Internet of Things
(Assignment)

Q-A) Explain Architecture and programming of Arduino.

Arduino

- Arduino is Hardware and Software company, project and user community that designs and manufactures single board Micro-Controllers and kits for building digital devices.

- Arduino board designs use a variety of Microprocessors and controllers.

- It's hardware products are licensed under a Creative commons license, while software is licensed under the GNU Lesser General Public License (LGPL) permitting the manufacture of Arduino boards and software distribution by anyone.

- The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards or breadboards and other circuits.

- The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers.

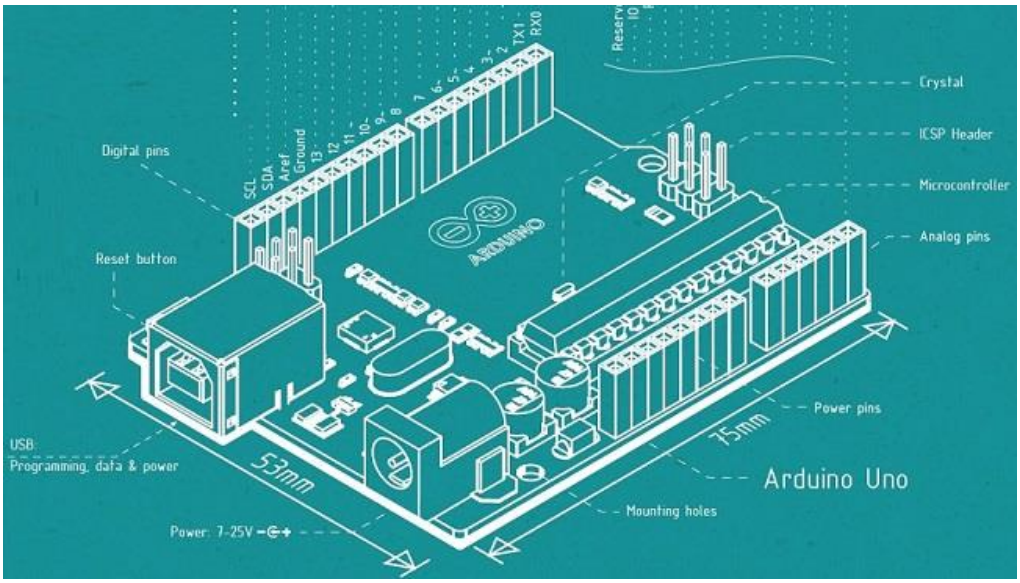


Fig.01-> Arduino Uno

- The microcontrollers can be programmed using the C and C++ programming languages, using a standard API which is also known as the "Arduino language".

- In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) and a command line tool (Arduino- cli) developed in Go.

Arduino Architecture

- Basically, the processor of the Arduino board uses the Harvard architecture where the program code and program data have separate memory. It consists of two memories such as program memory and data memory.
- Wherein the data is stored in data memory and the code is stored in the flash program memory. The Atmega328 microcontroller has 32kb of flash memory, 2kb of SRAM 1kb of EPROM and operates with a 16MHz clock speed.

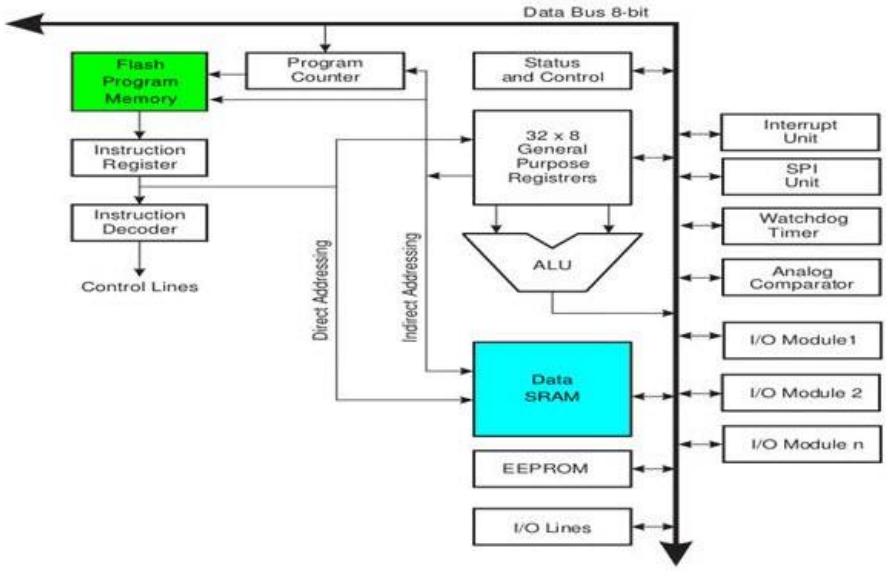
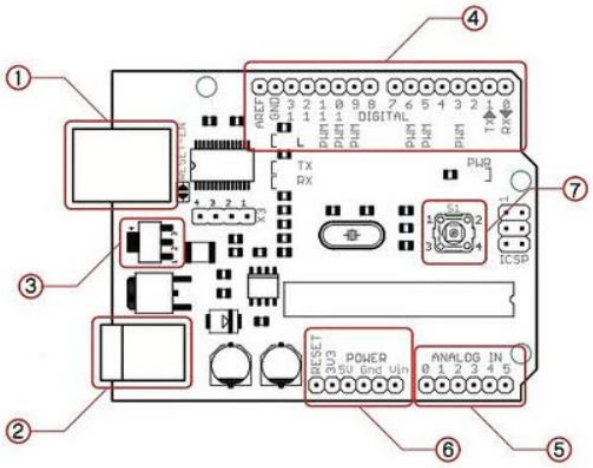


Fig.02 -> Arduino Architecture

Circuit Explanation:

Arduino board technology which is Arduino UNO and it contains ATmega328 with 28 pins. The following figure shows the pin diagram of Arduino UNO board architecture. The Arduino UNO consists of 14 digital input/output pins, from these pins the six pins are used for the o/p pins of pulse width modulation, and six pins are used for analogue input pins, ICSP header, power jack, USB connection, 16MHz crystal oscillator.



Arduino Board Specifications

The Arduino board can be powered from the personal computer by using the USB or external source like a battery or adapter. The operation of this board is by using the external power supply of 7-12V with the voltage reference of the IO Ref pin or through the Vin pin.

1) **Digital Input Output Pins:** There are 14 digital input, output pins and for each pin, there is a current of 40mA. For some pins there are special functions like pins 0 & 1, they are acting as a transmitter and receiver respectively. The serial communication ports have the external interrupts of pins 2 & 3 and the delivery of PWM o/p is from the 3,5,6,9,11 pins. The LED is connected to the pin 13.

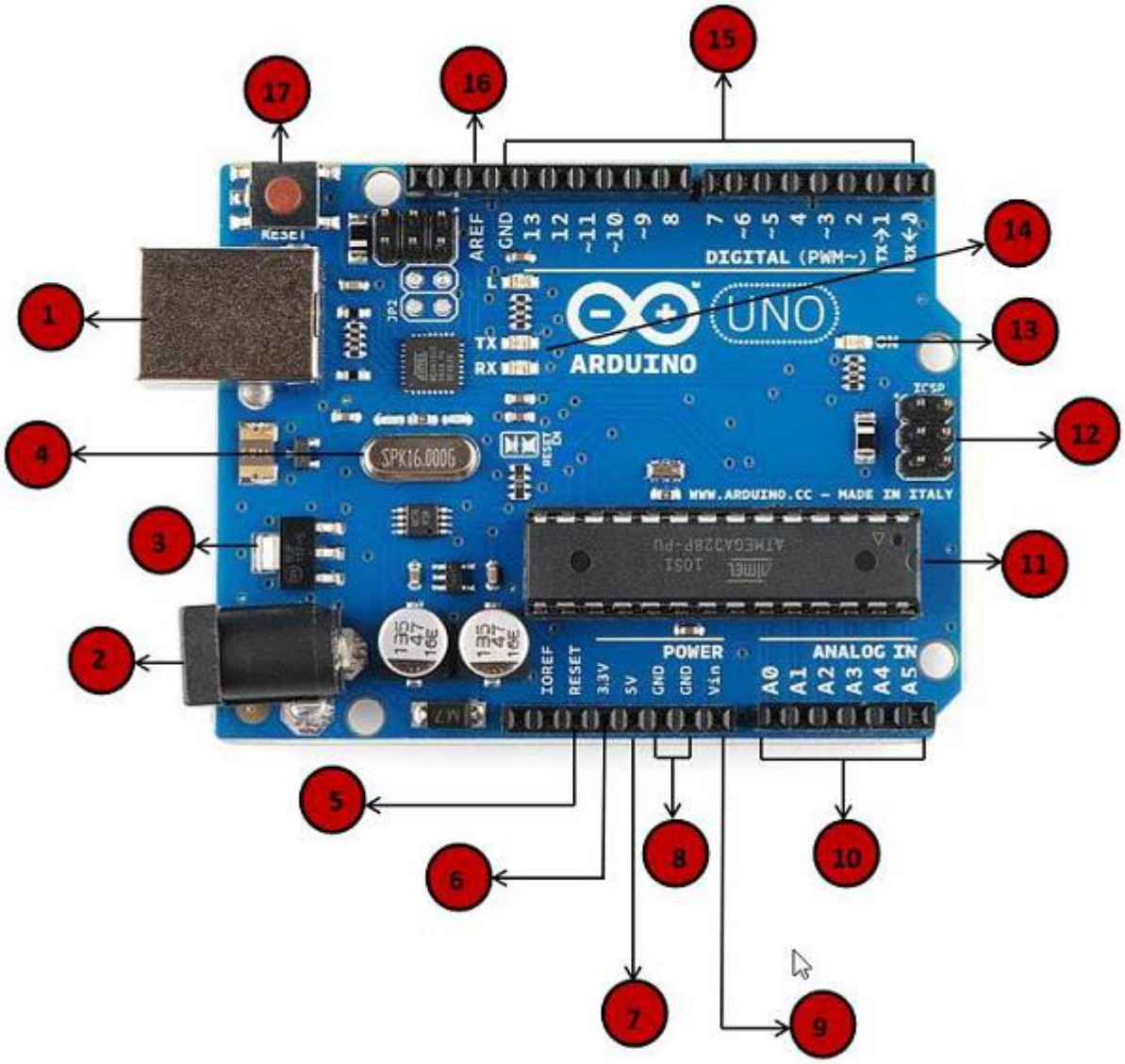
The serial communication ports have the external interrupts of pins 2 & 3 and the delivery of PWM o/p is from the 3,5,6,9,11 pins. The LED is connected to the pin 13.

2) **Analogue inputs:** The analogue input pins consist of 6 analogues I/O pin and for each pin are provided by the 10 bits resolution.

3) **AREF pin:** This pin will give the reference to the analogue I/P's.

4) **Reset Pin:** The reset pin resets the Microcontroller when the pin is low.

Arduino Circuit Pin IN/OUT:



Label -> Instructions	
1-> Power USB	10-> Analog pins
2-> Power (Barrel Jack)	11-> Main microcontroller
3-> Voltage Regulator	12-> ICSP pin
4-> Crystal Oscillator	13-> Power LED indicator
5,17-> Arduino Reset	14-> TX and RX LEDs
6,7,8,9 -> Pins (Supply 3.3 output volt)	15-> Digital I/O
16-> AREF stands for Analog Reference	

Application of Arduino:

1) Security System
2) Traffic Light Count Down Timer
3) Parking Lot Counter
4) Weighing Machines
5) Medical Instrument
6) Emergency Light for Railways
7) Animatronics Hand
8) Smart Irrigation System using Arduino
9) Weather Monitoring System
10) Automatic Solar Tracker
11) Biometric Authentication System
12) Smart Energy Meter using GSM
13) Wi-Fi Controlled Robot
14) Robotic Arm
15) Persistence of Vision
16) Health Monitoring Wearable Glove
17) GPS & GSM based Tracker

And many more IOT device that's uses sensor to program using Arduino.

➢ Advantage

The biggest advantage of Arduino is its ready to use structure.

As Arduino comes in a complete package form which includes the 5V regulator, a burner, an oscillator, a micro-controller, serial communication interface, LED and headers for the connections.

You don't have to think about programmer connections for programming or any other interface.

Arduino Programming

The Arduino Software (IDE) runs on Windows, Macintosh OSX and Linux Operating Systems. But most microcontroller system software requirements are limited to Windows only. The Arduino Software is provided as an open-source tool for the beginners and students to write and upload the program onto the microcontroller. It is similar to the processing programming environment so you can get used to the interface immediately.

The Arduino programming language can be extended to multiple C++ libraries and you can also use AVR-C programming on this system. With this, you will be able to add AVR-C code onto the Arduino Programme directly.

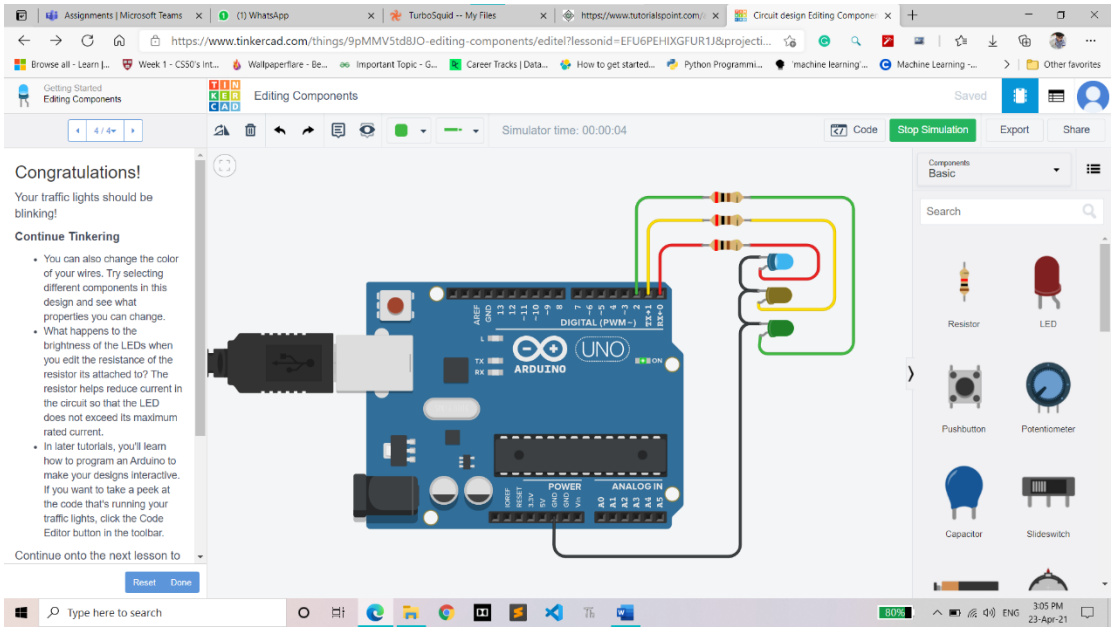
Sketches' is the term used to denote the programming of the Arduino Board. Each sketch contains three parts - Variables Declaration, Initialization and Control code. Following are the steps to programme the Arduino Board:

- Initialization is written in the setup function and Control code is written in the loop function.
- The sketch is saved with .ino format and operations like opening a sketch, verifying and saving can be done using the tool menu.
- The sketch must be stored in the sketchbook directory.
- Select the suitable board from the serial port numbers and tools menu.
- Select the tools menu and click on the upload button, then the boot loader uploads the code on the microcontroller.

IDE

Now let's take one example and implement that's by Arduino circuits

1) LED BLINKING

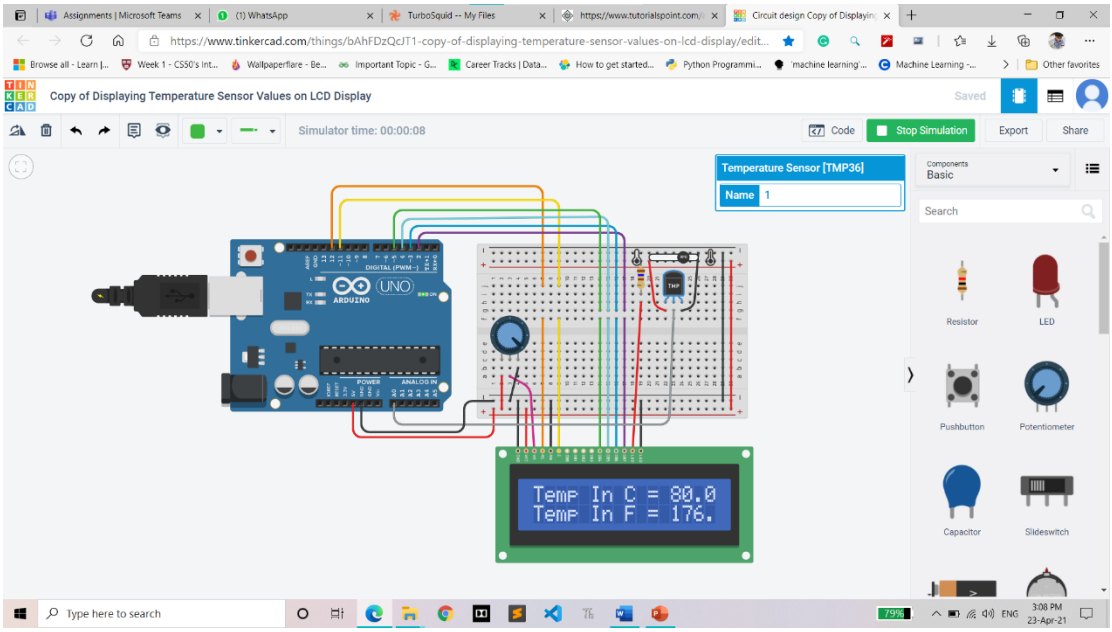


CODE:

```
int led_red = 0; // the red LED is connected to Pin 0 of the Arduino
int led_yellow = 1; // the yellow LED is connected to Pin 1 of the Arduino
int led_green = 2; // the green LED is connected to Pin 2 of the Arduino

void setup() {
 // set up all the LEDs as OUTPUT
 pinMode(led_red, OUTPUT);
 pinMode(led_yellow, OUTPUT);
 pinMode(led_green, OUTPUT);
}

void loop() {
 // turn the green LED on and the other LEDs off
 digitalWrite(led_red, LOW);
 digitalWrite(led_yellow, LOW);
 digitalWrite(led_green, HIGH);
 delay(2000);    // wait 2 seconds

 // turn the yellow LED on and the other LEDs off
 digitalWrite(led_red, LOW);
 digitalWrite(led_yellow, HIGH);
 digitalWrite(led_green, LOW);
 delay(1000);   // wait 1 second
```

```
 // turn the red LED on and the other LEDs off
 digitalWrite(led_red, HIGH);
 digitalWrite(led_yellow, LOW);
 digitalWrite(led_green, LOW);
 delay(3000);  // wait 3 seconds
}
```

2) TEMPERATURE DETECTION



CODE:

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int tempsensor = A0; // Assigning analog pin A0 to variable
'tempsensor'
float tempc;  //variable to store temperature in degree Celsius
float tempf;  //variable to store temperature in Fahreinheit
float vout;   //temporary variable to hold sensor reading

void setup() {
  pinMode(tempsensor,INPUT); // Configuring pin A0 as input
  Serial.begin(9600);
```

```
  lcd.begin(16,2);
  delay(500);
}

void loop() {
  vout = analogRead(tempsensor);
  vout = map(((analogRead(A0) - 20) * 3.04), 0, 1023, -40, 125);
  tempc = vout; // Storing value in Degree Celsius
  tempf = (vout*1.8)+32; // Converting to Fahrenheit
  lcd.setCursor(0,0);
  lcd.print("Temp In C = ");
  lcd.print(tempc);
  lcd.setCursor(0,1);
  lcd.print("Temp In F = ");
  lcd.print(tempf);
  delay(1000); //Delay of 1 second for ease of viewing in serial
monitor
}
```

Q-2) Explain Architecture and programming of Raspberry Pi.

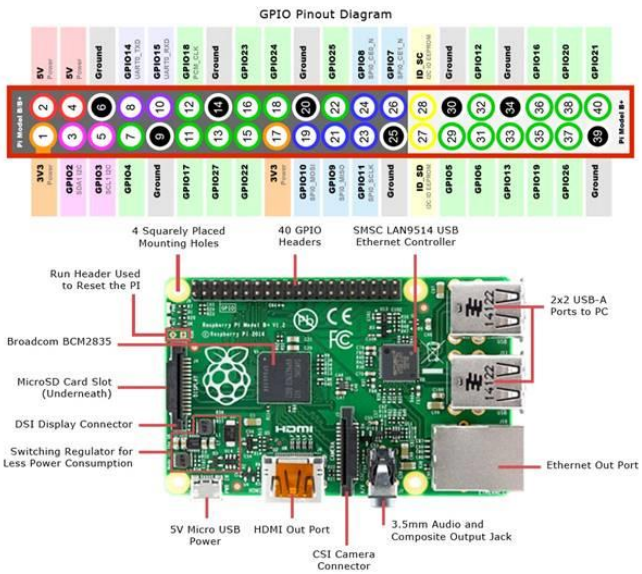
➢

Raspberry Pi (Mini Computer)

- Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom.
- Early on, the Raspberry Pi project leaned towards the promotion of teaching basic computer science in schools and in developing countries.
- Later, the original model became far more popular than anticipated, selling outside its target market for uses such as robotics.
- It is now widely used in many areas, such as for weather monitoring, because of its low cost, modularity, and open design.

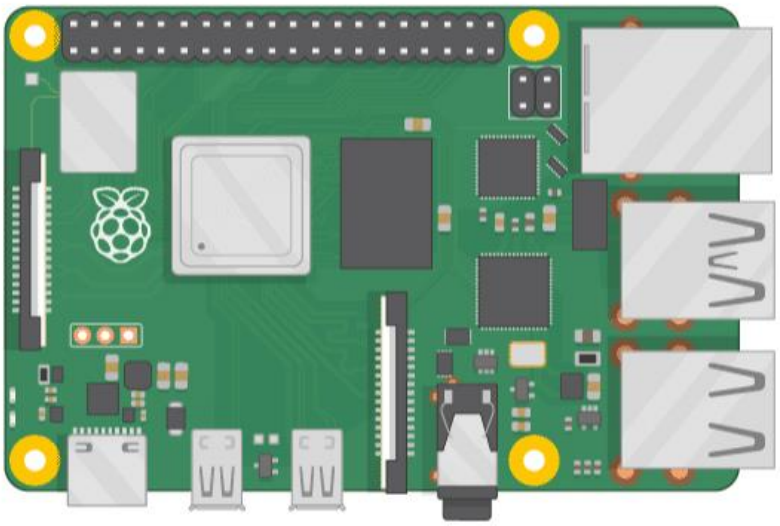
- After the release of the second board type, the Raspberry Pi Foundation set up a new entity, named Raspberry Pi.
- The Foundation was rededicated as an educational charity for promoting the teaching of basic computer science in schools and developing countries.

Raspberry Pi Circuit



Connect Raspberry

- The Raspberry P ngs. You plug it into a monitor and attach a keyboard and mouse.



- We list down all the hardware can be joined with Raspberry Pie circuit and Used OS within the SD card.

Hardware

- A Raspberry Pi computer with an SD card or micro-SD card
- A monitor with a cable (and, if needed, an HDMI adaptor)
- A USB keyboard and mouse
- A power supply
- Headphones or speakers (optional)
- An ethernet cable (optional)

Software

- Raspberry Pi OS, installed using the Raspberry Pi Imager

❖ **Install Raspberry Pi OS on your SD card with the Raspberry Pi Imager**

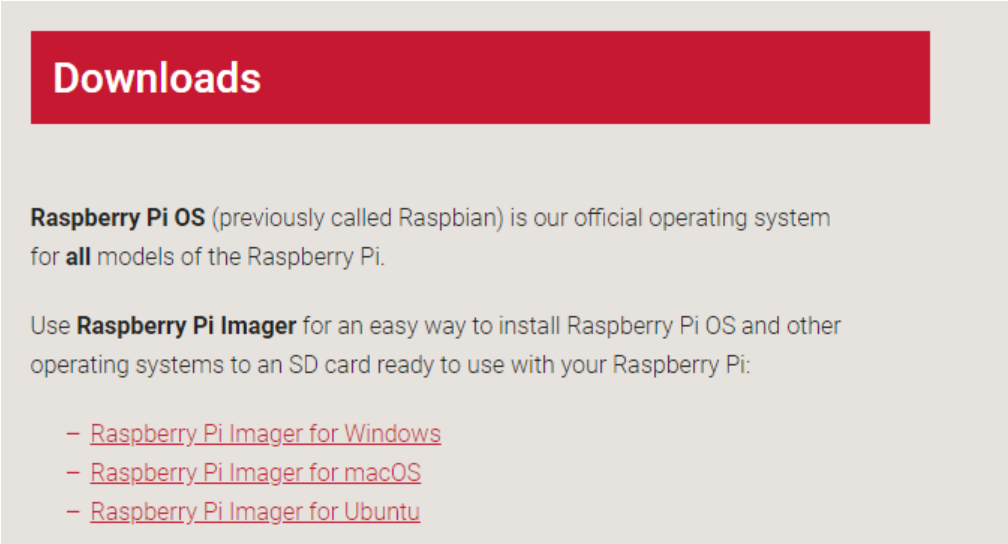
Many vendors sell SD cards with a simple Raspberry Pi OS installer called NOOBS preinstalled but you can really easily install Raspberry Pi OS yourself using a computer that has an SD card port or using an SD card reader.

Using the Raspberry Pi Imager is the easiest way to install Raspberry Pi OS on your SD card.

installing operating system images

Download and launch the Raspberry Pi Imager

- Visit the Raspberry Pi downloads page.
- Click on the link for the Raspberry Pi Imager that matches your operating system.



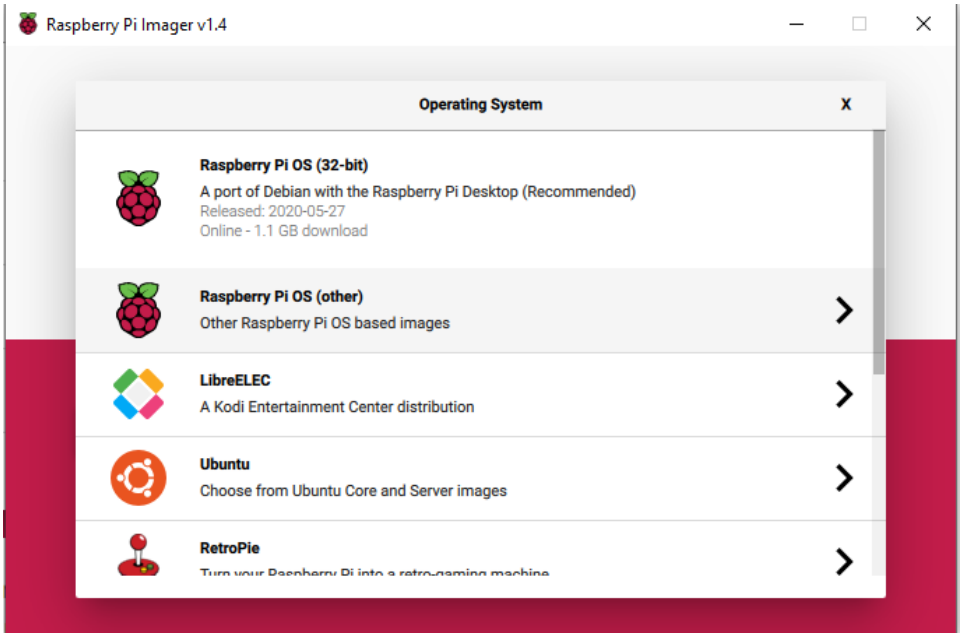
- When the download finishes, click on it to launch the installer.

Using the Raspberry Pi Imager

All data stored on the SD card will be overwritten during formatting and lost permanently, so make sure that you back up the card or any files you want to keep before running the installer.

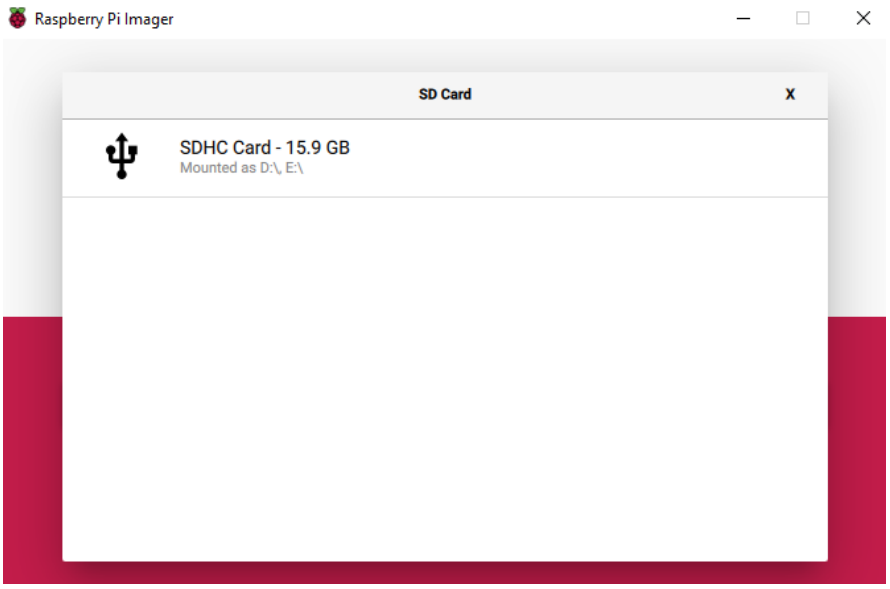
When you launch the installer, your operating system may try to block you from running it. For example, Windows may give the following message:

- If you get this, click on More info and then Run anyway.
- Insert your SD card into the computer or laptop's SD card slot.
- In the Raspberry Pi Imager, select the OS that you want to install. The first option, Raspberry Pi O, is the recommended OS.



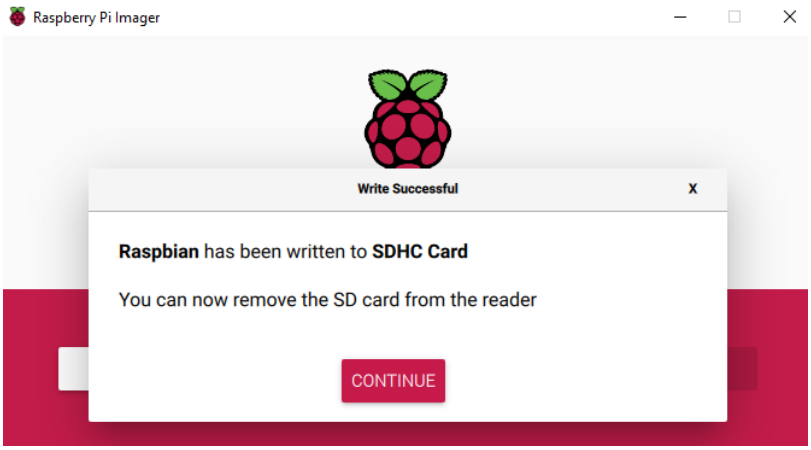
- Select the SD card you would like to install it on. Different platforms will display the drives in different ways. Mac OS, for example, will show you all drives including you main operating system.

Note: Make sure you are selecting the correct drive. The drives memory capacity can be a useful indication of which drive you are selecting.

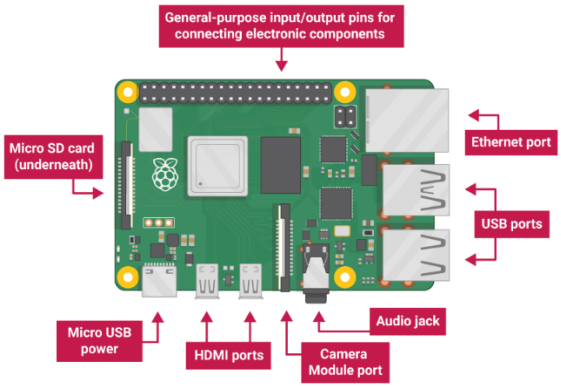


Once you have selected both the OS and the SD card, a new WRITE button will appear.

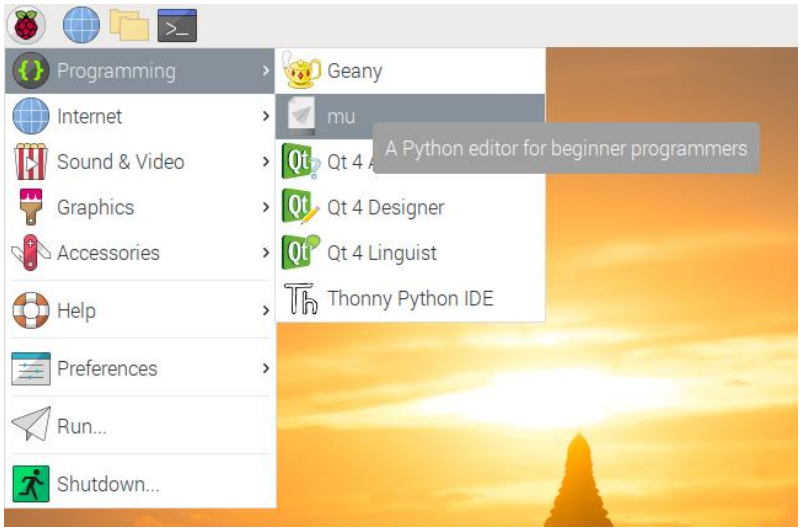
- Then simply click the WRITE button.
- Wait for the Raspberry Pi Imager to finish writing.
- Once you get the following message, you can eject your SD card.



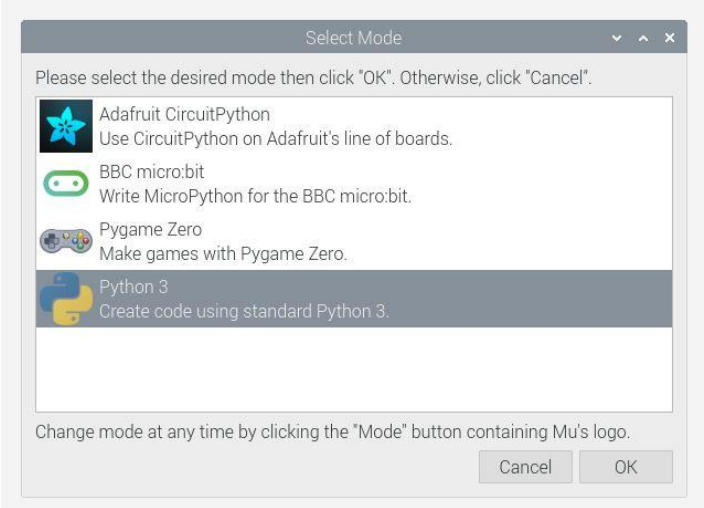
Circuit Diagram



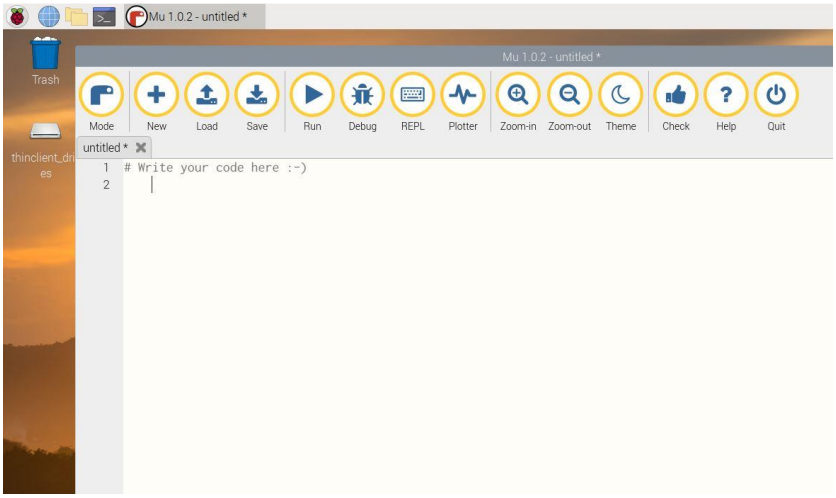
We now open the Python console, in which we can immediately enter program code. Open this by firstly clicking the Raspberry Pi symbol in the top left and then under Programming on “mu”. If you don’t see this option, you can either install it (Preferences -> Recommended Software) or also use the Thonny Python IDE.



First, you are asked for the Editor Mode. Select Python 3.

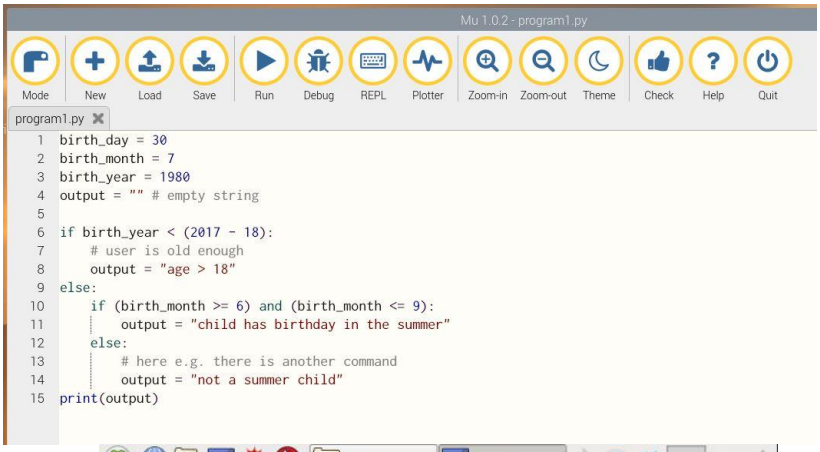


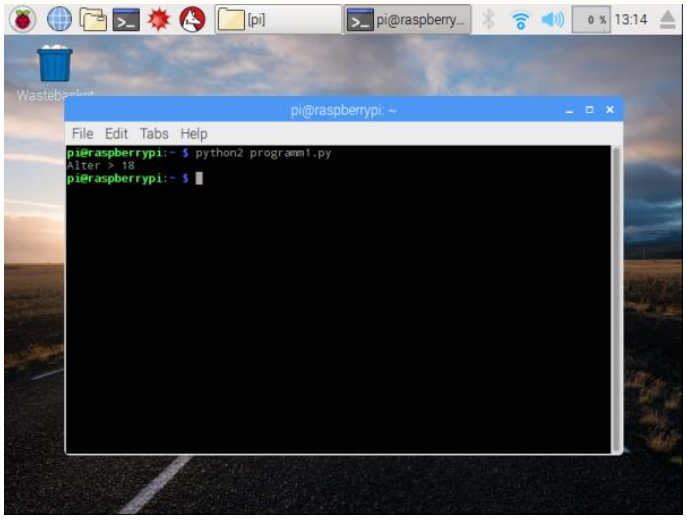
The Python console now opens and it looks like this. We will write our code in this:



Click on “REPL” on the Top bar, so that we can see our output immediately.

If you are wondering why we took **Python 3** and whether you can use Python 2, the answer is that both are possible. There are a few differences between those two versions, but they do not matter to get started. The code we write is executable on both versions.





Applications of Raspberry Pi

- The raspberry pi boards are used in many applications like Media streamer, Arcade machine, Tablet computer, Home automation, Carputer, Internet radio, controlling robots, Cosmic Computer, hunting for meteorites, Coffee and also in raspberry pi-based projects.

Advantage of Raspberry Pi

- Low cost
- Huge processing power in a compact board
- Many interfaces (HDMI, multiple USB, Ethernet, onboard Wi-Fi and Bluetooth, many GPIOs, USB powered, etc.)
- Supports Linux, Python (making it easy to build applications)
- Readily available examples with community support

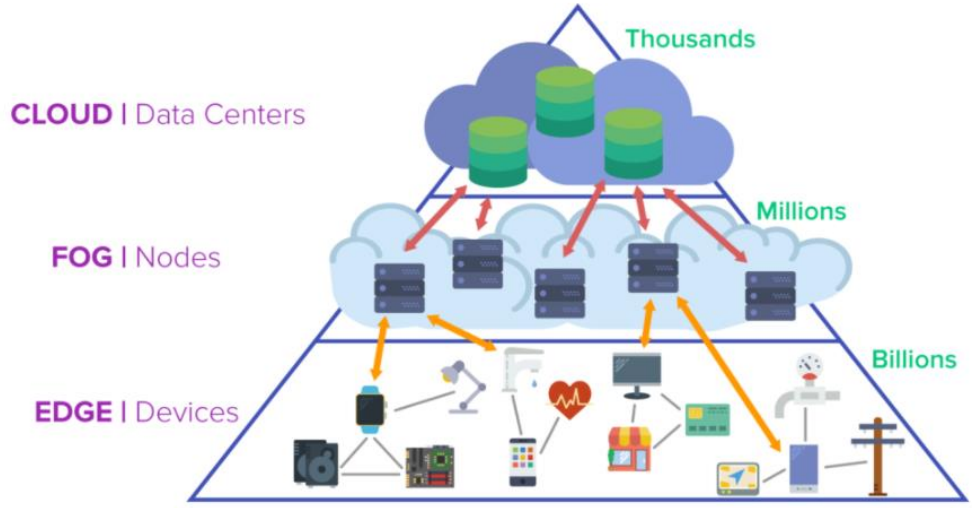
- Developing such an embedded board is going to cost a lot of money and effort.

Q-3) What is Fog Computing. Explain with example.

- The term fog computing, originated by Cisco, refers to an alternative to cloud computing. This approach seizes upon the dual problem of the proliferation of computing devices and the opportunity presented by the data those devices generate by locating certain resources and transactions at the edge of a network.
- By locating these closer to devices, rather than establishing in-cloud channels for utilization and storage, user's aggregate bandwidth at access points such as routers. This in turn reduces the overall need for bandwidth, as less data can be transmitted away from data centres, across cloud channels and distances.
- Data storage is another important difference between cloud computing and fog computing. In fog computing less data demands immediate cloud storage, so users can instead subject data to

strategic compilation and distribution rules designed to boost efficiency and reduce costs.

- By moving real time analytics into a cloud computing fog located closer to devices, it is easier to capitalize on the existing computing power present in those devices. This improves user experience and reduces burdens on the cloud as a whole. Fog computing is especially important to devices connected to the internet of things (IoT).



Fog Computing:

- Decentralization and flexibility are the main difference between fog computing and cloud computing. Fog computing, also called fog networking or fogging, describes a decentralized computing structure located between the cloud and devices that produce data. This flexible structure enables users to place resources, including applications and the data they produce, in logical locations to enhance performance.
- The structure's goal is to locate basic analytic services at the edge of the network, closer to where they are needed. This reduces the distance across the network that users must transmit data, improving performance and overall network efficiency.

- Fog computing security issues also provide benefits for users. The fog computing paradigm can segment bandwidth traffic, enabling users to boost security with additional firewalls in the network.
- Fog computing maintains some of the features of cloud computing, where it originates. Users may still store applications and data offsite, and pay for not just offsite storage, but also cloud upgrades and maintenance for their data while still using a fog computing model. Their teams will still be able to access data remotely, for example.

Advantages and Disadvantages

Advantages of fog computing include:

- Minimize latency. Keeping analysis closer to the data source, especially in verticals where every second counts, prevents cascading system failures, manufacturing line shutdowns, and other major problems. The ability to conduct data analysis in real-time means faster alerts and less danger for users and time lost.
- Conserve network bandwidth. Many data analytics tasks, even critical analyses, do not demand the scale that cloud-based storage and processing offers. Meanwhile, connected devices constantly generate more data for analysis. Fog computing eliminates the need to transport most of this voluminous data, saving bandwidth for other mission critical tasks.
- Reduce operating costs. Processing as much data locally as possible and conserving network bandwidth means lower operating costs.
- Enhance security. Whether in transmission or being stored, it is essential to protect IoT data. Users can monitor and protect fog nodes using the same controls, policies, and procedures deployed across the entire IT environment and attack continuum to provide enhanced cybersecurity.
- Improve reliability. Because IoT devices are often deployed under difficult environmental conditions and in times of emergencies, conditions can be harsh. Fog computing can improve reliability under these conditions, reducing the data transmission burden.
- Deepen insights, without sacrificing privacy. Instead of risking a data breach sending sensitive data to the cloud for analysis, your team can analyse it locally to the devices that collect, analyse, and store that

data. This is why the nature of data security and privacy in fog computing offers smarter options for more sensitive data.

- Boost business agility. Only by knowing what resources customers need, where they need those resources, and when the support is needed can businesses respond to consumer demand quickly. Fog computing allows developers to develop fog applications rapidly and deploy them as needed. Fog computing technology also allows users to offer more specific services and solutions to their customers and locate data and data tools where they are best processed—all based on existing computing capabilities and infrastructure.
- Fog computing challenges include a heavy reliance on data transport. The rollout of the 5G network has improved this issue, but limited availability, lower speeds, and peak congestion are all issues. Both speed and security at fog nodes are other potential issues that demand attention.

Platform and Applications:

A smart electrical grid is a use case for fog computing. In order to function efficiently, smart cities must respond to rising and falling demands, reducing production as needed to stay cost-effective. This means that smart grids demand real time electrical consumption and production data. These kinds of smart utility systems often aggregate data from many sensors, or need to stand up to remote deployments. Either way, fog computing architecture is an ideal solution.

Smart transportation networks are another example of a fog computing application. Each connected vehicle, traffic device, and even street on this kind of grid generates a stream of data. Obviously, this means a tremendous amount of data analysis in real-time is critical to avoid accidents, and a fog computing approach is essential to sharing the limited mobile bandwidth that is available.

Fog computing and IoT generally is a rich area. Connected manufacturing devices with cameras and sensors provide another great example of fog computing implementation, as do systems that make use of real-time analytics.

Fog Computing Working:

Fog computing implementation involves either writing or porting IoT applications at the network edge for fog nodes using fog computing software, a package fog computing program, or other tools. Those nodes closest to the edge, or edge nodes, take in the data from other edge devices such as routers or modems, and then direct whatever data they take in to the optimal location for analysis.

In connecting fog and cloud computing networks, administrators will assess which data is most time-sensitive. The most critically time-sensitive data should be analysed as close as possible to where it is generated, within verified control loops.

The system will then pass data that can wait longer to be analysed to an aggregation node. The characteristics of fog computing simply dictate that each type of data determines which fog node is the ideal location for analysis, depending on the ultimate goals for the analysis, the type of data, and the immediate needs of the user.

Fog Computing Example:

1) **Mining**: Today's mining companies are moving more towards using autonomous trucks and trains as well as tunnelling and boring machines to ensure worker safety and increase productivity. This sophisticated type of mining equipment can generate terabytes of data in a single day through the course of normal operations. When you're working in harsh conditions, not to mention 100s of feet underground, reliable networks and cloud connectivity are typically unavailable. That's where "the fog" comes into play. Do all that processing locally and then send a small upload up to the cloud at the end of the day.
2) **Wind Farms**: We talked about wind turbine blades before and realized that a wind turbine is a very sophisticated piece of equipment. Depending on the weather, you can make changes to your wind farm turbines in real-time to optimize electricity output. The problem is of course that most wind farms are located in remote areas. "Edge intelligence" lets you analyze data locally in real-time without relying on continuous wide area network availability.

3) **Trains**: As part of the rise in the Industrial Internet of Things or IIoT, trains and tracks are being equipped with a new generation of instruments and sensors with locomotives acting as the central hub for all the data gathered from these sensors. The problem is that trains move fast and it's difficult to maintain a connection with "the cloud". Install some fog computing nodes in your locomotive and you're golden.
4) **Pipelines**: Did you know that a 100-mile-long section of oil pipeline generates gigabytes of data during operation? This amounts to many terabytes of data over time, given that there are 2.5 million miles of oil and gas pipelines in the United States alone. This example of fog computing seems a bit tougher to visualize but one would imagine that you install a fog computing node at regular intervals and they speak to each other all the way down the line. In the end, the cloud only needs to know what's important at a high level so the bandwidth needs are much less.
5) **Oil Wells**: The Electric Submersible Pump (ESP) lies at the centre of an oil well extracting oil and pumping it to the surface. That far under the ground, there isn't a whole lot of connectivity available to talk to "the cloud". An ESP failure will close a well and can be very costly to replace. A real-time edge intelligence solution can monitor the operational data gathered from the ESP and apply advanced analytics in real-time to predict failures and increase uptime for oil wells.

Q-4) Discuss security concerns in Cloud.

IoT Cloud Security

- Cloud computing offers several advantages to businesses, including greater technological flexibility, reduced operational costs and easy scalability.
- When cloud computing is implemented in an IoT network, the cloud platform and connected applications become highly vulnerable to cyber threats. Here are some ways in which you can ensure IoT cloud security using holistic security principles:

Encryption of data at rest

- Businesses embracing IoT for the first time lay a lot of focus on the security of the cloud infrastructure. So, it is crucial to deploy encryption technologies to secure the cloud.
- Encryption is a process in which legible data (plaintext) is converted into an output (ciphertext) that does not reveal any information about the input plaintext.
- An encryption algorithm is employed for this conversion. Encryption ensures that even if an attacker obtains access to storage devices with sensitive data, they would not be able to decipher it.

- Encryption of data at rest implies that an encryption algorithm is used to safeguard data that is stored on any kind of disk, including backup devices and solid-state drives.
- Several layers of encryption can be used to protect data at rest. An example of this is the encryption of sensitive information prior to storage along with the encryption of the storage drive itself.

Encryption of data in transit

- Data in transit is considered to be at higher risk for security breaches. So, whether the data is being communicated over the internet or between data centers, it is crucial to ensure that an end-to-end security strategy is in place.
- In order to protect data in transit, encryption is enabled prior to moving the data. Encrypted connections such as HTTPS, FTPS, SSL, TLS, etc. can also be used.

Device identity

- Each device in an IoT implementation should have a unique device identity. When a device comes online, this identity is used to authenticate it and authorize secure communication with other components of the IoT ecosystem.

Device authentication using OAuth 2.0

- OAuth 2.0 is a powerful open standard that can be used by API developers to protect an IoT ecosystem. It is a token-based authentication and authorization solution that also offers a framework for the decisions associated with authentication.

User role and policy

- As part of access management, a privileged user management system can be deployed to ensure that stringent authentication processes are followed for user access to IoT data.
- It is also possible to create policies that can be attached to identities/resources to define their permissions. The administrator defines the policies and specifies the access level of resources.

Certificate based authentication

- A certificate is essentially a signed digital document that includes attributes identifying its issuer and owner (also referred to as subject).
- It contains two important fields – a public key that belongs to the owner/subject and a digital signature from the issuer. The issuer is usually a Certificate Authority (CA) and X.509 is a popular digital certificate standard.
- The public key can establish a secure communication channel with the subject. The signature is proof that the subject's identity is verified by the issuer.
- The subject also possesses a private key that matches the public key, but this is not a part of the certificate. The private key is used for proving the identity of the subject once a communication session is established.

Certificate based authentication is more powerful than password-based authentication.

There are several other IoT cloud security mechanisms that can be adopted as well, i.e., MQTT token-based authentication, maintaining access control lists and IP Whitelisting/blacklisting.