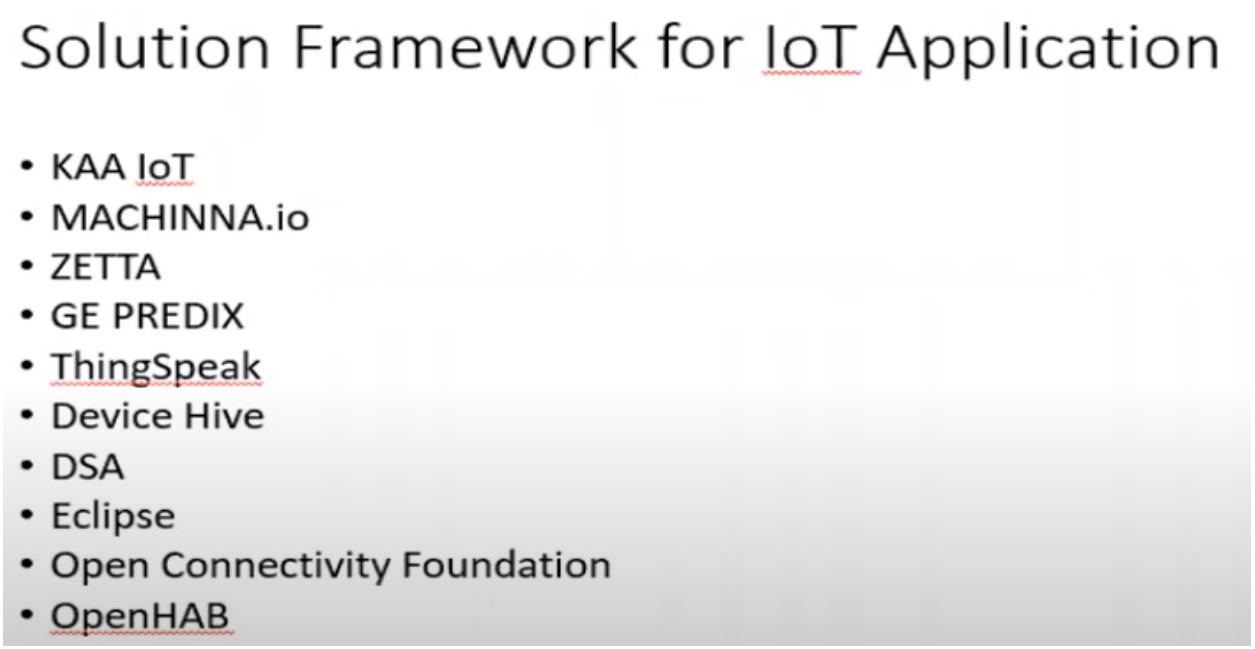
Unit-4

Different solution frameworks for to implement an IOT applications



features

- **Connectivity**
- **Device management**
- **Data collection**
- **Data processing and analytics**
- **Data visulization**
- **Alert trigger and notifications**
- **Configuration management**
- **Command execution**
- **multitenancy**

Kaa IoT

- Kaa is an end-to-end IoT platform applicable for any scale of enterprise IoT projects.

MACHINNA.IO

- Macchina.io EDGE is an open source software toolkit for quickly building embedded applications for the Internet of Things that run on Linux-based devices like the Raspberry Pi, Beaglebone or similar.

- It implements a web-enabled, modular and extensible JavaScript and C++ runtime environment and provides readily available, easy to use building blocks that allow your application to talk to various sensors and devices, as well as cloud services.

ZETTA

- Zetta is an open source platform built on Node.js for creating Internet of Things servers that run across geo-distributed computers and the cloud.
- Zetta combines REST APIs, WebSockets and reactive programming – perfect for assembling many devices into data-intensive, real-time applications.

GE-PREDIX

- It provides secure and scalable edge-to-cloud data connectivity, analytics processing, and many other underlying services that power your GE Digital applications and IoT initiatives.

Thingspeak

- ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualization of live data, and send alerts.

implement the device integration in IoT Application and Deployment

Scenarios in different domains

- **Integration** means making independently designed applications and data work well together. IoT integration means making the mix of new IoT devices, IoT data, IoT platforms and IoT applications — combined with IT assets (business applications, legacy data, mobile, and SaaS) — work well together in the context of implementing end-to-end IoT business solutions.
- **Device integration**

The connection of various devices in order to allow them to communicate, interact, and interoperate. The main issue to face to achieve integration is the heterogeneity of communication protocols and data formats

Device integration has the potential to revolutionize business processes in a multitude of domains and ways.

Let's look at five of the most important advantages that device integration can provide in terms of driving more efficient procedures.

1. Improved data accuracy

An automated approach increases the accuracy of data collected by an organization, in part by removing the danger of data being erroneously transferred or lost in transit to a centralized system. Furthermore, an integrated approach allows for automated data validation, resulting in higher-quality data and less opportunities for human error.

2. Greater efficiency

A process with connected devices is substantially more efficient when properly executed, allowing a team to focus on higher-value activities rather than manually transmitting data. Because data is processed as soon as it is collected, not only does the data collection process become more efficient, but an organization may make judgments or act on more up-to-date data.

3. More effective decentralized team

A decentralized team can have regular access to relevant data with effective device integration, allowing the support team to monitor and analyze data from any location. This enhanced coordination can be a major benefit for large or small firms dispersed Across the country or the globe.

4. More reliable response strategies

Automated monitoring and extensive logging are possible because to thoughtful device integration, which means that a system could trigger alert conditions in particular instances. Rather than depending solely on human inspection, firms may verify that automated responses to specific mistake scenarios are consistent and trustworthy.

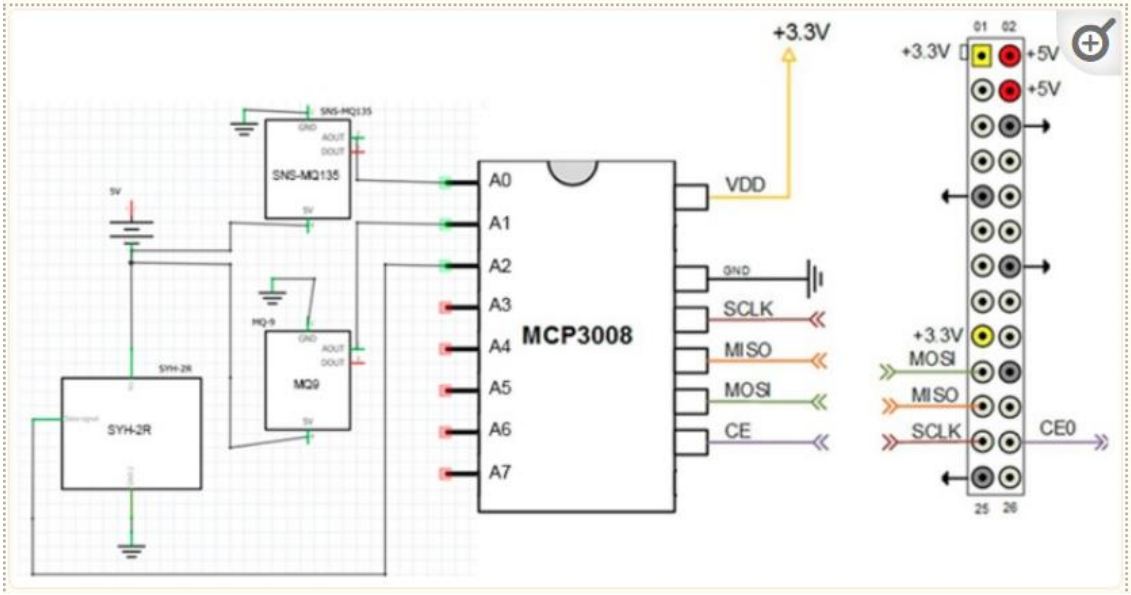
5. An increasingly rich archive of data

Collecting a body of logged data allows firms the capacity to examine it afterwards, possibly even refining their business processes further based on the data they collect, much like any database system. Device integration entails creating a growing and interconnected repository of usable data, which can be quite valuable over time.

Implement the air pollution monitoring system using the Web Socket approach, using MCP3008 - A/D converter

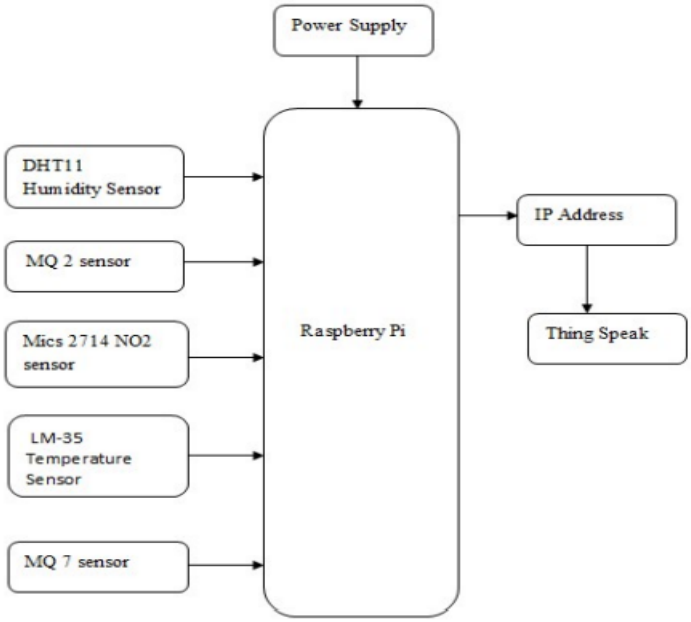
- The MCP 3008 is an analog-to-digital converter, with which an analogue size can be accepted at the input, and a number representing an approximation of the analog signal of the input signal is outputted.
- This model works with eight analogue channels, allowing up to eight analog sensors to be connected. The working voltage is 3.3 V and the operating temperature is between −40 and 85 °C.
- The SNS-MQ135 is an electrochemical sensor that measures air quality and is capable of detecting levels of carbon dioxide (CO_2), ammonium (NH_4), ethanol (C_2H_6O), and toluene (C_7H_8).
- The sensor has an analogue and digital output, uses a working voltage of 5 V, and has a current consumption of ~40 mA. The working temperature is between −20 and 50 °C.
- MQ9 is a gas sensor that measures the level of carbon monoxide (CO) and methane (CH4). Like SNS-MQ135, this sensor has analogue and digital output, a working voltage of 5 V, and a current consumption of ~40 mA. The working temperature is between −20 and 50 °C.
- The two sensors in the MQ series (MQ135, MQ9) require a pre-heating period of at least 24 h, when commissioned for calibration.
- SYH-2R is an analog sensor for the detection of relative humidity levels. The working voltage is 5 V and the operating temperature is between −20 and 85 °C.
- BMP280 is a sensor designed by Bosch to detect temperature (±1 °C accuracy) and barometric pressure (±1 hPa accuracy). It contains a voltage regulator and can use a working voltage of 3.3 to 5 V.

- The operating temperature is between −40 and 85 °C. This sensor can communicate through both the Serial Peripheral Interface (SPI) interface and Inter-Integrated Circuit (I2C) bus.
- The analogue sensors are connected to the MCP3008 converter as follows: The SNS-MQ135 is connected to the analogue channel, A0; the MQ9 is connected to the analogue channel, A1; and the humidity sensor, SYH-2R, is connected to the A2 channel. The converter connects through the SPI interface to the Raspberry Pi board (Figure).



- The connection of the analog-digital converter to Raspberry Pi is done via the SPI interface (Figure 3) as follows: VDD pin connects to 3.3 V—GPIO 01, the GND pin connects to the ground pin—GPIO 06, the SCLK pin connects at SPI_CLK—GPIO 11, the MISO pin connects to SPI_MISO—GPIO 09, the MOSI pin connects to SPI_MOSI—GPIO 10, and the CE pin connects to SPI_CE0—GPIO 08.
- Interfacing the BMP280 temperature and pressure sensor with Raspberry Pi is done by the I2C protocol as follows: The SDI pin on the sensor connects to SDA1—GPIO 02, the CS pin connects to SCL1—GPIO 03, the GND pin to the ground-board on the breadboard, and the pin 3.3 V to the power plug on the breadboard.
- Connecting the status LEDs is done through the following pins: Green LED connects to GPIO 21, blue LED connects to GPIO 12, and red LED connects to GPIO 05. All LEDs connect to the line ground on the breadboard.

Using web socket approach



- The entire system is connected through various connectors. the sensors are connected to the Arduino Mega which acts as the interface.
- The Arduino Mega needs a power source so that the sensors start working. The Raspberry Pi provides the power supply to the Arduino Mega as well as the sensors.
- The Raspberry Pi can be connected to the power supply either through a USB Cable or a charger. Once the power supply is given, the sensors start to detect data.
- The data detected by the sensors is sent to the Arduino Mega, when the sensor specific code is run on it. We can actually see the data coming through the serial monitor in the Arduino IDE.
- A UDP connection is created between the Arduino and Raspberry Pi through a Ruby Code which is run on the Raspberry Pi. Once the data reaches the Raspberry Pi, the only thing left out is to send the data to the web server hosted by Ruby on Rails.
- The Ruby on Rails framework implements Web Socket through Action Cable. Once this action cable is established, there will be full duplex communication channel between the web server and the Raspberry Pi. This channel exist until it is manually stopped, Hence the real time data reading can be achieved

Various steps in data acquiring and storing data for an IoT application.What is the use of GPIO pins in a IoT device?

Data Acquisition

• Data acquisition means acquiring the data from IOT/M2M devices

•The data communicate after the interactions with a Data acquisition system (Application)

•The Application interacts and communicates with number of devices for acquiring the needed data

•The devices send data on demand or at the programmed intervals

• Data of devices communicate using the network, transport and security layers

Data store: The acquired data stores in the databases at a server

• Database

•Relational database

• Flat file

• Spreadsheet

• Mail server

•Web server

Data Storage Three Categories

1. On-line or real time or streaming data needing the processing, and only the results of processing and analysis need storage
2. Data called once, only the results of processing at a later time and of analysis store
3. Data needing repeated calls store for reference or audit in future

- VMware at one node or distributed multiple nodes • A Data Store is a data repository of a set of objects which integrate into the Store.
- Objects in a Data Store model using Classes which the database schemas define.
- Data Store may be distributed over multiple nodes, (Apache Cassandra is example of distributed Data Store.)
- A Data Store may consist of multiple schemas or may consist of data in only one scheme. (Example of only one scheme Data Store is relational database.)

Data Store at Server

• For short reaction times, Optimised performance and high security

Data Centre Data Store

• Data security and protection using the advanced tools, full data backups along with data recovery, redundant data communication connections and full system power

Data Store Management

• Data Store requires Data Centre management or Server management.

Spatial storage

• Spatial database optimised to store, enables querying the data objects defined in a geometric space, and which is a database for 2D and 3D objects

•Topological coverage, linear networks, triangular irregular networks or other complex structures.

Use of GPIO pins in IoT

- GPIOs are used in a diverse variety of applications, limited only by the electrical and timing specifications of the GPIO interface and the ability of software to interact with GPIOs in a sufficiently timely manner.
- GPIOs usually employ standard logic levels and cannot supply significant current to output loads.
- When followed by an appropriate high-current output buffer (or mechanical or solid-state relay), a GPIO may be used to control high-power devices such as lights, solenoids, heaters, and motors (e.g., fans and blowers). Similarly, an input buffer, relay or opto-isolator is often used to translate an otherwise incompatible signal (e.g., high voltage) to the logic levels required by a GPIO.
- Integrated circuit GPIOs are commonly used to control or monitor other circuitry (including other ICs) on a board.

Write a note on: 1) Trust for IoT 2) Security and Privacy for IoT 3) Physical IoT Security

Trust for IoT

- Light weight public key infrastructure
- Light weight key management systems
- Quality of information
- Decentralized and self configuring systems
- Novel methods for assessing trust in people

Security for IoT

Dos and DDOS prevention

- General attack detection and recovery
- Cyber situation tools and techniques
- Variety of access control and associated accounting schemes
- Handling virtually all modes of operation

privacy for IoT

- Cryptographic techniques
- Techniques to supports privacy by design concepts
- Fine-grained and self-configuring access control mechanism
- Preserving location privacy
- Prevention of personal information inference
- Use of soft identities

(iii) physical iot security

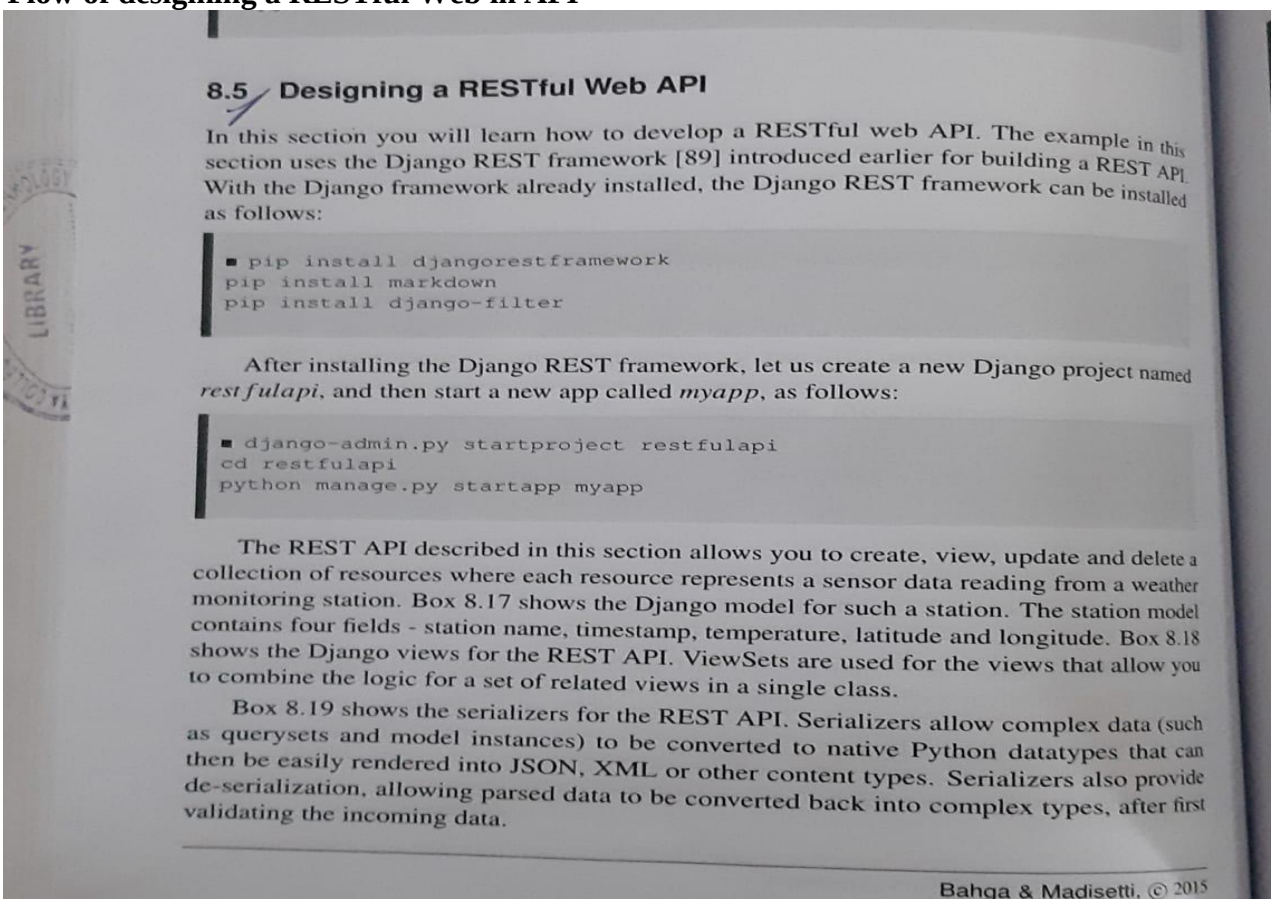
- From the moment of its creation, any physical device is liable to be tampered with in a way not intended by the manufacturer or retailer.
- IT devices in particular are a target for those people who are just plain curious, hackers seeking a new challenge to their technical skills, people trying to steal corporate knowledge about products & services, those seeking financial gain and a multitude of others pursuing an assortment of malicious intent.

IoT devices are a particular target as they have many features that can be of interest to others:

- They can be in private, remote or unattended locations so physical access is almost unrestricted with no time constraints.
- Some IoT devices can small in size and therefore easy to conceal if stolen.
- Many such devices are technically limited, for example limited processing capability or restricted power supply, which in turn generally means that security measures are similarly limited, potentially making a compromise easier to achieve.

- It is often clear by the nature and location of a device that if it were compromised then it could be used to attack a very specific target.
- It may be possible that once one device is compromised then a multitude of similar devices could also be compromised more easily.

Flow of desigining a RESTful Web in API



8.5 Designing a RESTful Web API

In this section you will learn how to develop a RESTful web API. The example in this section uses the Django REST framework [89] introduced earlier for building a REST API. With the Django framework already installed, the Django REST framework can be installed as follows:

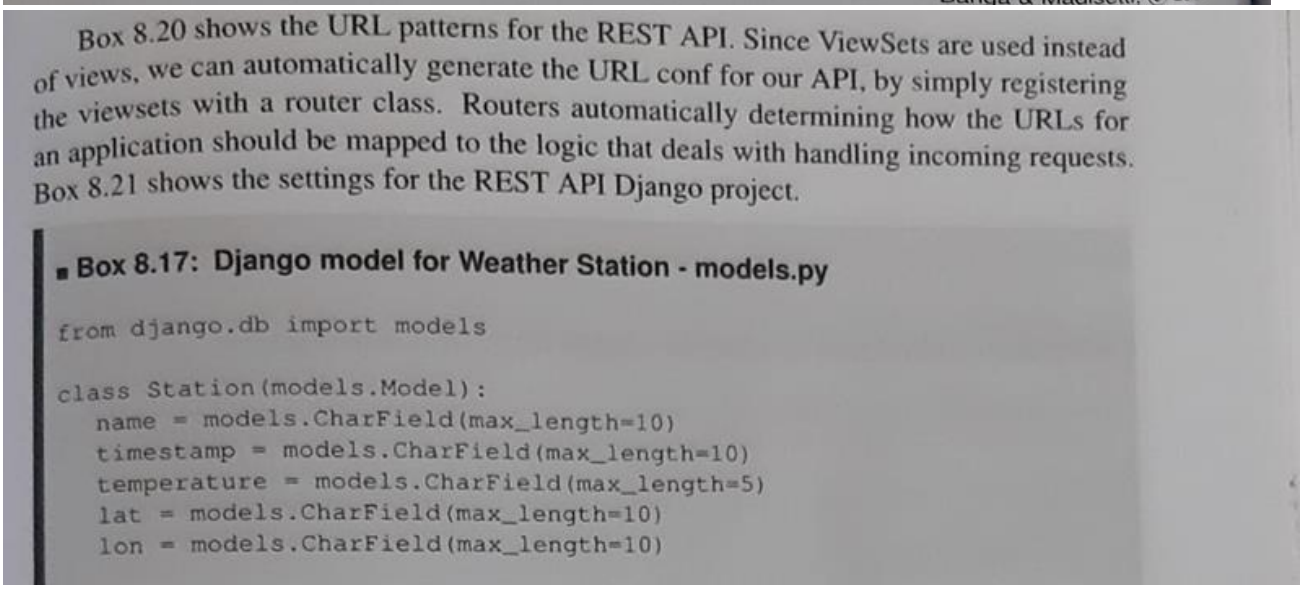
```
pip install djangorestframework
pip install markdown
pip install django-filter
```

After installing the Django REST framework, let us create a new Django project named *restfulapi*, and then start a new app called *myapp*, as follows:

```
django-admin.py startproject restfulapi
cd restfulapi
python manage.py startapp myapp
```

The REST API described in this section allows you to create, view, update and delete a collection of resources where each resource represents a sensor data reading from a weather monitoring station. Box 8.17 shows the Django model for such a station. The station model contains four fields - station name, timestamp, temperature, latitude and longitude. Box 8.18 shows the Django views for the REST API. ViewSets are used for the views that allow you to combine the logic for a set of related views in a single class.

Box 8.19 shows the serializers for the REST API. Serializers allow complex data (such as querysets and model instances) to be converted to native Python datatypes that can then be easily rendered into JSON, XML or other content types. Serializers also provide de-serialization, allowing parsed data to be converted back into complex types, after first validating the incoming data.



Box 8.20 shows the URL patterns for the REST API. Since ViewSets are used instead of views, we can automatically generate the URL conf for our API, by simply registering the viewsets with a router class. Routers automatically determining how the URLs for an application should be mapped to the logic that deals with handling incoming requests. Box 8.21 shows the settings for the REST API Django project.

Box 8.17: Django model for Weather Station - models.py

```
from django.db import models

class Station(models.Model):
    name = models.CharField(max_length=10)
    timestamp = models.CharField(max_length=10)
    temperature = models.CharField(max_length=5)
    lat = models.CharField(max_length=10)
    lon = models.CharField(max_length=10)
```

How the Authorization and Authentication differs in Cloud model and write down the necessity of Address Resolution Protocol (ARP) works in the system

Authentication vs. Authorization

- Despite the similar-sounding terms, authentication and authorization are separate steps in the login process. Understanding the difference between the two is key to successfully implementing an IAM solution.

  Let's use an analogy to outline the differences.

- Consider a person walking up to a locked door to provide care to a pet while the family is away on vacation. That person needs:

  - **Authentication**, in the form of a key. The lock on the door only grants access to someone with the correct key in much the same way that a system only grants access to users who have the correct credentials.
  - **Authorization,** in the form of permissions. Once inside, the person has the authorization to access the kitchen and open the cupboard that holds the pet food. The person may not have permission to go into the bedroom for a quick nap.

Authentication and authorization work together in this example. A pet sitter has the right to enter the house (authentication), and once there, they have access to certain areas (authorization).

	Authentication	**Authorization**
What does it do?	Verifies credentials	Grants or denies permissions
How does it work?	Through passwords, biometrics, one-time pins, or apps	Through settings maintained by security teams
Is it visible to the user?	Yes	No
It is changeable by the user?	Partially	No
How does data move?	Through ID tokens	Through access tokens

Address Resolution Protocol (ARP)

- Address Resolution Protocol (ARP) is a procedure for mapping a dynamic IP address to a permanent physical machine address in a local area network (LAN). The physical machine address is also known as a media access control (MAC) address.
- The job of ARP is essentially to translate 32-bit addresses to 48-bit addresses and vice versa. This is necessary because IP addresses in IP version 4 (IPv4) are 32 bits, but MAC addresses are 48 bits.
- ARP works between Layers 2 and 3 of the Open Systems Interconnection model (OSI model). The MAC address exists on Layer 2 of the OSI model, the data link layer. The IP address exists on Layer 3, the network layer.
- ARP can also be used for IP over other LAN technologies, such as token ring, fiber distributed data interface (FDDI) and IP over ATM.

8.7 SkyNet IoT Messaging Platform

SkyNet is an open source instant messaging platform for Internet of Things. The SkyNet API supports both HTTP REST and real-time WebSockets. SkyNet allows you to register devices (or nodes) on the network. A device can be anything including sensors, smart home devices, cloud resources, drones, etc. Each device is assigned a UUID and a secret token. Devices or client applications can subscribe to other devices and receive/send messages.

Box 8.39 shows the commands to setup SkyNet on a Linux machine. Box 8.40 shows a sample configuration for SkyNet. Box 8.41 shows examples of using SkyNet. The first step is to create a device on SkyNet. The POST request to create a device returns the UUID and token of the created device. The box also shows examples of updating a device, retrieving last 10 events related to a device, subscribing to a device and sending a message to a device. Box 8.42 shows the code for a Python client that performs various functions such as subscribing to a device, sending a message and retrieving the service status.

■ Box 8.39: Commands for Setting up SkyNet

```
#Install DB Redis:
sudo apt-get install redis-server

#Install MongoDB:
```

```
sudo apt-key adv -keyserver keyserver.ubuntu.com -recv 7F0CEB10
echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart
dist 10gen' |
sudo tee /etc/apt/sources.list.d/10gen.list
sudo apt-get update
sudo apt-get install mongodb-10gen

#Install dependencies
sudo apt-get install git
sudo apt-get install software-properties-common
sudo apt-get install npm

#Install Node.JS
sudo apt-get update
sudo apt-get install -y python-software-properties python g++ make
sudo add-apt-repository ppa:chris-lea/node.js
sudo apt-get update
sudo apt-get install nodejs

#Install Skynet:
git clone https://github.com/skynetim/skynet.git
npm config set registry http://registry.npmjs.org/
npm install
```

Box 8.40: Sample SkyNet configuration file

```
module.exports = {
  databaseUrl:  "mongodb://localhost:27017/skynet",
  port:  3000,
  log:  true,
  rateLimit:  10, // 10 transactions per user per secend
  redisHost:  "127.0.0.1",
  redisPort:  "6379"
};
```