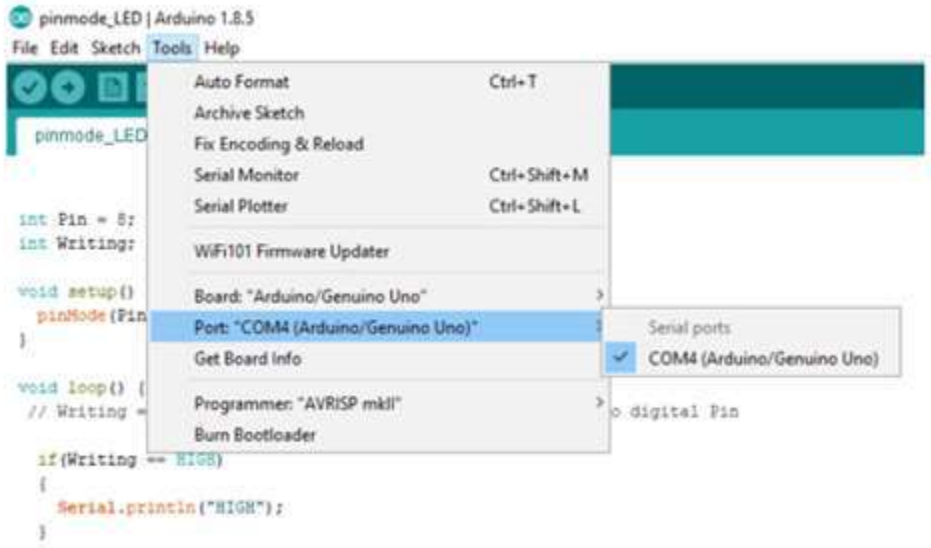
- New/Open/Save: The next three buttons, as their names suggest, let you open a new code window, open an existing code file, or save the currently open code. Arduino code files have an *.ino extension.

- Serial/Monitor: The last button on the right lets you open the Serial Monitor window.

How to Select the Board



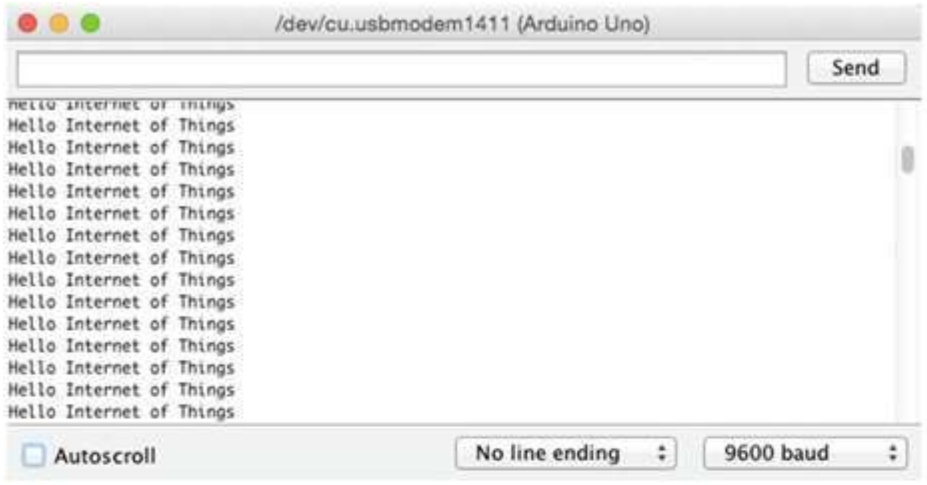
```
int Pin = 8;
int Writing;

void setup()
  pinMode(Pin
}

void loop() {
  Writing = d

  if(Writing == HIGH)
  {
    Serial.println("HIGH");
  }

  if(Writing == LOW)
  {
    Serial.println("LOW");
  }

}
```


pinmode_LED | Arduino 1.8.5
File Edit Sketch Tools Help
pinmode_LED
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM4 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
✓ COM4 (Arduino/Genuino Uno)

```
int Pin = 8;
int Writing;

void setup()
  pinMode(Pin
}

void loop() {
  // Writing =                o digital Pin

  if(Writing == HIGH)
  {
    Serial.println("HIGH");
  }
```

Just go to the "Board" section and select the board you aim to work on. Similarly, COM1, COM2, COM4, COM5, COM7 or higher are reserved for the serial and USB board. You can look for the USB serial device in the ports section of the Windows Device Manager.

Serial Monitor Window



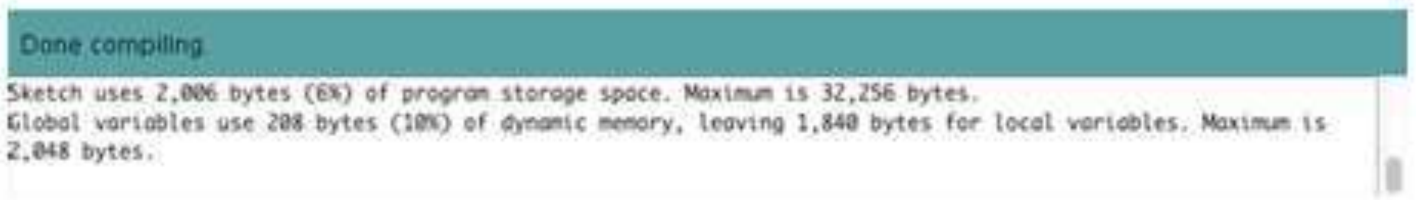
```
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
Hello Internet of Things
```

Autoscroll, No line ending, 9600 baud

Prints all log messages generated by the Serial.print() and Serial.println() functions in the code.

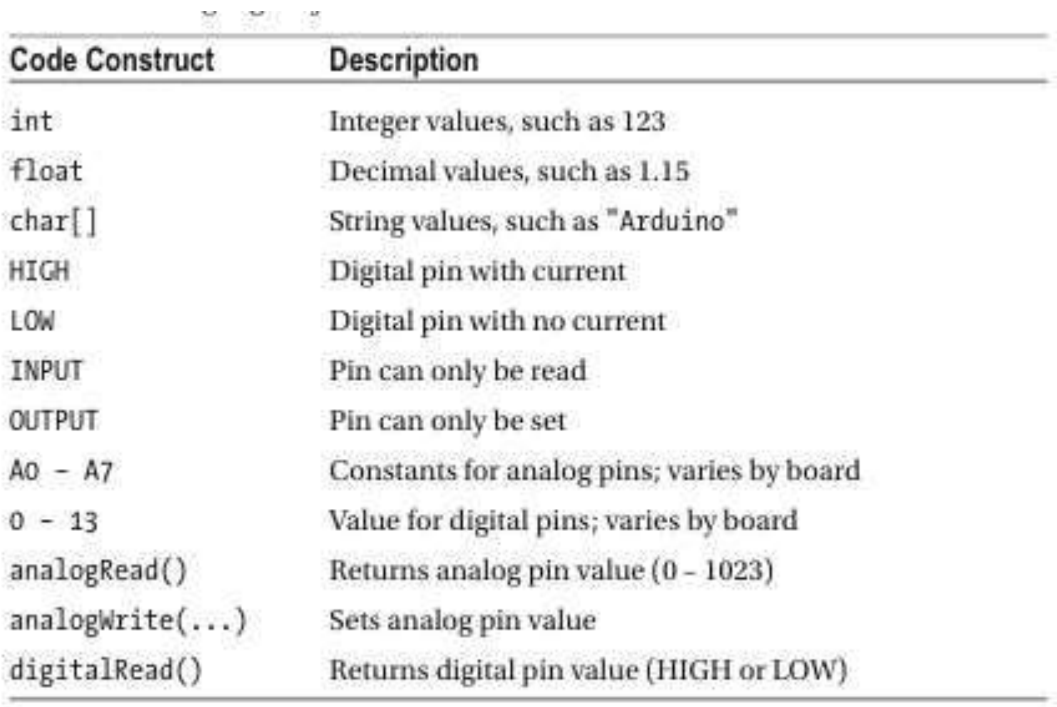
Status Window

When you verify the code or upload it to a board, the Status window shown in Figure lists all the results. Any errors that occur during code verification or uploading will be shown in the Status window .

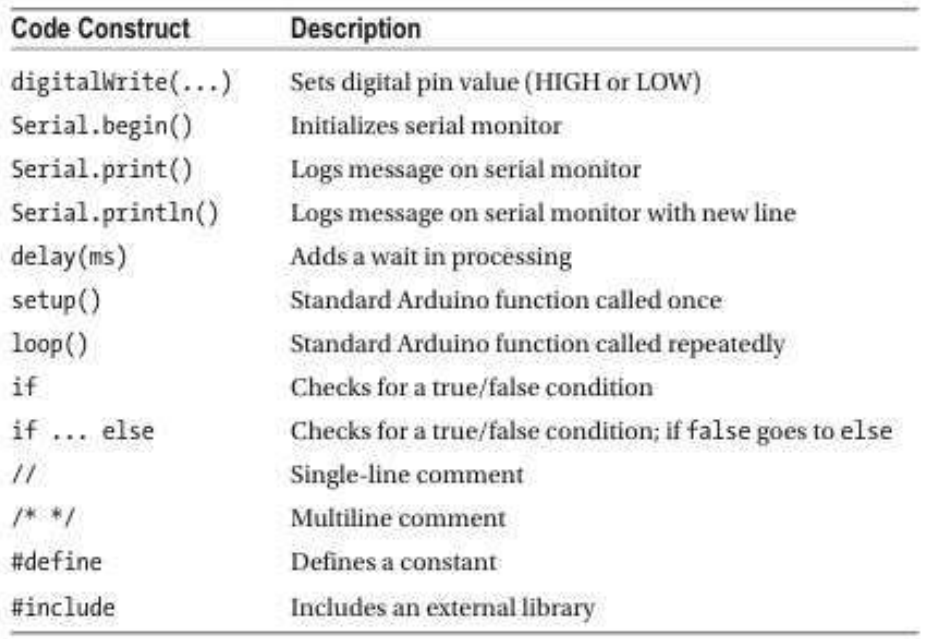


```
Done compiling.
Sketch uses 2,006 bytes (6%) of program storage space. Maximum is 32,256 bytes.
Global variables use 208 bytes (10%) of dynamic memory, leaving 1,840 bytes for local variables. Maximum is
2,048 bytes.
```

Arduino Programming Language Reference



Code Construct	Description
`int`	Integer values, such as 123
`float`	Decimal values, such as 1.15
`char[]`	String values, such as `"Arduino"`
`HIGH`	Digital pin with current
`LOW`	Digital pin with no current
`INPUT`	Pin can only be read
`OUTPUT`	Pin can only be set
`A0 - A7`	Constants for analog pins; varies by board
`0 - 13`	Value for digital pins; varies by board
`analogRead()`	Returns analog pin value (0 – 1023)
`analogWrite(...)`	Sets analog pin value
`digitalRead()`	Returns digital pin value (HIGH or LOW)



Code Construct	Description
`digitalWrite(...)`	Sets digital pin value (HIGH or LOW)
`Serial.begin()`	Initializes serial monitor
`Serial.print()`	Logs message on serial monitor
`Serial.println()`	Logs message on serial monitor with new line
`delay(ms)`	Adds a wait in processing
`setup()`	Standard Arduino function called once
`loop()`	Standard Arduino function called repeatedly
`if`	Checks for a true/false condition
`if ... else`	Checks for a true/false condition; if `false` goes to `else`
`//`	Single-line comment
`/* */`	Multiline comment
`#define`	Defines a constant
`#include`	Includes an external library

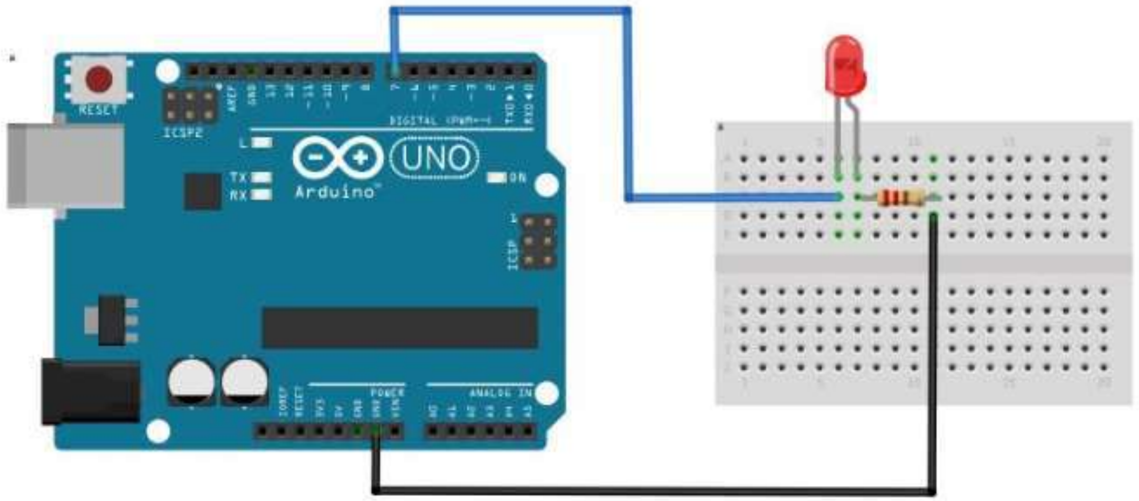
Arduino Sample Code

3. Aim:-Program to blink a LED Infinity time using Arduino board.

```
void setup()
{                                   // put your setup code here, to run once:
 pinMode(13,OUTPUT);

}


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

 digitalWrite(13,HIGH);
 delay(1000);
 digitalWrite(13,LOW);
 delay(1000);
}
```

OUTPUT:-

Connect LED of Anode to pin number 13 and cathode to ground terminal. Observe the LED blinking.



What is Arduino

- Arduino is a **microcontroller development board**. It has a microcontroller as its heart and some digital and analog Input Output pins. We can give some information/data in the form of voltage to Arduino via digital or analog input pins. And in the output pin, Arduino generates some control signal.

What is Microcontroller?

A Small Computer on a ChipA microcontroller is a compact integrated circuit (IC) that contains:

1. Processor: A central processing unit (CPU) that executes instructions.

2. Memory: Random-access memory (RAM) and read-only memory (ROM) for storing data and programs.

3. Input/Output (I/O) Peripherals: Interfaces for connecting to external devices, such as sensors, actuators, and communication modules.

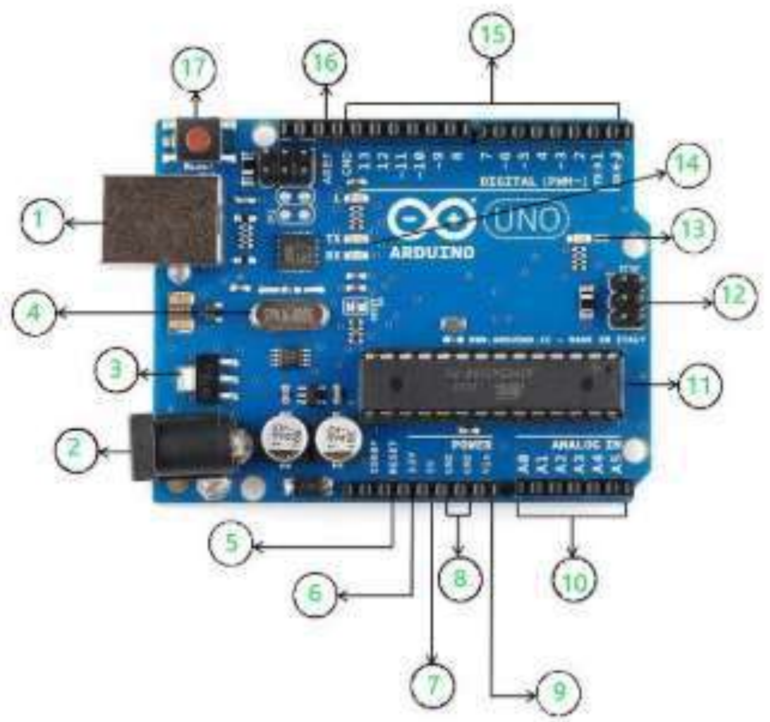
Examples of Microcontrollers:

1. Arduino Uno (ATmega328P)

2. ESP32 (Espressif Systems)

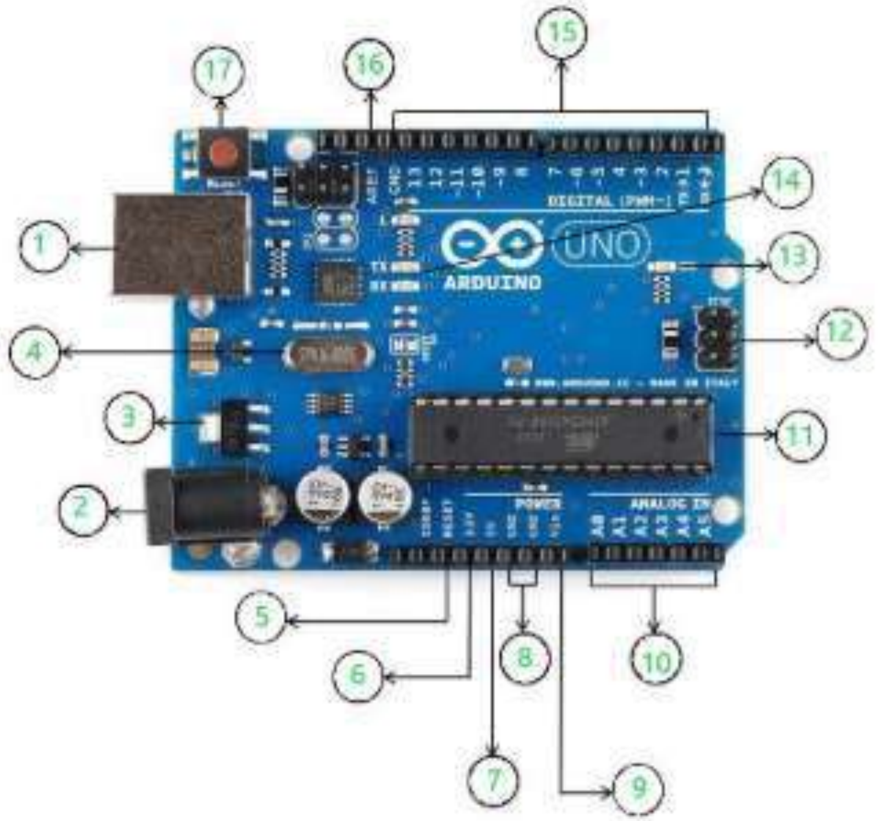
3. ESP8266 (Tensilica Diamond Micro CPU)

Arduino UNO Specs



1.USB: can be used for both power and communication with the IDE
2.Barrel J ack: used for power supply
3.Voltage Regulator: regulates and stabilizes the input and output voltages
4.Crystal Oscillator: keeps track of time and regulates processor frequency
5.Reset Pin: can be used to reset the Arduino Uno
6.3.3V pin: can be used as a 3.3V output
7.5V pin: can be used as a 5V output
8.GND pin: can be used to ground the circuit.
9.Vin pin: can be used to supply power to the board
10.Analog pins (A0-A5): can be used to read analog signals to the board

Arduino UNO Specs



11.Microcontroller(ATMega328): the processing and logical unit of the board

12.ICSP pin: a programming header on the board also called SPI

13.Power indicator LED: indicates the power status of the board

14.RX and TX LEDs: receive(RX) and transmit(TX) LEDs, blink when sending or receiving serial data respectively

15.Digital I/O pins: 14 pins capable of reading and outputting digital signals; 6 of these pins are also capable of PWM

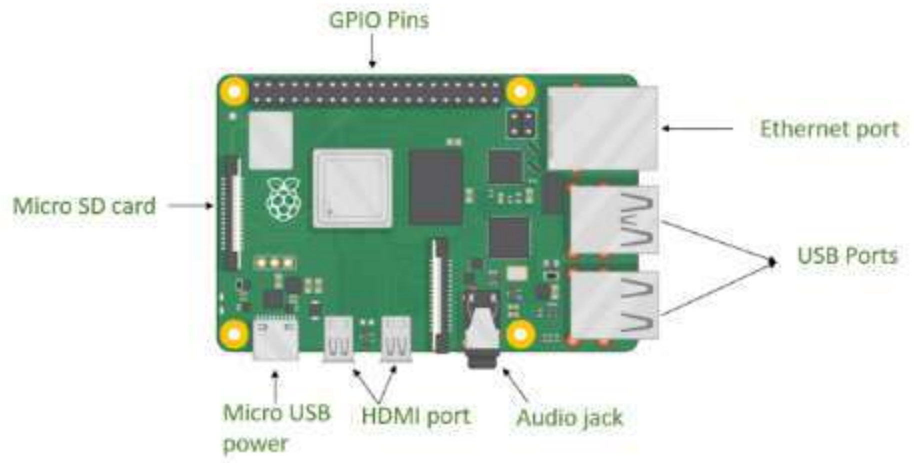
16.AREF pins: can be used to set an external reference voltage as the upper limit for the analog pins

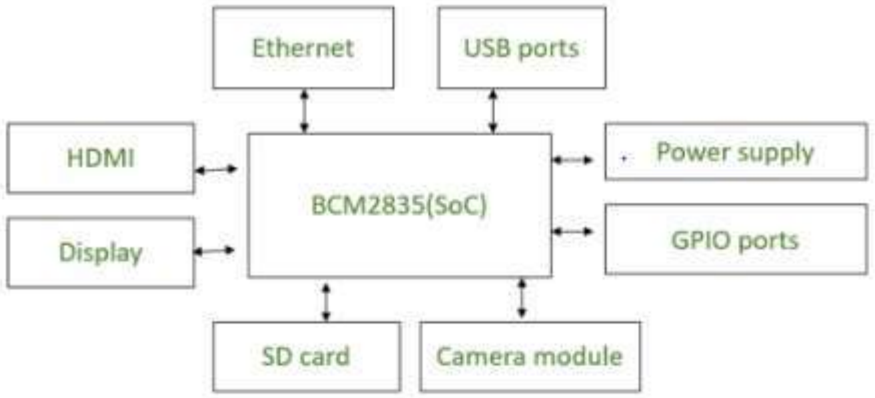
17.Reset button: can be used to reset the board

Raspberry PI

Architecture of Raspberry Pi

Raspberry Pi is a small single-board computer (SBC). It is a credit card-sized computer that can be plugged into a monitor. It acts as a minicomputer by connecting the keyboard, mouse, and display. Raspberry Pi has an ARM processor and 512MB of RAM.





- **Processor:** Raspberry Pi uses Broadcom BCM2835 system on chip which is an ARM processor and Video core Graphics Processing Unit (GPU). It is the heart of the Raspberry Pi which controls the operations of all the connected devices and handles all the required computations.

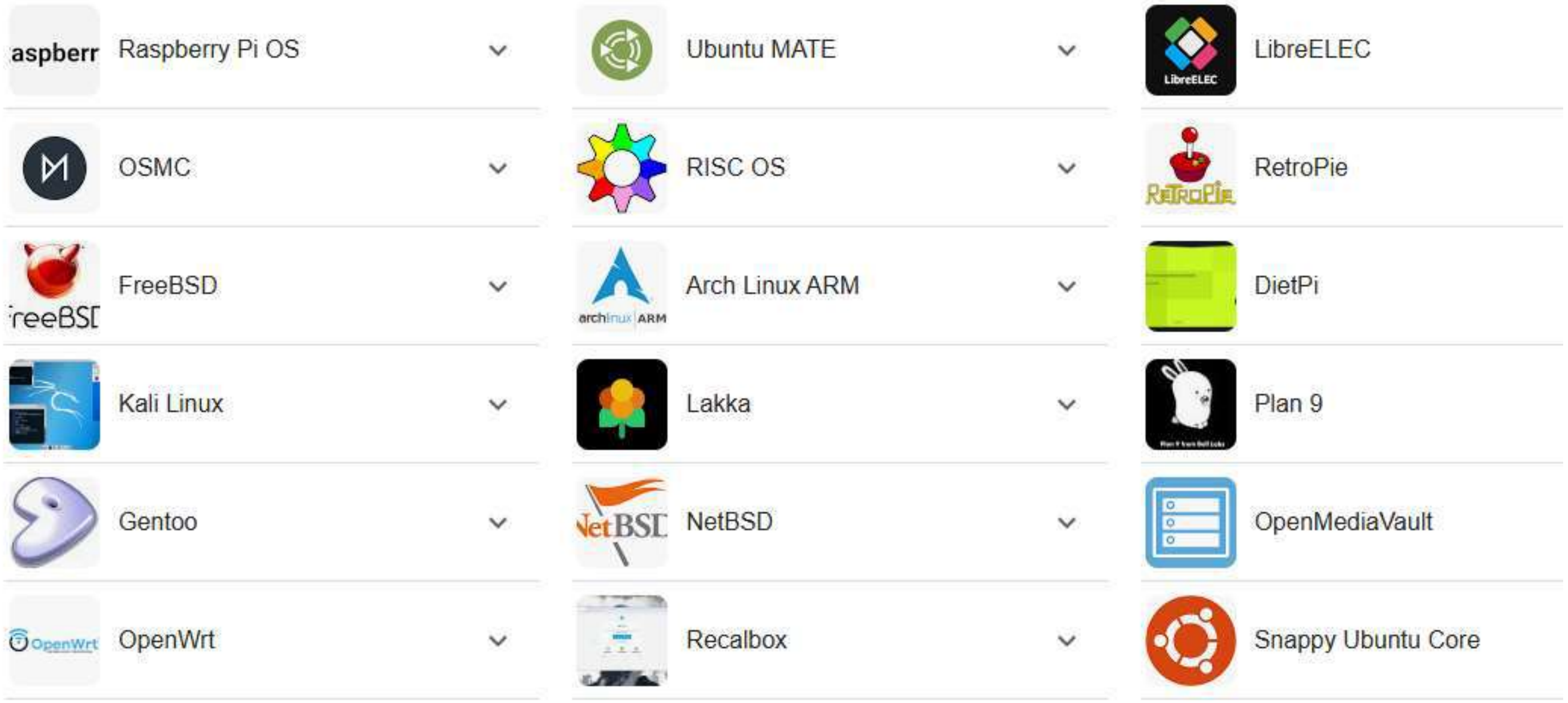
- **HDMI:** High Definition Multimedia Interface is used for transmitting video or digital audio data to a computer monitor or to digital TV. This HDMI port helps Raspberry Pi to connect its signals to any digital device such as a monitor digital TV or display through an HDMI cable.

- **GPIO ports:** General Purpose Input Output ports are available on Raspberry Pi which allows the user to interface various I/P devices.

- **Audio output**: An audio connector is available for connecting audio output devices such as headphones and speakers.

USB ports: This is a common port available for various peripherals such as a mouse, keyboard, or any other I/P device. With the help of a USB port, the system can be expanded by connecting more peripherals.
SD card: The SD card slot is available on Raspberry Pi. An SD card with an operating system installed is required for booting the device.
Ethernet: The ethernet connector allows access to the wired network, it is available only on the model B of Raspberry Pi.
Power supply: A micro USB power connector is available onto which a 5V power supply can be connected.
Camera module: Camera Serial Interface (CSI) connects the Broadcom processor to the Pi camera.
Display: Display Serial Interface (DSI) is used for connecting LCD to Raspberry Pi using 15 15-pin ribbon cables. DSI provides a high-resolution display interface that is specifically used for sending video data.

OS Supported by Raspberry Pi



Raspberry Pi OS	Ubuntu MATE	LibreELEC
OSMC	RISC OS	RetroPie
FreeBSD	Arch Linux ARM	DietPi
Kali Linux	Lakka	Plan 9
Gentoo	NetBSD	OpenMediaVault
OpenWrt	Recalbox	Snappy Ubuntu Core

Programming Raspberry Pi using Python

☑ 1. Set Up Raspberry Pi

•Install Raspberry Pi OS (formerly Raspbian) on a **microSD card**.

•Boot up and connect to a display, keyboard, and mouse (or use SSH/VNC remotely).

☑ 2. Open Python

•Raspberry Pi OS comes with **Python pre-installed** (usually Python 3).

•You can open:

- **Thonny Python IDE** (Beginner-friendly, found under Programming menu)
- Or open a **Terminal** and type:

```
python3
```

☑ 3. Write Your First Python Script

Example: Blink an LED using GPIO

4. Run the Script

Save the file as blink.py and run it in the terminal:

```
python3 blink.py
```

Common Python Libraries for Raspberry Pi Projects:

- RPi.GPIO – Control GPIO pins
- gpiozero – Higher-level GPIO control (easier for beginners)
- picamera – Control Pi Camera
- smbus – For I2C communication
- paho.mqtt – For IoT and MQTT communication

Wireless Temperature Monitoring System using Raspberry Pi

Basic Components:
•**Raspberry Pi** (any model with Wi-Fi)
•**DHT11 or DHT22** Temperature & Humidity sensor
•**Python** for programming
•**MQTT** or **HTTP** for wireless data transmission
Server/Cloud or a dashboard (like Node-RED, ThingsBoard, or Blynk)

📶 **How It Works:Sensor** reads temperature and humidity.
Raspberry Pi collects data using Python.
Data is **sent wirelessly** (via Wi-Fi) to a remote server or dashboard.
You can **monitor real-time temperature** on a smartphone, laptop, or display.

Wireless Temperature Monitoring System using Raspberry Pi

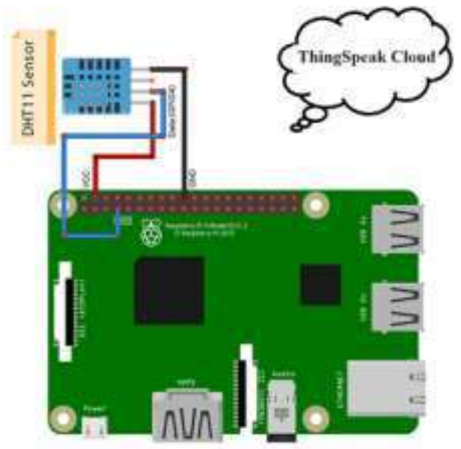
3) Thingspeak communication with the Hardware

1) Write about Thingspeak cloud set up.
2) How you integrate DHT11 sensor with Raspberry and Thingspeak.

HTTP Wireless Data Transmission used

1) Get Request from Server to client (Thingspeak to Raspberry)
2) Post Request from (Raspberry to Thingspeak).
3) Write the Code taught in lab.

Circuit Diagram



Raspberry Pi Interface with DHT-11 Sensor and Raspberry Pi is connected to the cloud

www.reva.edu.in

Wireless Temperature Monitoring System using MQTT Data Transmission

```
import Adafruit_DHT
import paho.mqtt.client as mqtt
import time

sensor = Adafruit_DHT.DHT11
pin = 4  # GPIO4

broker = "broker.hivemq.com"
topic = "iot/temperature"

client = mqtt.Client()
client.connect(broker, 1883, 60)
```

```
while True:
    humidity, temperature = Adafruit_DHT.read_retry(sensor, 
pin)
    if humidity and temperature:
        payload = f"Temp: {temperature}C, Humidity: {humidity}%"
        client.publish(topic, payload)
        print(payload)
    else:
        print("Failed to read from sensor")
    time.sleep(5)
```

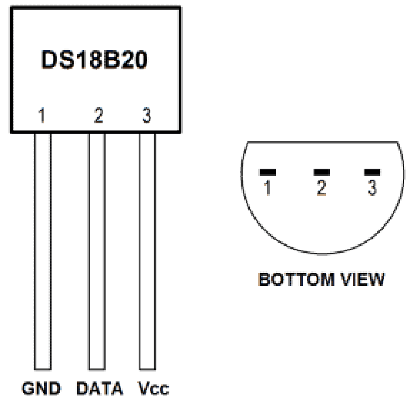
DS18B20 TEMPERATURE SENSOR

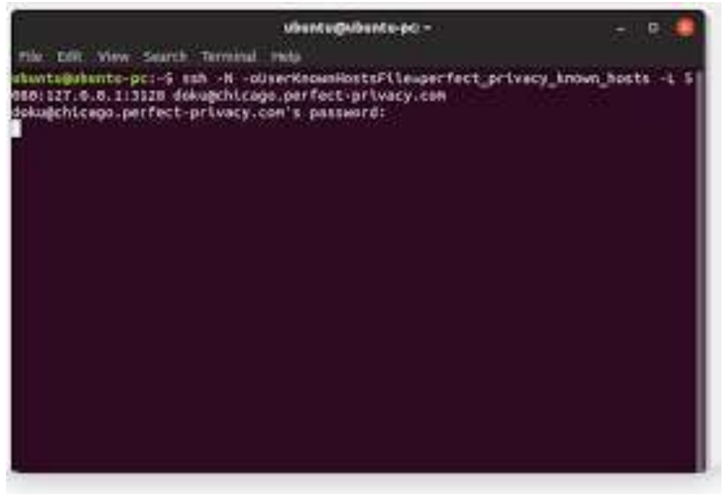
The **DS18B20** is a digital temperature sensor that communicates over a **1-Wire protocol**, allowing multiple devices to be connected to the same GPIO pin. It's widely used with Raspberry Pi and Arduino for accurate temperature measurement.

Key Features of DS18B20:

- Digital output (no analog pin needed)
- Temperature range: **–55°C to +125°C**
- Accuracy: ±0.5°C (typical)
- 9 to 12-bit resolution
- Unique 64-bit serial code for each sensor
- Can be powered by **parasite power** or an external 3.3V/5V supply

How to connect the DS18B20 to your Raspberry Pi and display the temperature readings on the SSH terminal





```
ubuntu@ubuntu-pc:~$ ssh -N -oUserKnownHostsFile=perfect_privacy_known_hosts -L 5
080:127.0.0.1:3128 doku@chicago.perfect-privacy.com
doku@chicago.perfect-privacy.com's password:
```

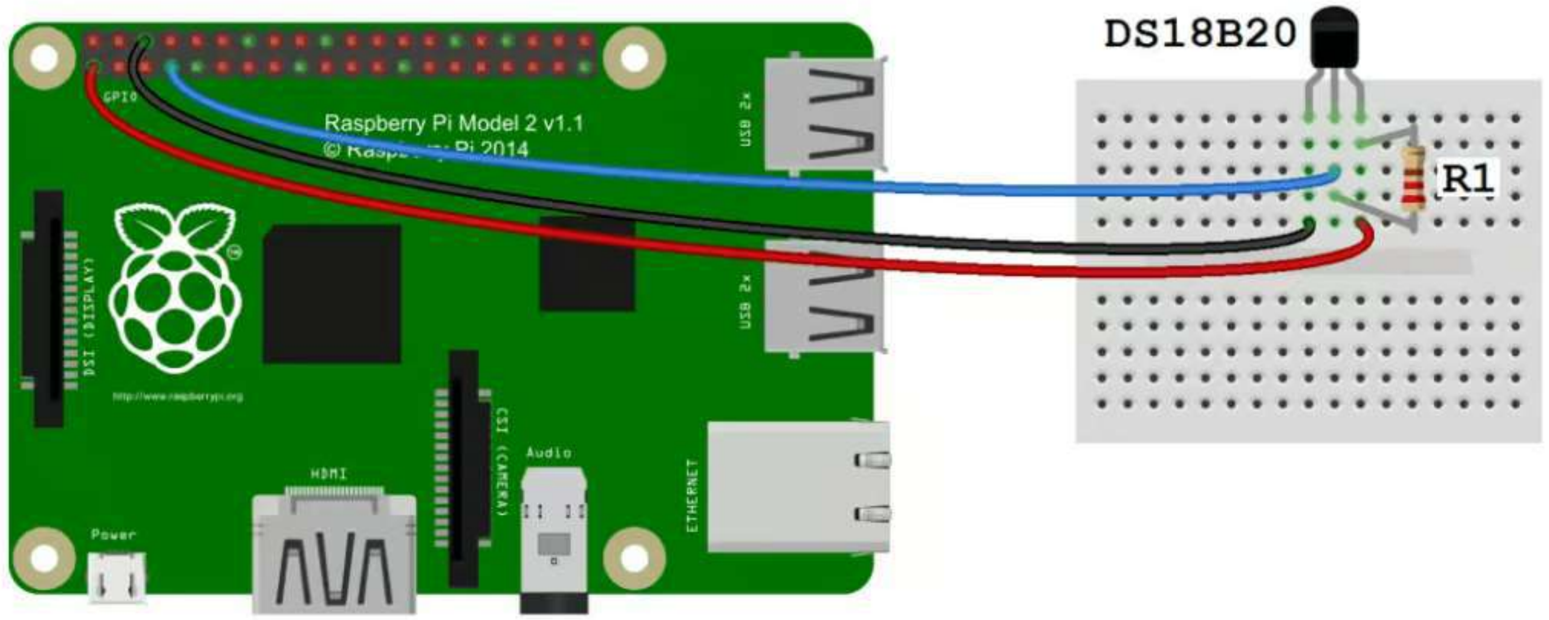
SSH Terminal

About the DS18B20

- The DS18B20 communicates with the “One-Wire” communication protocol, a proprietary serial communication protocol that uses only one wire to transmit the temperature readings to the microcontroller.
- . Normally the DS18B20 needs three wires for operation: the Vcc, ground, and data wires.
- The DS18B20 can be operated in what is known as parasite power mode. Where only the ground and data lines are used, and power is supplied through the data line.
- The DS18B20 also has an alarm function that can be configured to output a signal when the temperature crosses a high or low threshold that’s set by the user.
- A 64 bit ROM stores the device’s unique serial code. This 64 bit address allows Raspberry to receive temperature data from a virtually unlimited number of sensors at the same pin.

Wiring for SSH Terminal Output

Follow this wiring diagram to output the temperature to an SSH terminal:



R1: 4.7K Ohm or 10K Ohm resistor

Steps to Enable the One-Wire Interface

1) At the command prompt, enter sudo nano /boot/config.txt, then add this to the bottom of the file: **dtoverlay=w1-gpio**

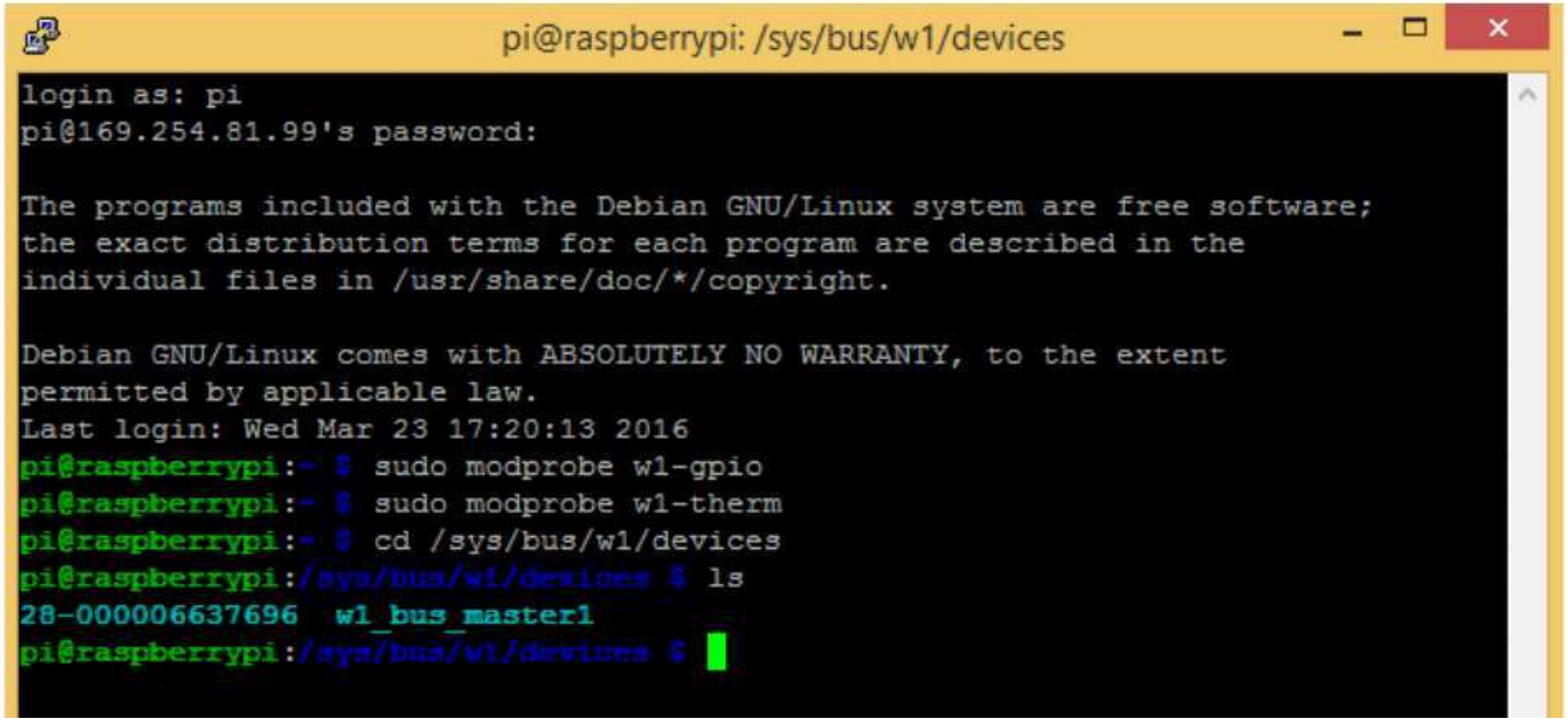
2)Exit Nano, and reboot the Pi with sudo reboot

3) Log in to the Pi again, and at the command prompt enter **sudo modprobe w1-gpio**

4) Then enter **sudo modprobe w1-therm**

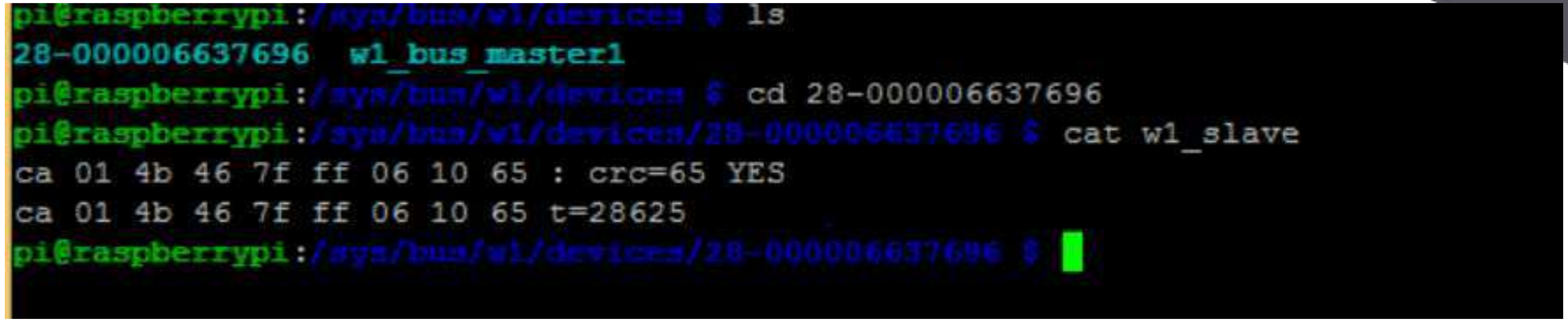
5) Change directories to the **/sys/bus/w1/devices** directory by entering **cd /sys/bus/w1/devices**

6) Now enter ls to list the devices:



```
login as: pi
pi@169.254.81.99's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 23 17:20:13 2016
pi@raspberrypi:~ $ sudo modprobe w1-gpio
pi@raspberrypi:~ $ sudo modprobe w1-therm
pi@raspberrypi:~ $ cd /sys/bus/w1/devices
pi@raspberrypi:/sys/bus/w1/devices $ ls
28-000006637696  w1_bus_master1
pi@raspberrypi:/sys/bus/w1/devices $
```

28-000006637696 w1_bus_master1 is displayed.



```
pi@raspberrypi:/sys/bus/w1/devices $ ls
28-000006637696  w1_bus_master1
pi@raspberrypi:/sys/bus/w1/devices $ cd 28-000006637696
pi@raspberrypi:/sys/bus/w1/devices/28-000006637696 $ cat w1_slave
ca 01 4b 46 7f ff 06 10 65 : crc=65 YES
ca 01 4b 46 7f ff 06 10 65 t=28625
pi@raspberrypi:/sys/bus/w1/devices/28-000006637696 $
```

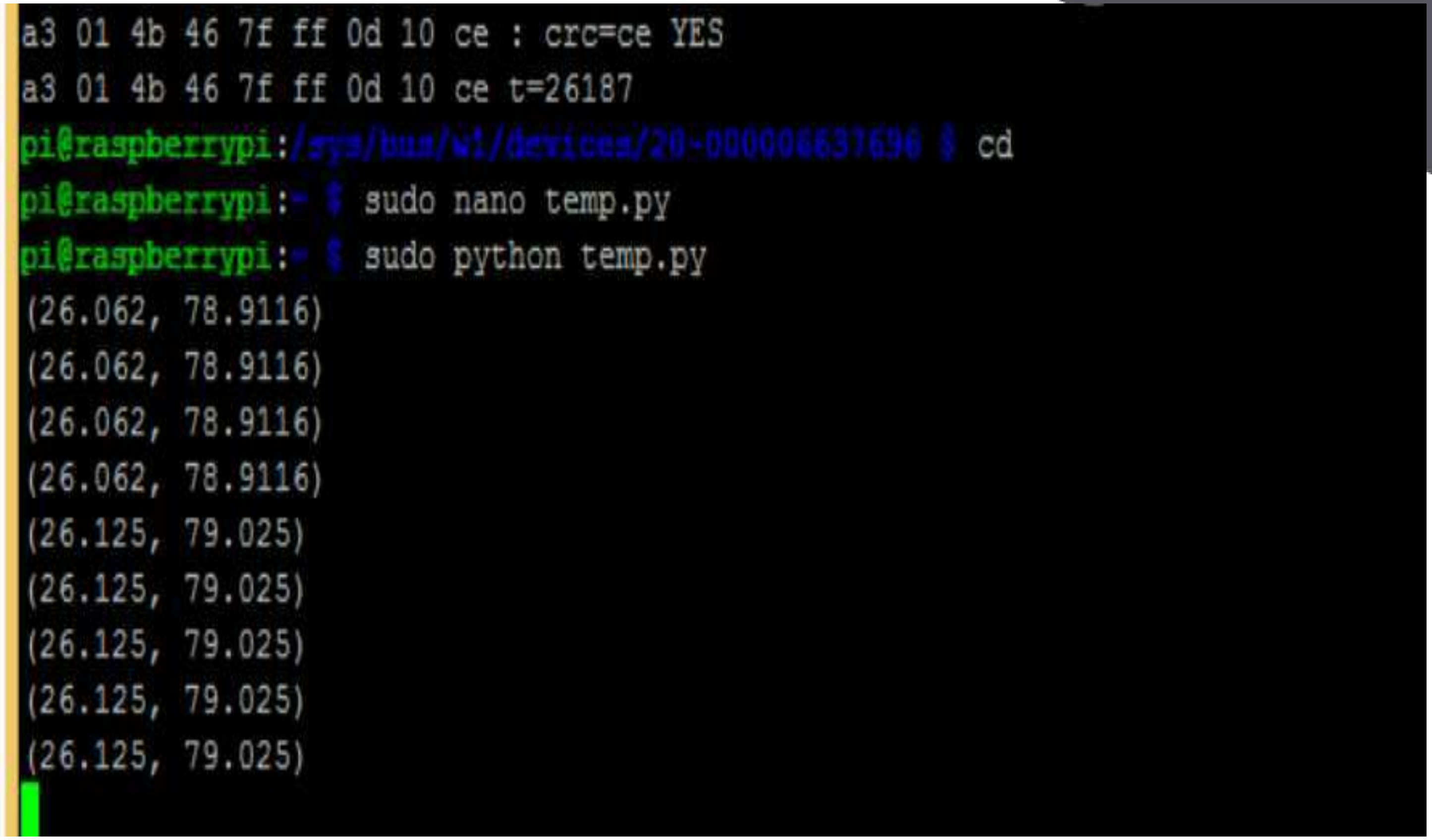
Enter cat w1_slave which will show the raw temperature reading output by the sensor:

Programming the Temperature Sensor

```
import os
import glob
import time

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines
```

Programming the Temperature Sensor

```
def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c, temp_f

while True:
        print(read_temp())
        time.sleep(1)
```



```
a3 01 4b 46 7f ff 0d 10 ce : crc=ce YES
a3 01 4b 46 7f ff 0d 10 ce t=26187
pi@raspberrypi:/sys/bus/w1/devices/28-000006637696 $ cd
pi@raspberrypi:~ $ sudo nano temp.py
pi@raspberrypi:~ $ sudo python temp.py
(26.062, 78.9116)
(26.062, 78.9116)
(26.062, 78.9116)
(26.062, 78.9116)
(26.125, 79.025)
(26.125, 79.025)
(26.125, 79.025)
(26.125, 79.025)
(26.125, 79.025)
```

Methods to Access Raspberry Pi Remotely

Method 1:- SSH (Secure Shell):

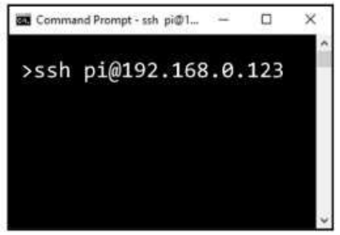
Purpose: Provides secure command-line access to your Raspberry Pi from another device.

How it works: SSH uses encryption to create a secure connection, allowing you to run commands and manage your Raspberry Pi as if you were physically present.

Requirements: You need to enable SSH on your Raspberry Pi and know its IP address.

Benefits: A powerful tool for automating tasks, making configuration changes, and managing your Raspberry Pi without a physical keyboard or mouse.

Drawbacks: Requires more technical knowledge and is primarily a text-based interface.



```
>ssh pi@192.168.0.123
```

Method 2:- VNC (Virtual Network Computing):

Purpose: Allows you to view and control the graphical user interface (GUI) of your Raspberry Pi from another device.

How it works: VNC uses a protocol that streams the display of your Raspberry Pi to a client device, allowing you to interact with it as if it were a normal remote desktop.

Requirements: You need to install a VNC server on your Raspberry Pi and a VNC viewer client on your other device.

Benefits: Provides a graphical interface, making it easier to manage your Raspberry Pi and use its applications.

Drawbacks: Can be more resource-intensive than SSH and may not work as well on low-bandwidth connections.

- **Method 3:- Raspberry Pi Connect:**

Purpose: Offers a simple, browser-based way to access your Raspberry Pi from anywhere.

How it works: Raspberry Pi Connect provides a free, cloud-based service that allows you to connect to your Raspberry Pi's shell or screen share its desktop from a web browser.

Requirements :You need to enable Raspberry Pi Connect on your Raspberry Pi and create a Raspberry Pi ID.

Benefits: User-friendly, accessible from any internet-connected device, and requires minimal setup.

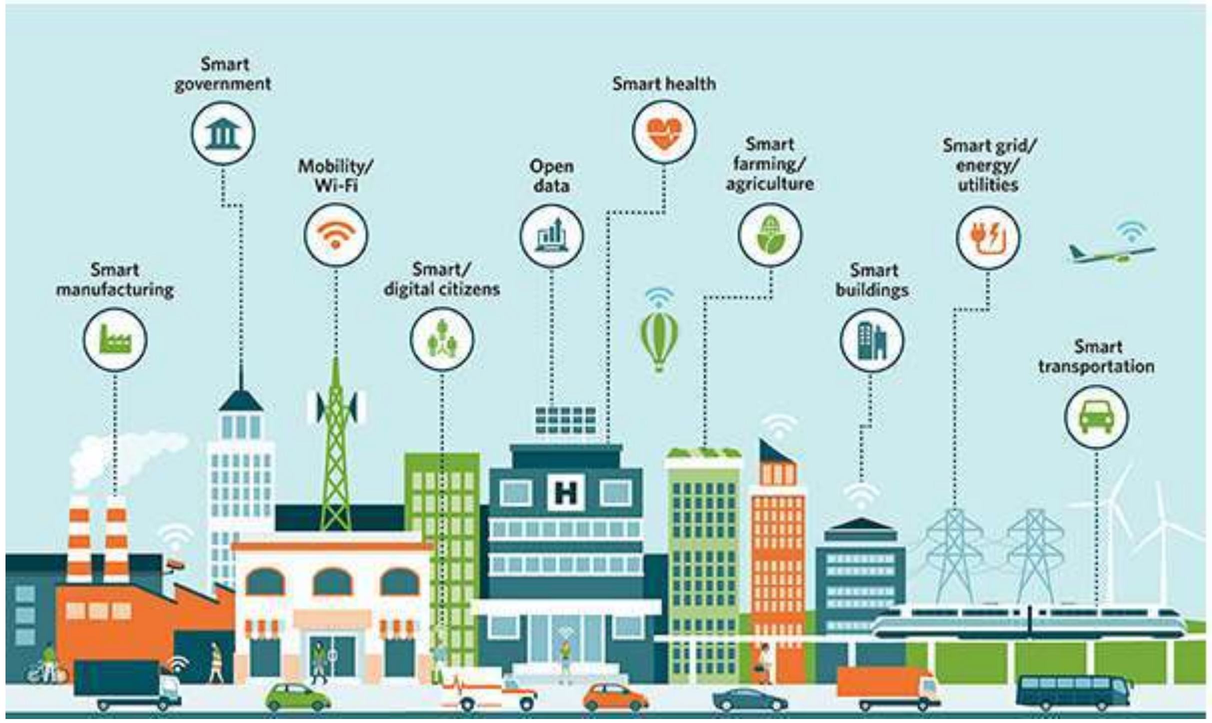
Drawbacks: Relies on a cloud service and may not be as secure or powerful as SSH or VNC.

Arduino Vs Raspberry Pi

Parameters	Arduino	Raspberry Pie
Control Unit	The Control Unit of the Arduino is from the ATmega family.	The Control Unit of the Raspberry Pi is from the ARM family.
Basis	Arduino works on the basis of a microcontroller.	Raspberry Pi, on the other hand, works on the basis of a microprocessor.
Use	The Arduino basically helps in controlling all the electrical components that connect to a system's circuit board.	The Raspberry Pi primarily computes data and info for producing valuable outputs. It also controls the various components in any given system on the basis of the outcome (of the computation).
Structure of Hardware and Software	The Arduino boards have a very simple structure of software and hardware.	The Raspberry Pi boards consist of comparatively complex software and hardware architecture.

Parameters	Arduino	Raspberry Pie
Type of CPU Architecture	Arduino has an 8-bit architecture.	Raspberry Pi has a 64-bit architecture.
RAM Usage	Arduino makes use of very little RAM of about 2 kB (Kilobytes).	Raspberry Pi always requires more RAM than Arduino of about 1 GB (Gigabytes).
Processing Speed	Arduino clocks 16 MHz (Megahertz) of processing speed in a system.	The Raspberry Pi clocks 1.4 GHz (Gigahertz) of processing speed in a system.
Cost Efficiency	It has a higher cost-efficiency because it is comparatively cheaper.	It has a lower cost-efficiency because it is comparatively more expensive.
I/O Drive Strength	The I/O current drive strength in the case of Arduino is higher.	The I/O current drive strength in the case of Raspberry Pi is lower.
Power Consumption	Arduino consumes power of about 200 MW (Megawatts).	Raspberry Pi consumes about 700 MW.

Smart & Connected Cities



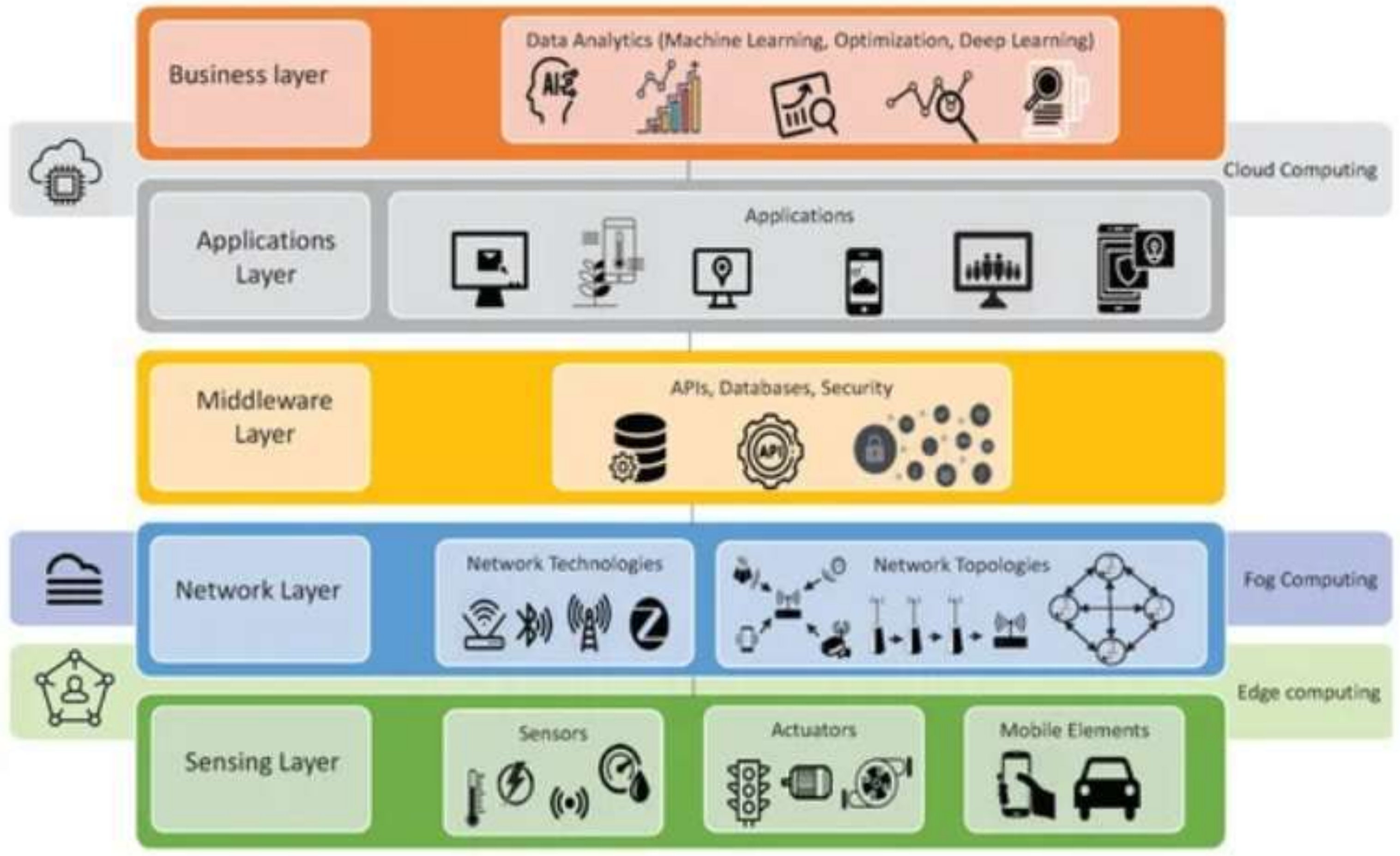
What makes a City considered as Smart?
Importance of Smart Cites
IOT Strategies for Smart Cites

1) https://blog.getmyparking.com/2021/02/09/iot-in-smart-cities-how-iot-is-raising-the-standard-of-city-living/

2) https://www.linkedin.com/pulse/definition-development-smart-cities-seven-ren/

3) https://www.linkedin.com/pulse/smart-city-pavithra-h/

Smart City IOT Architecture



Smart City IOT Architecture

1. **Sensing / Device Layer:**
- IoT Devices: Sensors, actuators, wearables, and other smart devices that collect and transmit data.
- Connectivity: Communication protocols, network technologies (like Wi-Fi, cellular, LPWAN), and device management for connecting devices to the network.

2. **Network / Transport Layer:**
- Communication Networks: Wired and wireless networks that enable data transmission between devices, gateways, and central systems.
- Protocols: MQTT, CoAP, HTTP, LoRaWAN, NB-IoT, and more for efficient and secure data exchange.

Smart City IOT Architecture

3. Middleware / Data Processing Layer:
- Data Ingestion: Receiving, validating, and transforming data from devices for further processing.
- Data Storage: Storing collected data in databases, data lakes, or cloud storage for analysis.
- Stream Processing: Real-time data processing to extract insights, detect anomalies, and trigger actions.

4. Application / Service Layer:
- IoT Applications: Developing applications that utilize IoT data to deliver value to users and organizations.
- Dashboard and Visualization: Displaying insights and data through user-friendly interfaces.
- Alerts and Notifications: Notifying users or systems based on predefined conditions.

5. Business / Business Logic Layer:
- Business Logic: Implementing logic to process IoT data based on business rules and requirements.
- Workflow Automation: Automating processes and workflows based on IoT data triggers.

Smart City Use Case Example

https://www.scnsoft.com/blog/iot-for-smart-city-use-cases-approaches-outcomes

Thank You