

INTERNET OF THINGS (IoT) FROM IDEA TO IMPLEMENTATION



Dr. Ritu Raj Lamsal, PhD Engg., FIE

University of Malaga, Spain

Founder President, CEO

Medhayantra Company Pvt.Ltd

Overview of the session

- IoT: What does it really mean?
- Smart things and their applications
- The Future of IoT: How It Will Change Our Lives
- IoT Applications in various sectors like Industry, Healthcare, Agriculture, manufacturing and Others



Overview ..

- Introduction to IoT
- Key Components of IoT
- Application of IoT
- IoT Architecture
- Challenges and Security in IoT
- IoT Data Analytics and Visualization
- Future Trends in IoT
- Implementation
- Q&A Session



Introduction to IoT

- Definition and concept of IoT
- Importance and real-world applications
- How IoT is changing industries

Key Components of IoT

- Sensors and actuators
- Microcontrollers and microprocessors
- Communication protocols (Wi-Fi, Bluetooth, MQTT, etc.)
- Cloud platforms and data storage

IoT Architecture

- Device layer
- Communication layer
- Cloud layer
- Application layer

Challenges and Security in IoT (15 minutes)

- Data privacy and security concerns
- Securing IoT devices
- Best practices for IoT security

IoT Data Analytics and Visualization

- Importance of data analytics in IoT
- Tools for data analysis and visualization
- Creating meaningful insights from IoT data

- MS Excel
- SQL
- R
- SAS
- Tableau

Basic data
analytics tools



- SPSS
- MATLAB
- Python
- Hadoop
- Spark
- Hive

Advanced data
analytics tools



What is IoT ?

Internet

Things

- Internet: is a network of networks
 - **worldwide system of computer networks**
- Things: Any physical things
- The network of everyday objects (Things) using the enabling them to send and receive data.



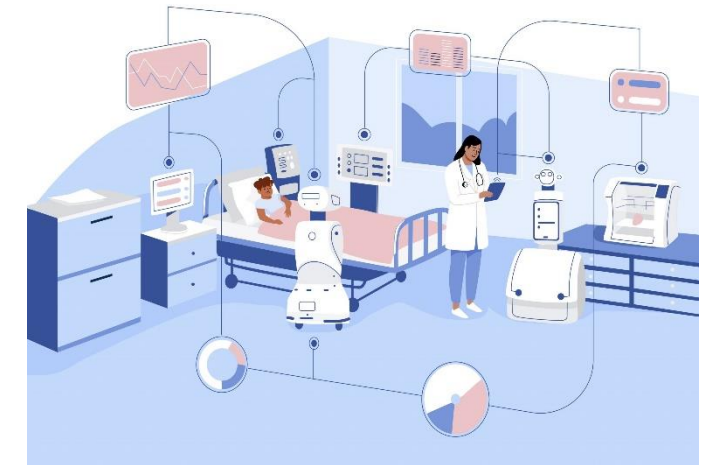
SMART THINGS, What do they do?

- Smartphones
- Smart speakers
- Smart cars
- Smart doorbells
- Smart locks
- Smart refrigerators
- Smart watches, smart bands,
- Smart keychains,
- Smart glasses, Smart bulbs and many others.



Discussion: What possible things could be in ...

- Smart Home
- Smart City
- Smart Agriculture
- Smart manufacturing
- Smart healthcare



Overview: What is IoT is and how it works



- IoT stands for the Internet of Things. It refers to the connection of everyday physical devices to the internet, allowing them to send and receive data.
- The idea behind IoT is to make devices "smart" by enabling them to communicate with each other and with the internet, enabling a range of new applications and services.

How IoT works



Sensors
Collecting data



Connectivity
Sending data to cloud



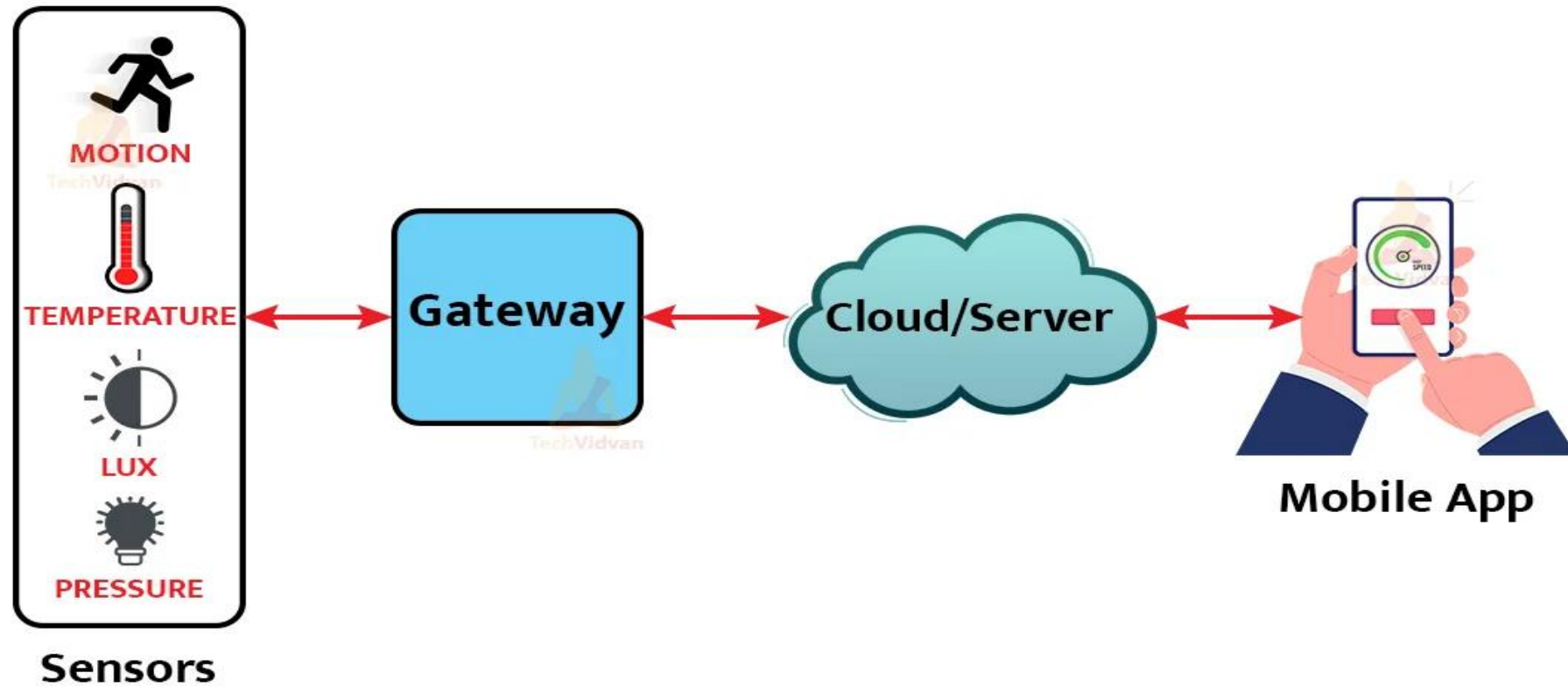
Data Processing
Making data useful



User Interface
Delivering information to user

How IoT Works

Working of IoT



CLOUD

Big Data processing
Business Logic
Data Warehousing

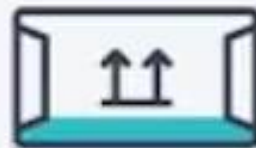
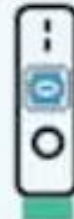
INTERNET

EDGE

Realtime data processing
At source/on premises
data visualization
Basic analytics
Data caching, buffering
Data filtering, optimization
M2M comms

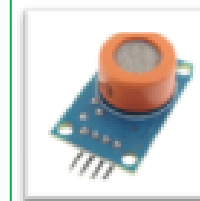
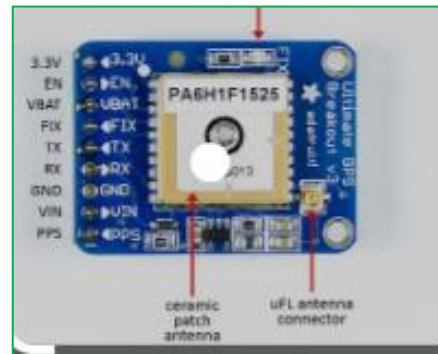
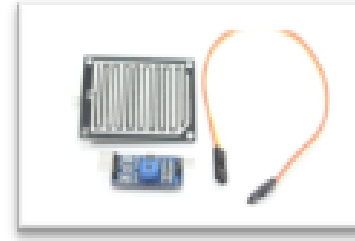
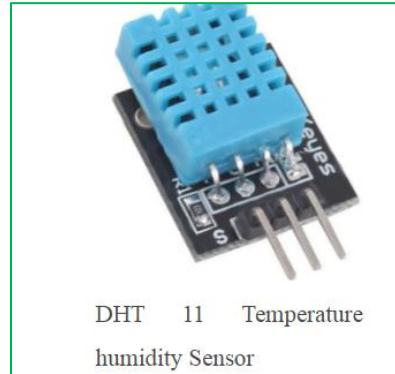
LAN/WAN

SENSORS AND CONTROLLERS



Overview of the various components of an IoT system

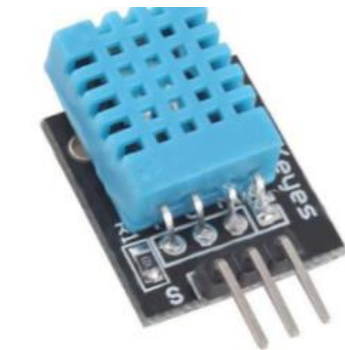
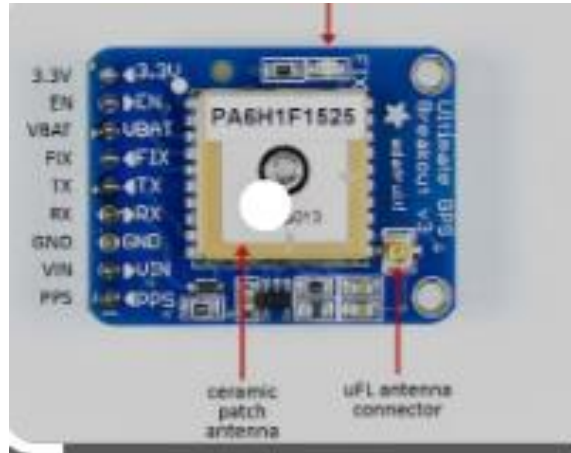
- Sensors
- Actuators
- Connectivity
- Cloud computing



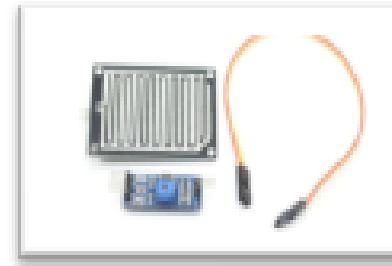
- **Sensors** => to collect data from the environment,
such as temperature, humidity, or motion.
- **Actuators** => to control physical devices,
 - Such as turning on a light or opening a door.
- **The connectivity** components allow the device to communicate with other devices or with the internet.
- IoT devices often use wireless communication protocols, such as Wi-Fi, Bluetooth, or Zigbee, to transmit data to other devices or to the cloud.

Overview of the various components of an IoT system

- Sensors
- Actuators
- Connectivity
- Cloud computing



DHT 11 Temperature
humidity Sensor



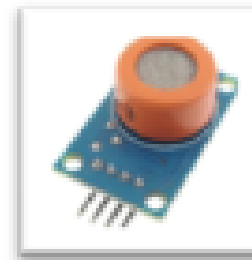
Rain Sensor



Proximity Sensor



PIR Sensor



Alcohol Sensor



Ultrasonic Sensor



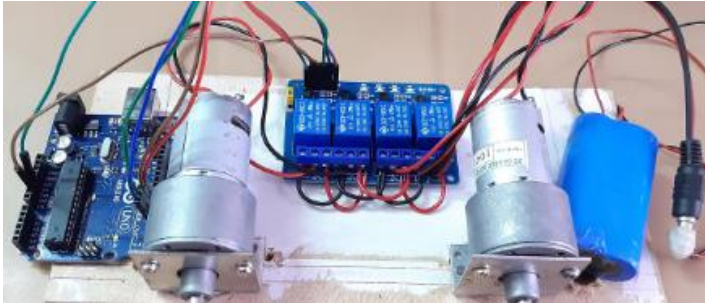
IR optical Sensor

Widely used sensors

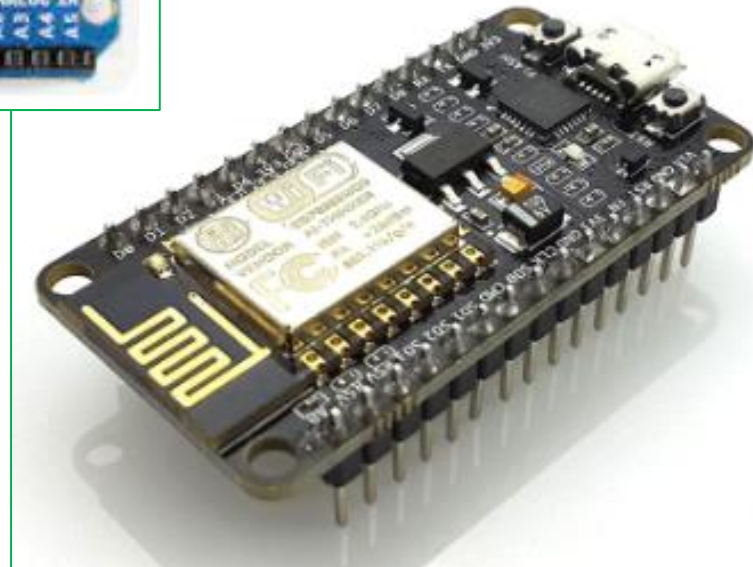
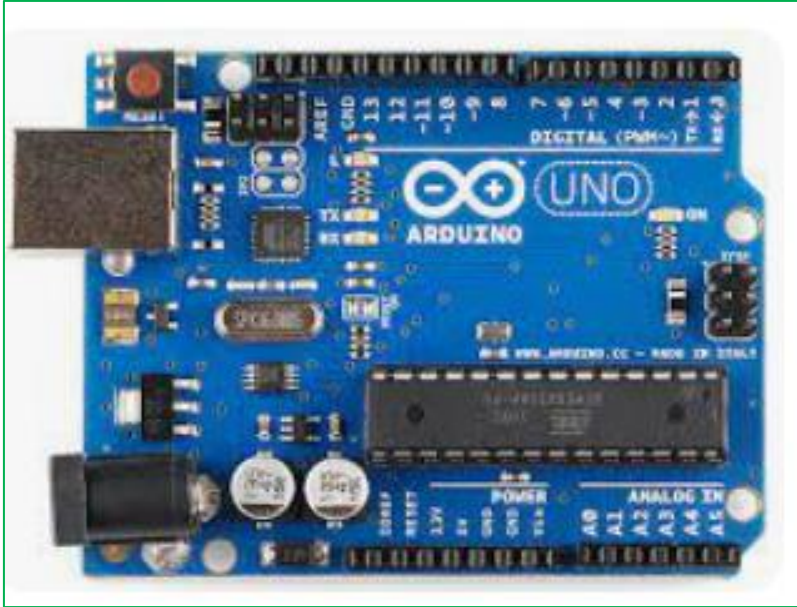
- **Temperature Sensor**
- **Humidity Sensors**
- **Proximity Sensor**
- **Pressure Sensor**
- **Water Quality Sensor**
- **Chemical/Smoke & Gas Sensor**
- **Level Sensor**
- **IR Sensor**
- **Ultrasonic Sensor**
- **Image sensors**
- **Motion Detection Sensors**
- **Accelerometer Sensors**
- **Gyrometer Sensors**
- **Optical Sensors**

Actuators

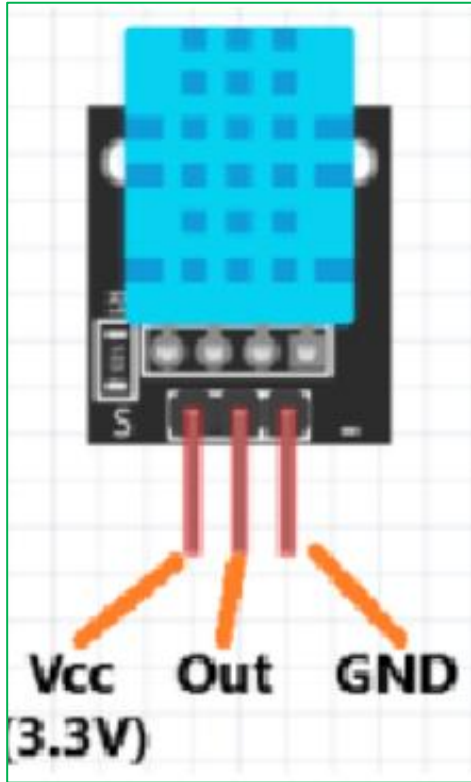
- Actuators are devices that are used to control the physical environment based on data collected by sensors. They can be used to trigger actions such as turning on/off lights, opening/closing doors, adjusting thermostats, and many more.



Commonly used Controllers and Boards



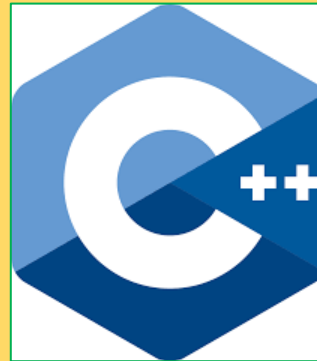
DHT sensor, Ultrasonic, PIR, Gas/Air quality sensor



Programming Languages used in IoT

Common programming languages used in IoT system development include

- C,
- C++,
- Python,
- Java, and JavaScript.



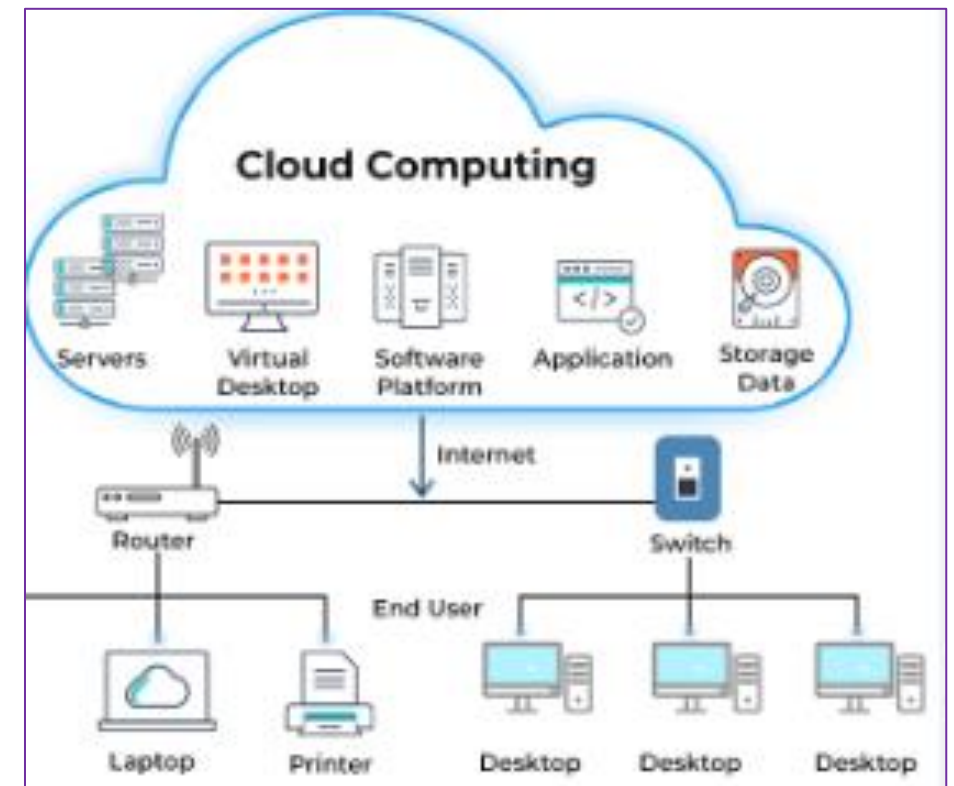
- The choice of programming language depends on the application's requirements, such as the device's hardware capabilities, the complexity of the application, and the required data processing speed.

Commonly Used Actuators

- Electric motor:
- **Solenoid valve**: A solenoid valve is an actuator that controls the flow of fluids or gases by opening or closing a valve using an electromagnetic coil. It is commonly used in irrigation systems, HVAC systems, and industrial automation.
- **Linear actuator**: A linear actuator is an actuator that converts electrical energy into linear motion. It is used in applications such as opening and closing doors, windows, or blinds, or in automated machinery.
- **Piezoelectric actuator**: A piezoelectric actuator is an actuator that converts electrical energy into mechanical motion using a piezoelectric material. It is used in applications such as precision positioning, inkjet printers, and microfluidics.
- **Thermal actuator**: A thermal actuator is an actuator that uses thermal expansion to generate motion. It is used in applications such as thermostats, HVAC systems, and automotive systems.

Connectivity

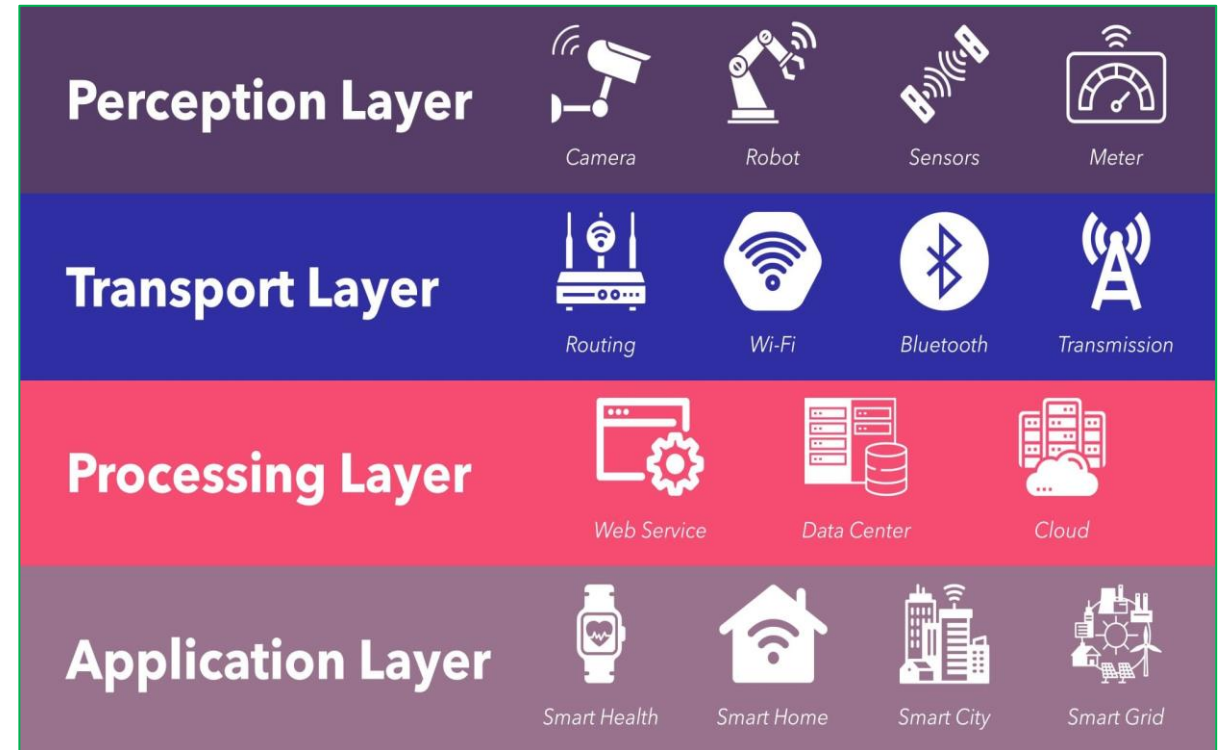
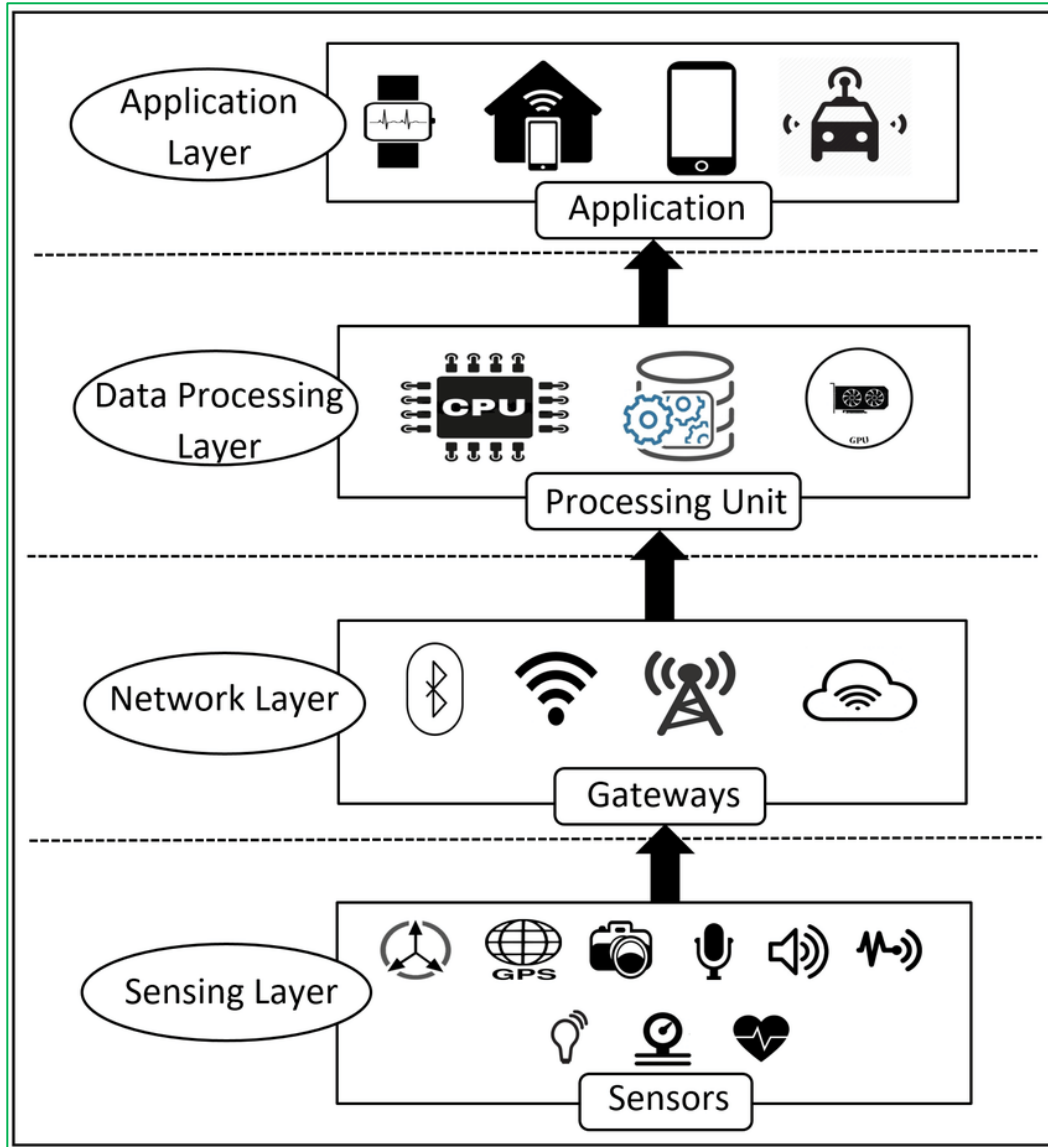
- **Connectivity**=>backbone of an IoT system
- Wi-Fi, cellular networks, Bluetooth, Zigbee, and LoRaWAN.
- **Cloud Computing**=> Cloud computing is used to store, process, and analyze the data collected by IoT devices.
- Anywhere / Any time
- Easy scalability
- Data analytics.



How IoT can be used to solve problems and improve efficiencies

- **Predictive maintenance:** IoT sensors can be used to monitor the condition of machines and equipment in real-time, allowing for early detection of issues before they lead to equipment failure. This can improve maintenance scheduling and reduce downtime, saving time and money.
- **Supply chain optimization:** IoT can help optimize the supply chain by tracking goods and assets in real-time, improving inventory management, reducing waste, and enhancing supply chain visibility.
- **Environmental monitoring:** IoT sensors can be used to monitor the environment, including air quality, water quality, and noise levels. This can help identify potential hazards and enable proactive measures to be taken.
- **Smart homes:** IoT can make homes smarter and more efficient by enabling control of lights, temperature, and appliances remotely. This can improve energy efficiency and reduce costs.
- **Healthcare:** IoT can improve healthcare by enabling remote monitoring of patients, improving medication adherence, and automating routine tasks such as medication dispensing.
- **Smart cities:** IoT can make cities smarter by enabling intelligent traffic management, reducing waste and energy consumption, and improving public safety.

IoT Architectures

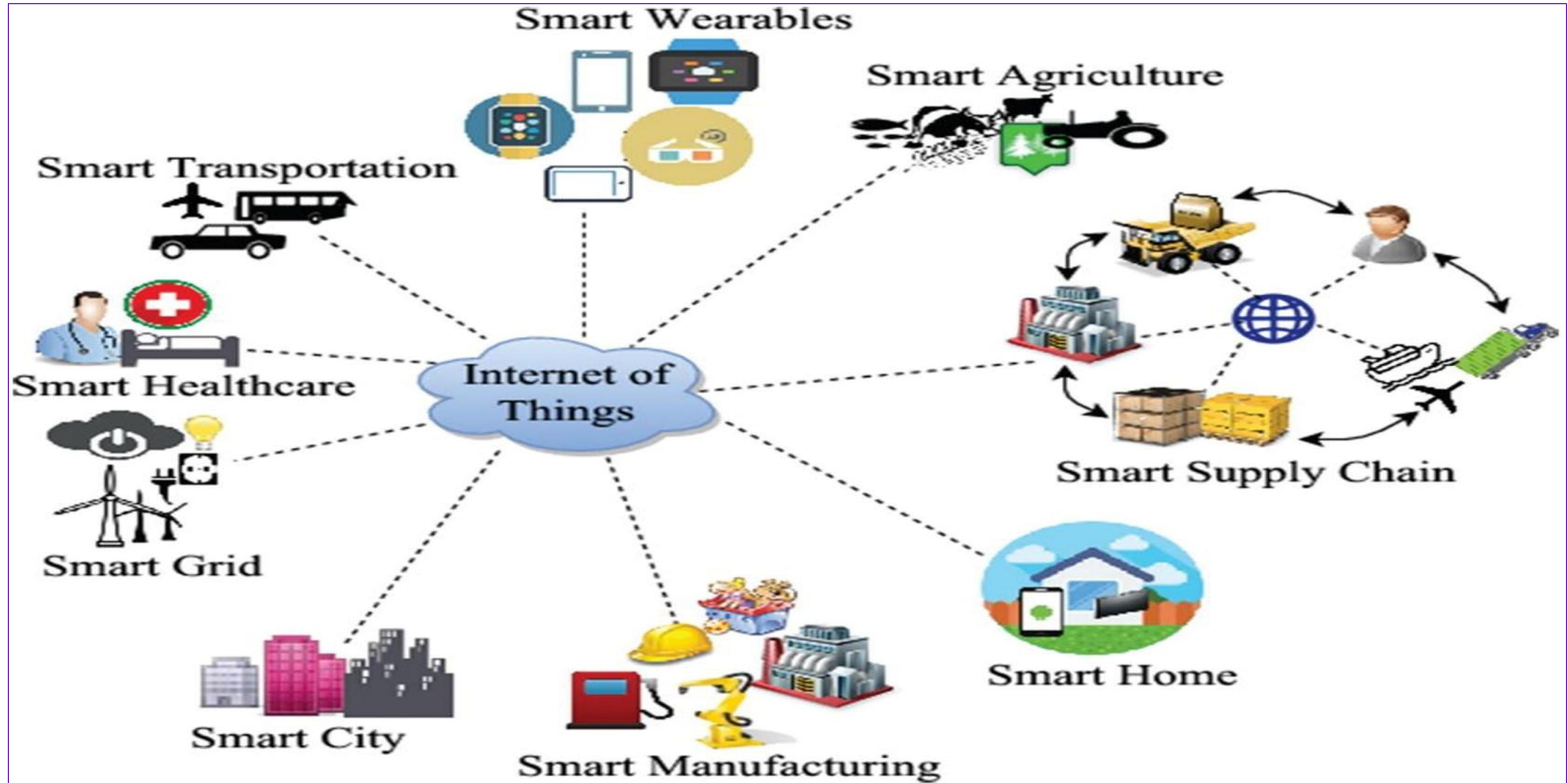


IoT-Communication Protocols:



- Communication protocols are essential in IoT as they enable devices to communicate with each other and with the cloud. There are various communication protocols available for IoT systems, each with its advantages and disadvantages.
- Some common communication protocols includes
- Wi-Fi, Bluetooth, Zigbee, and LoRaWAN.
- The choice of communication protocol depends on the application's requirements, such as range, data rate, power consumption, and security.

IoT Application Scope



Discussion of how IoT is transforming these industries and enabling new use cases •

- In Agriculture, IoT is being used to monitor soil moisture and temperature, track livestock, and optimize crop management.
- In Healthcare, IoT is being used to monitor patients remotely, improve medication adherence, and automate routine tasks.
- In Manufacturing, IoT is being used to monitor equipment, predict maintenance needs, and optimize production processes.
- In Transportation, IoT is being used to optimize logistics, improve safety, and reduce fuel consumption.

Real-world Case Studies

- **Agriculture: John Deere** - Precision Agriculture platform, Combines data from multiple sources, including sensors, weather reports, and satellite imagery, to optimize crop management. Enables farmers to reduce waste, increase yields, and improve efficiency.
- **Healthcare: Philips Healthcare** develop its eICU platform. Remote monitoring of patients in the ICU. Sensors and cameras to monitor patients' vital signs, alerting clinicians
- **Manufacturing: Rolls-Royce** TotalCare platform, which monitors the performance of its jet engines in real-time.
- enables Rolls-Royce to optimize maintenance schedules, reduce downtime, and improve efficiency.
- **Transportation: UPS** ORION platform, uses sensors, GPS, and other sources to optimize its delivery routes.
- to reduce fuel consumption, improve delivery times, and reduce costs.



Case study

- **Manufacturing: Harley-Davidson** manufacturing process of motorcycle engines, which reduced defects by 40%.
- The Internet of Things And Harley-Davidson, August 23, 2019 by [mac](#)
- Healthcare: IoT-enabled devices and wearables allow for remote patient monitoring and improve patient outcomes.
- hospital readmissions by 50% and reduced mortality rates by 45%.
- **Agriculture: Monsanto** uses IoT sensors to monitor soil moisture and crop growth, which has increased crop yields by 20%.
- Transportation: The city of Barcelona uses IoT sensors to monitor traffic flow and reduce congestion, which has reduced travel times by 25%.
- **Energy: Nest's** smart thermostat uses IoT sensors to learn users' preferences and adjust heating and cooling settings accordingly, which has resulted in energy savings of up to 20%.



- **Retail:** IoT sensors and analytics can track customer behavior, preferences, and buying patterns to improve inventory management, reduce waste, and enhance the customer experience. For example, **Walmart** uses IoT sensors to track inventory levels, which has reduced out-of-stock items by 30%.
- **Construction:** IoT sensors can monitor construction sites to improve safety, optimize resource usage, and reduce waste. For example, the construction company, **Skanska**, uses IoT sensors to monitor construction sites, which has reduced energy consumption by 30%.
- **Smart Cities:** IoT sensors and analytics can optimize city infrastructure, reduce congestion, and improve public safety. For example, **the city of Amsterdam uses IoT sensors to monitor air quality and reduce pollution**, which has improved the health of residents.
- **Financial Services:** IoT sensors and analytics can help financial institutions monitor transactions and prevent fraud. For example, **Mastercard uses IoT sensors to monitor user behavior and identify fraudulent transactions**, which has reduced fraud by 40%.
- **Hospitality:** IoT sensors and analytics can enhance the guest experience and improve operational efficiency in hotels and resorts. For example, **the Marriott hotel chain uses IoT sensors to monitor guest preferences and adjust room settings accordingly**, which has improved guest satisfaction.

IoT security

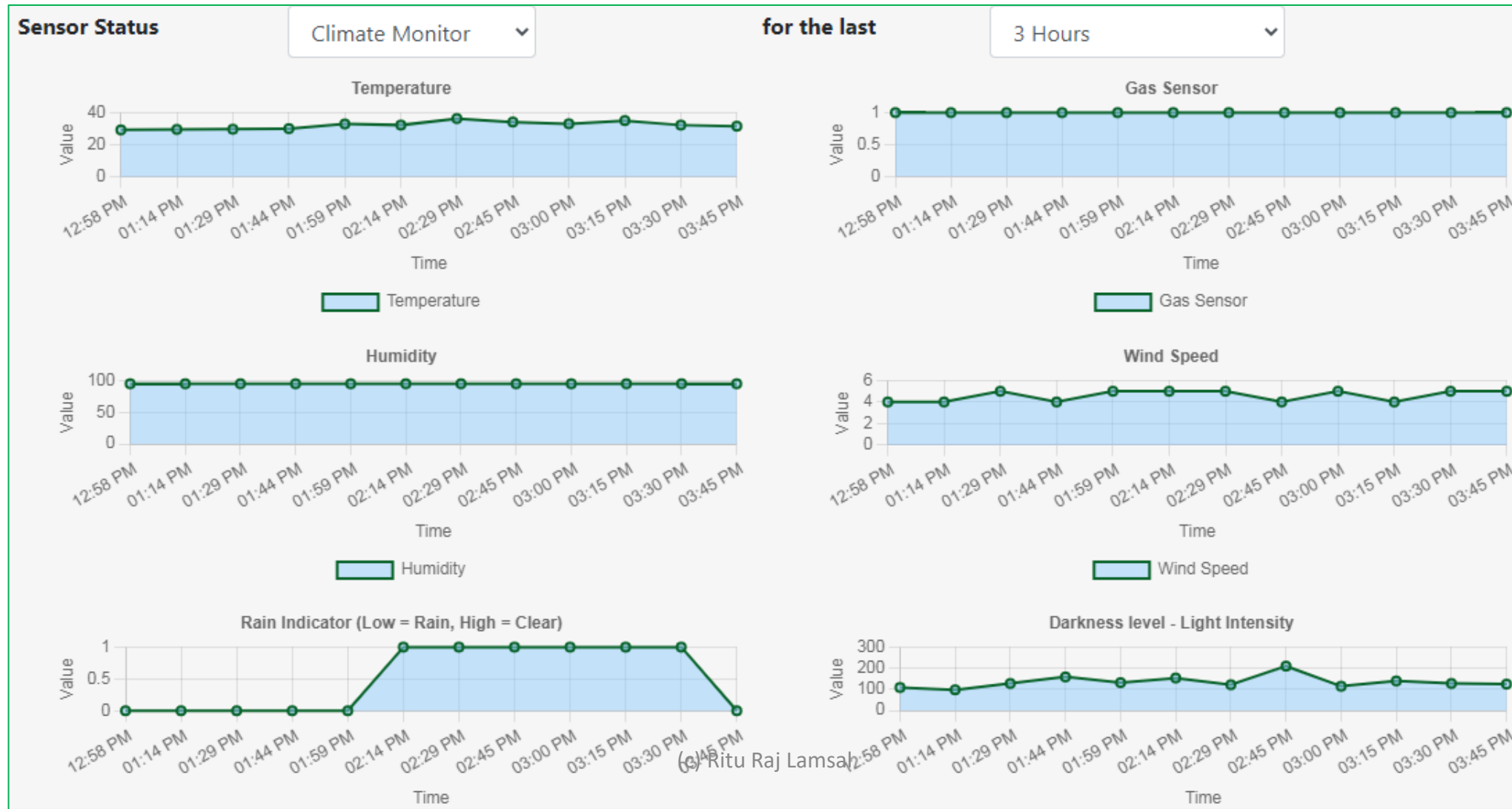
- Data Encryption: Converting data into a coded language
- Access Control: Granting or denying access to resources



- Secure communication :ensuring that data transmitted between devices is secure and cannot be intercepted or tampered with.
- Firmware Updates: Firmware updates are essential for fixing security vulnerabilities and ensuring that devices are up to date with the latest security patches.

Implementation

Implementation- Time series data Visualization



Implementation –Dashboard

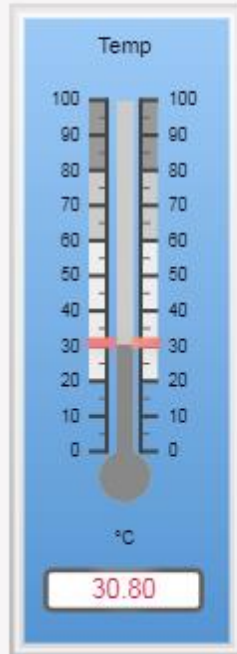
Weather Dashboard

Temperature

Last Updated On: 2023-06-14 03:33 PM

Last 24 Hours

Min: 27.1 Max: 32.2 Avg: 29.8



Humidity

Last Updated On: 2023-06-14 03:33 PM

Last 24 Hours

Min: 49 Max: 62 Avg: 56.2



Water Level

Last Updated On: 2023-06-14 03:42 PM



Water Pump



Humble beginning



Implementation: Grafting



Off Season Grafting of Citrus Fruits



Mushroom : Maintaining Temp and Humidity



(c) Ritu Raj Lamsal

Nursery



(c) Ritu Raj Lamsal

Hitech Tunnel Internal microclimate control



Fogger System



(c) Ritu Raj Lamsal

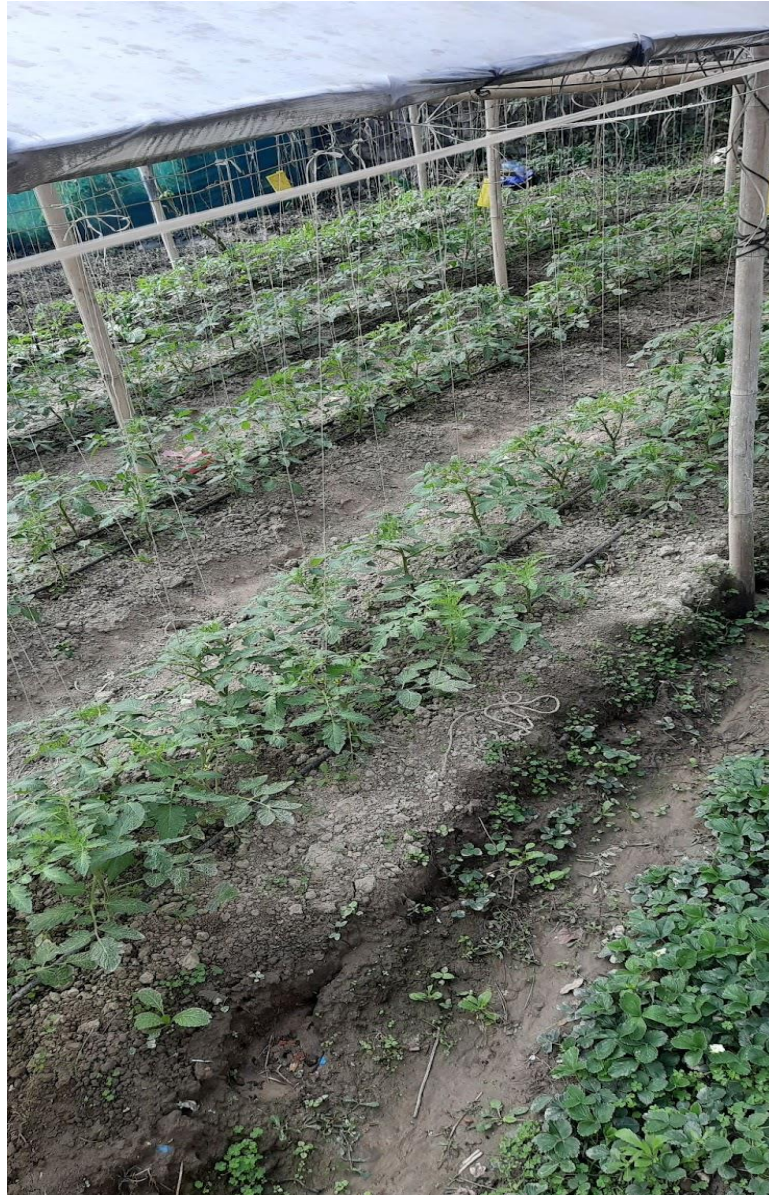
Tunnel Drier



Glass House: Internal Microclimate Management



Drip Irrigation

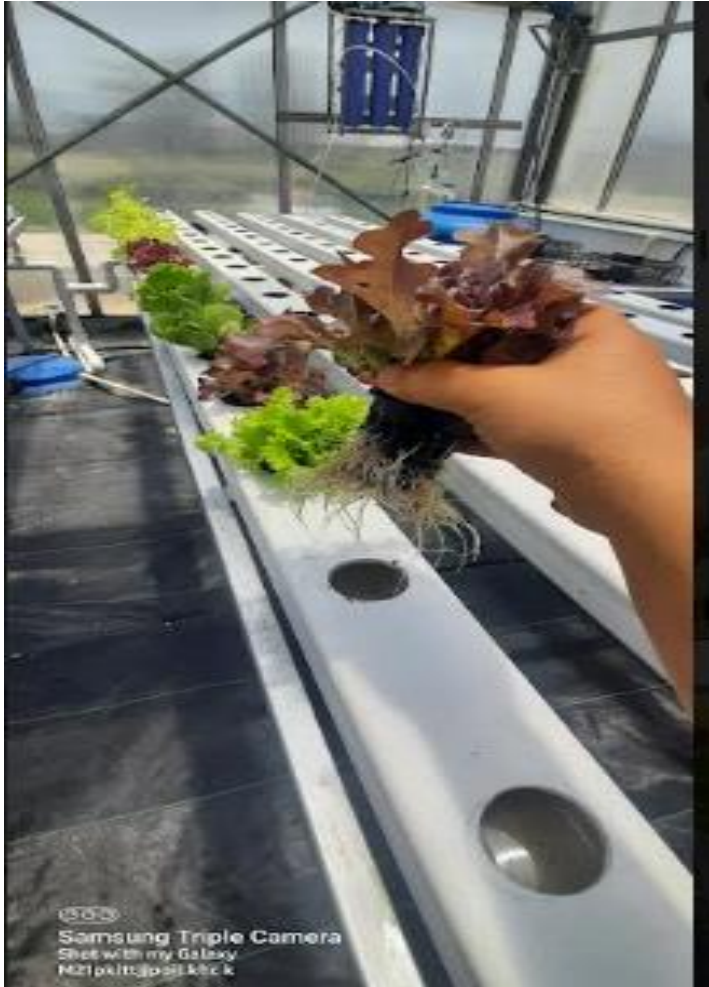


(c) Ritu Raj Lamsal

Drip Irrigation in Tunnel : Tomato



Hydroponics



(c) Ritu Raj Lamsal

Biofloc Fish Farming



(c) Ritu Raj Lamsal

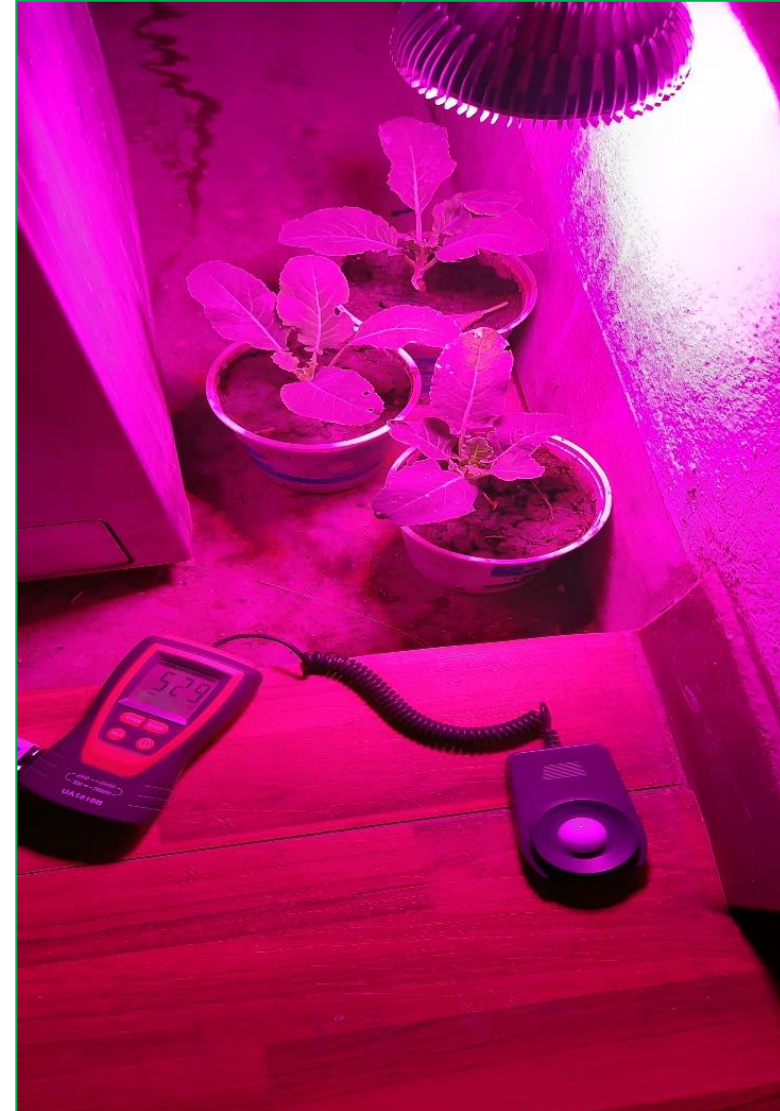
Auto Irrigation -Orange Farming



Grow Light Based Farming



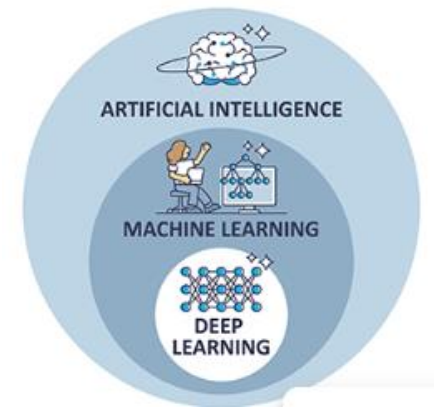
(c) Ritu Raj Lamsal



- Thank you.

A.I, Machine learning and IoT

- Machine Learning: Machine learning is a subset of data analytics that involves using algorithms and statistical models to enable machines to learn from data without being explicitly programmed.
- Machine learning algorithms can be used to analyze large amounts of data generated by IoT devices and extract valuable insights that can be used to optimize system performance and improve efficiency.



Machine Learning Steps

- Collecting Data: IoT System can collect data from various sensors over the period of time. Machines initially learn from the data that you give them.
- Preparing the Data: The data need to refine and prepare for the training and test.
- Choosing a Model: There are several Models to choose
- Training the Model: ...
- Evaluating the Model: ...
- Parameter Tuning: ...
- Making Predictions.

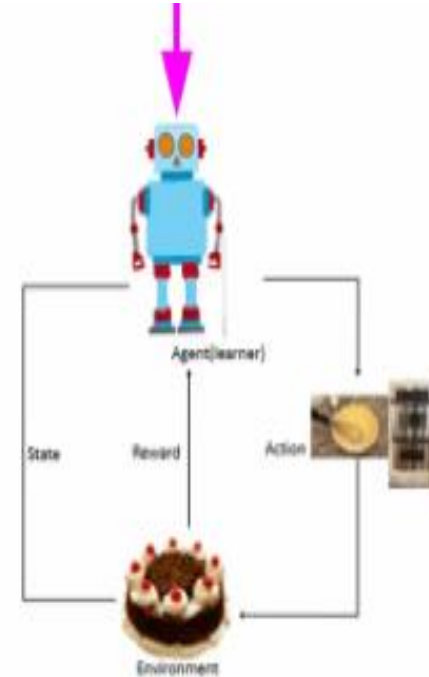
Most Common Machine Learning Types



Supervised Learning



Unsupervised Learning



Reinforcement Learning

Semi Supervised: Used mixed approach

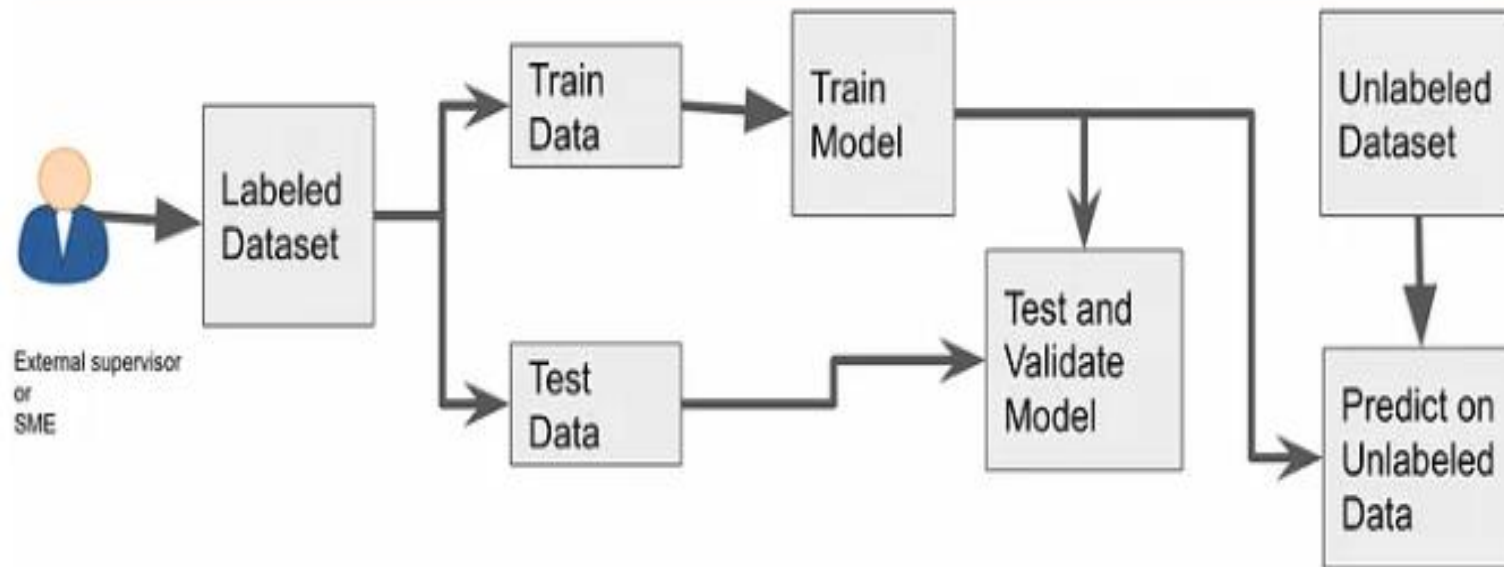
Most Common Machine Learning Types

- supervised: If you're learning a task under supervision, someone is present judging whether you're getting the right answer. Similarly, in supervised learning, that means having a full set of labeled data while training an algorithm.
- Unsupervised: In unsupervised learning, a deep learning model is handed a dataset without explicit instructions on what to do with it. The training dataset is a collection of examples without a specific desired outcome or correct answer. The [neural network](#) then attempts to automatically find structure in the data by extracting useful features and analyzing its structure.
- and reinforcement learning: Reinforcement learning is the closest to human learning as digital systems and machines can get. Through this training, machine learning models can be taught to follow instructions, conduct tests, operate equipment, and much more

Supervised Learning Model

Supervised Model

- Classification -Logistics Regression, Image Classification, Decision Tree, SVM, KNN
- Regression-Linear Regression, Decision Tree Regression

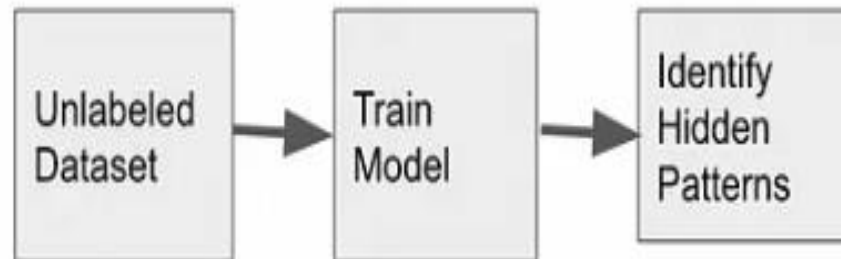


Unsupervised

A child playing with toys can arrange them by identifying patterns based on colors, shapes, sizes, or just based on their interests. The kid discovers new ways to cluster the toys without needing external supervision is similar to unsupervised learning.

Unsupervised Model

- Clustering-K-Means Clustering, Hierarchical Clustering, Agglomerative Clustering
- Association - Recommendation system
- Dimensionality Reduction - Principal Component Analysis(PCA)
- Anomaly detection

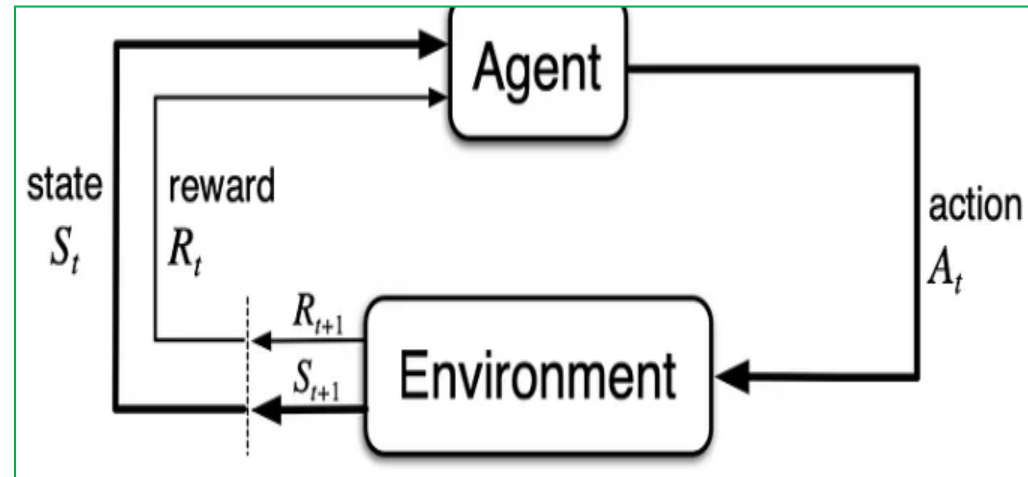


Reinforcement

- A chef explores different ingredients by exploring and experimenting with different recipes in the hope of creating that perfect recipe that wows everyone. This is similar to Reinforcement Learning, where the chef tries a variety of actions like trying different ingredients in different proportions to progressively favors those that appear to taste the best.

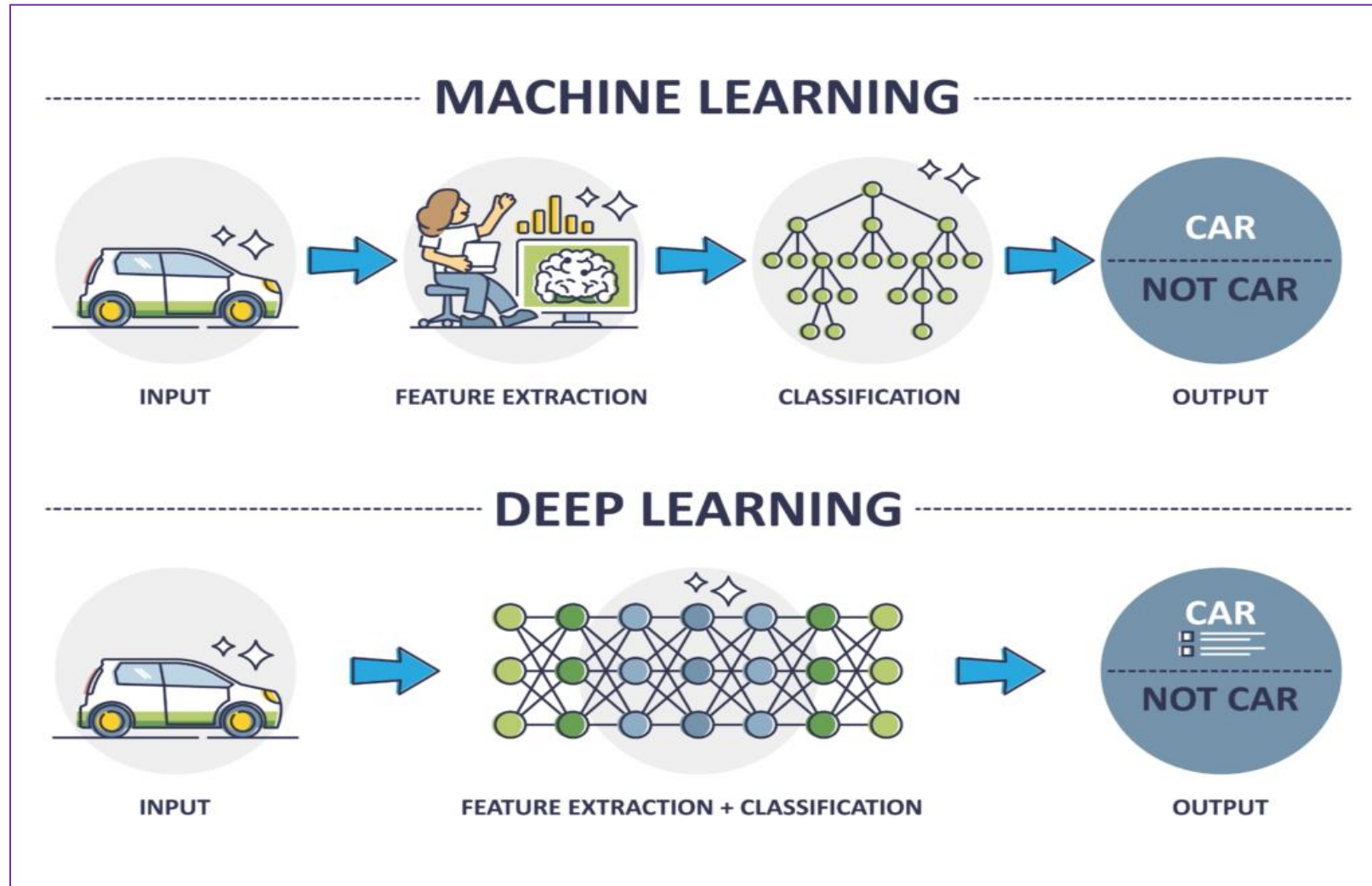
Reinforcement learning is applied in Robotics, Self-driving cars, evaluating trading strategies and adaptive controls.

Reinforcement



In Reinforcement learning an agent interacts with the environment by sensing its state and learns to take actions in order to maximize long-term reward. As the agent takes actions it needs to maintain a balance between exploration and exploitation by performing a variety of actions using trial and error to favor the actions that yield the maximum reward in the future.

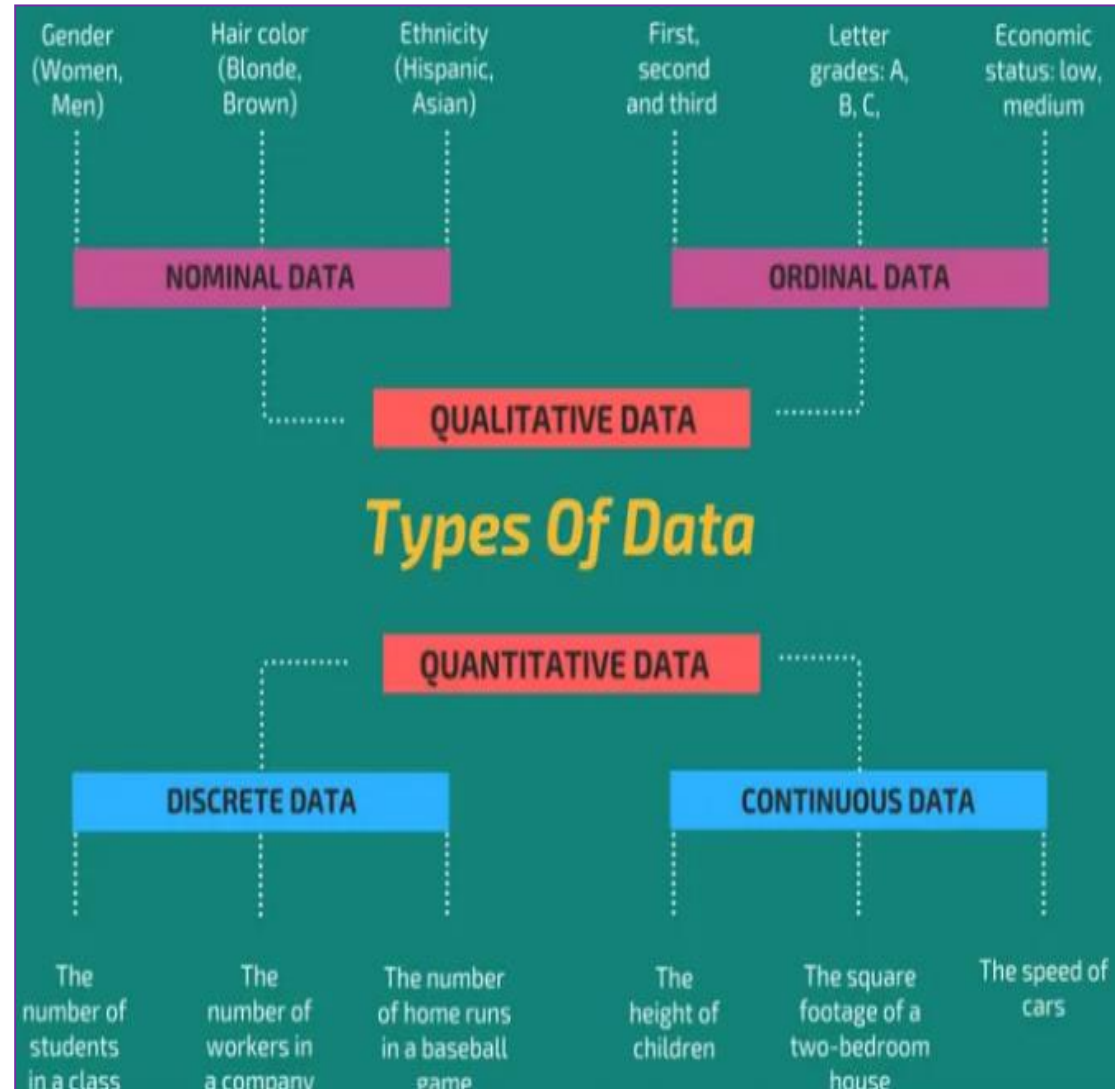
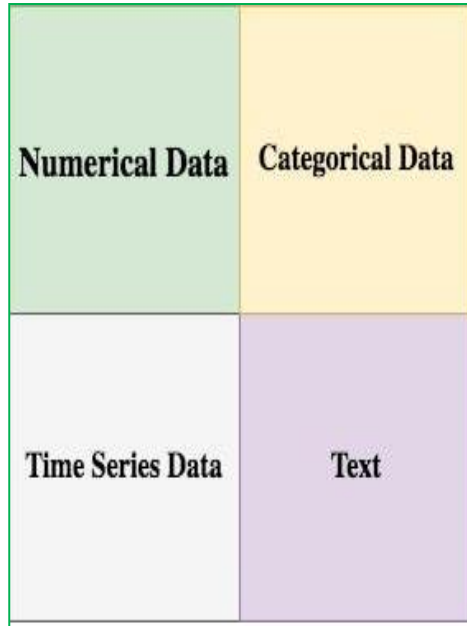
Machine learning and deep learning



Algorithms Types and Applications

- Supervised learning consists of Classification and Regression. Classification algorithms are applied to detect fraud, spam detection, and classify images, and Regression algorithms help predict sales, house prices, etc.
- Unsupervised learning consists of Clustering, Association, Anomaly detection, and dimensionality reduction. Unsupervised learning applications are customer segmentation, market basket analysis, fraud detection, network security analysis, etc.
- Reinforcement learning algorithms are either Value-based, Policy-based, or Model-based. Deep Q-Network(DQN), state-Action-Reward-State-Action(SARSA), Asynchronous Advantage Actor-Critic Algorithm(A3C), and Deep Deterministic Policy Gradient(DDPG) are a few Reinforcement algorithms used in Robotics, Gamings, developing business strategies.

Types of Data In ML



[illegible]

(c) Ritu Raj Lamsal

Big Data -3 Vs

- Variety
 - Volume
 - Velocity
-
- Big data is data that contains greater variety, arriving in increasing volumes and with more velocity.

BIG DATA CHARACTERISTICS

Big Data is the term describing large sets of structured, unstructured, and semi-structured data, continuously generated at a high speed and in high volumes.

Volume

Measure describing the size of generated data

Velocity

Speed at which the data is generated and processed

Variety

Vector showing that Big Data is diverse – structured and unstructured

Veracity

Measure of how truthful, accurate, and reliable data is

Big Data Analytics

- Big data analytics is the process of collecting, storing, analyzing, and visualizing large and complex data sets to gain insights that help organizations make better decisions.
- Big data analytics can be used to improve a wide range of business operations, including customer service, marketing, product development, and fraud detection.

Steps in Big Data

- Collect
- Process
- Clean
- Analyze
 - **Data mining** sorts through large datasets to identify patterns and relationships by identifying anomalies and creating data clusters.
 - **Predictive analytics** uses an organization's historical data to make predictions about the future, identifying upcoming risks and opportunities.
 - **Deep learning** imitates human learning patterns by using artificial intelligence and machine learning to layer algorithms and find patterns in the most complex and abstract data.

Steps in Big data

- The first step in big data analytics is to collect data from a variety of sources, such as social media, sensors, and transactional systems. Once the data is collected, it needs to be stored in a way that makes it easy to access and analyze. Big data analytics platforms are used to store and process large data sets.

- The next step in big data analytics is to clean and prepare the data. This involves removing errors and inconsistencies from the data. Once the data is clean, it can be analyzed using a variety of statistical and machine learning techniques. Big data analytics can be used to identify patterns and trends in the data. These insights can be used to make better decisions about business operations.

Benefits

- Improved decision-making: Big data analytics can help organizations make better decisions by providing them with insights into their customers, operations, and markets.
- Increased efficiency: Big data analytics can help organizations improve their efficiency by identifying areas where they can save time and money.
- Reduced risk: Big data analytics can help organizations reduce risk by identifying potential problems before they occur.
- Enhanced customer service: Big data analytics can help organizations improve their customer service by providing them with a better understanding of their customers' needs.
- Increased innovation: Big data analytics can help organizations innovate by providing them with new ideas and insights.

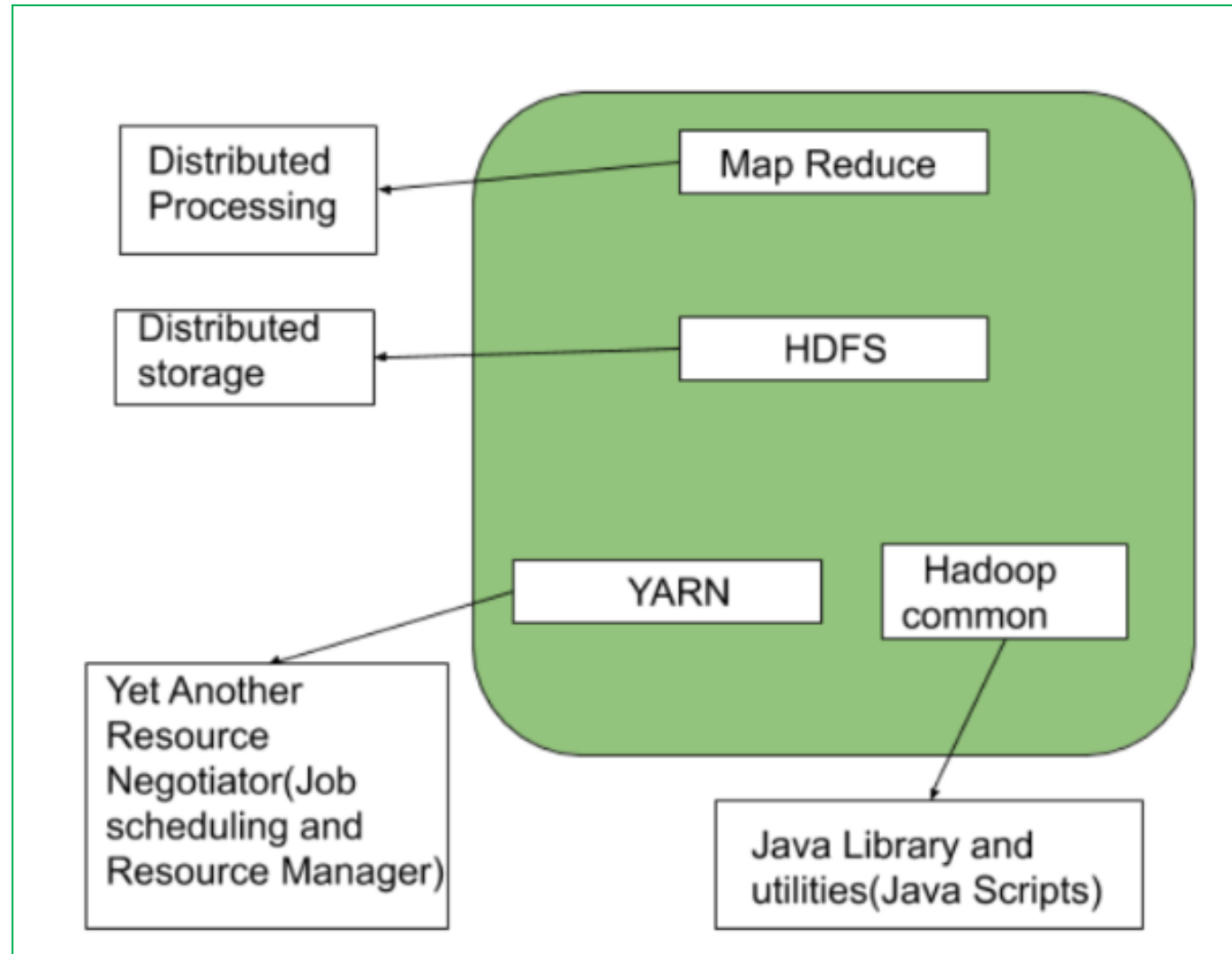
Big data characteristics and challenges

- Data volume: Big data sets can be very large and complex, which can make them difficult to store and analyze.
- Data velocity: Big data sets can be generated very quickly, which can make it difficult to keep up with the data flow.
- Data variety: Big data sets can be very heterogeneous, which can make it difficult to integrate and analyze the data.
- Data quality: Big data sets can contain errors and inconsistencies, which can make it difficult to trust the results of the analysis.
- Data security: Big data sets can be valuable targets for cyberattacks, which can make it important to protect the data from unauthorized access.

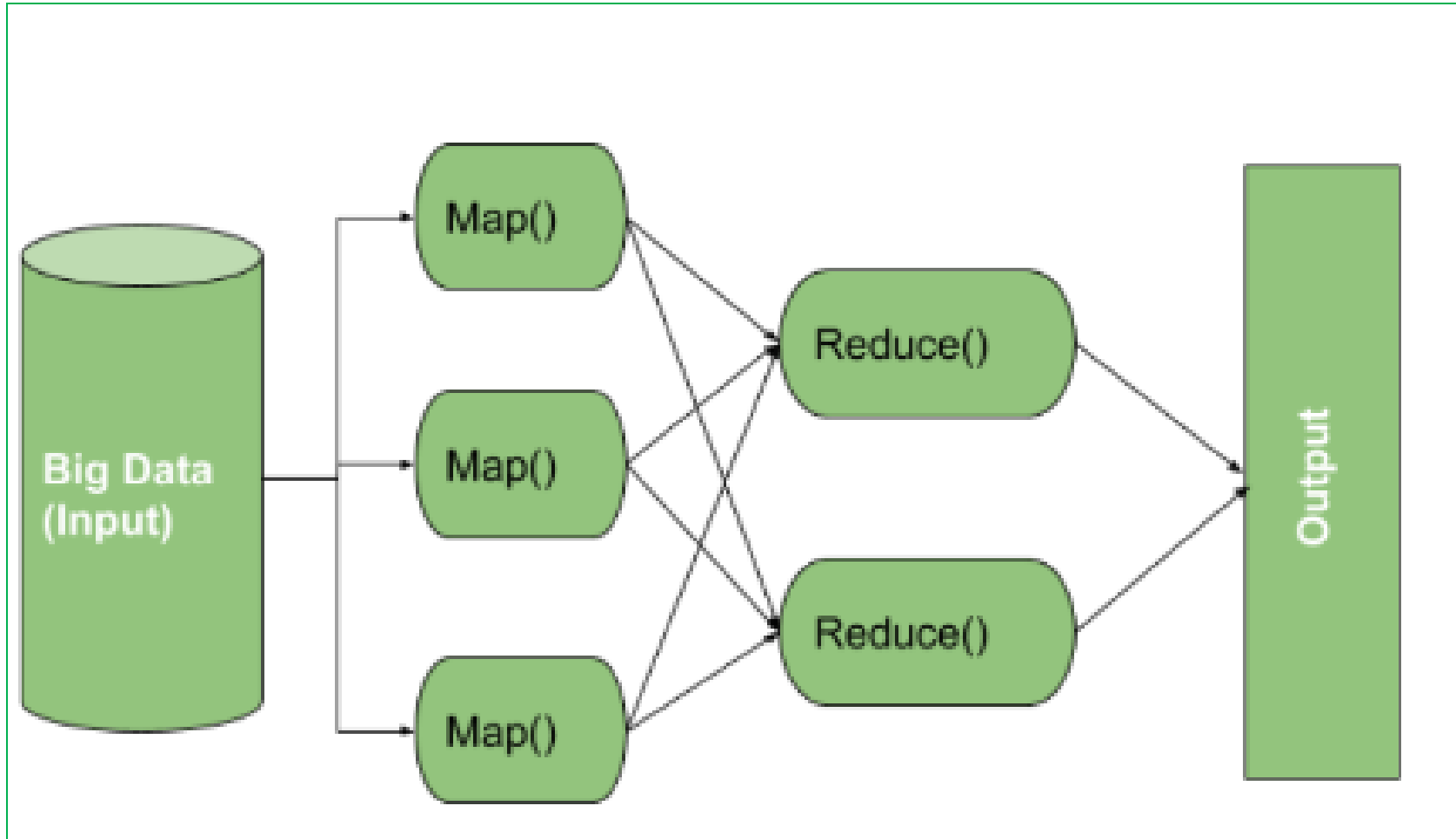
Big Data Tools and platforms

- Apache Hadoop: Hadoop is an open-source framework for storing and processing large data sets. It is a distributed system that can be used to process data on a large scale.
- Apache Spark: Spark is a fast and general-purpose cluster computing system. It can be used to process both structured and unstructured data.
- Apache Hive: Hive is a data warehouse infrastructure built on top of Hadoop. It provides SQL-like queries to access data stored in Hadoop.
- Apache Pig: Pig is a high-level language for expressing data manipulation operations on large data sets. It is used to process data stored in Hadoop.
- Apache Storm: Storm is a real-time distributed computation system. It can be used to process streaming data in real time.
- Apache Flink: Flink is a distributed stream and batch processing framework. It can be used to process both streaming and batch data.
- Amazon Web Services (AWS): AWS offers a wide range of big data tools and services, including Hadoop, Spark, Hive, Pig, Storm, and Flink.
- Microsoft Azure: Azure offers a similar range of big data tools and services to AWS.
- Google Cloud Platform (GCP): GCP also offers a range of big data tools and services, including Hadoop, Spark, Hive, Pig, Storm, and Flink.

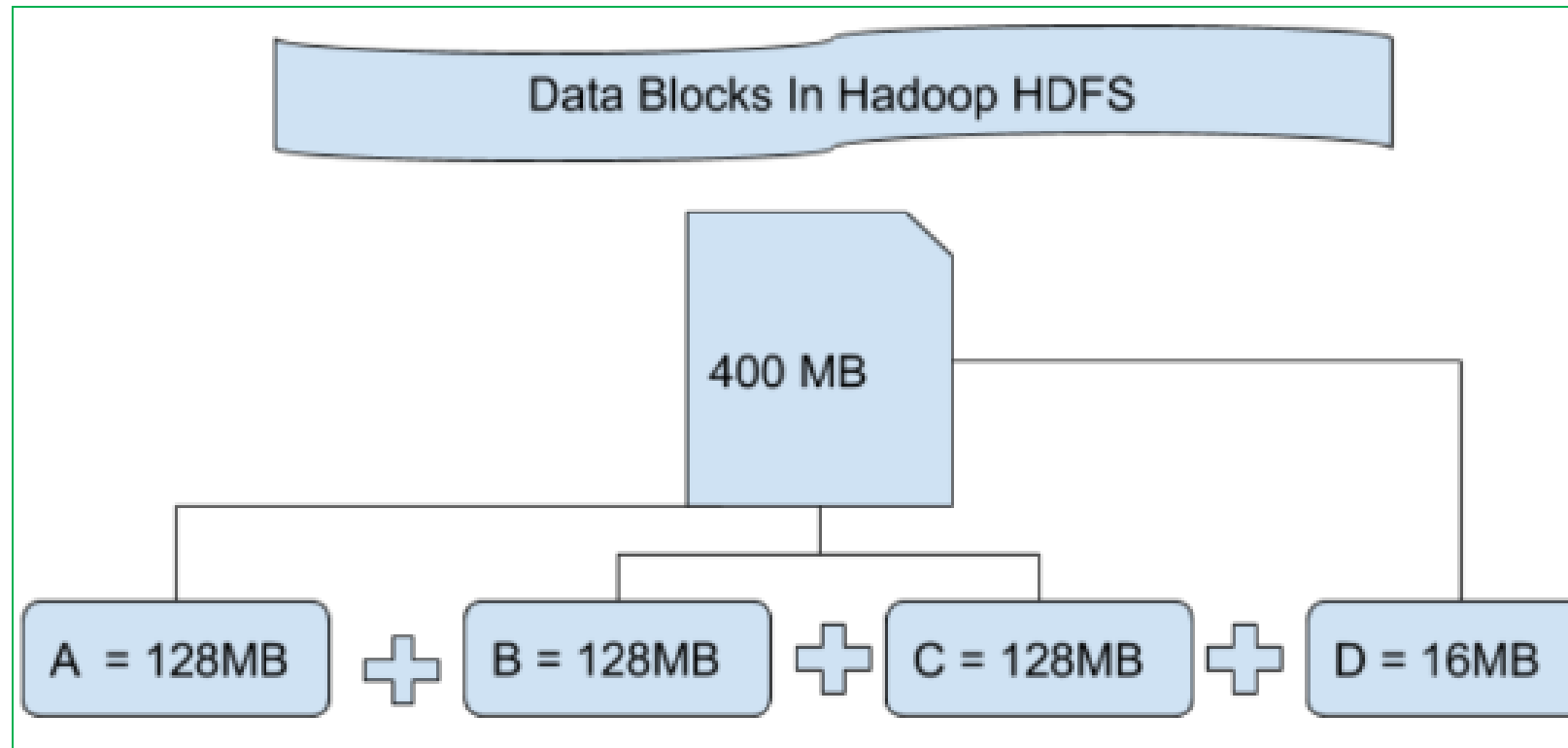
Architecture of Hadoop



Hadoop working- MAP Reduce

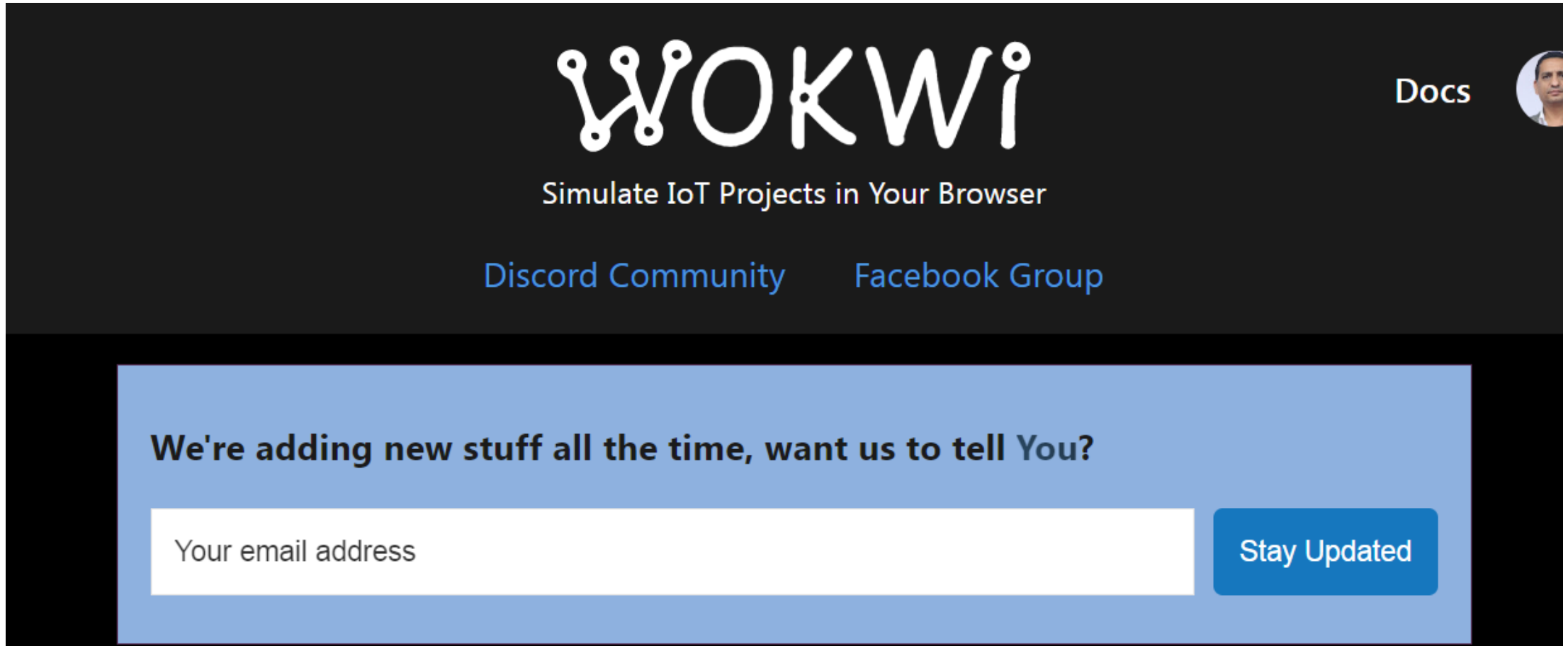


File storage

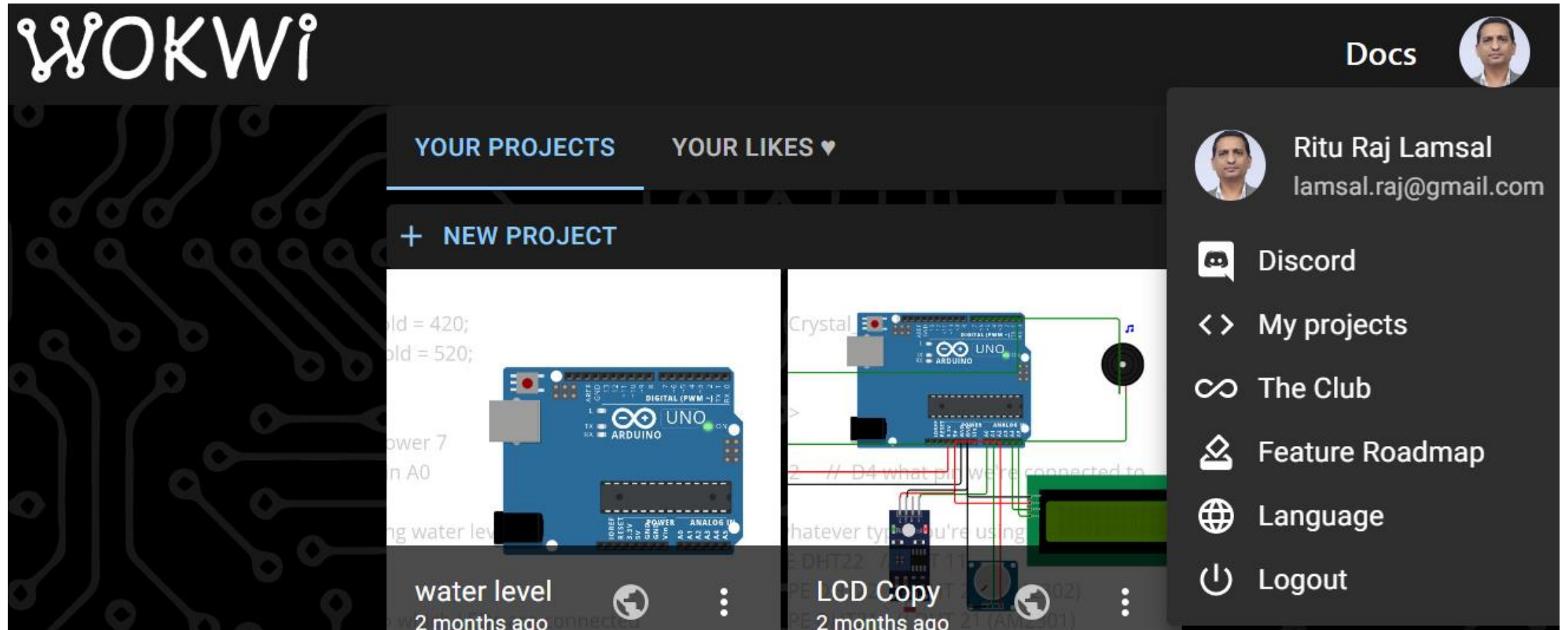


- Thank You

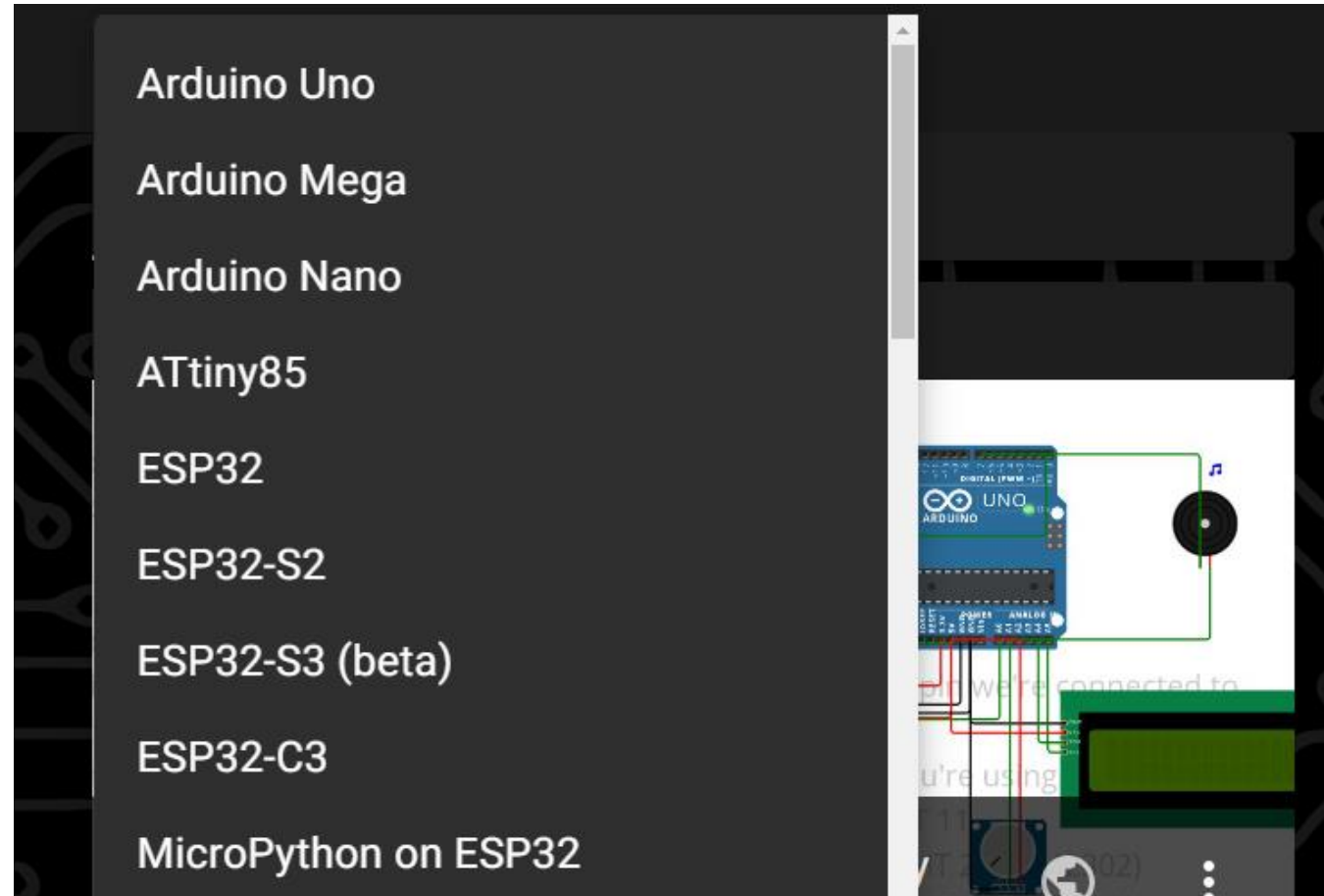
Simulator Wokwi



Go to My Project- Select New project



Select new Project




Select Arduino uno

WOKWI **SAVE** **SHARE** **Docs**

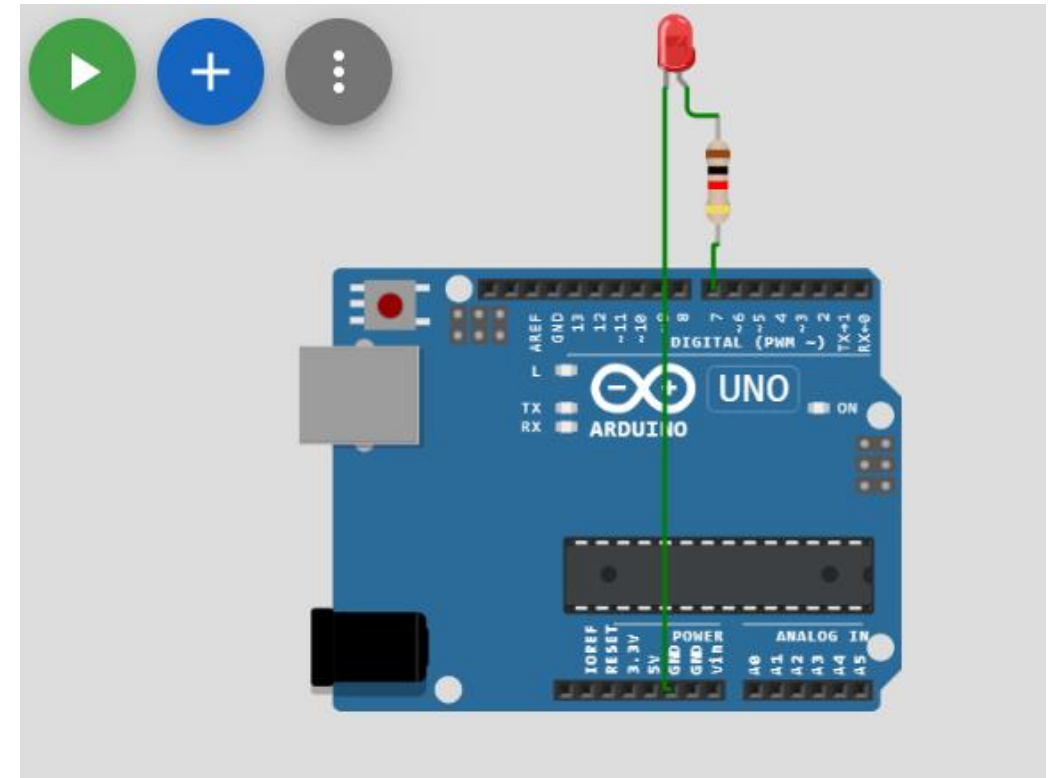
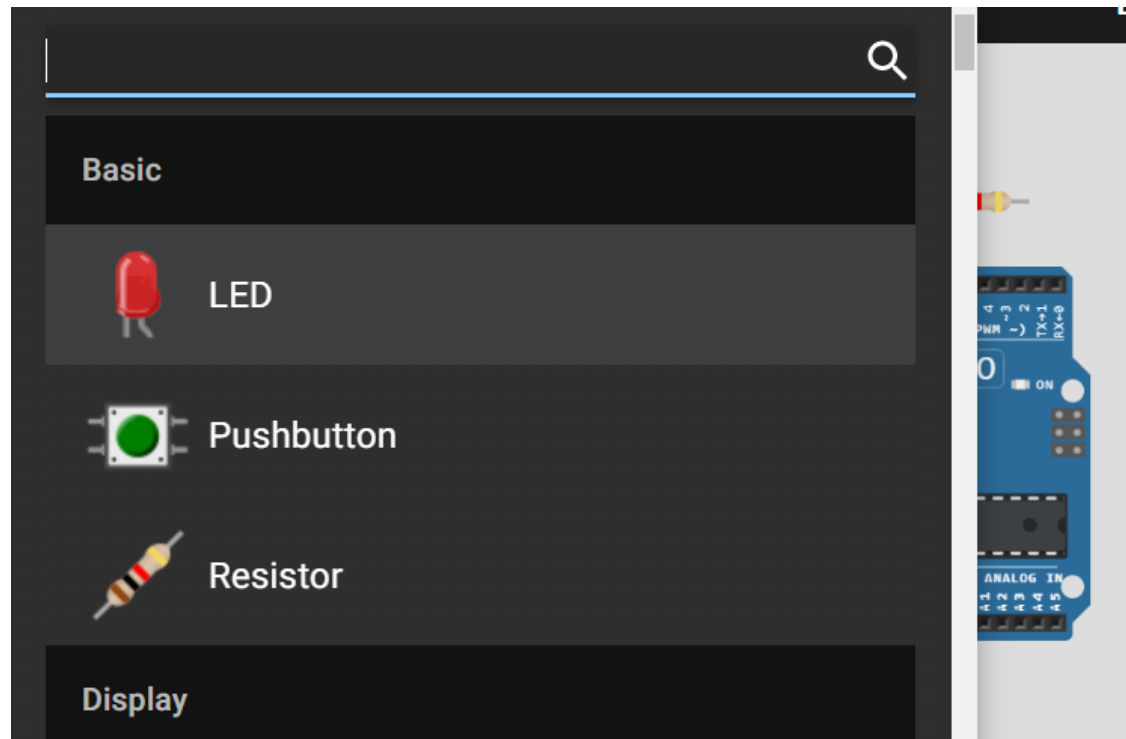
sketch.ino **diagram.json** ● **Library Manager** ▼

```
1 void setup() {  
2   // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7   // put your main code here, to run repeatedly:  
8  
9 }  
10
```

Simulation



Add new part and connect



Write a code to blink led

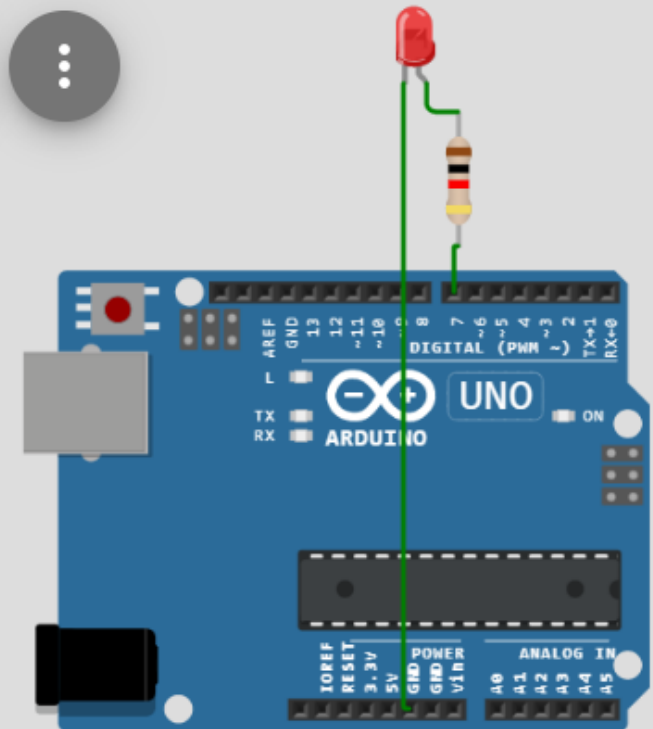
sketch.ino ●

diagram.json ●

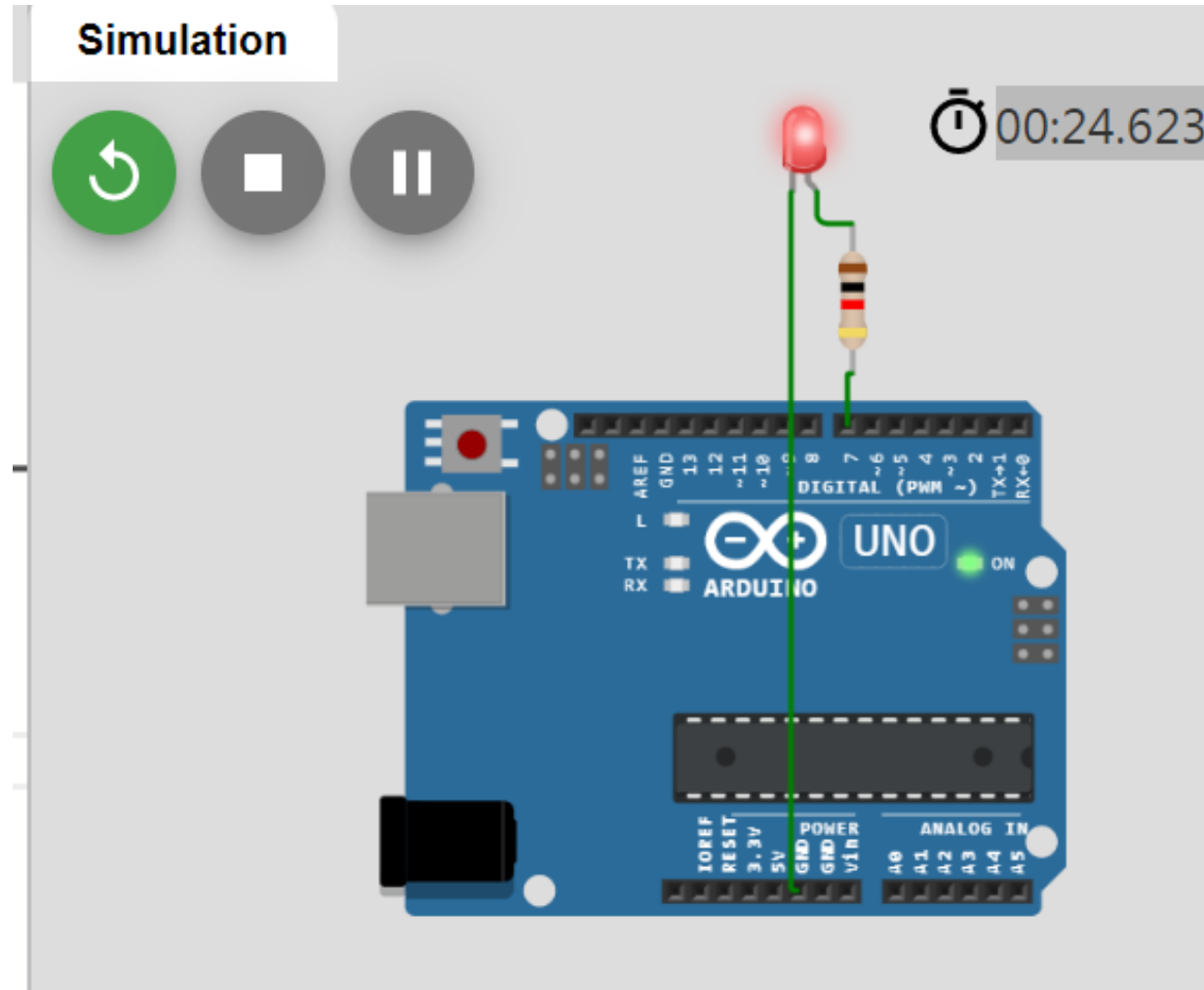
Library Manager ▼

```
1 void setup() {  
2   // put your setup code here, to run once:  
3   pinMode(7,OUTPUT);  
4 }  
5  
6 void loop() {  
7   // put your main code here, to run repeatedly:  
8   digitalWrite(7,HIGH);  
9   delay(200);  
10  digitalWrite(7,LOW);  
11  delay(500);  
12 }
```

Simulation



Run Simulation

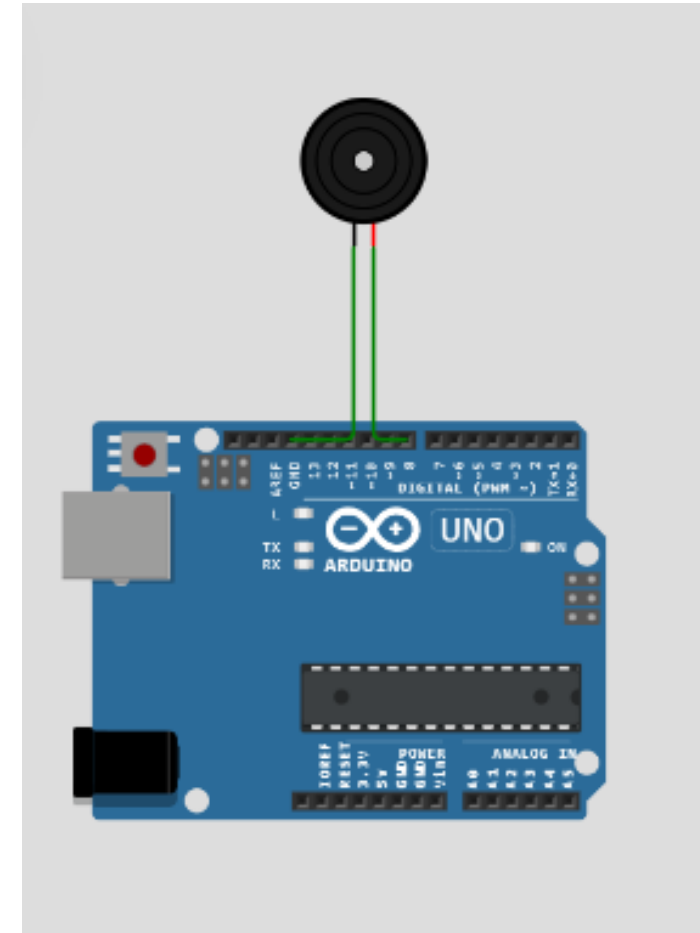


Simple Tune generator

```
void setup() {  
  pinMode(8, OUTPUT);  
}
```

```
void loop() {  
  tone(9,260);  
  delay(500);  
  noTone(9);  
  delay(500);  
}
```

```
void setup() {  
  pinMode(8, OUTPUT);  
}  
void loop() {  
  tone(8,260,500); // if notone is used  
  delay(1000);  
}
```



Happy birthday Tune

```
int speakerPin = 8;

int length = 28; // the number of notes

char notes[] = "GGAGcB GGAGdc GGxecBA yyecdc";

int beats[] = { 2, 2, 8, 8, 8, 16, 1, 2, 2, 8, 8,8, 16, 1, 2,2,8,8,8,16,
1,2,2,8,8,8,16 };

int tempo = 150;

void playTone(int tone, int duration) {
  for (long i = 0; i < duration * 1000L; i += tone * 2) {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration) {
  char names[] = {'C', 'D', 'E', 'F', 'G', 'A', 'B',

               'c', 'd', 'e', 'f', 'g', 'a', 'b',

               'x', 'y' };

  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014,

                956, 834, 765, 593, 468, 346, 224,

                655 , 715 };

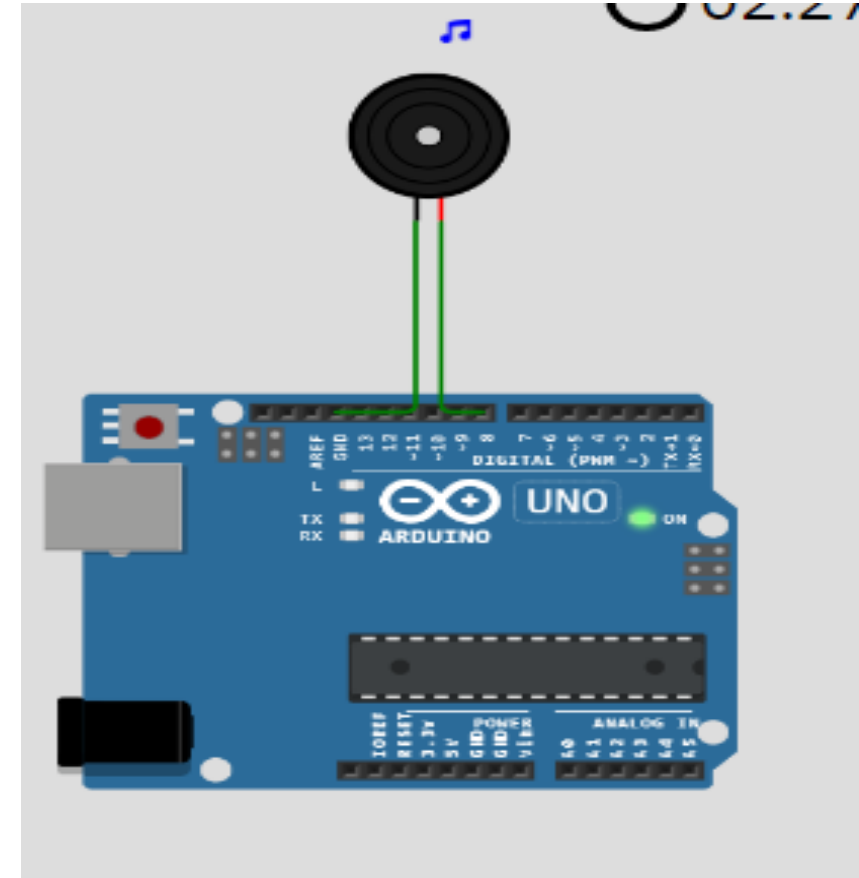
  int SPEE = 5;
```

```
// play the tone corresponding to the note name
for (int i = 0; i < 17; i++) {
  if (names[i] == note) {
    int newduration = duration/SPEE;
    playTone(tones[i], newduration);
  }
}

void setup() {
  pinMode(speakerPin, OUTPUT);
}

void loop() {
  for (int i = 0; i < length; i++) {
    if (notes[i] == ' ') {
      delay(beats[i] * tempo); // rest
    } else {
      playNote(notes[i], beats[i] * tempo);
    }

    // pause between notes
    delay(tempo);
  }
}
```



Cloud server for data visualization Thingspeak

- 1. Go to <https://thingspeak.com/> and create an account if you do not have one. Login to your account.
2. Create a new channel by clicking on the button.
- Enter the basic details of the channel. Then Scroll down and save the channel. You can follow the video guide below.
- 3. Then go to API keys copy and paste this key to a separate notepad file. You will need it later while programming.

Sign in



To send data faster to ThingSpeak or to send more data from more devices, consider the [paid license options](#) for commercial, academic, home and student usage.

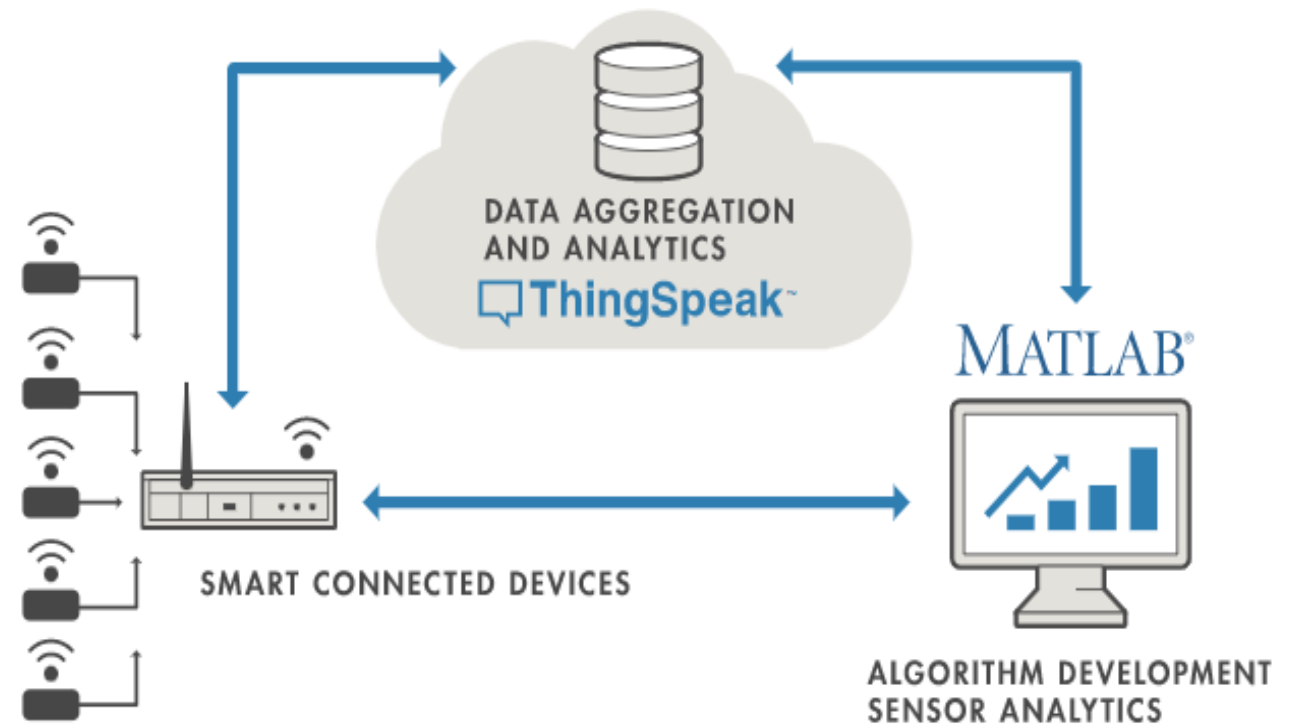


Email

No account? [Create one!](#)

By signing in, you agree to our [privacy policy](#).

Next



Signed in successfully.



My Channels

New Channel

Search by tag



Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to [create channels](#), explore and transform data.

Create Channel

New Channel

Name	<input type="text"/>	
Description	<input type="text"/>	
Field 1	<input type="text" value="Field Label 1"/>	<input checked="" type="checkbox"/>
Field 2	<input type="text"/>	<input type="checkbox"/>
Field 3	<input type="text"/>	<input type="checkbox"/>
Field 4	<input type="text"/>	<input type="checkbox"/>
Field 5	<input type="text"/>	<input type="checkbox"/>

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.

Create field and save channel

New Channel

Name	<input type="text" value="My weather station"/>	
Description	<input type="text" value="This displays Temperature and humidity"/>	
Field 1	<input type="text" value="Temp"/>	<input checked="" type="checkbox"/>
Field 2	<input type="text" value="Humidity"/>	<input checked="" type="checkbox"/>
Field 3	<input type="text"/>	<input type="checkbox"/>
Field 4	<input type="text"/>	<input type="checkbox"/>
Field 5	<input type="text"/>	<input type="checkbox"/>

Save channel

My weather station

Channel ID: **2144698**

Author: **mwa0000030122890**

Access: Private

This displays Temperature and humidity

Private View

Public View

Channel Settings

Sharing

API Keys

Data Import / Export

+ Add Visualizations

+ Add Widgets

Export recent data

MATLAB Analysis

MATLAB Visualization

Channel Stats

Created: less than a minute ago

Entries: 0

Save API Keys

Write API Key

Key

5MH4SDZG8226MSBX

Generate New Write API Key

Read API Keys

Key

NUVKFSBB33IT10UQ

Note

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=5MH4SDZG8226MSBX&field
```

API key directly can be used with browser

```
https://api.thingspeak.com/update?api_key=5MH4SDZG8226MSBX&field1=0
```

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=5MH4SDZG8226MSBX&field1=0
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/2144698/feeds.json?api_key=5MH4SDZG8226MSBX
```

API response

← → ↻ 🏠 api.thingspeak.com/update?api_key=5MH4SDZG8226MSBX&field1=20

1

← → ↻ 🏠 api.thingspeak.com/update?api_key=5MH4SDZG8226MSBX&field1=15&field2=50

← → ↻ 🏠 api.thingspeak.com/update?api_key=5MH4SDZG8226MSBX&field1=25&field2=26

3

Real Time Visualization

My weather station

Channel ID: 2144698

Author: mwa0000030122890

Access: Private

This displays Temperature and humidity

Private View

Public View

Channel Settings

Sharing

API Keys

Data Import / Export

+ Add Visualizations

+ Add Widgets

Export recent data

MATLAB Analysis

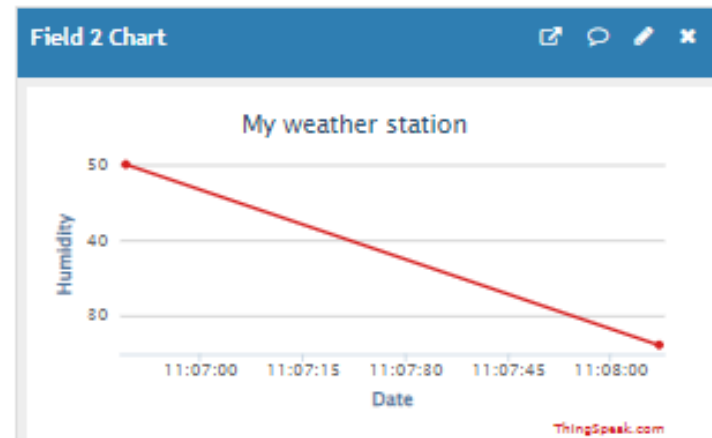
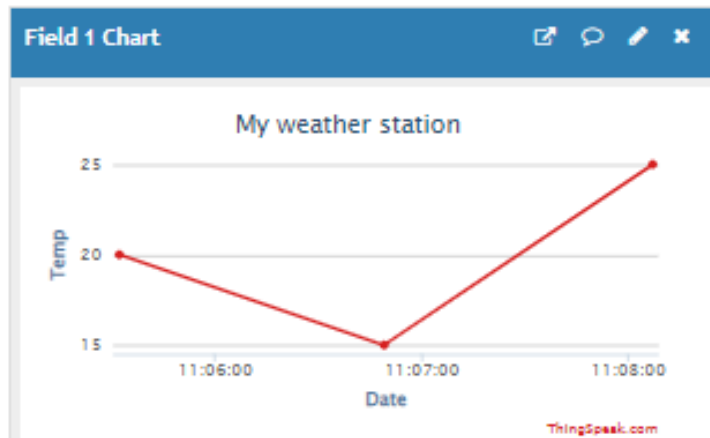
MATLAB Visualization

Channel Stats

Created: 8 minutes ago

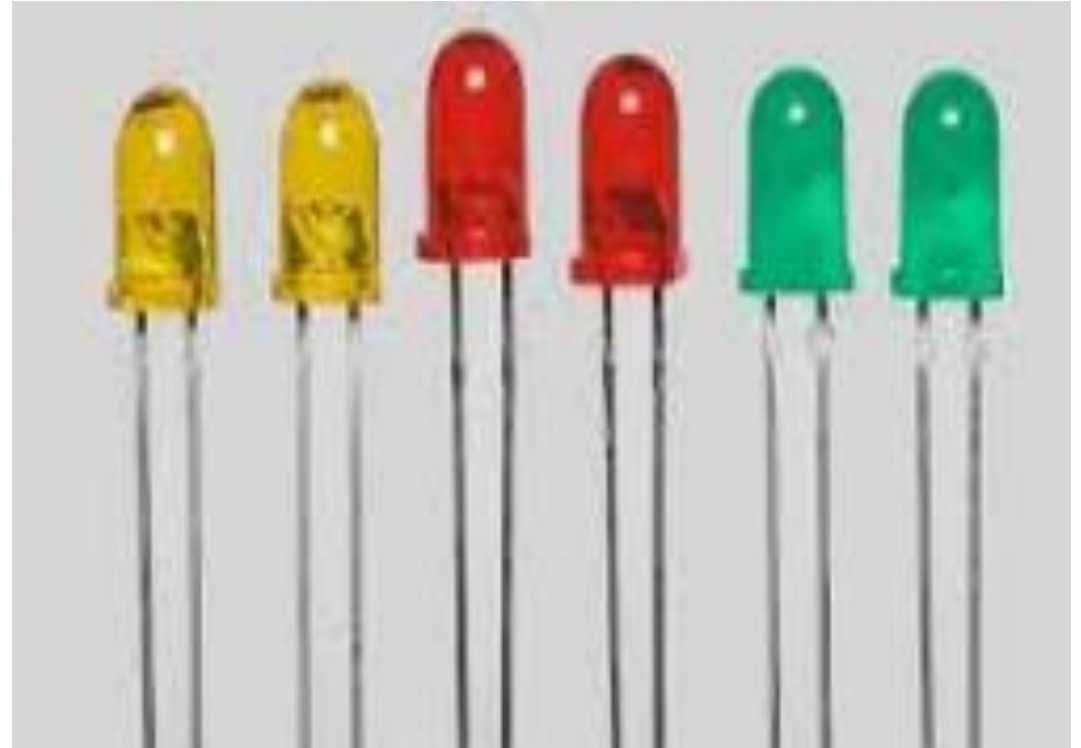
Last entry: 2 minutes ago

Entries: 3



Blinking Leds

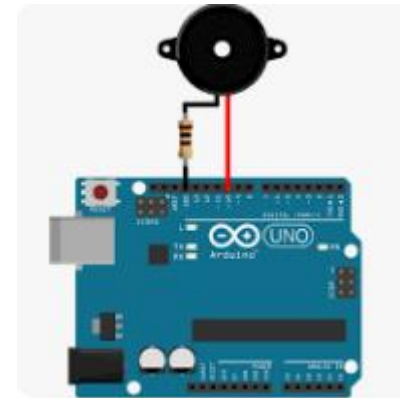
```
void setup() {  
  pinMode(13, OUTPUT); // set pin 13 as output  
  pinMode(12, OUTPUT); // set pin 12 as output  
}  
  
void loop() {  
  digitalWrite(13, HIGH); // turn on LED connected to pin 13  
  digitalWrite(12, LOW); // turn off LED connected to pin 12  
  delay(500); // wait for 500 milliseconds  
  digitalWrite(13, LOW); // turn off LED connected to pin 13  
  digitalWrite(12, HIGH); // turn on LED connected to pin 12  
  delay(500); // wait for 500 milliseconds  
}
```



Tone generator

```
int speakerPin = 8; // Speaker is connected to digital pin 8
```

```
void setup() {  
  pinMode(speakerPin, OUTPUT); // Set the speaker pin as output  
}  
void loop() {  
  tone(speakerPin, 440); // Generate a 440 Hz tone on the speaker pin  
  delay(1000); // Play the tone for one second  
  noTone(speakerPin); // Stop the tone  
  delay(1000); // Wait for one second before playing the next tone  
}
```



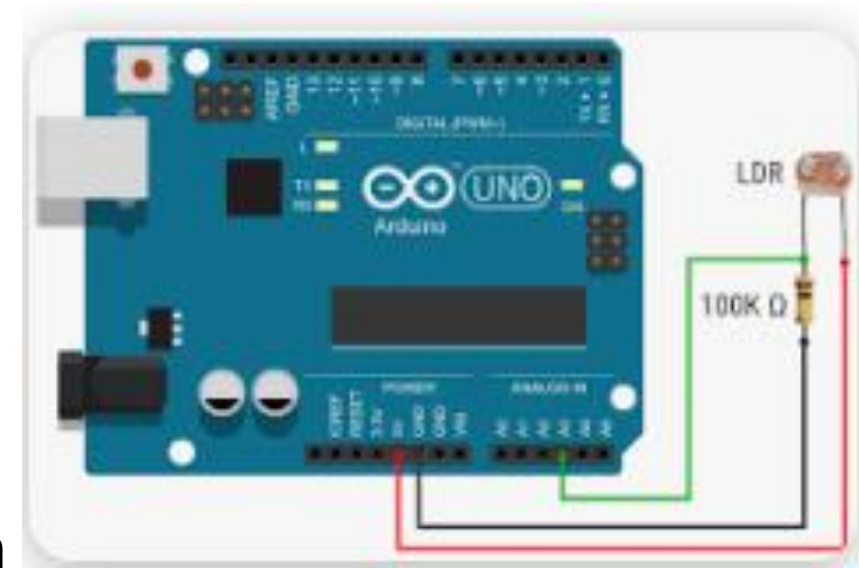
Happy Birthday Tone Generator

```
int speakerPin = 8; // Speaker is connected to digital pin 8
int melody[] = { // Notes in the melody of "Happy Birthday"
    NOTE_C4, NOTE_C4, NOTE_D4, NOTE_C4, NOTE_F4, NOTE_E4,
    NOTE_C4, NOTE_C4, NOTE_D4, NOTE_C4, NOTE_G4, NOTE_F4,
    NOTE_C4, NOTE_C4, NOTE_C5, NOTE_A4, NOTE_F4, NOTE_E4,
    NOTE_D4, NOTE_A4,
    NOTE_A4, NOTE_G4, NOTE_F4, NOTE_E4, NOTE_F4, NOTE_C5,
    NOTE_C5,
    NOTE_C5, NOTE_A4, NOTE_F4, NOTE_G4, NOTE_F4, NOTE_C5,
    NOTE_C5,
    NOTE_C5, NOTE_A4, NOTE_F4, NOTE_G4, NOTE_F4, NOTE_C5,
    NOTE_C5
};
int noteDurations[] = { // Durations of each note in the melody
    4, 4, 2, 2, 2, 1,
    4, 4, 2, 2, 2, 1,
    4, 4, 2, 2, 2, 2, 2, 1,
    4, 4, 2, 2, 2, 2, 1,
    4, 4, 2, 2, 2, 2, 1,
    4, 4, 2, 2, 2, 2, 1,
    4, 4, 2, 2, 2, 2, 1
};
```

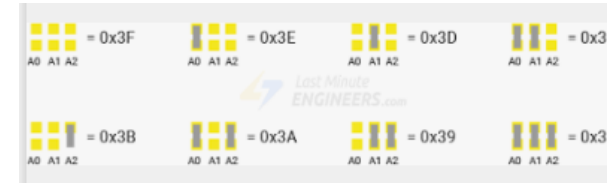
```
};
void setup() {
    pinMode(speakerPin, OUTPUT); // Set the speaker pin as output
}
void loop() {
    for (int i = 0; i < sizeof(melody)/sizeof(melody[0]); i++) {
        int noteDuration = 1000/noteDurations[i]; // Calculate the duration of
        the current note
        tone(speakerPin, melody[i], noteDuration); // Play the current note on
        the speaker
        delay(noteDuration*1.3); // Add a small delay before playing the next
        note
        noTone(speakerPin); // Stop the current note
    }
    delay(5000); // Wait 5 seconds before playing the melody again
}
```









Interfacing LDR

```
int ldrPin = A0; // define LDR pin as analog input
void setup() {
  Serial.begin(9600); // initialize serial communication
}
void loop() {
  int ldrValue = analogRead(ldrPin); // read LDR value from analog input pin
  Serial.print("LDR Value: ");
  Serial.println(ldrValue); // print LDR value to serial monitor
  delay(500); // wait for 500 milliseconds before taking the next reading
}
```



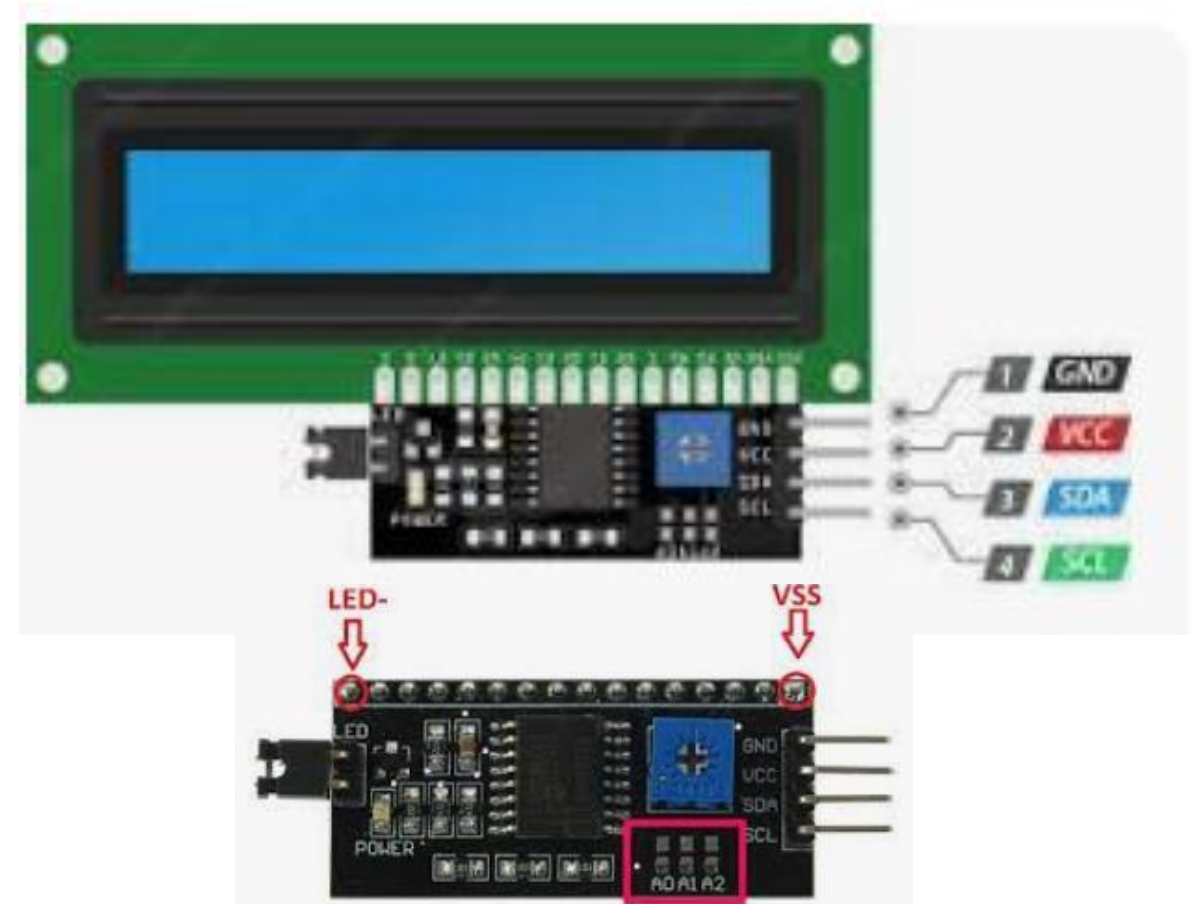
LCD interface 16*2 I2c



 = 0x3F	 = 0x3E	 = 0x3D	 = 0x3C
 = 0x3B	 = 0x3A	 = 0x39	 = 0x38

A0	A1	A2	HEX ADDRESS
1	1	1	0x27
0	1	1	0x26
1	0	1	0x25
0	0	1	0x24
1	1	0	0x23
0	1	0	0x22
1	0	0	0x21
0	0	0	0x20

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2); // 0x3F
void setup() {
  lcd.begin();
}
void loop() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Hi how are you");
  delay(1000);
}
```



Interfacing Relay

```
int relayPin = 8; // define relay pin
```

```
void setup() {  
    pinMode(relayPin, OUTPUT); // set relay pin as output  
}
```

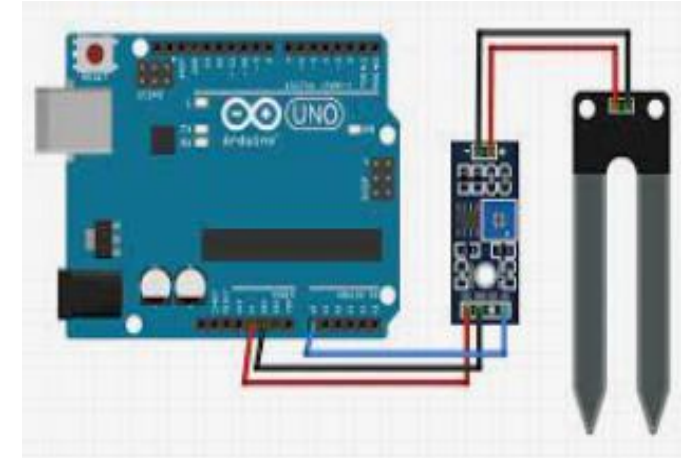
```
void loop() {  
    digitalWrite(relayPin, HIGH); // turn on relay  
    delay(1000); // wait for 1 second  
    digitalWrite(relayPin, LOW); // turn off relay  
    delay(1000); // wait for 1 second  
}
```



Soil Moisture sensor

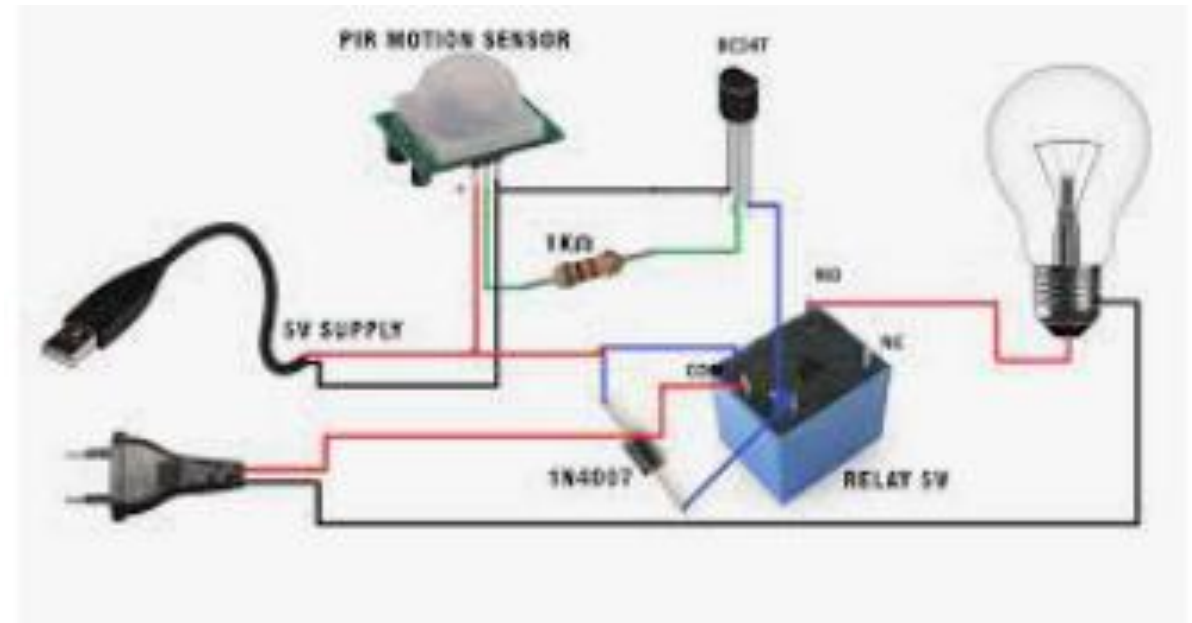
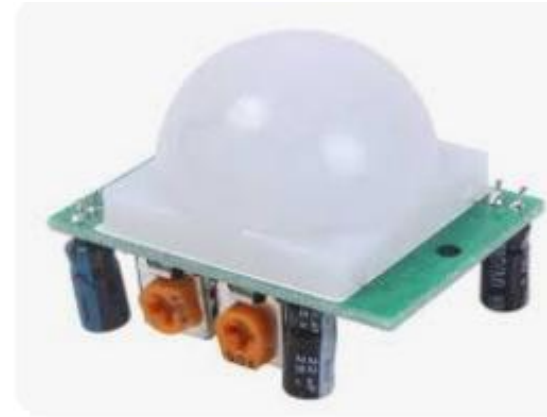
```
int soilMoisturePin = A0; // Soil moisture sensor is connected to analog pin A0
int soilMoistureValue; // Variable to store the soil moisture value
void setup() {
  pinMode(soilMoisturePin, INPUT); // Set the soil moisture sensor pin as input
  Serial.begin(9600); // Start the serial communication
}
void loop() {
  soilMoistureValue = analogRead(soilMoisturePin); // Read the soil moisture value from
the sensor
  Serial.print("Soil Moisture: ");
  Serial.print(soilMoistureValue); // Print the soil moisture value to the serial monitor
  Serial.println();

  delay(1000); // Wait for a second before taking the next reading
}
```



PIR motion Detection

```
int pirPin = 2; // PIR sensor is connected to pin 2
int ledPin = 13; // LED is connected to pin 13
void setup() {
  pinMode(pirPin, INPUT); // Set the PIR sensor pin as input
  pinMode(ledPin, OUTPUT); // Set the LED pin as output
  Serial.begin(9600); // Start the serial communication
}
void loop() {
  int motion = digitalRead(pirPin); // Read the value of the PIR sensor
  if (motion == HIGH) { // If motion is detected
    digitalWrite(ledPin, HIGH); // Turn on the LED
    Serial.println("Motion detected!");
    delay(1000); // Wait for a second
  } else {
    digitalWrite(ledPin, LOW); // Turn off the LED
  }
}
```



Ultrasonic sensor for distance measurement



```
#define trigPin 9 // define trig pin
#define echoPin 10 // define echo pin
void setup() {
  Serial.begin(9600); // initialize serial communication
  pinMode(trigPin, OUTPUT); // set trig pin as output
  pinMode(echoPin, INPUT); // set echo pin as input
}
void loop() {
  long duration, distance; // define variables
  // send a 10 microsecond pulse to the trig pin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
```

```
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // measure the duration of the pulse from the echo pin
  duration = pulseIn(echoPin, HIGH);
  // calculate the distance in centimeters based on the
  duration
  distance = duration / 58;
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  delay(500); // wait for half a second
}
```

Mq gas sensor

```
int gasPin = A0; // Gas sensor is connected to analog pin A0
```

```
void setup() {
```

```
  Serial.begin(9600); // Initialize serial communication at  
  9600 baud rate
```

```
}
```

```
void loop() {
```

```
  int gasValue = analogRead(gasPin); // Read the analog  
  value from the gas sensor
```

```
  Serial.print("Gas value: "); // Print the gas value to the  
  serial monitor
```

```
  Serial.println(gasValue);
```

```
  delay(1000); // Wait for 1 second before taking another  
  reading
```

```
}
```




```
int alcoholPin = A0; // Alcohol sensor is connected to analog pin A0
int buzzerPin = 9; // Buzzer is connected to digital pin 9
void setup() {
  Serial.begin(9600); // Initialize serial communication at 9600 baud rate
  pinMode(buzzerPin, OUTPUT); // Set the buzzer pin as an output
}
void loop() {
  int alcoholValue = analogRead(alcoholPin); // Read the analog value from the alcohol sensor
  Serial.print("Alcohol value: "); // Print the alcohol value to the serial monitor
  Serial.println(alcoholValue);
  if (alcoholValue > 500) { // If alcohol value is greater than 500 (you can adjust this threshold to suit your needs)
    digitalWrite(buzzerPin, HIGH); // Turn on the buzzer
    delay(1000); // Wait for 1 second
    digitalWrite(buzzerPin, LOW); // Turn off the buzzer
  }
  delay(1000); // Wait for 1 second before taking another reading
}
```

Temperature Humidity Sensor

```
#include <DHT.h>

#define DHTPIN 2    // define DHT sensor pin
#define DHTTYPE DHT11 // set DHT sensor type to DHT11
DHT dht(DHTPIN, DHTTYPE); // initialize DHT sensor

void setup() {
    Serial.begin(9600); // initialize serial communication
    dht.begin();        // start DHT sensor
}

void loop() {
    float temperature, humidity; // define variables for temperature and humidity
    // read temperature and humidity from DHT sensor
    humidity = dht.readHumidity();
    temperature = dht.readTemperature();

    // check if the readings are valid
    if (isnan(humidity) || isnan(temperature)) {
        Serial.println("Failed to read from DHT sensor!");
    }
    else {
        Serial.print("Temperature: ");
        Serial.print(temperature);
        Serial.print(" °C, Humidity: ");
        Serial.print(humidity);
        Serial.println(" %");
    }
    delay(2000); // wait for two seconds before taking the next reading
}
```



Temp Humidity Webserver

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <DHT.h>
#define DHTPIN D4    // define DHT sensor pin
#define DHTTYPE DHT11 // set DHT sensor type to DHT11
const char* ssid = "YOUR_SSID"; // your network SSID (name)
const char* password = "YOUR_PASSWORD"; // your network password
ESP8266WebServer server(80); // create web server object
DHT dht(DHTPIN, DHTTYPE); // initialize DHT sensor

void setup() {
    Serial.begin(115200); // initialize serial communication
    WiFi.begin(ssid, password); // connect to WiFi network
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting to WiFi...");
    }
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    server.on("/", handleRoot); // set root page handler
    server.begin(); // start web server
    Serial.println("Web server started");
    dht.begin(); // start DHT sensor
}

void loop() {
    server.handleClient(); // handle web server requests

    float temperature, humidity; // define variables for temperature and humidity
    // read temperature and humidity from DHT sensor
    humidity = dht.readHumidity();
    temperature = dht.readTemperature();
    // check if the readings are valid
    if (isnan(humidity) || isnan(temperature)) {
        Serial.println("Failed to read from DHT sensor!");
    }
    else {
        Serial.print("Temperature: ");
        Serial.print(temperature);
        Serial.print(" °C, Humidity: ");
        Serial.print(humidity);
        Serial.println(" %");
        // format the temperature and humidity values into a string
        String temperatureString = String(temperature, 1);
        String humidityString = String(humidity, 1);
        String response = "Temperature: " + temperatureString + " °C<br>Humidity: " + humidityString + " %";
        // send the response to the client
        server.send(200, "text/html", response);
    }
}

// read temperature and humidity from DHT sensor
humidity = dht.readHumidity();
temperature = dht.readTemperature();
// check if the readings are valid
if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
}
else {
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.print(" °C, Humidity: ");
    Serial.print(humidity);
    Serial.println(" %");
}
delay(2000); // wait for two seconds before taking the next reading
}

// function to handle root page request
void handleRoot() {
    float temperature, humidity; // define variables for temperature and humidity
```

HTML and Webserver Temp and humidity

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <DHT.h>
#define DHTPIN D4 // define DHT sensor pin
#define DHTTYPE DHT11 // set DHT sensor type to DHT11
const char* ssid = "YOUR_SSID"; // your network SSID (name)
const char* password = "YOUR_PASSWORD"; // your network password
ESP8266WebServer server(80); // create web server object
DHT dht(DHTPIN, DHTTYPE); // initialize DHT sensor

void setup() {
  Serial.begin(115200); // initialize serial communication
  WiFi.begin(ssid, password); // connect to WiFi network
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("WiFi connected");
  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());

  server.on("/", handleRoot); // set root page handler
  server.begin(); // start web server
  Serial.println("Web server started");
  dht.begin(); // start DHT sensor
}

void loop() {
  server.handleClient(); // handle web server requests
}

// function to handle root page request
void handleRoot() {
  float temperature, humidity; // define variables for temperature and humidity
  // read temperature and humidity from DHT sensor
  humidity = dht.readHumidity();
  temperature = dht.readTemperature();
  // check if the readings are valid
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");

    server.send(200, "text/plain", "Failed to read from DHT sensor!");
  }
  else {
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.print(" °C, Humidity: ");
    Serial.print(humidity);
    Serial.println(" %");
    // format the temperature and humidity values into a string
    String temperatureString = String(temperature, 1);
    String humidityString = String(humidity, 1);
    String htmlPage = "<html><body><h1>Temperature and Humidity</h1>";
    htmlPage += "<p>Temperature: " + temperatureString + " &deg;C</p>";
    htmlPage += "<p>Humidity: " + humidityString + " %</p></body></html>";
    // send the HTML page to the client
    server.send(200, "text/html", htmlPage);
  }
}
```

