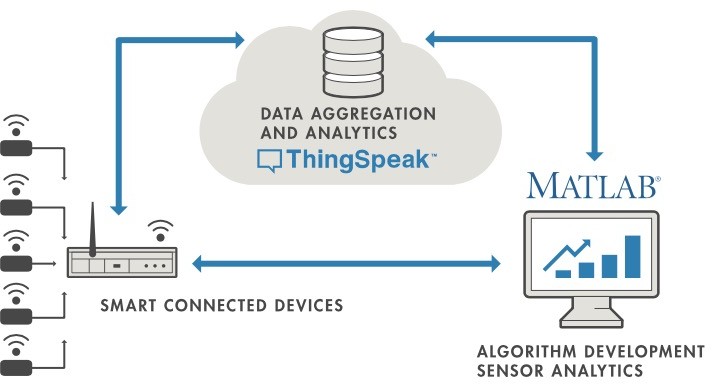
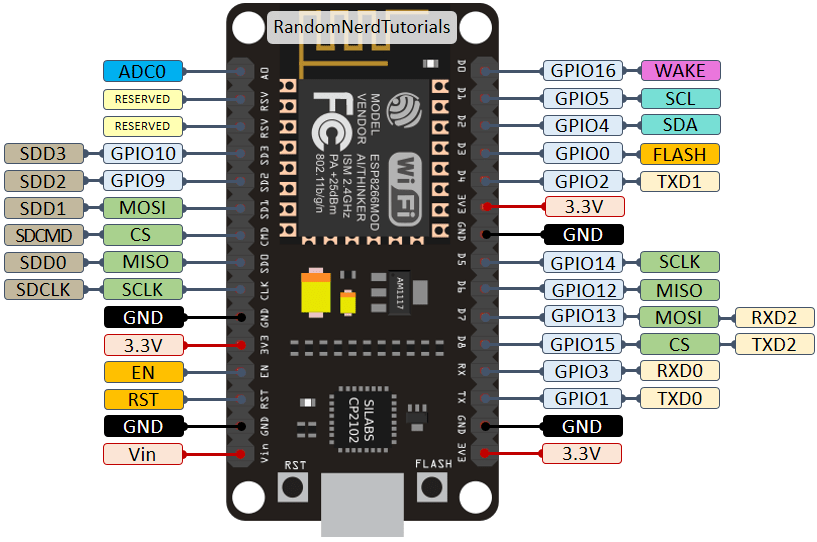
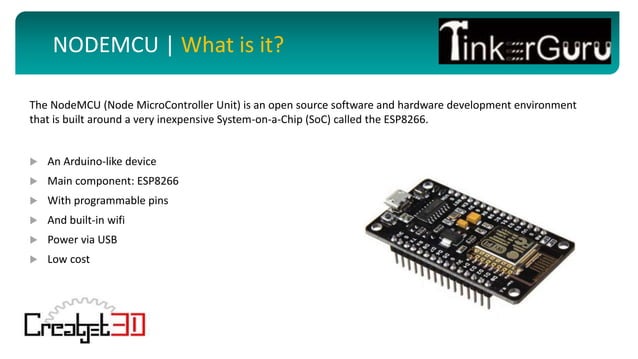
**1. Explain about the thingspeak iot cloud application**

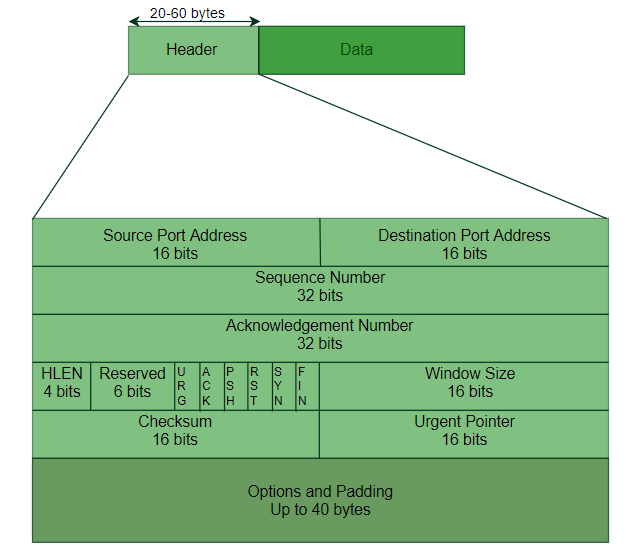
* A. ThingSpeak is a Web Service (REST API) that lets you collect and store sensor data in the cloud and develop Internet of Things applications.
* It works with Arduino, Raspberry Pi and MATLAB (premade libraries and APIs exists).
* But it should work with all kind of Programming Languages, since it uses a REST API and HTTP.



**2. Draw neat labelled diagram of NodeMCU**

**A.** 

**3. Explain briefly about TCP header format(with diagram)**

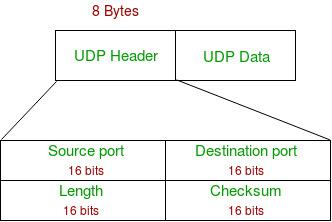
**A.** 

The header of a TCP segment can range from 20-60 bytes. 40 bytes are for options. If there are no options, a header is 20 bytes else it can be of upmost 60 bytes.   
Header fields: 

* Source Port Address –   
  A 16-bit field that holds the port address of the application that is sending the data segment.
* Destination Port Address –   
  A 16-bit field that holds the port address of the application in the host that is receiving the data segment.
* Sequence Number –   
  A 32-bit field that holds the sequence number, i.e, the byte number of the first byte that is sent in that particular segment. It is used to reassemble the message at the receiving end of the segments that are received out of order.
* Acknowledgement Number –   
  A 32-bit field that holds the acknowledgement number, i.e, the byte number that the receiver expects to receive next. It is an acknowledgement for the previous bytes being received successfully.
* Header Length (HLEN) –   
  This is a 4-bit field that indicates the length of the TCP header by a number of 4-byte words in the header, i.e if the header is 20 bytes(min length of TCP header), then this field will hold 5 (because 5 x 4 = 20) and the maximum length: 60 bytes, then it’ll hold the value 15(because 15 x 4 = 60). Hence, the value of this field is always between 5 and 15.
* Control flags –   
  These are 6 1-bit control bits that control connection establishment, connection termination, connection abortion, flow control, mode of transfer etc. Their function is:
  + URG: Urgent pointer is valid
  + ACK: Acknowledgement number is valid (used in case of cumulative acknowledgement)
  + PSH: Request for push
  + RST: Reset the connection
  + SYN: Synchronize sequence numbers
  + FIN: Terminate the connection
* Window size –   
  This field tells the window size of the sending TCP in bytes.
* Checksum –   
  This field holds the checksum for error control. It is mandatory in TCP as opposed to UDP.
* Urgent pointer –   
  This field (valid only if the URG control flag is set) is used to point to data that is urgently required that needs to reach the receiving process at the earliest. The value of this field is added to the sequence number to get the byte number of the last urgent byte.

**4. Explain briefly about UDP header format (with diagram)**

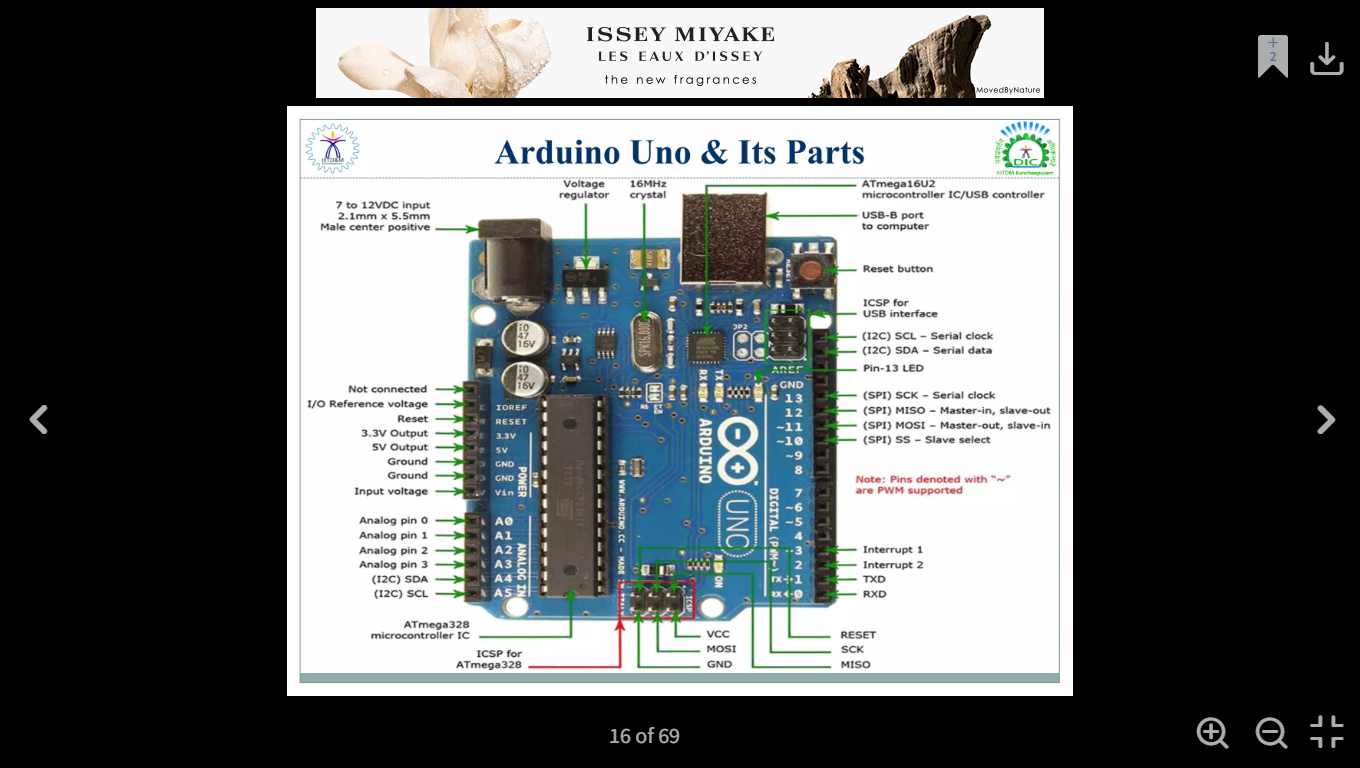
**A.** UDP header is an 8-byte fixed and simple header, while for TCP it may vary from 20 bytes to 60 bytes. The first 8 Bytes contain all necessary header information and the remaining part consists of data. UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535; port number 0 is reserved. Port numbers help to distinguish different user requests or processes.



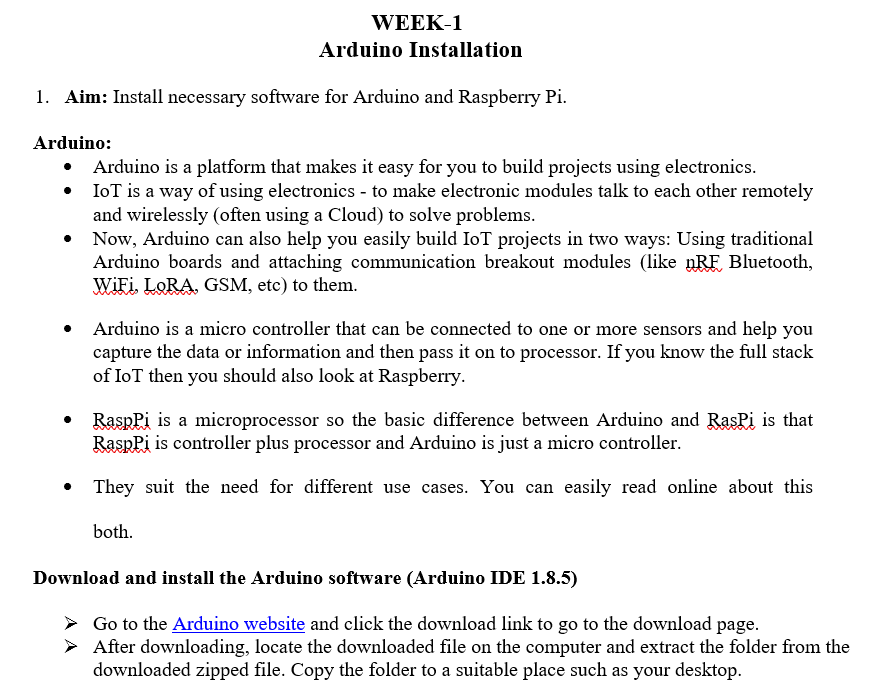
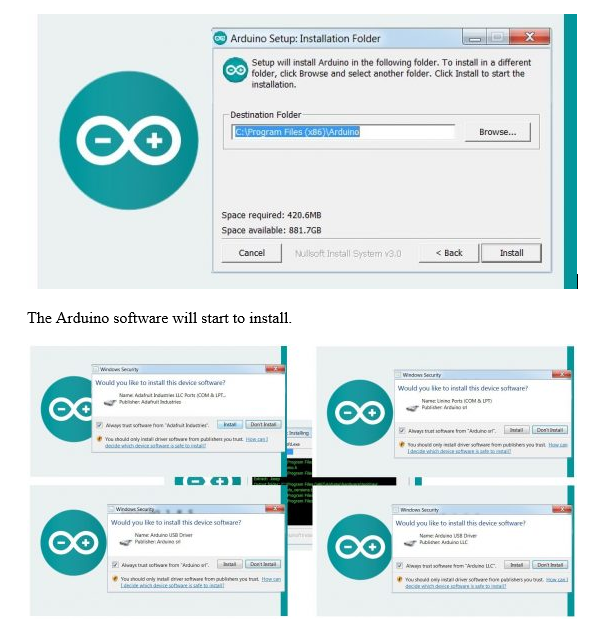
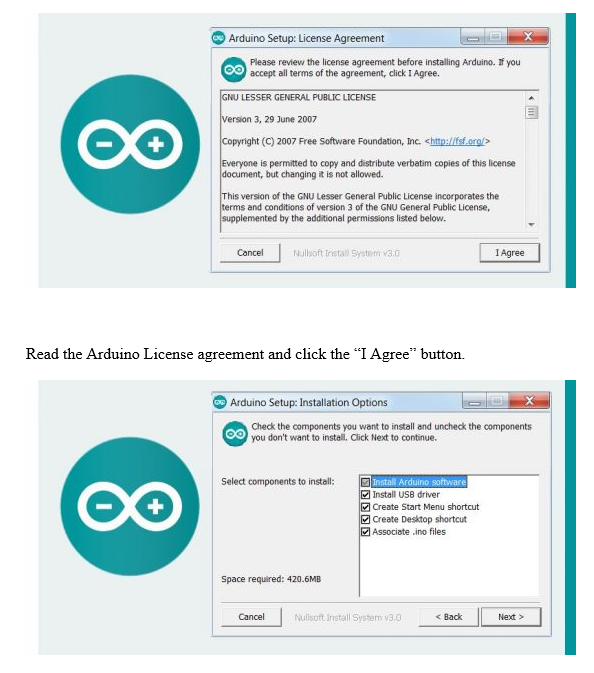
*UDP Header*

* Source Port: Source Port is a 2 Byte long field used to identify the [port number](https://www.geeksforgeeks.org/what-is-ports-in-networking/)of the source.
* Destination Port: It is a 2 Byte long field, used to identify the port of the destined packet.
* Length: Length is the length of UDP including the header and the data. It is a 16-bits field.
* Checksum: Checksum is 2 Bytes long field. It is the 16-bit one’s complement of the one’s complement sum of the UDP header, the pseudo-header of information from the IP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

**5. Draw a neat labelled pin diagram of Arduino UNO board**

**A.** 

**6. Write the steps for the installation of Arduino UNO**

**A.**  

**7. What are the applications of DHT sensor**

**A.** The DHT11 is a digital temperature and humidity sensor that provides calibrated output via a single-wire communication protocol.

* It is commonly used in
* weather monitoring systems,
* home automation, and
* IoT applications.

1. Weather Monitoring Systems

🌤️ Used in DIY and professional weather stations to measure temperature and humidity levels.

2. Home Automation & Smart Homes

🏡 Integrated into IoT-based smart home systems for automatic climate control, triggering fans, air purifiers, or dehumidifiers.

3. Greenhouse Monitoring

🌱 Helps maintain optimal temperature and humidity levels for plant growth in agricultural environments.

4. Industrial Environmental Monitoring

🏭 Used in factories and warehouses to monitor environmental conditions that affect machinery and stored goods.

5. HVAC (Heating, Ventilation, and Air Conditioning) Systems

❄️ Helps in climate control systems by providing real-time temperature and humidity data for automated adjustments.

**8. What are the features of ThingSpeak and How does it work?**

**A. ThingSpeak** is a cloud-based IoT platform designed for collecting, storing, analyzing, and visualizing real-time data from connected devices. It is widely used in academic and research projects due to its simplicity and powerful features.

**Features of ThingSpeak:**

1. **Real-time Data Collection:**  
   ThingSpeak allows devices like Arduino or NodeMCU to send sensor data (e.g., temperature, humidity, pressure) over the internet and store it in channels.
2. **Customizable Visualizations:**  
   Users can create line charts, bar graphs, and other plots to monitor sensor readings live, helping them analyze trends and behavior over time.
3. **Multiple Data Fields:**  
   Each channel can have up to 8 data fields, along with location and status updates, making it suitable for multi-sensor systems.
4. **MATLAB Integration:**  
   ThingSpeak integrates with MATLAB, enabling users to write custom scripts for data analysis, filtering, and even triggering alerts or control actions.
5. **RESTful and MQTT APIs:**  
   It supports both HTTP (REST) and MQTT protocols, allowing flexible and secure data transfer between devices and the cloud.
6. **Alerts and Triggers:**  
   Users can set up email alerts, ThingHTTP, and webhooks to notify or control devices based on specific data conditions.
7. **Data Export and Sharing:**  
   Data can be exported for offline analysis or shared with others using public or private access options.

**How ThingSpeak Works:**

1. **Channel Creation:**  
   Users begin by creating a "channel" on ThingSpeak to store their data. Each channel has a unique Channel ID, Write API Key, and Read API Key.
2. **Sending Data:**  
   Devices like ESP8266 or Arduino collect data from sensors and send it to ThingSpeak using the Write API Key via HTTP requests or MQTT messages.
3. **Storing Data:**  
   ThingSpeak receives the incoming data and logs it into the appropriate fields of the designated channel.
4. **Data Visualization:**  
   The platform automatically plots this data on interactive graphs, which can be customized in terms of type, time range, and appearance.
5. **Data Analysis:**  
   MATLAB code can be written in the "Apps" section to process incoming data—perform calculations, identify anomalies, or apply logic to trigger alerts.
6. **Triggering Actions:**  
   Based on analyzed data, ThingSpeak can interact with other services or devices using webhooks or ThingHTTP, enabling real-time automation.

**Conclusion:**  
ThingSpeak is a powerful yet user-friendly IoT platform that simplifies cloud integration for sensor-based projects. With real-time data visualization, built-in analytics, and automation features, it is highly effective for developing smart IoT systems.

**9. How to create My Channel on ThingSpeak**

**A. How to Create My Channel on ThingSpeak (5-Mark Answer):**

Creating a channel on ThingSpeak is the first step to store and visualize data from your IoT devices. Here's a step-by-step guide to create your own channel:

**1. Sign In or Create an Account:**

* Visit <https://thingspeak.com>.
* If you don’t have an account, sign up using your email or MathWorks account.
* Log in to access the dashboard.

**2. Navigate to Channels:**

* On the dashboard, click on the **"Channels"** tab at the top.
* Select **"My Channels"** to view or manage your channels.

**3. Create a New Channel:**

* Click the **"New Channel"** button.
* Fill in the following details:
  + **Name:** Give a descriptive name for your channel (e.g., "Weather Station").
  + **Description:** Briefly describe what the channel will monitor.
  + **Field 1 to Field 8:** Tick the boxes for fields you want to use and name them (e.g., Temperature, Humidity).
  + **Location Info (Optional):** Add latitude, longitude, and elevation.
  + You can also enable public view if you want others to access your data.

**4. Save the Channel:**

* Click **"Save Channel"** at the bottom of the page.
* Your new channel is now created.

**5. Get API Keys:**

* After saving, go to the **"API Keys"** tab.
* Note the **Write API Key** (used to send data) and **Read API Key** (used to view data).
* These keys are required when programming your Arduino, ESP8266, or other devices to interact with ThingSpeak.

**6. Start Using Your Channel:**

* Begin sending data using HTTP or MQTT protocols with your device.
* Go to the **"Private View"** or **"Public View"** tab to see real-time data visualized as charts.

**Conclusion:**  
Creating a ThingSpeak channel is simple and crucial for managing IoT data. By properly setting up fields and using the API keys, you can build powerful data-driven projects with visualization, analysis, and automation features.

**10. What is the difference between ReadAPI keys and WriteAPI key?**

**A. Difference between Read API Key and Write API Key (5-Mark Answer):**

In ThingSpeak, **API Keys** are used to control access to your channel. The two main types are the **Read API Key** and the **Write API Key**, each serving a different purpose:

**1. Read API Key:**

* **Purpose:**  
  Used to **read or retrieve** data from a ThingSpeak channel.
* **Functionality:**  
  It allows users or devices to access and view the data stored in your channel’s fields, status updates, or location info.
* **Usage Example:**  
  If you have a web app or mobile app that displays sensor values, it will use the Read API key to fetch that data securely.
* **Security:**  
  Keeps your channel data protected by allowing only authorized users/devices to view the data.

**2. Write API Key:**

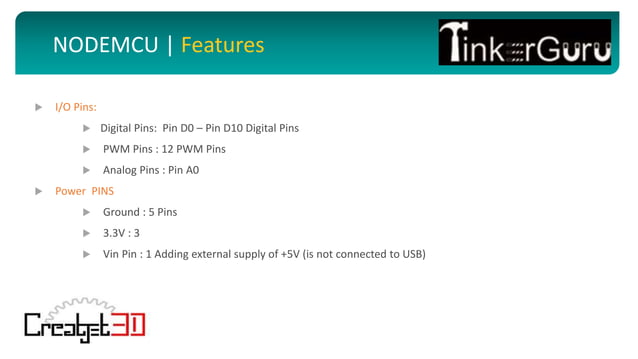
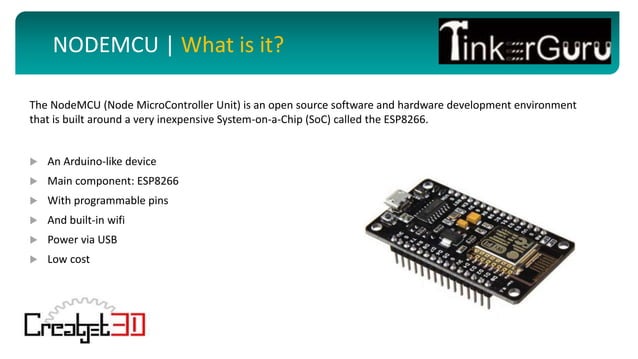
* **Purpose:**  
  Used to **send or write** data to a ThingSpeak channel.
* **Functionality:**  
  Devices like Arduino, NodeMCU, or Raspberry Pi use the Write API key to update sensor readings into the channel.
* **Usage Example:**  
  If you want to send temperature and humidity data from a DHT sensor to ThingSpeak, the code on your microcontroller must include the Write API key.
* **Security:**  
  Ensures that only authorized devices or applications can send data to your channel.

**Summary Table:**

| **Feature** | **Read API Key** | **Write API Key** |
| --- | --- | --- |
| **Used For** | Reading/viewing data | Writing/sending data |
| **Permission** | Read-only | Write-only |
| **Used By** | Apps, users, or scripts fetching data | Devices/sensors updating data |
| **Example Use** | Dashboard display | Upload sensor data |

**Conclusion:**  
The Read API key is for accessing existing data, while the Write API key is for updating or adding new data. Keeping these keys secure ensures proper and safe use of your ThingSpeak channels.

**11. What are the features of NodeMCU?**

**A.** 

**12. Raspberry Pi board pin diagram.**

**A.** 