IoT APPLICATIONS FOR SMART CITIES

Smart City

Smart city applications and services connect people, process, data and things. A smart city can be defined as a vision which integrates multiple ICT and IoT solutions in a secure fashion to manage a city's assets such as information systems, schools, libraries, transportation systems, hospitals, power plants, water supply networks, waste management, law enforcement and other community services. Sectors that have been developing smart city technology include government services, transport and traffic management, energy, health care, water, innovative urban agriculture and waste management.

Smart-city solutions integrate a number of city services and can include the following:

- Smart parking spaces
- Smart street lightings and smart lighting solutions, such as SimplySNAP smart lighting solution which enables the development, control, and optimisation of a smart lighting implementation, developed by Synapse Wireless partnering with ThingWorx
- Smart traffic solutions, such as smart energy management, smart parking, smart waste bins, smart street lighting, and security and surveillance, developed by Tech Mahindra partnering with ThingWorx
- Smart water management, such as AquamatiX for monitoring and optimizing a city's water and sewage services
- Smart connected bike share services
- Smart health services
- Smart structures (building, bridges and historical monuments) health, vibrations and material conditions monitoring, analysing and managing structures health data to improve energy usage, maintenance, operations, and comfort solution. For example using solutions developed by WiseUp partnering with ThingWorx platform
- Smart city system integrator, such as Pactera from ThingWorx

Architectural View

It shows four-layer architectural framework developed at CISCO cloud IoTfor a city. Layers consists of (i) devices network and distributed nodes, (ii) distribute data capture, processing and analysing, (iii) data centres and cloud and (iv) applications, such as waste containers monitoring. Figure 12.9 shows data-flow diagram and domain architecture reference model for the smart city applications and services.19

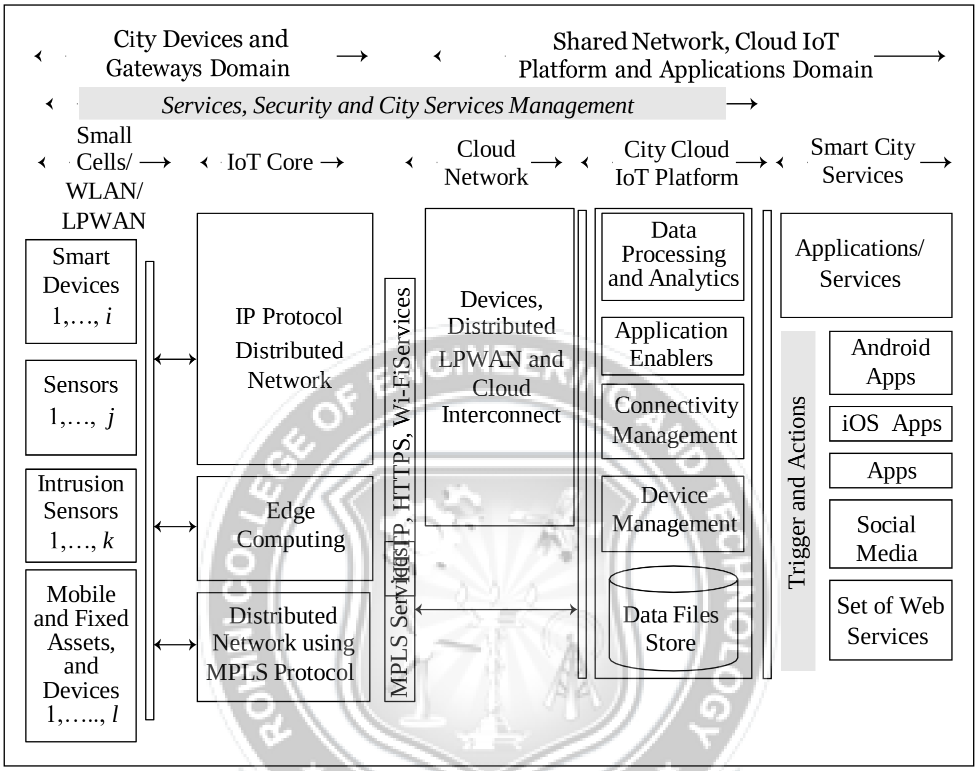
City Devices and Gateways Domain
Shared Network, Cloud IoT Platform and Applications Domain
Services, Security and City Services Management
Small Cells/ WLAN/ LPWAN
IoT Core
Cloud Network
City Cloud IoT Platform
Smart City Services
Smart Devices 1,…, i
Sensors 1,…, j
Intrusion Sensors 1,…, k
Mobile and Fixed Assets, and Devices 1,….., l
IP Protocol Distributed Network
Edge Computing
Distributed Network using MPLS Protocol
MPLS Services
HTTP, HTTPS, Wi-FiServices
Devices, Distributed LPWAN and Cloud Interconnect
Data Processing and Analytics
Application Enablers
Connectivity Management
Device Management
Data Files Store
Trigger and Actions
Applications/ Services
Android Apps
iOS Apps
Apps
Social Media
Set of Web Services

Figure Data flow diagram and domain architecture reference model for the smart City Applications and Services

Two domains are (i) City devices and Gateways domain, and (ii) shared network, cloud IoT platform and applications domains. Services, security and city services management are cross-domain functions. Assume the edge sensors and devices consist of, say *i*-smart devices, *j*- sensors, *k*-intrusion sensors and *l*- mobile and fixed assets and devices whereas *i*, *j*, *k* and *l* can be very large numbers.

The edge sensors and devices wirelessly connect within small cells, systems connect with WLAN (Wireless LAN). They communicate using LPWAN. The distributed network of edge-computing systems connects using IP protocols or using the Multiprotocol Label Switching (MPLS). The MPLS assigns the labels to data packets and forward the labels to city cloud IoT platform.

City cloud IoT platform collects messages, triggers, alerts and data files at data store. The platform does device and connectivity management functions, application enabler functions, and data processing and analytics. The platform generates triggers which follows actions, such as connect to social media, set of web services, applications, iOS apps and Android apps. The platform connects to a number of city applications and services.

Smart city applications and services can deploy CISCO IoT, IOx and Fog. This is because of usages of shared networks and distributed access point nodes, and the need of an ecosystem with ability to transform sensor data and perform the control functions within the distributed network nodes. This enables development of applications, such as site asset management, energy monitoring, and smart parking infrastructure and connected cities.

Alternatively, smart city solutions can deploy ThingWorx IoT platform, which is an intelligent management platform for all connected things (IMPACT). IMPACT platform provides devices and connectivity management, application enablement, data and analytics with secure end-to-end access.

Sensors, Devices, Hardware Edge Systems Deployment

It described the sensors which can be deployed for a number of city applications It described embedded platforms Arduino, Edison, RPi, BeagleBone and mBed which can be used for prototype development of edge systems.

An open-source prototyping platform, openHAB and hardware RPi or Arduino can be deployed at the edge systems. Bosch presented at CES 2017 a prototyping platform for IoT.[20] The platform connects the devices and provides a number of solutions. Edge hardware is Bosch XDK 110 development kit, which includes devices [eight number microelectromechanical system, accelerometer, magnetometer and gyroscope, coupled with relative-humidity (RH), pressure (P), temperature (T), acoustic and digital light-sensors], functional extendibility with a 26-pin extension port, Bluetooth and Wi-Fi connectivity.[21] XDK workbench includes its IDE.

Code Development Environment, Development, Debugging and Deployment

It described end-to-end IoT Solutions with Java and Eclipse IoT Stack. openHAB software platform for devices, sensor and edge systems at a smart home. Java, Eclipse IoT Stack and openHAB can be used for end-to-end solutions for smart city applications and services.

XDK workbench is for code development environment for the prototype kit XDK 110. The workbench includes the IDE, software for all system component drivers, a debugger port, and high and low level APIs for sensors. The bench also includes libraries for sensor readings, CoAP and LWM2M client and server, MQTT client and broker using Eclipse Paho connector, and HTTP client request and response server. Also included are virtual SDK applications for virtual mobile iOS and Android GUIs for monitoring and control of the sensors and WLAN and Bluetooth LE protocol stacks. XDK provides access to XDK developer community for online technical support. The community offers platform for discussions of ideas, exchange of information and innovation.

Smart City Parking

A growing problem in cities is of vehicular traffic congestion and parking spaces. A modern city, therefore, provides a number of multilevel parking spaces which spread all over the city. A driver needs a mobile app. Significant fuel saving can result from provisioning of smart parking spaces in a city.

A smart parking-service should enable the following:

1. Guides the drivers for the available parking slots and spaces
2. Provides a mobile app, and the app assists a driver and enables him/her to obtain the appropriate parking-slot information remotely. The information includes location of the parking utility, its cost, advance reservation facility, direction guidance and the time to reach an available slot. The app accesses the slots availability in real-time data in pub/sub mode.
3. Publishes messages in real time for available slots and alerts for slot unavailability at the parking utility
4. Consists of a central supervisory control and monitoring system (CSS) which connects the edge sensors and devices, accurately senses the slots available for occupancy of vehicles in real time, and predicts the expected availability time in case of non-availability of slots
5. Optimises the usages of parking spaces and reaching time
6. Provides display boards at road traffic junctions for status of availability
7. Provides good parking experience to users
8. Adds value for all parking stakeholders, drivers and service providers
9. Enables intelligent decisions using data and historical analytics reports at city cloud data store, and enables planning for traffic flow in the city by predictive analytics

Sensors play vital role in the smart parking. The application is ranked as topmost among 50 sensor-applications for a smarter world.

Domain Architecture Reference Model

data-flow diagram, domains and architecture reference model for smart parking applications and services.

The figure shows four layers at two domains. Parking spaces are at layer 1. They are sensed using coordinators at each level in multilevel parking spaces. An actuator for the light at each slot is used. Lighting control module at the coordinator actuates the parking space lights. The lightings can be switched on and off as per requirement for each space.

A Parking Assistance System (PAS) is at layer 2. This includes CCS and three modules for monitoring, control and display. Layers 1 and 2 communication protocols are ZigBee, LWM2M and UDP.

The CSS maintains a real-time database of time-series data of the parking spaces. The system connects layer 3, which includes the SMS gateway and cloud IoT platform.

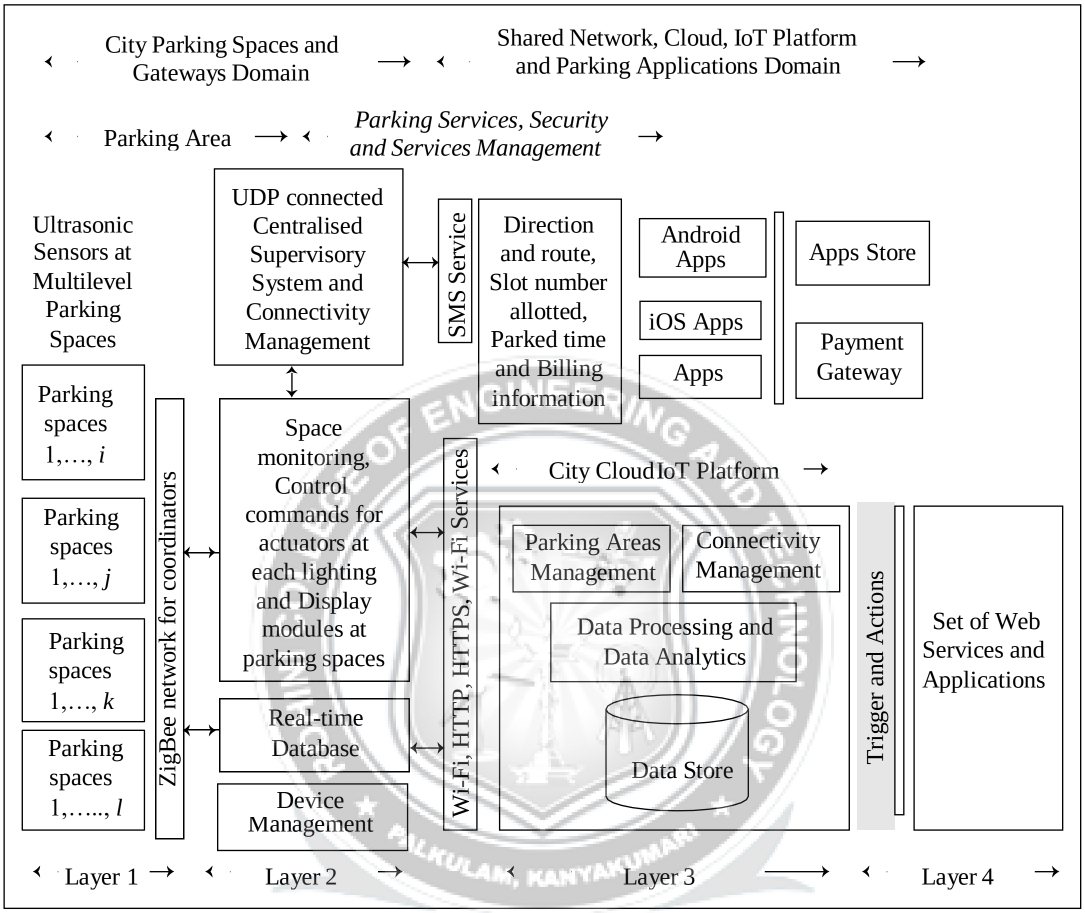
City Parking Spaces and Gateways Domain
Shared Network, Cloud, IoT Platform and Parking Applications Domain
Parking Area
Parking Services, Security and Services Management
Ultrasonic Sensors at Multilevel Parking Spaces
Parking spaces 1,…, i
Parking spaces 1,…, j
Parking spaces 1,…, k
Parking spaces 1,….., l
ZigBee network for coordinators
UDP connected Centralised Supervisory System and Connectivity Management
Space monitoring, Control commands for actuators at each lighting and Display modules at parking spaces
Real-time Database
Device Management
SMS Service
Wi-Fi, HTTP, HTTPS, Wi-Fi Services
Direction and route, Slot number allotted, Parked time and Billing information
Android Apps
iOS Apps
Apps
Apps Store
Payment Gateway
City CloudIoT Platform
Parking Areas Management
Connectivity Management
Data Processing and Data Analytics
Data Store
Trigger and Actions
Set of Web Services and Applications
Layer 1
Layer 2
Layer 3
Layer 4

Figure Data flow diagram, domain architecture reference model and four layers for the design of smart parking applications and services

Layer 2 connects CSS with all coordinators. Layer 2 includes a real-time time-series database. Layer 2 also connects with three modules for (i) displaying, (ii) space monitoring and (iii) control commands for actuators for each parking slot.

Layer 3 consists of SMS gateway and City cloud IoT platform that connects CSS, modules and database using the Wi-Fi, HTTP and HTTPS services. The platform has data store, data processing and analytics, and parking areas and connectivity management modules.

The CSS sends the UDP packets using MPLS and uses a SMS service to communicate with the mobile app. The SMS service communicates parking information. A packet provides information such as slot available, slot allocated, time parked, billing information and directional and parking space route details to the user’s mobile phone

The user downloads a PAS app from the App store. The user’s mobile also connects to a payment gateway for parking service bill payment.

Layer 4 web services connect the cloud data store, and use the PaaS cloud for the analytics.

Hardware Prototype Development and Deployment

Sensors for ultrasonic pulse detectors. When a car parking slot is occupied, then the parked car reflects back ultrasonic pulses to the source. The sensor measures the reflected directional intensity and delay period for the reflections. The advantage of ultrasonic waves is that the distance and therefore, identification of reflectorcar surface is also found from the delay in reflected pulses.

The coordinator updates the parked slots status on each alert from a circuit. The sensor associated circuit at coordinator alerts the CSS for status change and generates time-series messages from the sensor data and communicates to the CSS for saving at the real-time database.

The design principle for a set up for identifying vacant spacesand slot IDs using ultrasonic pulses and back reflections to the transceiver (emitter and sensors) at the coordinator.

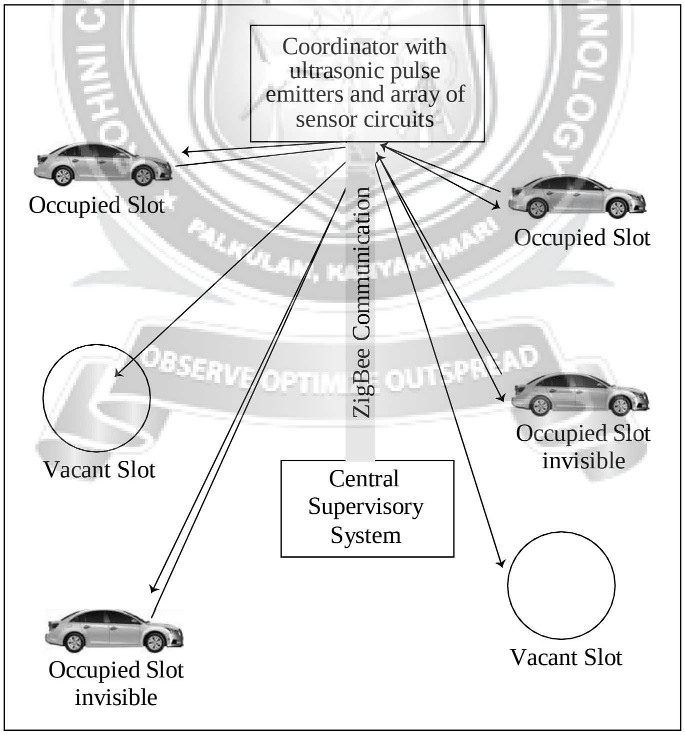

Figure Design principle for the set up for identifying vacant spaces using ultrasonic pulses andback reflections from cars to the transceiver at the coordinator

Assume that a set of four coordinators each are placed at each level. A coordinator consists of a transceiver, which emits ultrasonic pulses and receives reflected signals using an array of sensors. Sensors have associated circuits at each level of parking. Assume that each transceiver's-coverage area at each coordinator spans to one fourth of the slots at that level. Therefore, total sixteen coordinators are located at four levels 1, 2, 3, and 4. Four coordinators at level 1 are C_{i1}, C_{i2}, C_{i3} and C_{i4}, four at level 2 are C_{j1}, C_{j2}, C_{j3} and C_{i4}, four at level 3 are C_{k1}, C_{k2}, C_{k3} and C_{k4} and four at level 4 are C_{l1}, C_{l2}, C_{l3} and C_{l4}.

The coordinators process the status information and time-series data transmits to CSS. The CSS receives information of parking slots from the coordinators as UDP datagram. CSS then relays the parking information to city-cloud platform data-store and to users, which seeks that from mobiles or card dashboard computer. Each coordinator sends all messages with slot ID and observation instances onto the ZigBee network as time-series data. A real-time database updates at regular intervals, say of 1 m each.

The edge sensors and devices wirelessly connect within small cells, systems connect with ZigBee. They communicate using LPWAN. The distributed network of edge-computing systems connects using IP protocols or using the Multi Protocol Label Switching (MPLS). The MPLS assigns the labels to data packets and forward the labels to City cloud IoT-platform.

Code Development Environment, Development, Debugging and Deployment

End-to-end IoT Solutions with Java and Eclipse IoT Stack. openHAB software platform for devices, sensor and edge systems at a smart home. Java, Eclipse IoT Stack and openHAB can be used for end-to-end solutions for each parking utility in the city.

Smart city parking solutions can also be developed using the intelligent management platform IMPACT.