RegNo: 21RP09910

Class: IT Level 7 Y3 A

Module name: IoT

Module code: ITLIT701

**Assignment**

**Qn1. Describe IoT device deployment on IoT application domains using IoT levels.**

**IoT Levels:**

IoT levels refer to a framework that categorizes IoT applications based on the complexity and scope of the deployment. They range from Level 0 to Level 5, each representing a different stage of IoT maturity and sophistication.

**Level 0: Device-centric:** This level represents the basic stage of IoT, where devices are not connected and operate independently.

- *Deployment:* Simple, standalone devices with no connectivity to the internet or other devices.

- *Application Domains*: Basic consumer electronics like standalone sensors or simple home appliances.

**Level 1: Device-to-Device:** This level, devices are able to communicate with each other directly, but they might not connect to a central system.

- *Deploymen*t: Devices can share data with each other within a limited range.

- *Application Domains*: Localized systems such as smart home devices interacting with each other (e.g., smart lights communicating with a smart thermostat).

**Level 2: Device-to-Cloud:** This level,devices are connected to the cloud, allowing centralized data storage and processing.

- *Deployment:* Devices send data to a cloud server for storage, analysis, and access from anywhere.

- *Application Domains:* Industrial applications like remote monitoring of equipment, smart agriculture for crop management, or fleet management for logistics.

**Level 3: Device-to-Data Center:** In addition to the cloud, data centers are used for more extensive data processing and analytics.

- *Deployment:* Data is processed in both the cloud and local data centers for faster and more efficient analytics.

- *Application Domains:* Healthcare applications like patient monitoring systems that require real-time data processing, or smart cities for traffic management and public safety.

**Level 4: Device-to-Edge:** Edge computing is introduced to process data closer to the source, reducing latency and improving real-time decision-making.

- *Deployment:* Devices send data to both the cloud and local edge computing nodes for immediate processing.

- *Application Domains:* Autonomous vehicles, where split-second decisions are critical, or industrial automation for predictive maintenance and quality control.

**Level 5: Autonomous Systems:** Fully autonomous systems where devices make decisions independently with minimal human intervention.

- *Deployment*: Devices are interconnected and self-organize, leveraging AI and machine learning for complex decision-making.

- *Application Domains*: Advanced robotics, AI-driven smart grids for energy management, or fully automated smart buildings.

**IoT Device Deployment Examples:**

- *Smart Home (Level 1):* Smart thermostats communicating with smart plugs to manage energy consumption.

- *Fleet Management (Level 2):* GPS trackers in vehicles sending data to the cloud for route optimization and maintenance scheduling.

- *Healthcare Monitoring (Level 3):* Wearable health devices transmitting data to both the cloud for historical analysis and local data centers for real-time alerts.

- *Industrial IoT (Level 4):* Manufacturing machinery equipped with sensors sending data to edge devices for predictive maintenance analysis.

- *Autonomous Vehicles (Level 5):* Self-driving cars communicating with smart infrastructure and other vehicles for coordinated traffic flow.

**Qn2. Describe properly basic structure of Arduino sketch as per initialization rules.**

An Arduino sketch, which is essentially a program written for an Arduino board, typically follows a basic structure. Here's a breakdown of the essential parts:

Basic Structure of an Arduino Sketch:

**1. Comments:**

- Comments are lines in the code that are not executed, but provide information about the code. They start with `//` for single-line comments or `/\* \*/` for multi-line comments.

*// This is a single-line comment*

/\*

*This is a*

*multi-line comment*

\*/

2. Include Statements:

- These statements bring in external libraries or define constants.

#include <Servo.h> // **Example library**

#define LED\_PIN 13 // **Example constant**

3. Global Variables:

- Variables that are declared outside of any function, accessible throughout the sketch.

int sensorPin = A0;

int ledPin = 13;

4. Setup Function:

- The `setup()` function runs once when the Arduino board is powered on or reset. It is used for initializing variables, pin modes, libraries, etc.

void setup() {

pinMode(ledPin, OUTPUT);

}

5. Loop Function:

- The `loop()` function runs continuously after the `setup()` function. This is where the main logic of your program goes.

void loop() {

int sensorValue = analogRead(sensorPin);

if (sensorValue > 500) {

digitalWrite(ledPin, HIGH);

} else {

digitalWrite(ledPin, LOW);

}

delay(1000); // Wait for a second

}

6. Functions(Optional):

- Additional functions can be defined to modularize the code.

void myFunction() {

// Function code here

}

Example Sketch:

Putting it all together, here's an example sketch that blinks an LED based on a sensor reading:

// Include Libraries

#include <Arduino.h>

// Define Constants

#define LED\_PIN 13

#define SENSOR\_PIN A0

// Global Variables

int sensorValue = 0;

// Setup Function

void setup() {

pinMode(LED\_PIN, OUTPUT);

}

// Loop Function

void loop() {

sensorValue = analogRead(SENSOR\_PIN);

if (sensorValue > 500) {

digitalWrite(LED\_PIN, HIGH);

} else {

digitalWrite(LED\_PIN, LOW);

}

delay(1000); // Wait for a second

}