Om Shree
2006077
IT-5

Case Studies Illustrating IOT Design

Designs covered :-

1) Smart Lighting system
2) Home intrusion detection system
3) Smart parking system
4) Weather monitoring system
5) Weather reporting system
6) Air pollution monitoring system
7) Forest fire detection system
8) Smart irrigation system
9) IOT printer system

(1) Smart Lighting System

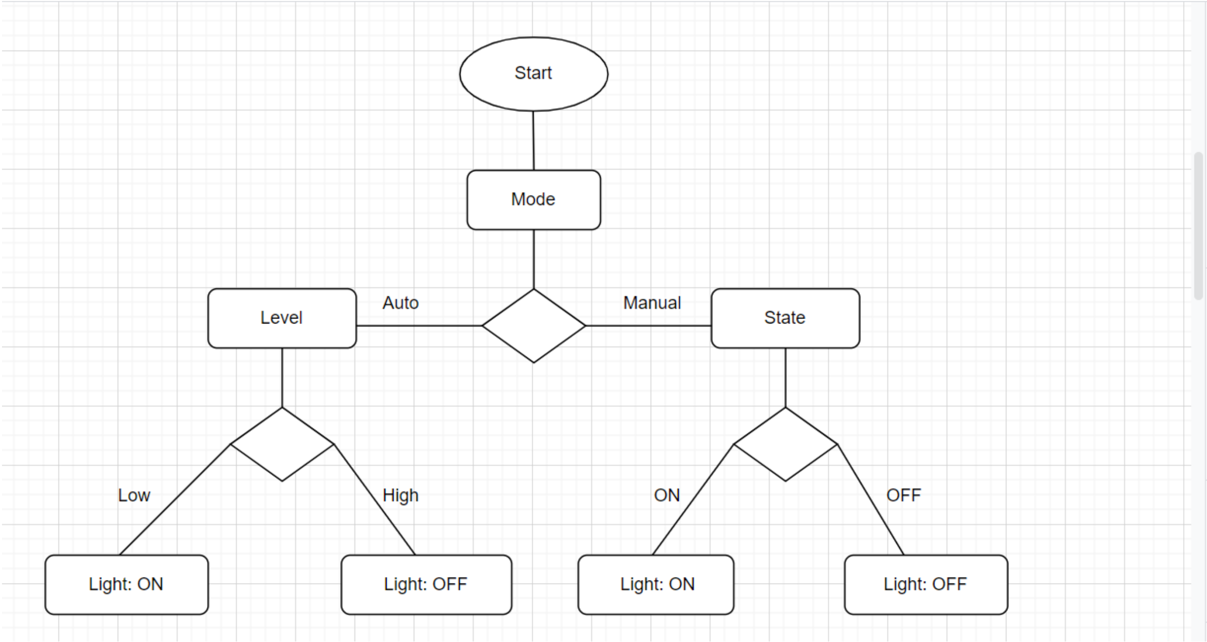
Step 1: Purpose and requirement specification

Purpose: Allow remote control of lighting in a room.

Requirements:-

Behaviour: 2 modes (auto and manual) implemented exclusively.
System management: remote monitoring and control
Data analytics: local data analytics
Security: user authentication
Deployment: local deployment on device with remote access

Step 2: Process specification



Step 3: Domain model specification

Physical entities: Room, light appliances

Virtual entities: Room, light appliances

Step 4: Information model specification

Records the various attributes of all entities involved in the IOT system.

Room: Light state(ON/OFF), light level(high/low)

Appliance: appliance state (ON/OFF)

Manual mode uses light state whereas the auto mode uses the light level.

Step 5: Service specification
Service to change the state of light appliances
Service to record the light level in a room

Step 6: IOT level specification
Home lighting automation system falls in IOT level 1

Step 7: Functional view specification
Device: to control and monitor the state of light appliances in a room
Communication: includes network layer protocols
Services: controller service, mode service, level service
Security: user authentication
Application: to be deployed on a device natively
Management: Django, MySQL

Step 8: Operational view specification
Devices: Raspberry pi controller, light sensor
API service: REST based APIs
Communication: Link layer, IPv4/v6, TCP, HTTP
Service: controller-native, mode and state service via REST based APIs
Application: Django application, MySQL database

Step 9: Device and component integration
Raspberry pi controller, LDR sensor and relay switch actuator

Step 10: Application deployment
Modes are exclusive in nature i.e. both modes can’t be switched on simultaneously. In auto mode the system is sensor driven whereas in manual mode the system is user input driven.

(2) Home Intrusion Detection System

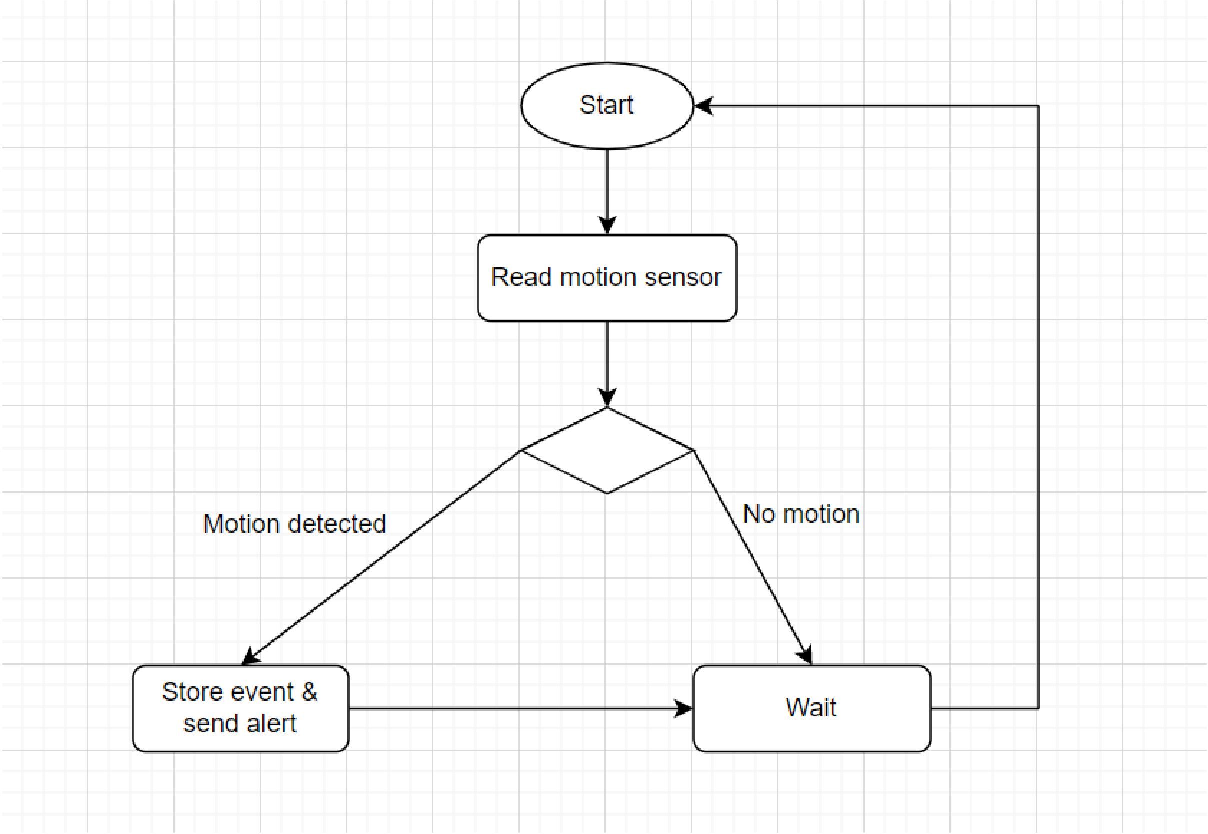
Step 1: Purpose and requirement specification

Purpose: Detect intrusions using sensors and generate alerts if necessary.

Requirements:-

Behaviour: 1 mode i.e. auto
System management: remote monitoring
Data analytics: cloud oriented
Security: user authentication
Deployment: cloud deployment with remote access

Step 2: Process specification



Step 3: Domain model specification

Physical entities: Room, door

Virtual entities: Room, door

Step 4: Information model specification

Records the various attributes of all entities involved in the IOT system.

Room: motion
Door: state

Step 5: Service specification
Service to retrieve the current motion in the room
Service to record the state of the door
Native service on the device to read the sensor data

Step 6: IOT level specification
Home intrusion detection system falls in IOT level 2

Step 7: Functional view specification
Device: to monitor the state of door as well as motion in a room
Communication: includes network layer protocols
Services: controller service, motion service, state service
Security: user authentication
Application: to be deployed on the cloud
Management: Django, MySQL

Step 8: Operational view specification
Devices: Raspberry pi controller, motion sensor
API service: REST based APIs
Communication: Link layer, IPv4/v6, TCP, HTTP
Service: controller-native, motion and state service via REST based APIs
Application: Django application, MySQL database

Step 9: Device and component integration
Raspberry pi controller, motion sensor and relay switch actuator

Step 10: Application deployment
In auto mode the system is sensor driven and generates alerts only when necessary.

(3) Smart Parking System

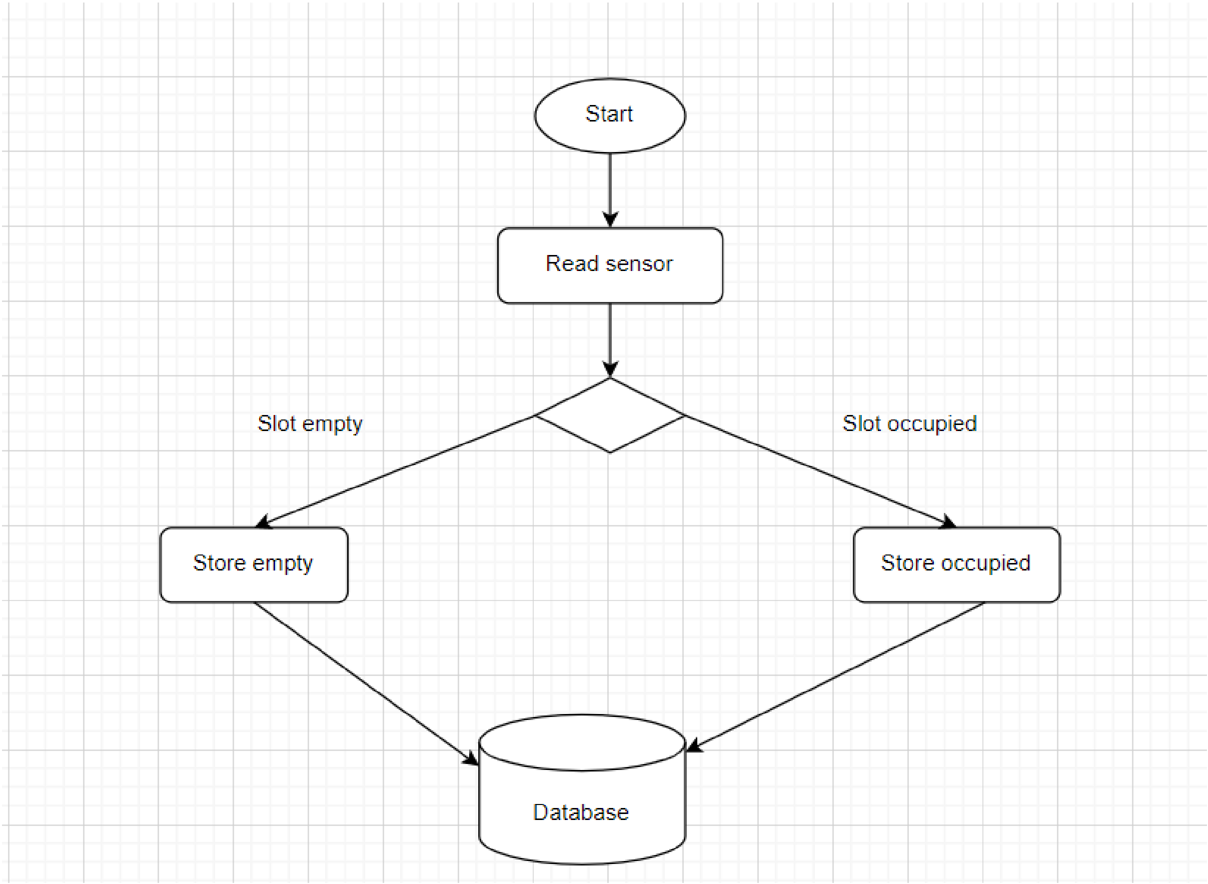
Step 1: Purpose and requirement specification

Purpose: Detection of empty parking slots and sending information over the internet to smart parking application back-end.

Requirements:-

Behaviour: 1 mode i.e. auto
System management: remote monitoring
Data analytics: cloud oriented
Security: user authentication, DDOS attack proofing
Deployment: cloud deployment with remote access

Step 2: Process specification



Step 3: Domain model specification

Physical entities: Parking slot

Virtual entities: Parking slot

Step 4: Information model specification

Records the various attributes of all entities involved in the IOT system.
Parking slot: state(empty or occupied)

Step 5: Service specification
Service to retrieve the current state of the parking slot
Service to record the state of the parking slot and store the data in the database
Controller service native to the device

Step 6: IOT level specification
Smart parking systems can fall in IOT level 4 or level 5 depending on the size of the city.

Step 7: Functional view specification
Device: to monitor the state of the parking slot.
Communication: includes network layer protocols
Services: controller service, retrieval service, state service
Security: DDOS proofing
Application: to be deployed on the cloud
Management: Django, MySQL

Step 8: Operational view specification
Devices: Raspberry pi controller, IR sensor or ultrasonic sensor
API service: REST based APIs
Communication: Link layer, IPv4/v6, TCP, HTTP
Service: controller-native, retrieval and state service via REST based APIs
Application: Django application, MySQL database

Step 9: Device and component integration
Raspberry pi controller, IR sensor and MySQL database

Step 10: Application deployment
The system is sensor driven and stores data in the cloud. This data can be accessed by end users of the city using a cloud based application. Cloud based applications can run on any device independent of its specifications.

(4) Weather monitoring system

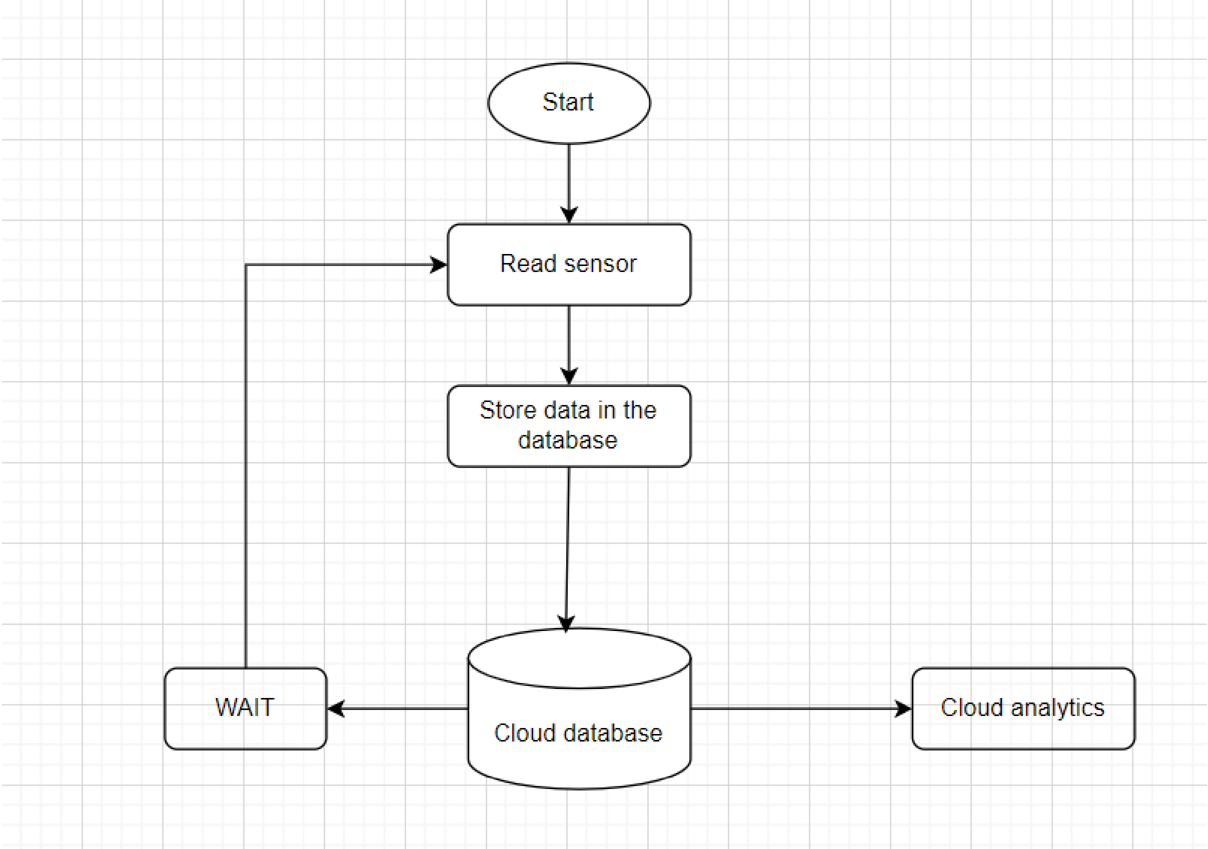
Step 1: Purpose and requirement specification

Purpose: Collect data on environmental conditions such as temperature, humidity, pressure and light in an area using multiple end nodes. The end nodes send this data to the cloud where it is aggregated and analysed.

Requirements:-

Behaviour: 1 mode i.e. auto
System management: remote monitoring
Data analytics: cloud oriented
Deployment: cloud deployment

Step 2: Process specification



Step 3: Domain model specification

Physical entities: Environment
Virtual entities: Environment

Step 4: Information model specification
Records the various attributes of all entities involved in the IOT system.
Environment: temperature, humidity and pressure

Step 5: Service specification
Service to read the sensors
Service to record and analyse the data in cloud databases
Controller service native to the device

Step 6: IOT level specification
Environment monitoring systems fall in IOT level 6

Step 7: Functional view specification
Device: multiple end nodes to monitor the environment conditions.
Communication: includes network layer protocols
Services: controller service, record service, store service and analytics service
Application: to be deployed on the cloud
Management: Django, Xively

Step 8: Operational view specification
Devices: Raspberry pi mini computer, temperature, humidity, pressure, LDR sensor and A/D convertor
API service: Xively services
Communication: Link layer, IPv4/v6, TCP, HTTP
Service: controller-native, recording and storing service via Xively APIs
Application: Django application, Xively database

Step 9: Device and component integration
Raspberry pi controller, LDR sensor, humidity sensor, pressure sensor, temperature sensor, A/D convertor and Xively database

Step 10: Application deployment
The system is sensor driven and stores as well as analyzes data in the cloud.

(5) **Weather Reporting Bot**

Step 1: Purpose and requirement specification

Purpose: Tweet weather updates

Requirements:-

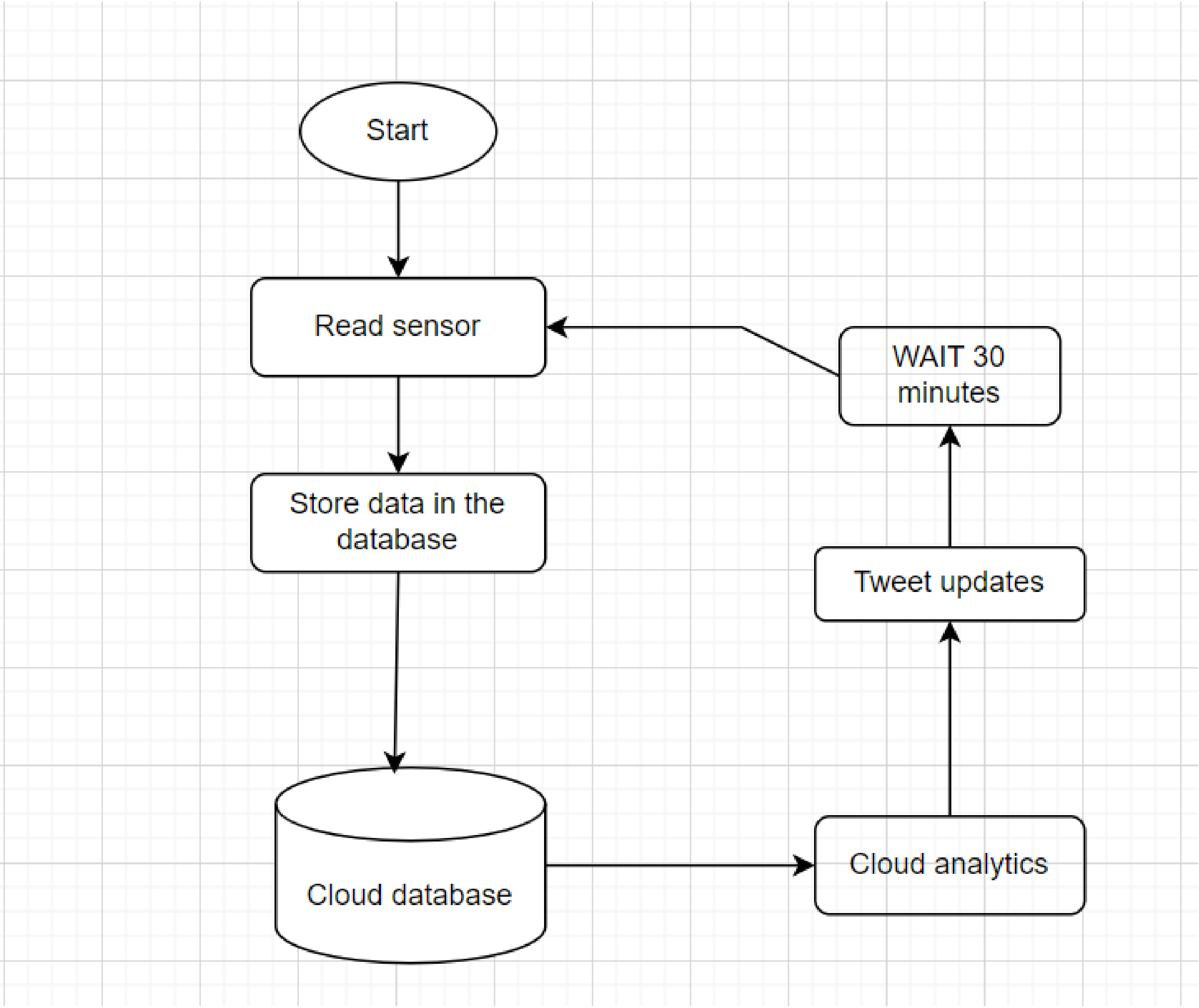
Behaviour: 1 mode i.e. auto

System management: automated tweets

Data analytics: cloud oriented

Deployment: cloud deployment

Step 2: Process specification



Step 3: Domain model specification

Physical entities: Environment

Virtual entities: Environment

Step 4: Information model specification

Records the various attributes of all entities involved in the IOT system.
Environment: temperature, humidity and pressure

Step 5: Service specification

Service to read the sensors
Service to record and analyze the data in cloud databases
Tweepy API service to tweet weather updates
Controller service native to the device

Step 6: IOT level specification

Weather reporting bot falls in IOT level 6

Step 7: Functional view specification

Device: multiple end nodes to monitor the environment conditions.
Communication: includes network layer protocols
Services: controller service, record service, store service, analytics service and tweet service using Tweepy
Application: to be deployed on the cloud
Management: Django, Xively and Tweepy

Step 8: Operational view specification

Devices: Raspberry pi mini computer, temperature, humidity, pressure, LDR sensor and A/D convertor
API service: Xively services, Tweepy API
Communication: Link layer, IPv4/v6, TCP, HTTP
Service: controller-native, recording and storing service via Xively APIs
Application: Django application, Xively database

Step 9: Device and component integration

Raspberry pi controller, LDR sensor, humidity sensor, pressure sensor, temperature sensor, A/D convertor, Tweepy API and Xively database

Step 10: Application deployment

The system is sensor driven and stores as well as analyzes data in the cloud. It generates tweets every 30 minutes.

(6) Air Pollution Monitoring

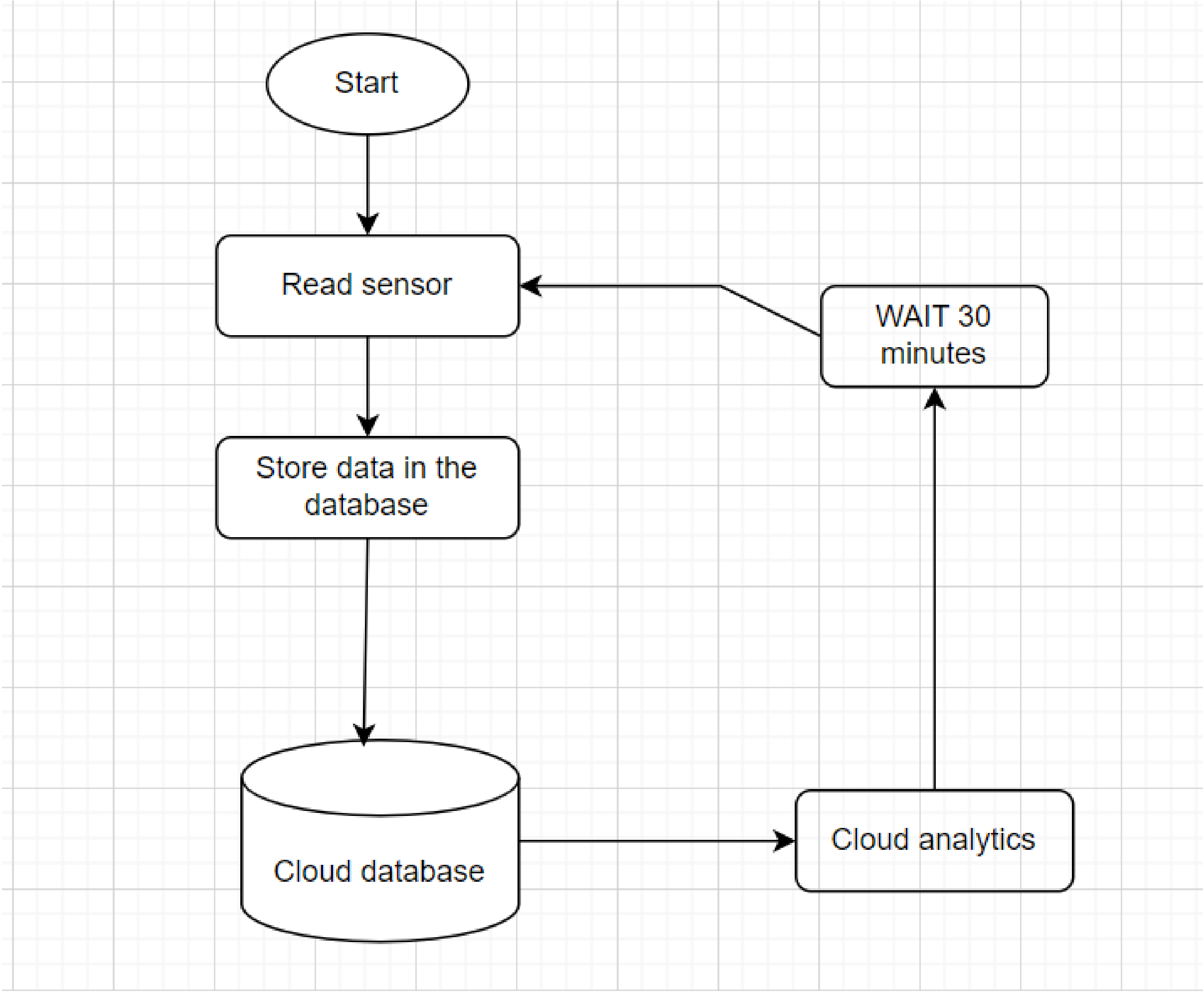
Step 1: Purpose and requirement specification

Purpose: Monitor the emission of harmful gases being emitted from factories using gaseous and meteorological sensors.

Requirements:-

Behavior: 1 mode i.e. auto
System management: automated tracking
Data analytics: cloud oriented
Deployment: cloud deployment

Step 2: Process specification



Step 3: Domain model specification

Physical entities: Factory
Virtual entities: Factory

Step 4: Information model specification
Records the various attributes of all entities involved in the IOT system.
Factory: Carbon monoxide sensors, Nitrous oxide sensors

Step 5: Service specification
Service to read the sensors
Service to record and analyze the data in cloud databases
Controller service native to the device

Step 6: IOT level specification
Air Pollution monitoring system falls in IOT level 6

Step 7: Functional view specification
Device: multiple end nodes to monitor the air pollution level.
Communication: includes network layer protocols
Services: controller service, record service, store service, analytics service and visualization service
Application: to be deployed on the cloud
Management: Django, Xively and AWS Lambda

Step 8: Operational view specification
Devices: Raspberry pi mini computer, carbon monoxide sensor, nitrous oxide sensor and A/D convertor
API service: Xively services, Lambda services
Communication: Link layer, IPv4/v6, TCP, HTTP
Service: controller-native, recording and storing service via Xively APIs
Application: Django application, Xively database

Step 9: Device and component integration
Raspberry pi controller, carbon monoxide sensor, nitrous oxide sensor, temperature sensor, A/D convertor and Xively database

Step 10: Application deployment
The system stores, analyzes and visualizes data in the cloud.

(7) Forest Fire Detection

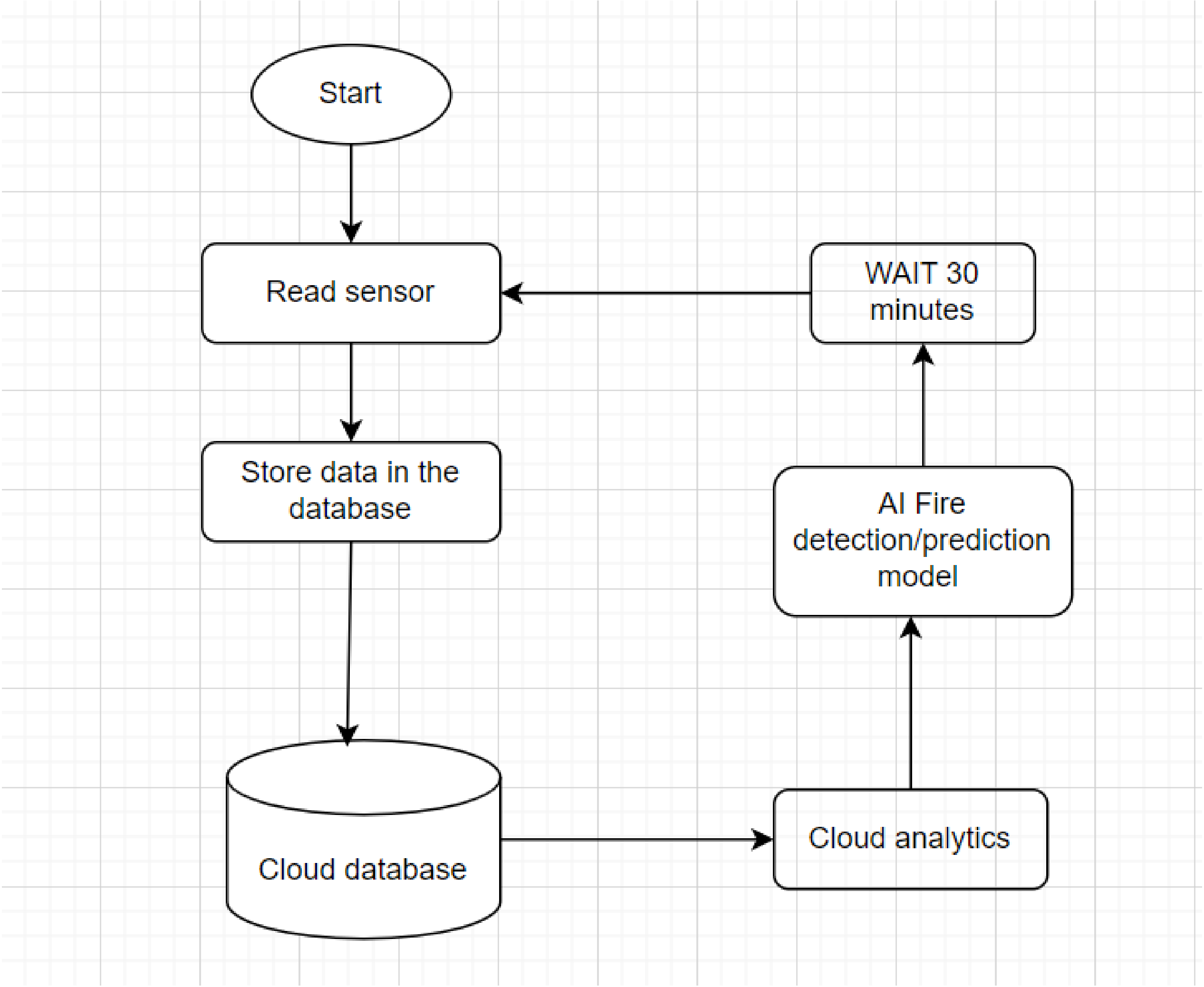
Step 1: Purpose and requirement specification

Purpose: Multiple end nodes to monitor the ambient conditions. Data is sent to the cloud where AI models predict whether a fire has broken out or not.

Requirements:-

Behavior: 1 mode i.e. auto
System management: automated tracking and analysis
Data analytics: cloud oriented
Deployment: cloud deployment

Step 2: Process specification


Start
Read sensor
WAIT 30 minutes
Store data in the database
AI Fire detection/prediction model
Cloud database
Cloud analytics

Step 3: Domain model specification

Physical entities: Forest
Virtual entities: Forest

Step 4: Information model specification

Records the various attributes of all entities involved in the IOT system.
Forest: temperature and humidity

Step 5: Service specification

Service to read the sensors
Service to record and analyze the data in cloud databases
AI prediction service
Controller service native to the device

Step 6: IOT level specification

Forest fire detection system falls in IOT level 5

Step 7: Functional view specification

Device: multiple end nodes to monitor the air pollution level. Co-ordinator node to aggregate data of end nodes and send it to the cloud.
Communication: includes network layer protocols
Services: controller service, co-ordinator service, record service, store service, analytics service, AI model service.
Application: to be deployed on the cloud
Management: Django, Xively and AWS Lambda

Step 8: Operational view specification

Devices: Raspberry pi mini computer, carbon monoxide sensor, nitrous oxide sensor and A/D convertor
API service: Xively services, Lambda services, AI model services
Communication: Link layer, IPv4/v6, TCP, HTTP
Service: controller-native, co-ordinator(remote), recording and storing service via Xively APIs
Application: Django application, Xively database

Step 9: Device and component integration

Raspberry pi controller, humidity sensor, temperature sensor, A/D convertor and Xively database

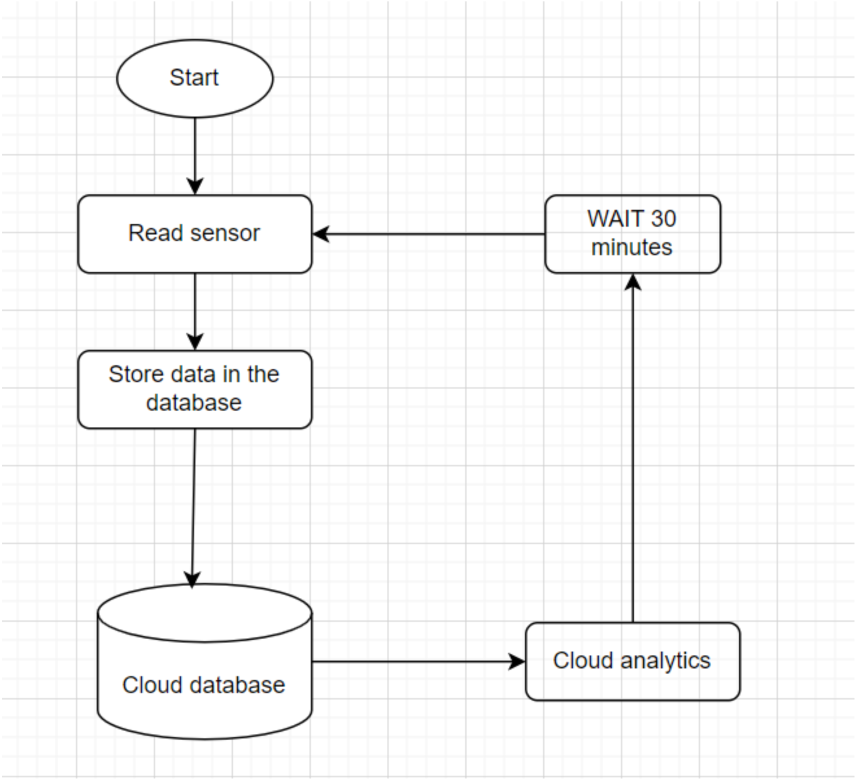
Step 10: Application deployment
The system stores and analyzes data in the cloud. The data is then fed into an AI model to predict whether fire has broken out or not. The process is repeated every 30 minutes.

(8) Smart Irrigation System

Step 1: Purpose and requirement specification
Purpose: Multiple end nodes to monitor soil moisture in a field and release water flow as necessary.
Requirements:-

Behavior: 1 mode i.e. auto
System management: automated tracking and analysis
Data analytics: cloud oriented
Deployment: cloud deployment

Step 2: Process specification



Step 3: Domain model specification
Physical entities: Soil
Virtual entities: Soil

Step 4: Information model specification
Records the various attributes of all entities involved in the IOT system.
Soil: moisture

Step 5: Service specification
Service to read the sensors
Service to record and analyze the data in cloud databases
Controller service native to the device

Step 6: IOT level specification
Smart irrigation system falls in IOT level 5

Step 7: Functional view specification
Device: multiple end nodes to monitor the air pollution level. Co-ordinator node to aggregate data of end nodes and send it to the cloud.
Communication: includes network layer protocols
Services: controller service, co-ordinator service, record service, store service and analytics service
Application: to be deployed on the cloud
Management: Django, Xively and AWS Lambda

Step 8: Operational view specification
Devices: Raspberry pi mini computer, moisture sensor and A/D convertor
API service: Xively services and Lambda services
Communication: Link layer, IPv4/v6, TCP, HTTP
Service: controller-native, co-ordinator(remote), recording and storing service via Xively APIs
Application: Django application, Xively database

Step 9: Device and component integration
Raspberry pi controller, moisture sensor, A/D convertor and Xively database.

Step 10: Application deployment

The system stores and analyzes data in the cloud. The application is automated and works without any human intervention.

(9) <u>IOT Printer</u>

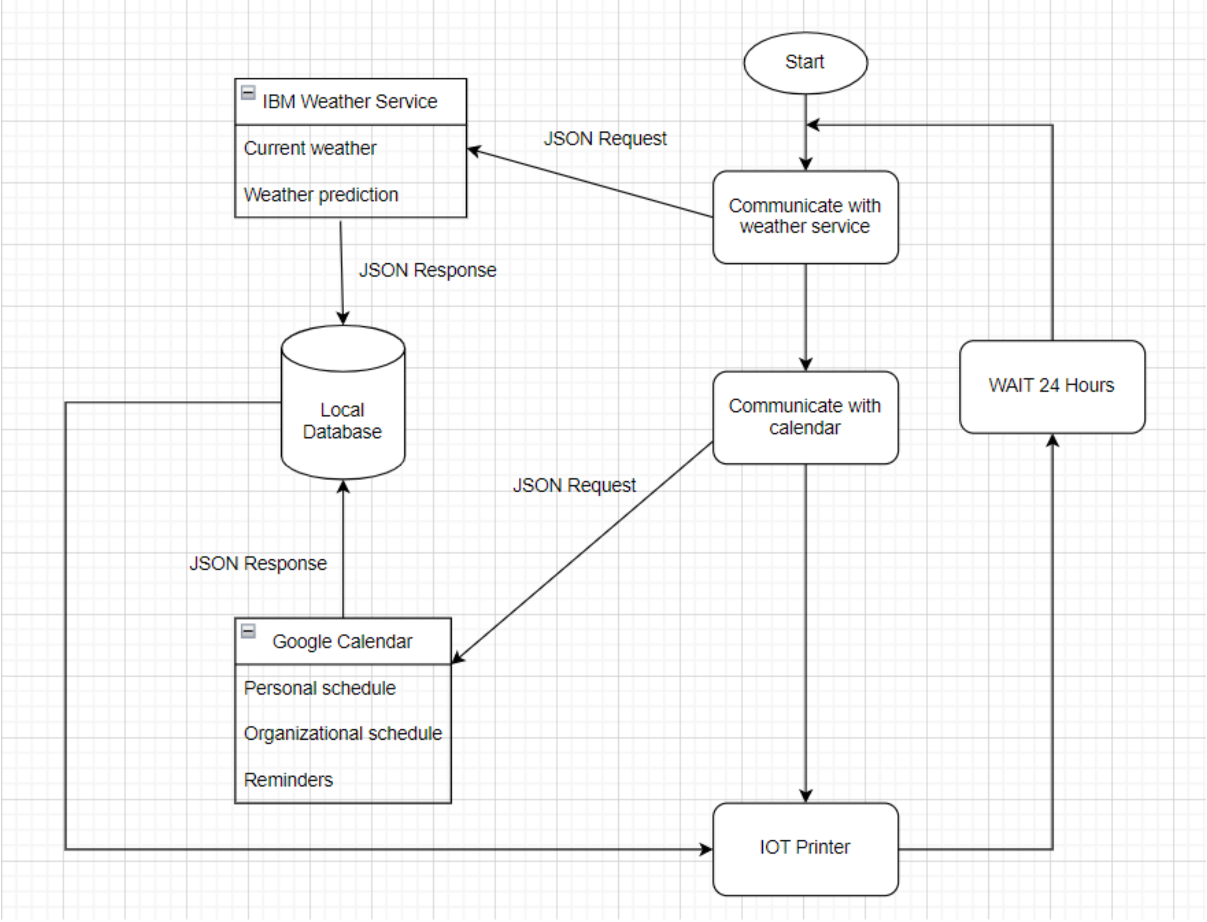
Step 1: Purpose and requirement specification

Purpose: Print daily briefings consisting of weather information and prediction for the day as well as the user's schedule for the day.

Requirements:-

Behavior: 1 mode i.e. auto
System management: automated tracking and printing
Data analytics: cloud oriented
Deployment: cloud deployment

Step 2: Process specification



Step 3: Domain model specification
Physical entities: Weather, schedule
Virtual entities: weather, schedule

Step 4: Information model specification
Records the various attributes of all entities involved in the IOT system.
Weather: temperature, humidity, overcast and forecast
Schedule: reminders, personal schedule and organizational schedule

Step 5: Service specification
Service to record and store the data in a local database.
Service to interact with IBM weather and Google calendar APIs.
Controller service native to the device(printer).

Step 6: IOT level specification
The IOT printer(the one I've designed) falls in IOT level 1 as it uses a local database and prints the data locally.

Step 7: Functional view specification
Device: Single printer to print the daily briefing.
Communication: includes network layer protocols
Services: controller service, store service and Google as well as IBM API services.
Application: To be deployed natively on the printer
Management: Django, LocalDB

Step 8: Operational view specification
Devices: Raspberry pi mini computer, Smart printer
API service: Google API and IBM weather API
Communication: Link layer, IPv4/v6, TCP, HTTP
Service: controller-native, LocalDB services
Application: Django application, LocalDB database

Step 9: Device and component integration
Raspberry pi controller, LocalDB database, JSON models(request and response) and Smart printer.

Step 10: Application deployment

The system is automated and works without any human intervention.
The cycle repeats every 24 hours(every day).