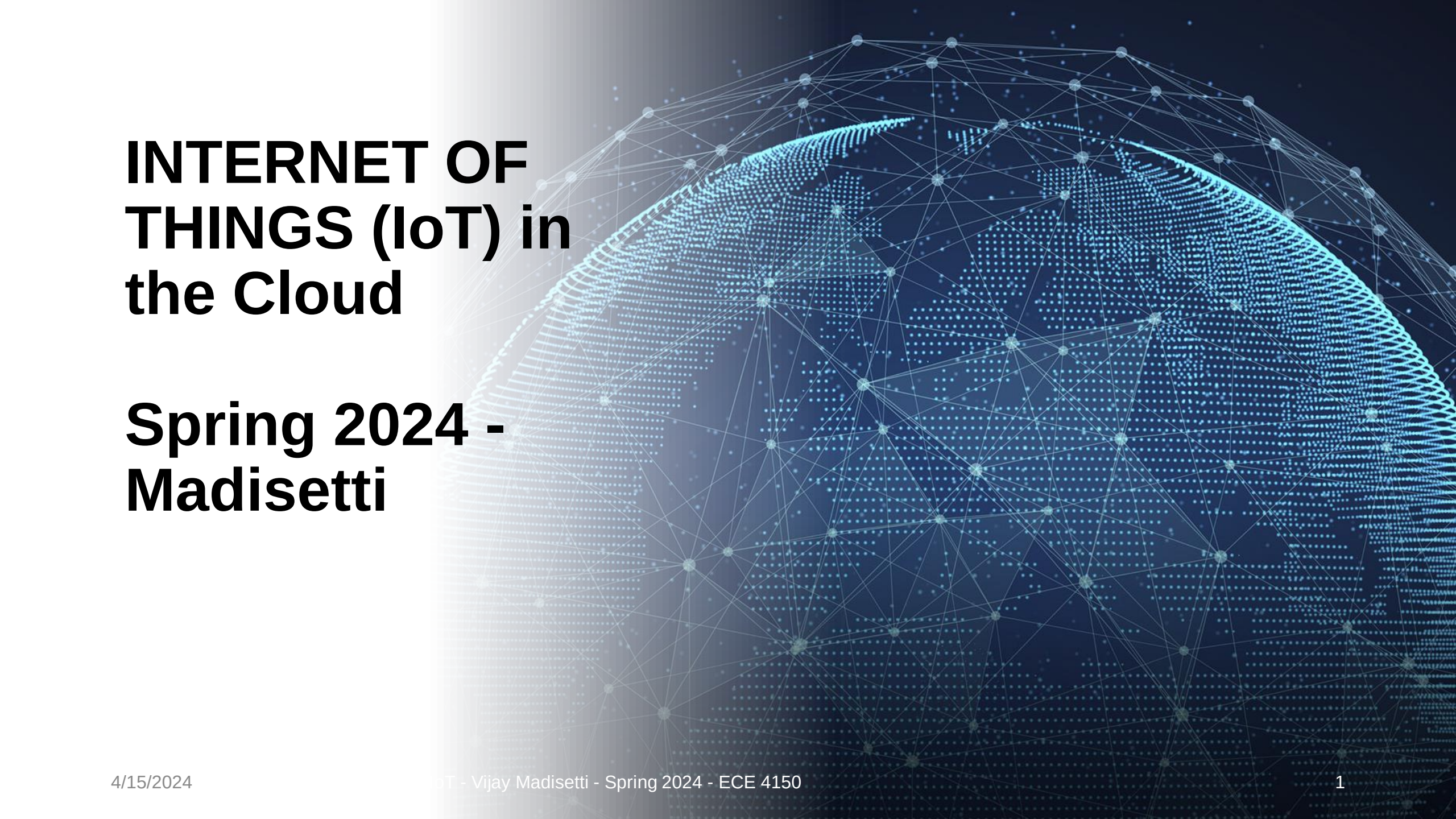
INTERNET OF THINGS (IoT) in the Cloud

Spring 2024 - Madisetti



The Four Industrial Revolutions

Navigating the next industrial revolution

WORLD ECONOMIC FORUM
COMMITTED TO IMPROVING THE STATE OF THE WORLD

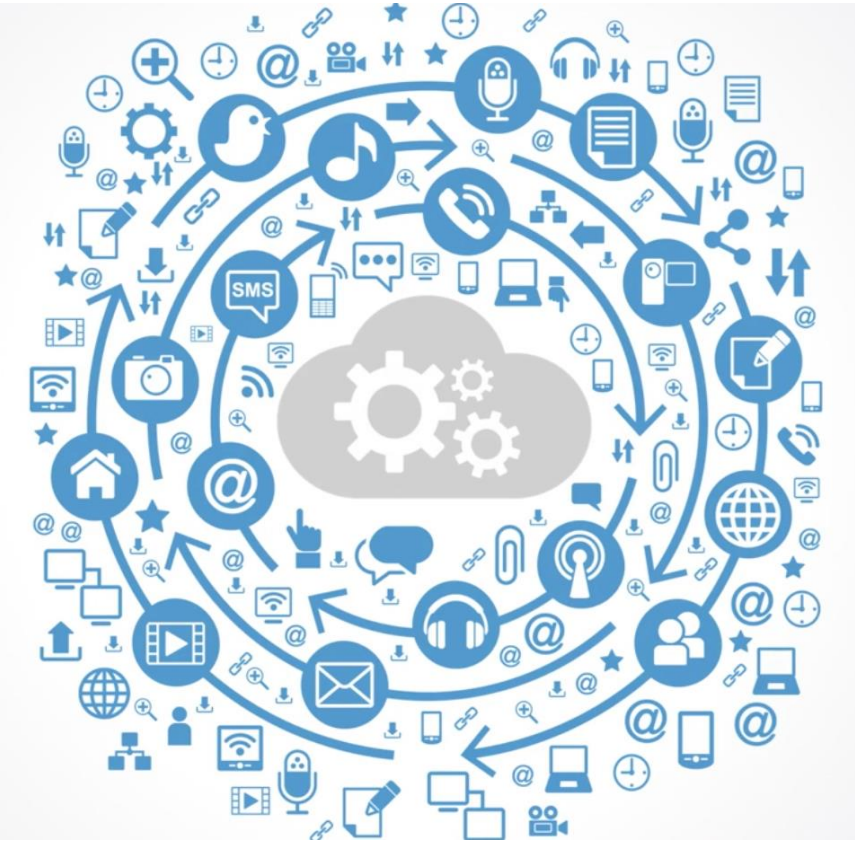
Revolution	Year	Information
1	1784	Steam, water, mechanical production equipment
2	1870	Division of labour, electricity, mass production
3	1969	Electronics, IT, automated production
4	?	Cyber-physical systems

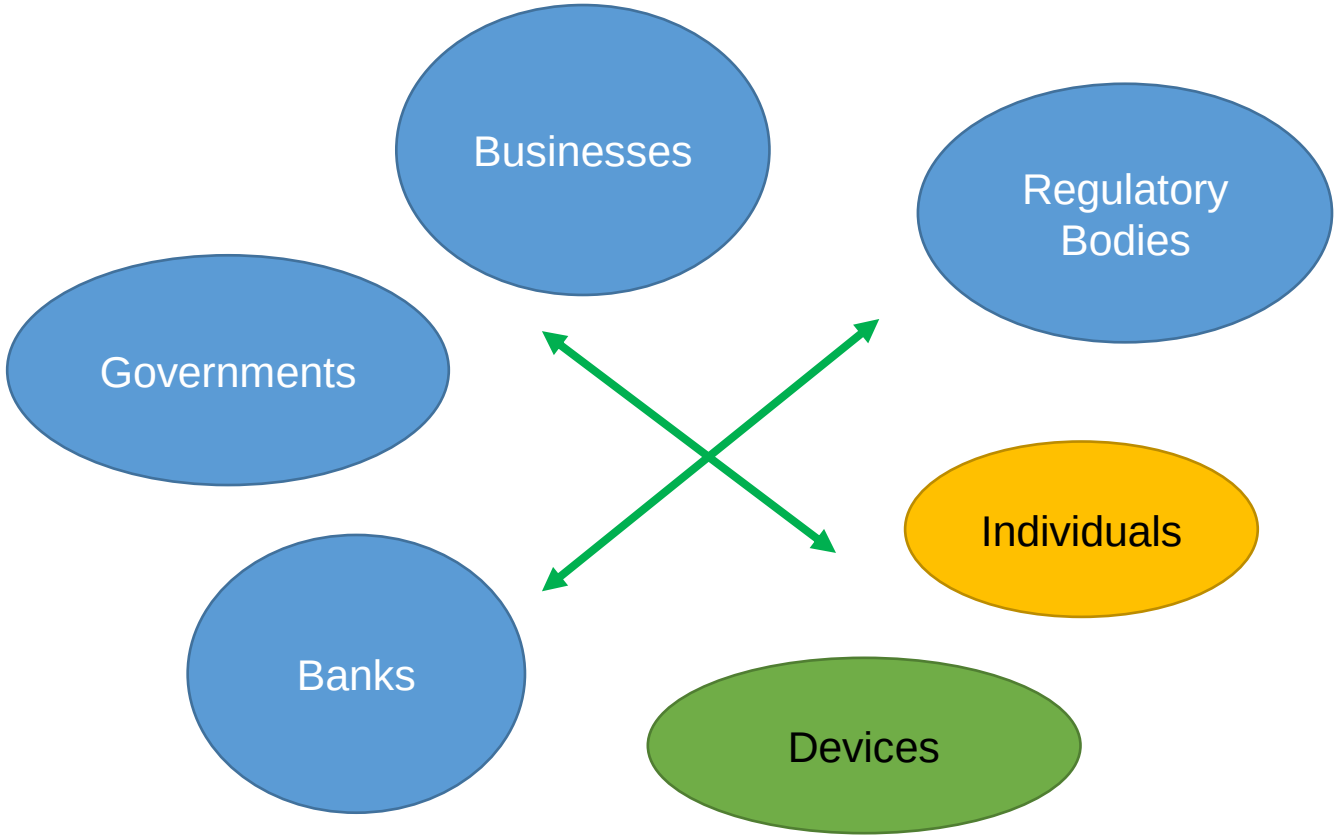
www.weforum.org

Internet of Things (IoT) Blockchain, Analytics will Play a Central Role in Cyber-Physical Systems

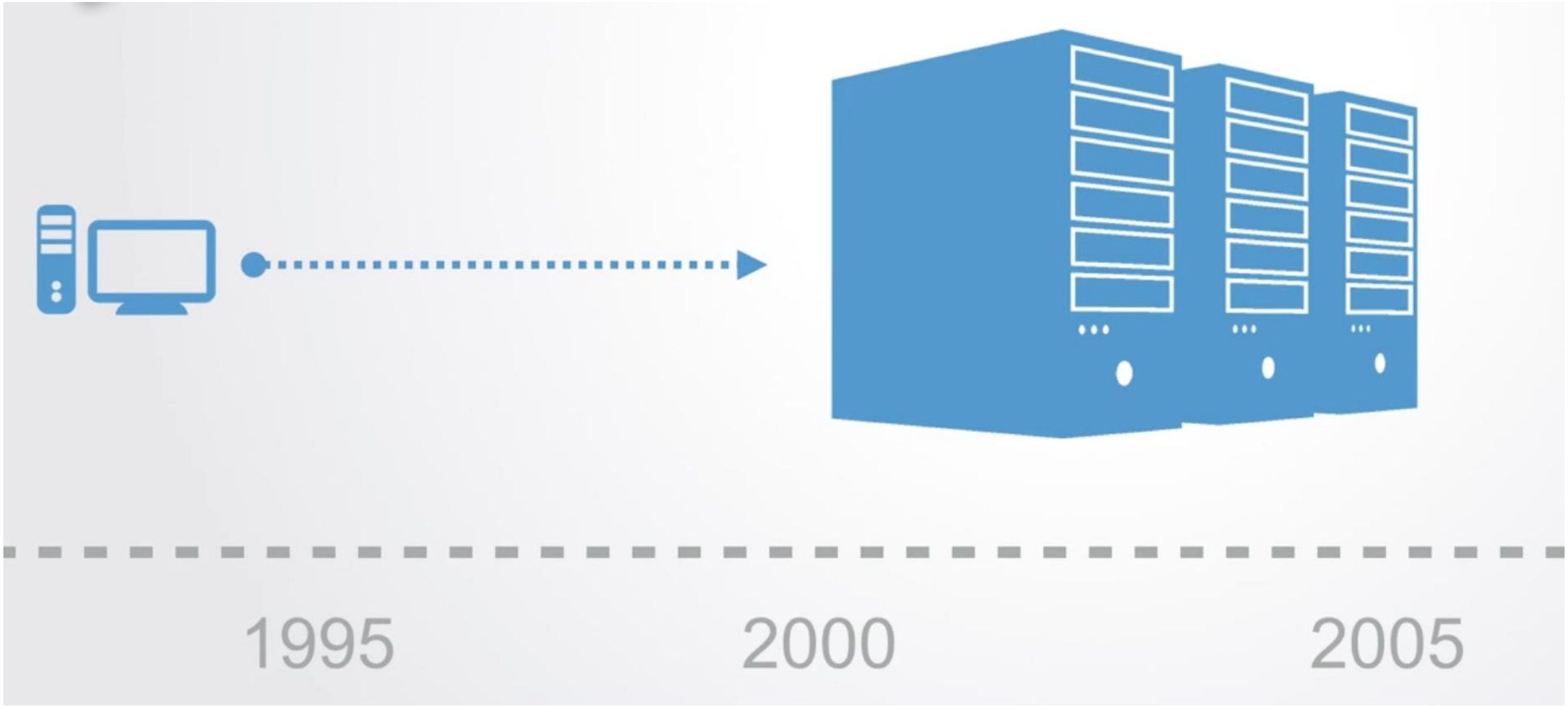
The new world – Web 3.0

Goal: Secure, transparent & efficient business and financial transactions between a large number of diverse and untrusting entities

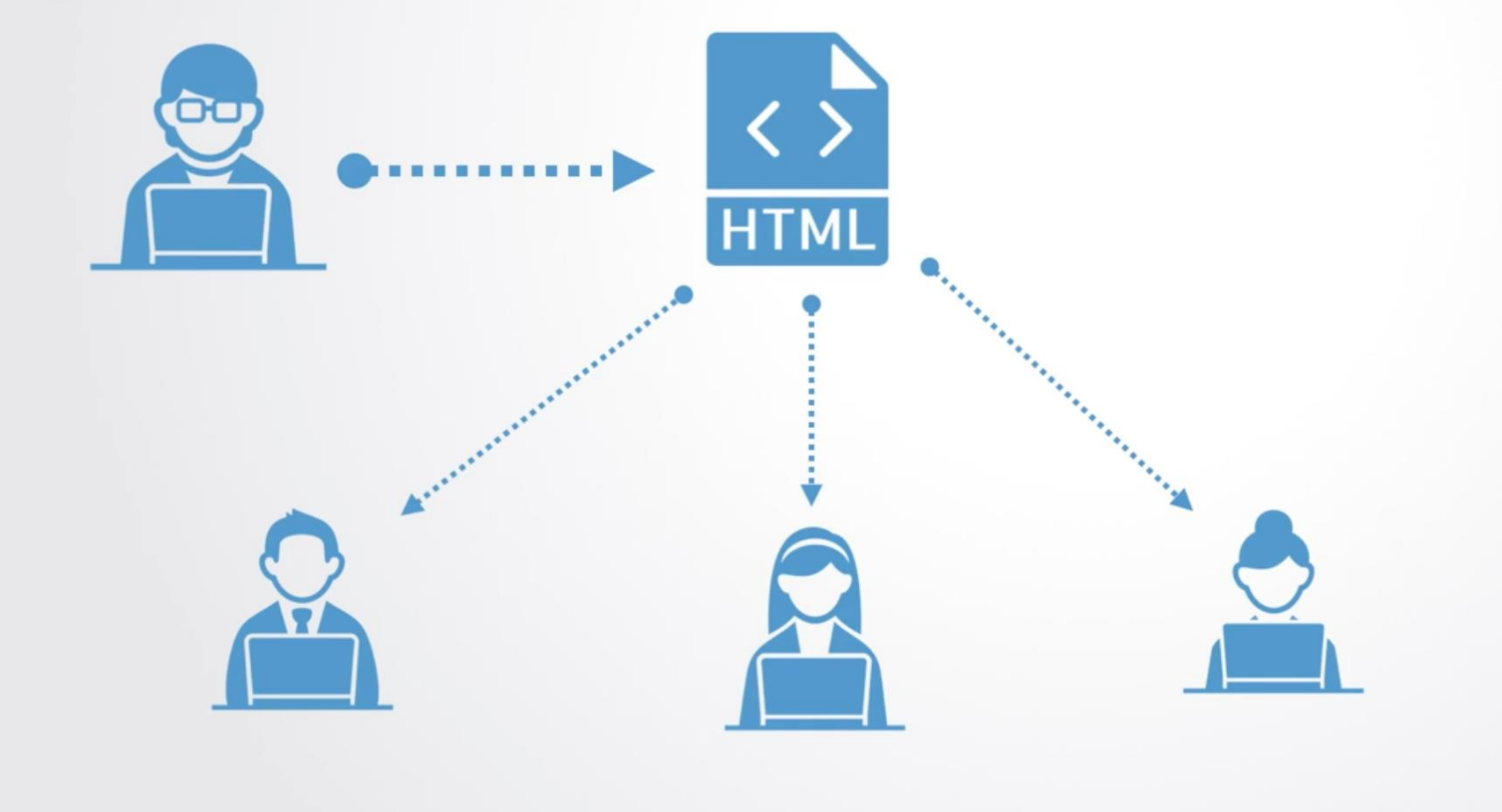




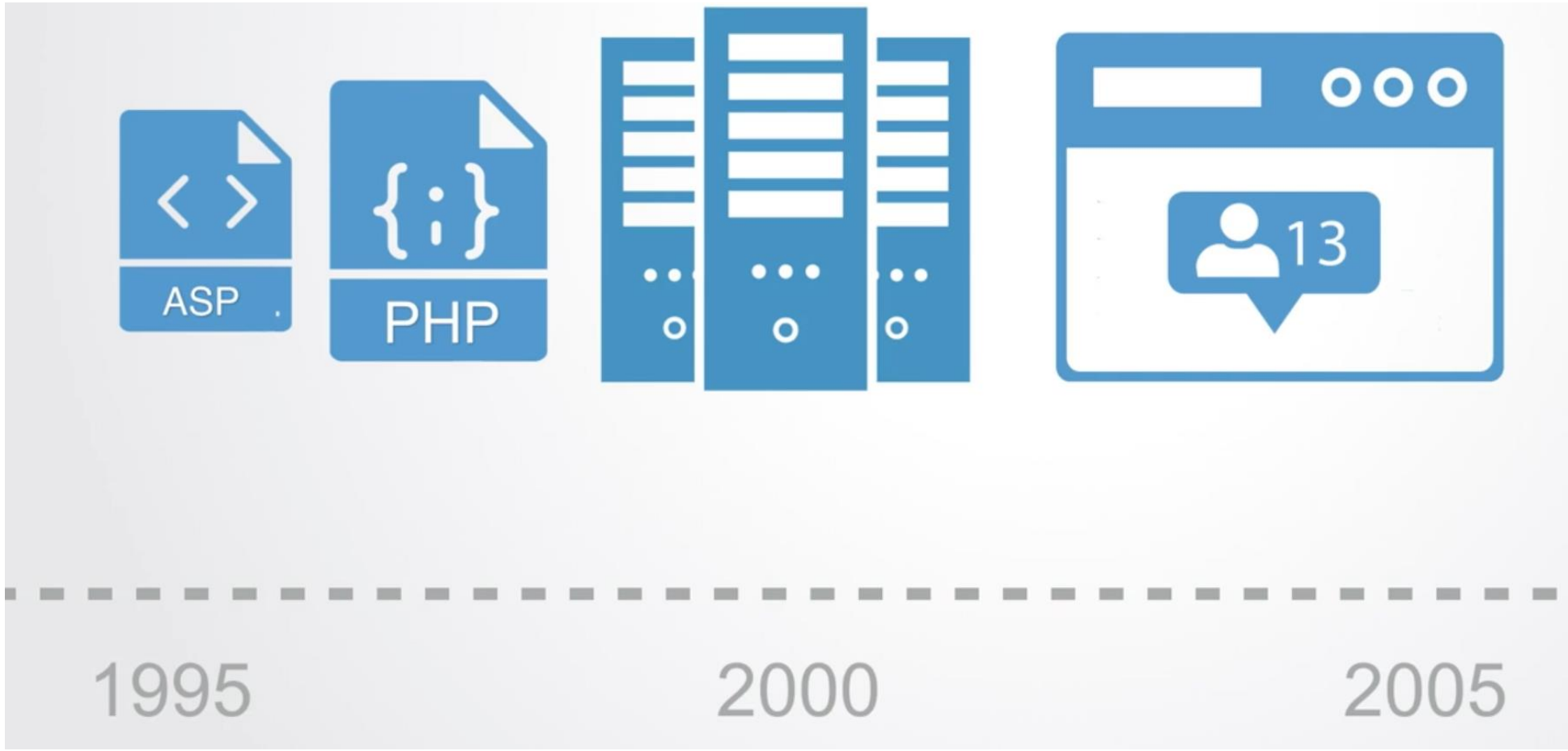
Evolution of the Internet & Web Technologies



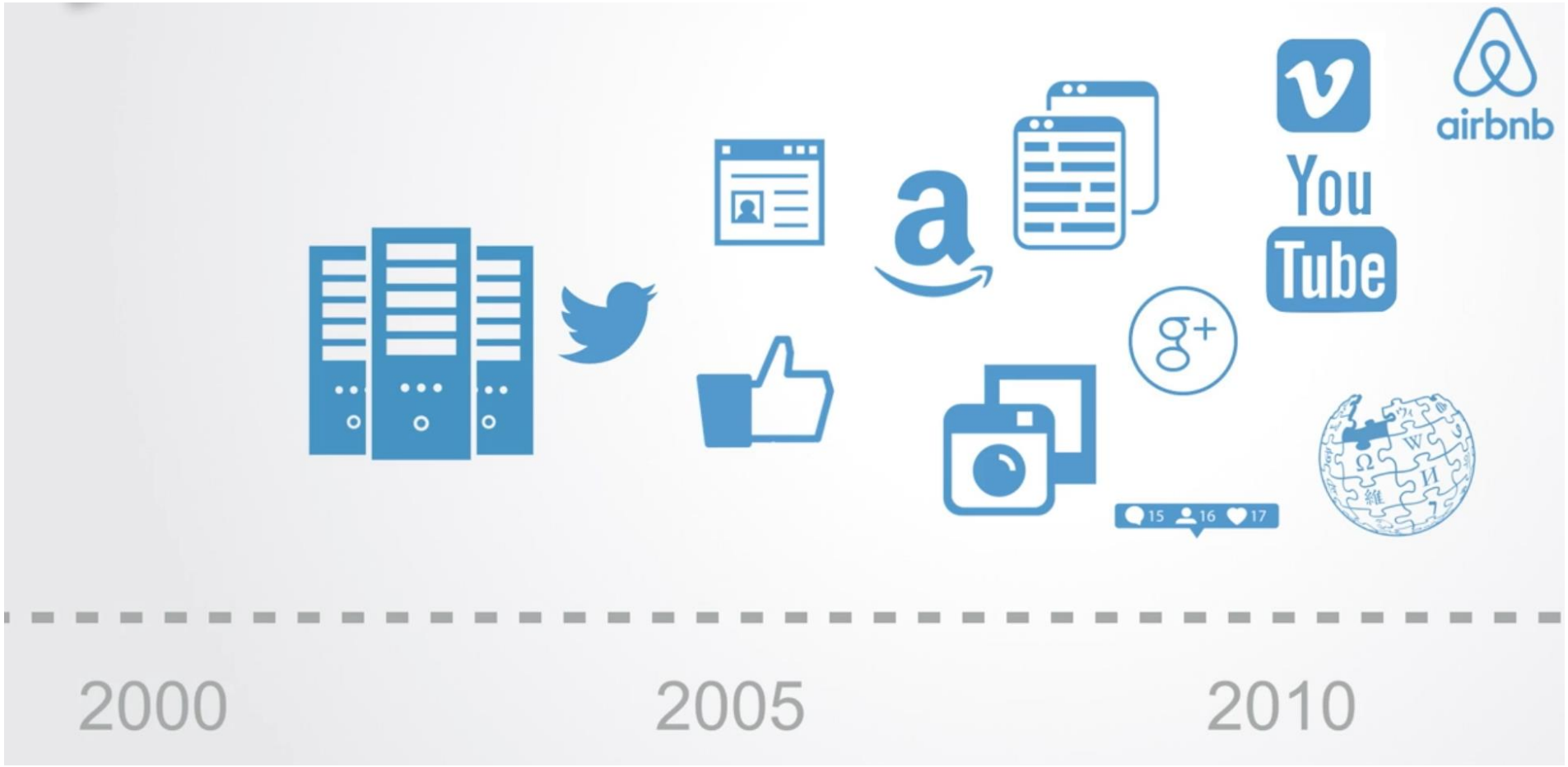
Web 1.0 - Simply Distributed Data to Clients



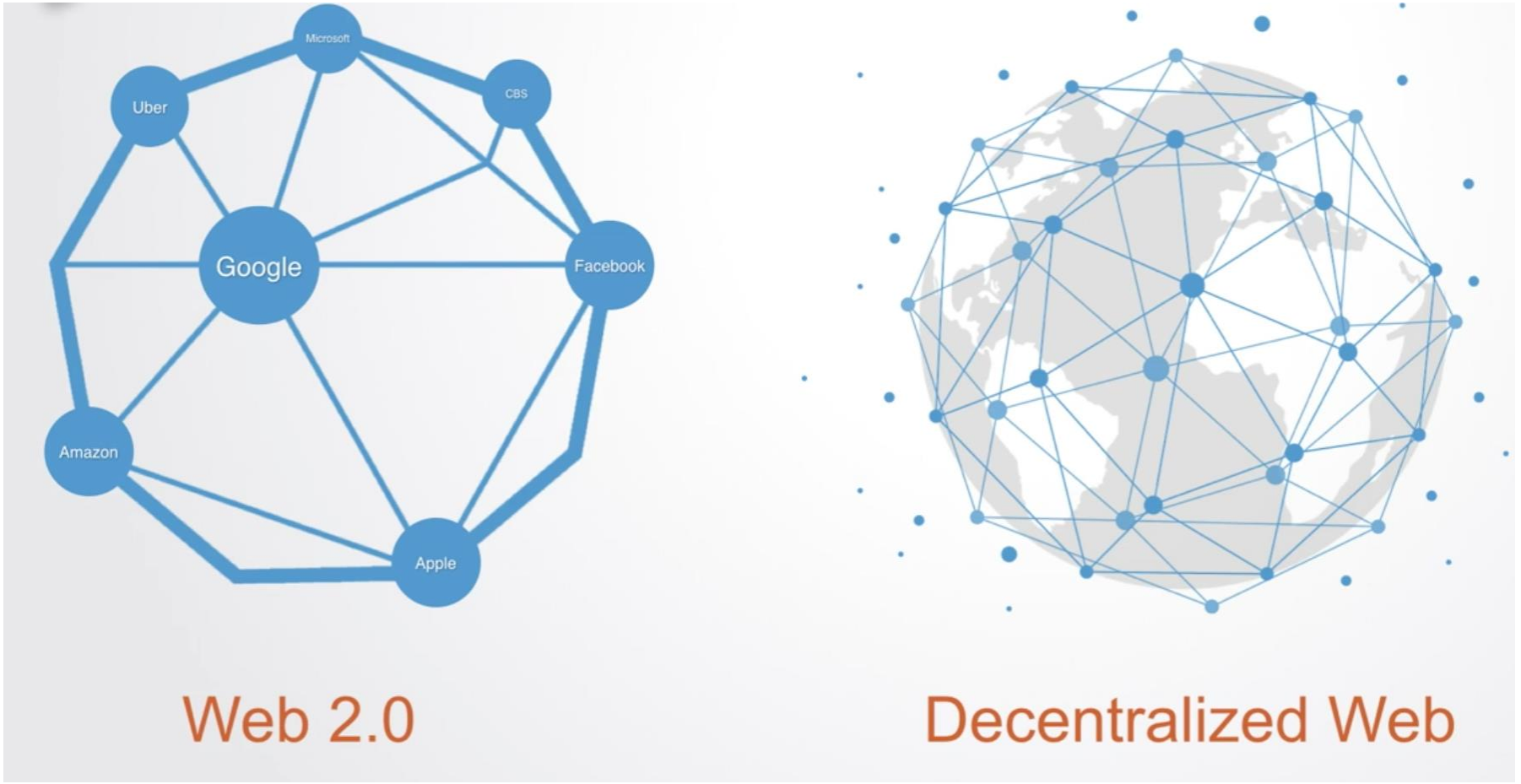
Web 2.0 - Evolution of the Internet & Web Technologies



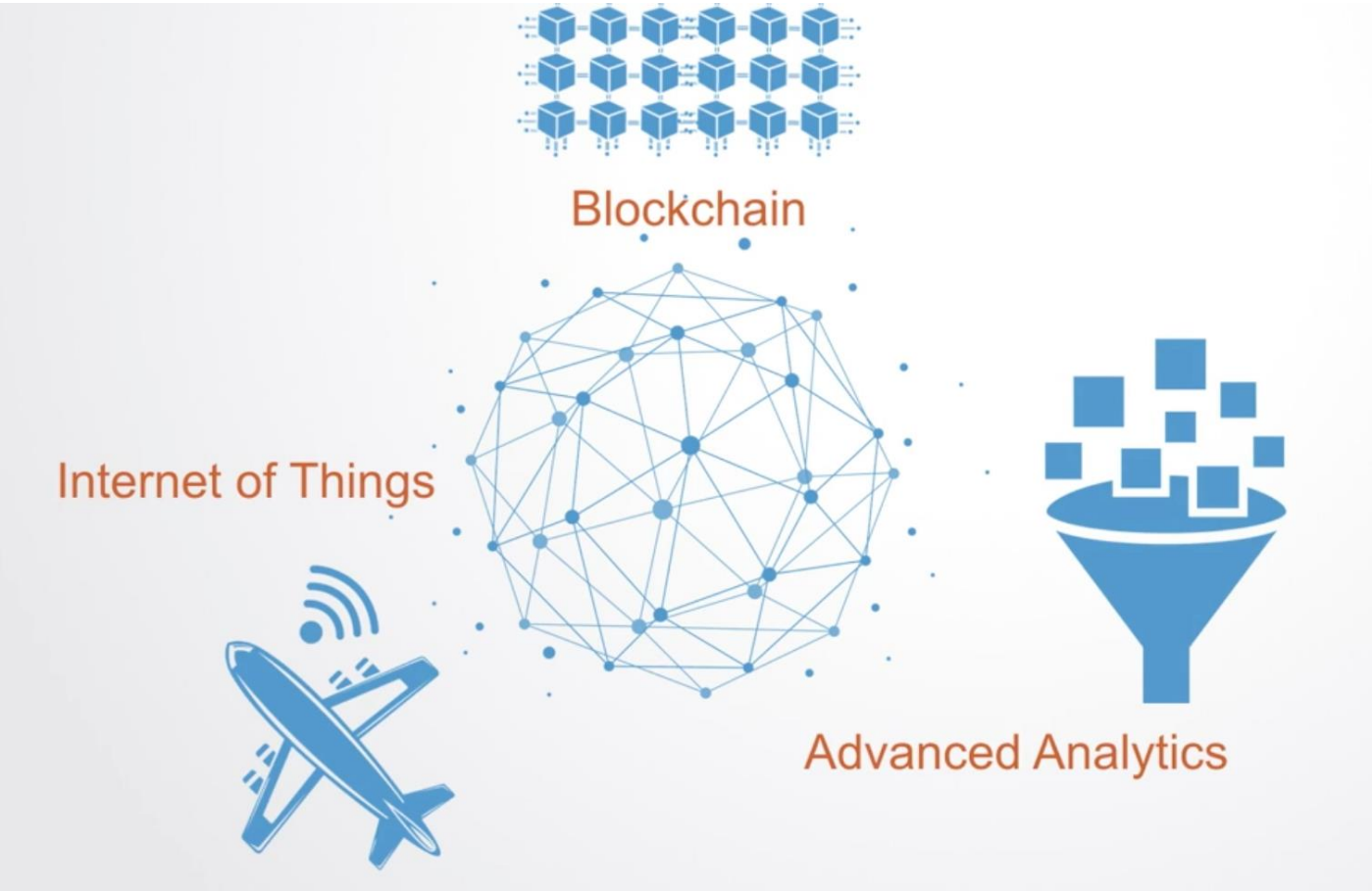
Large Servers & Platforms & Scalable Applications = Web 2.0



Big Centralized Servers to Fully Decentralized Nodes



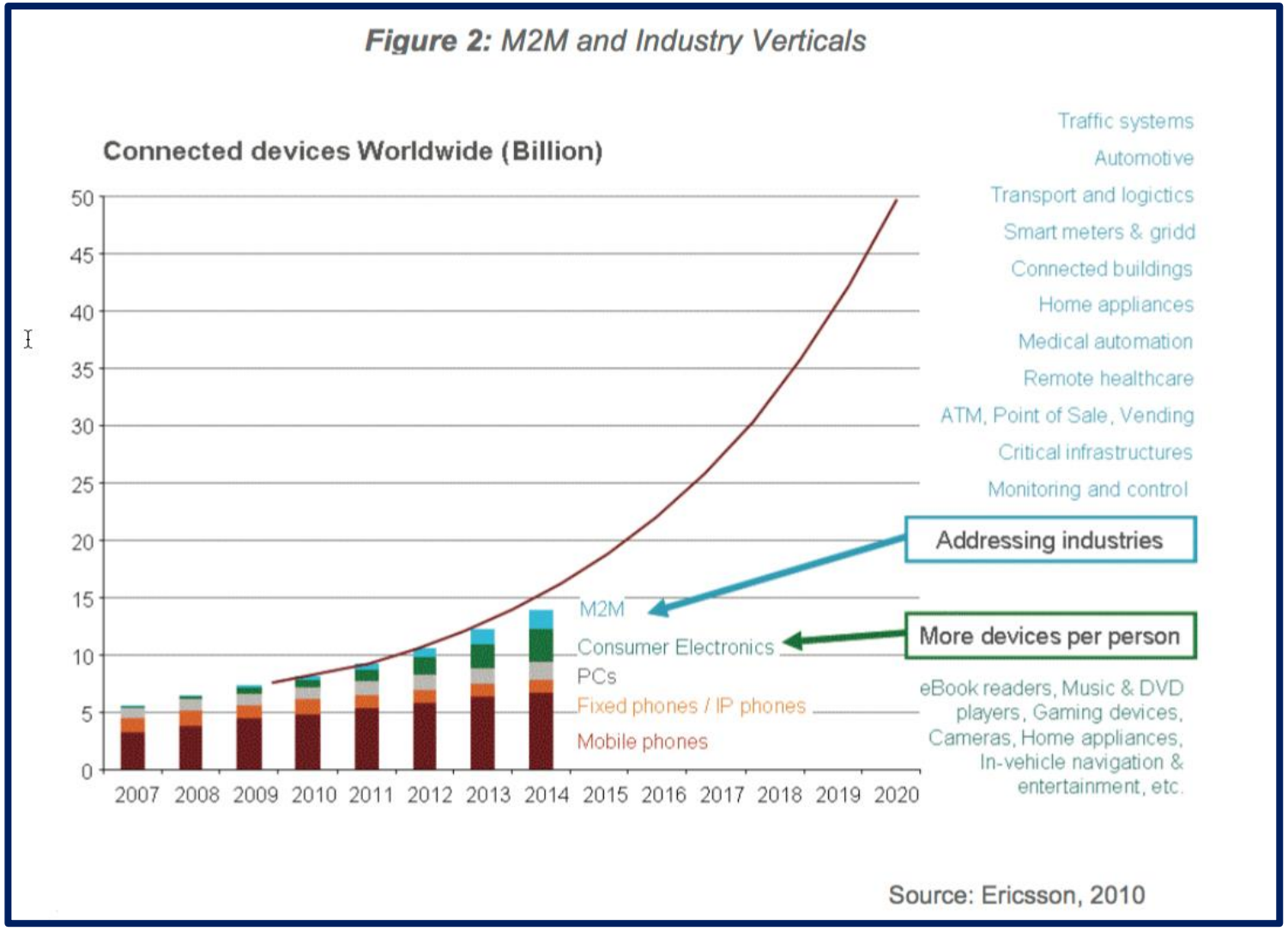
What are the building blocks of Web 3.0 ?



Definition of IoT

A **dynamic** global network infrastructure with **self-configuring** capabilities based on standard and **interoperable communication protocols** where physical and virtual "things" have **identities**, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly **integrated into the information network**, often communicate data associated with users and their environments.

IoT and Business Areas



Setting	Description	Examples
Human Health	Devices attached to or inside the human body	Devices (wearables and ingestibles) to monitor and maintain human health and wellness; disease management; increased fitness; higher productivity
Home	Buildings where people live	Home voice assistants; automated vacuums; security systems
Retail Environments	Spaces where consumers engage in commerce	Stores, banks, restaurants, arenas—buildings where consumers physically consider and purchase products and services; self-checkout; in-store offers; inventory optimization
Offices	Spaces where knowledge workers work	Energy management and security in office buildings; improved knowledge-worker productivity
Factories	Standardized production environments	Manufacturing plants, hospitals, and farms; operating efficiencies; optimizing equipment use and inventory
Work Sites	Custom production environments	Mining, oil and gas exploration and production, construction; operating efficiencies; predictive maintenance; health and safety
Vehicles	Systems inside moving vehicles	Vehicles, including cars, trucks, ships, aircraft, and trains; condition-based maintenance; usage-based design; presales analytics
Cities	Urban environments	Public spaces and infrastructure in urban settings; adaptive traffic control; smart meters; environmental monitoring; resource management
Outside	Between urban environments (and outside other settings)	Railroad tracks, autonomous vehicles (includes level 2 autonomy and up outside urban locations), and flight navigation; real-time routing; connected navigation; shipment tracking

The Factories setting, which includes all standardized production environments, accounts for the most potential economic value from IoT.

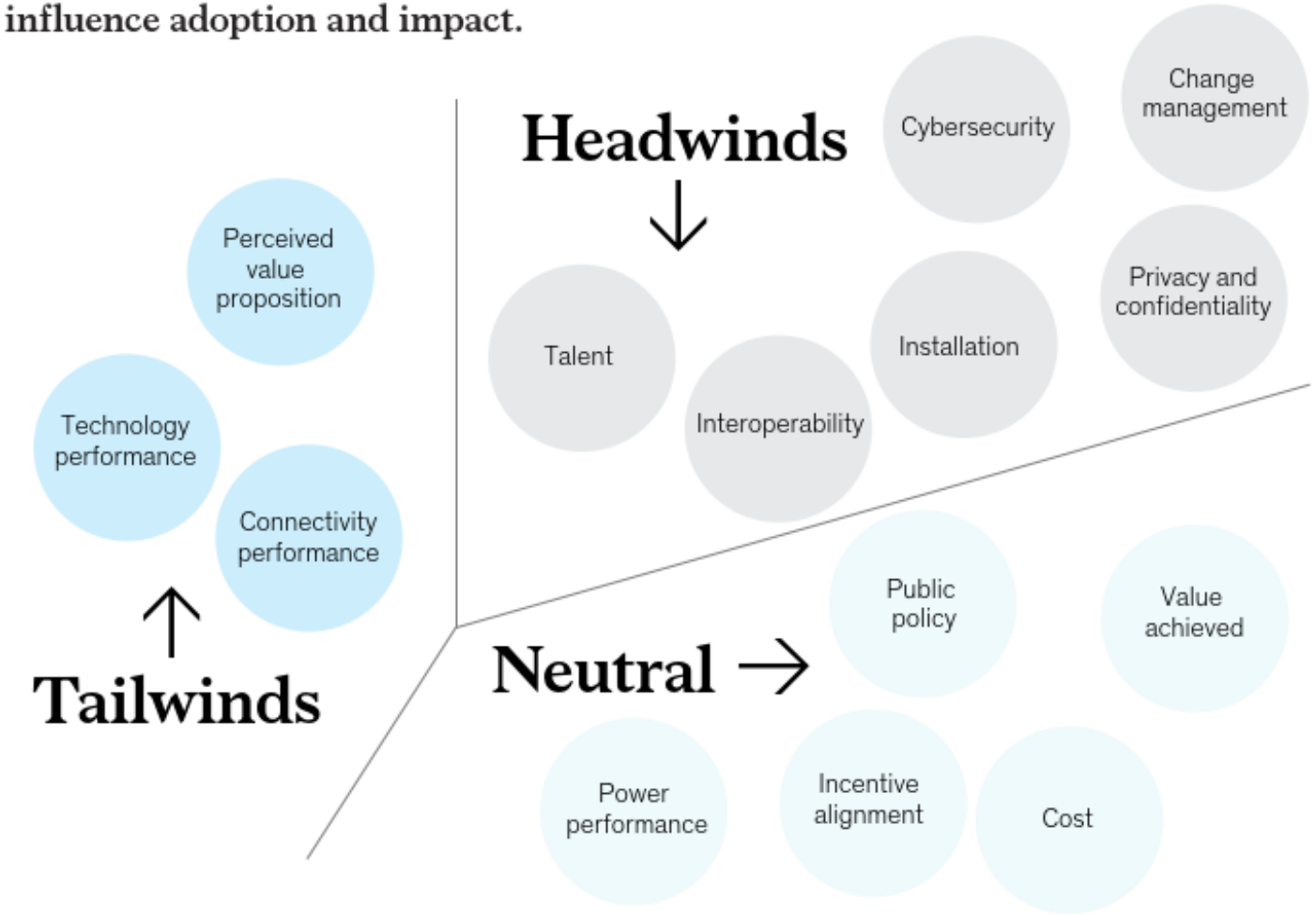
Estimated economic value, 2030, $ billions

Setting Top use cases	Total
Factories Operations management: Manufacturing, 460–1,290 Farm yield management, 250–520 Predictive maintenance: Manufacturing, 260–460	1,430–3,320
Human Health Monitoring and treating illness, 240–1,200 Improving wellness, 310–560	550–1,760
Work Sites Operations management: Construction, 70–540 Operations management: Oil and gas, 80–300 Improved equipment maintenance: Construction, 20–220	400–1,730
City Centralized and adaptive traffic control, 100–390 Autonomous vehicles, 240–300 Congestion lanes, 70–150	970–1,700
Retail Environments Self-checkout: Billing and material handling, 280–340 Real-time personalized promotions, 60–190 Self-checkout: Payments, 140–180	650–1,150
Outside Autonomous vehicles: Cars, 140–250 Operations management in defense, 60–190 Ship navigation, 80–160	400–930
Home Chore automation, 290–580 Energy management: Home, 130–230 Safety and security, 20	440–830
Vehicles Insurance: Personal transportation, 130–140 Passenger vehicles and trucks: After-sales service improvements (eg, condition-based maintenance), 90–140 Shipping: After-sales service improvements (eg, condition-based maintenance), 40–70	430–620
Offices Human productivity: HR redesign, 110–260 Human productivity: Augmented reality, 30–100 Human productivity: Activity monitoring, 60–80	240–500
Grand total	**5,500–12,600**

Note: Figures may not sum, because of rounding

A variety of factors influence at-scale IoT adoption and impact.

While nuances exist across settings, some cross-cutting headwinds and tailwinds influence adoption and impact.



Characteristics of IoT

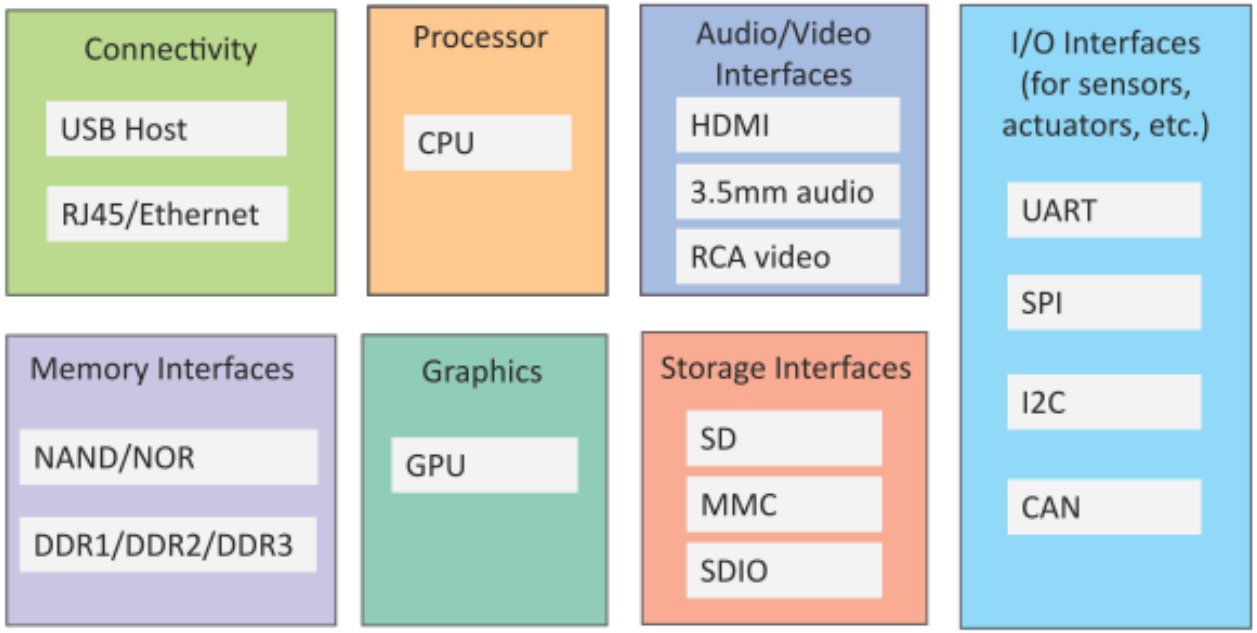
- Dynamic & Self-Adapting
- Self-Configuring
- Interoperable Communication Protocols
- Unique Identity
- Integrated into Information Network

Physical Design of IoT

- The "Things" in IoT usually refers to IoT devices which have unique identities and can perform remote sensing, actuation and monitoring.
- IoT devices can:
  - Exchange data with other connected devices and applications (directly or indirectly), or
  - Collect data from other devices and process the data locally or
  - Send the data to centralized servers or cloud-based application back-ends for processing the data, or
  - Perform some tasks locally and other tasks within the IoT infrastructure, based on temporal and space constraints

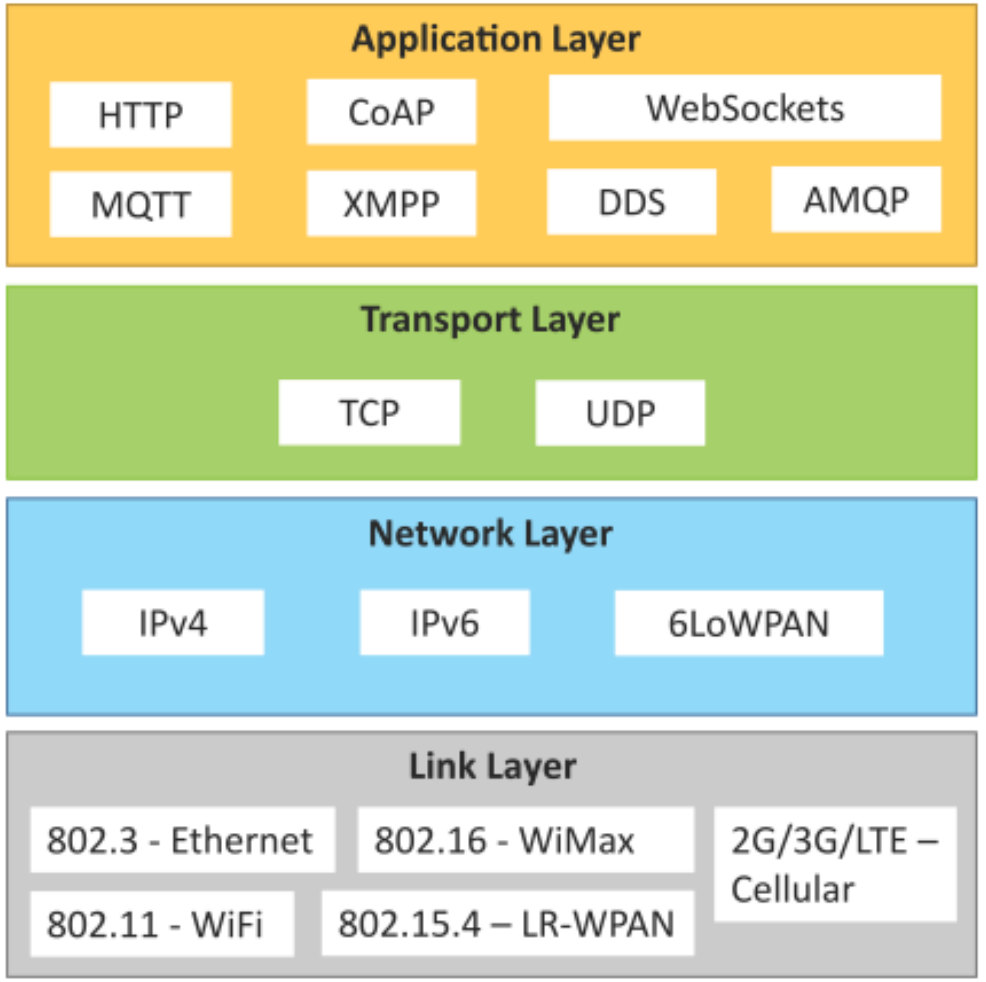
Generic block diagram of an IoT Device

- An IoT device may consist of several interfaces for connections to other devices, both wired and wireless.
  - I/O interfaces for sensors
  - Interfaces for Internet connectivity
  - Memory and storage interfaces
  - Audio/video interfaces.



IoT Protocols

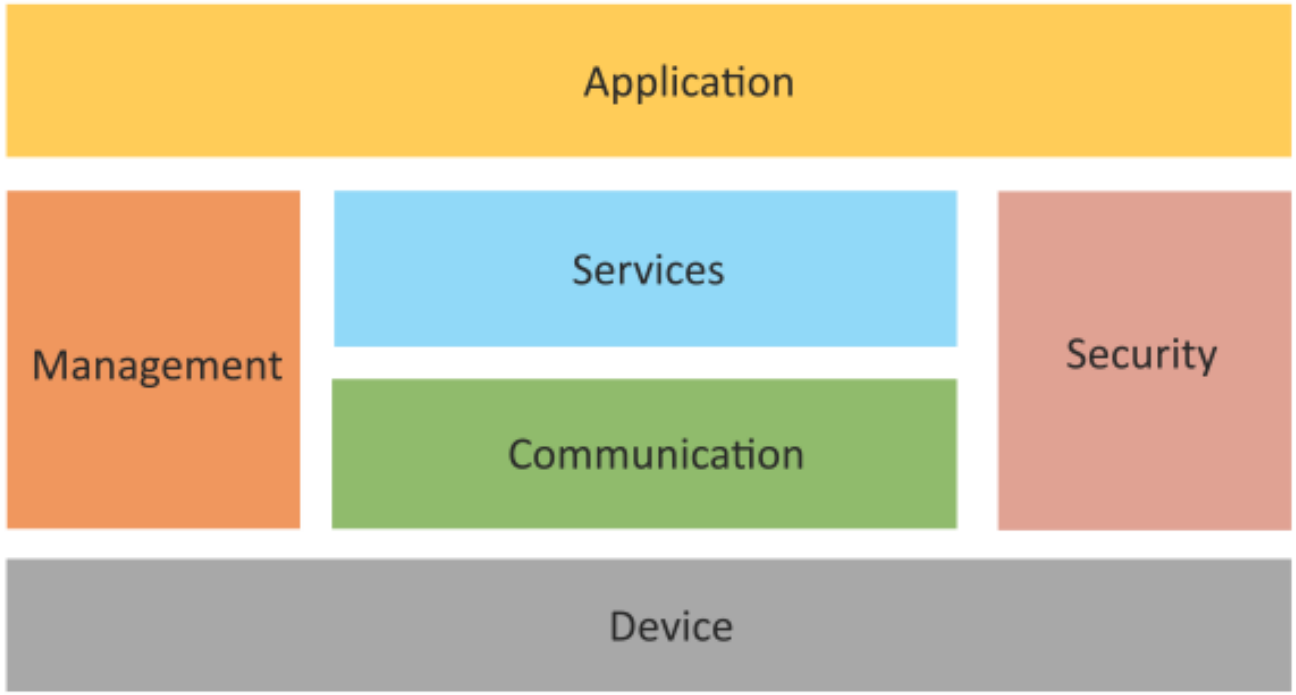
- Link Layer
  - 802.3 – Ethernet
  - 802.11 – WiFi
  - 802.16 – WiMax
  - 802.15.4 – LR-WPAN
  - 2G/3G/4G
- Network/Internet Layer
  - IPv4
  - IPv6
  - 6LoWPAN
- Transport Layer
  - TCP
  - UDP
- Application Layer
  - HTTP
  - CoAP
  - WebSocket
  - MQTT
  - XMPP
  - DDS
  - AMQP



Logical Design of IoT

- Logical design of an IoT system refers to an abstract representation of the entities and processes without going into the low-level specifics of the implementation.

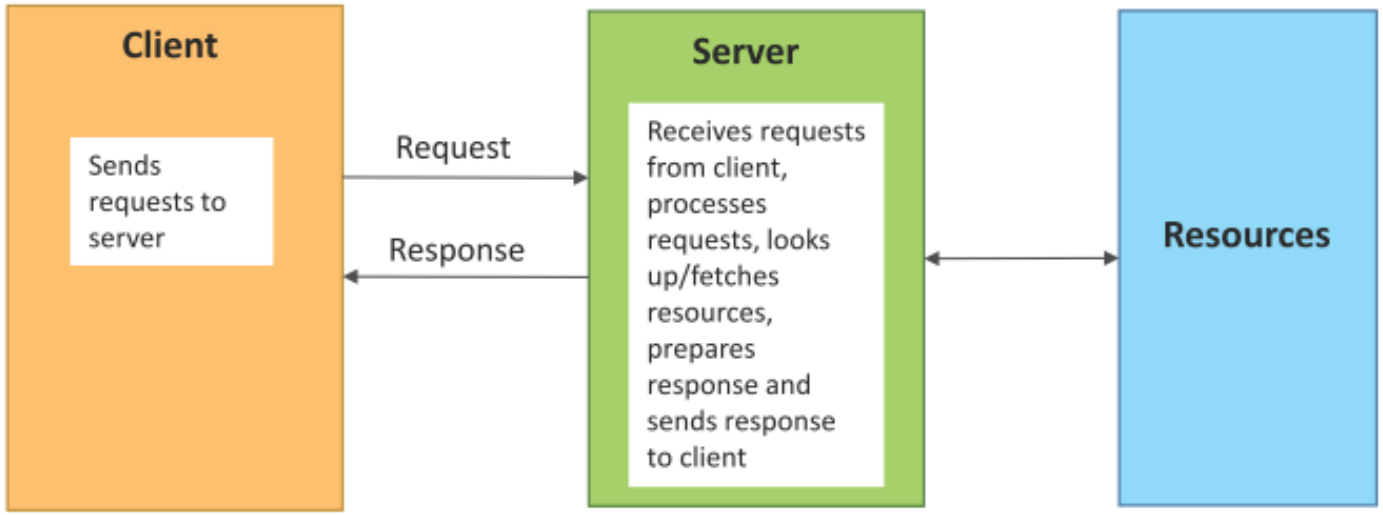
- An IoT system comprises of a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, communication, and management.



Request-Response communication model

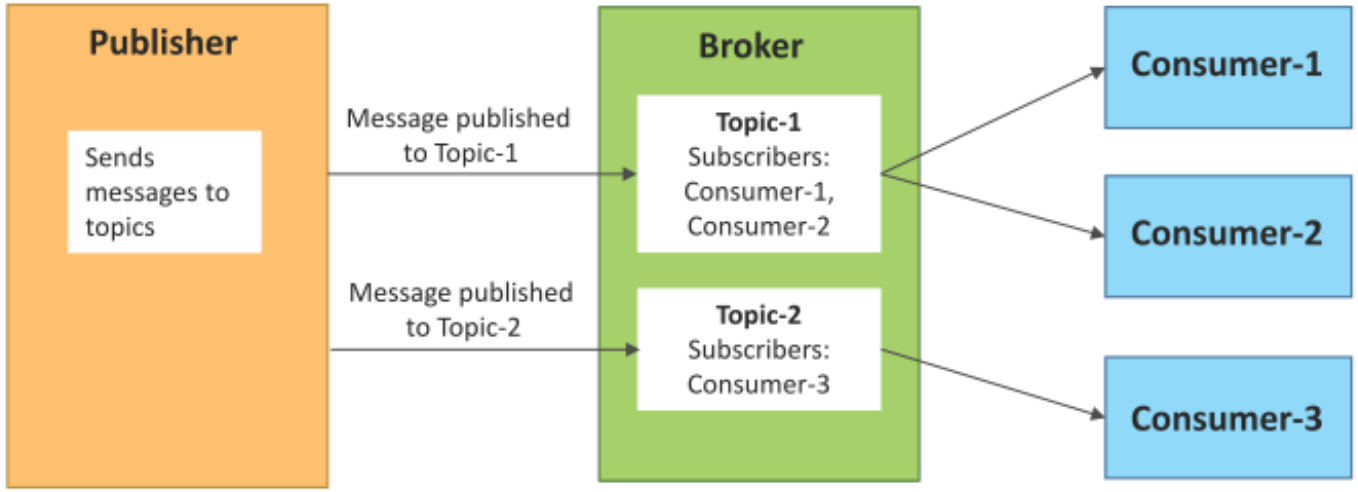
- Request-Response is a communication model in which the client sends requests to the server and the server responds to the requests.

- When the server receives a request, it decides how to respond, fetches the data, retrieves resource representations, prepares the response, and then sends the response to the client.



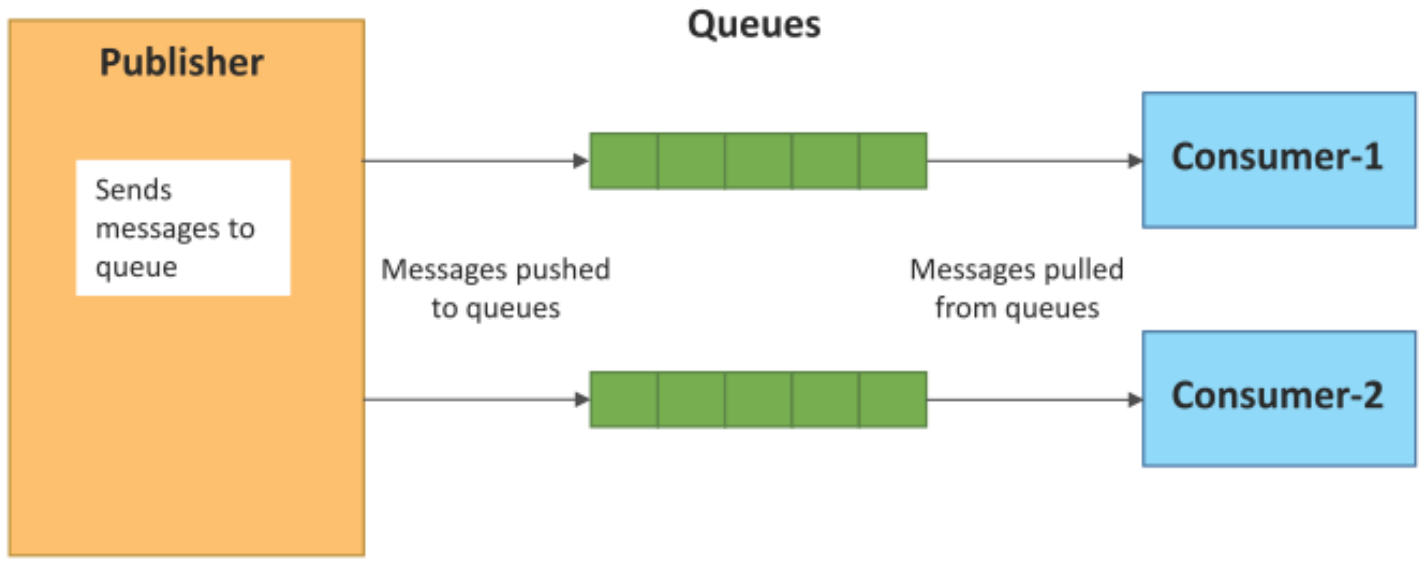
Publish-Subscribe communication model

- Publish-Subscribe is a communication model that involves publishers, brokers and consumers.
- Publishers are the source of data. Publishers send the data to the topics which are managed by the broker. Publishers are not aware of the consumers.
- Consumers subscribe to the topics which are managed by the broker.
- When the broker receives data for a topic from the publisher, it sends the data to all the subscribed consumers.



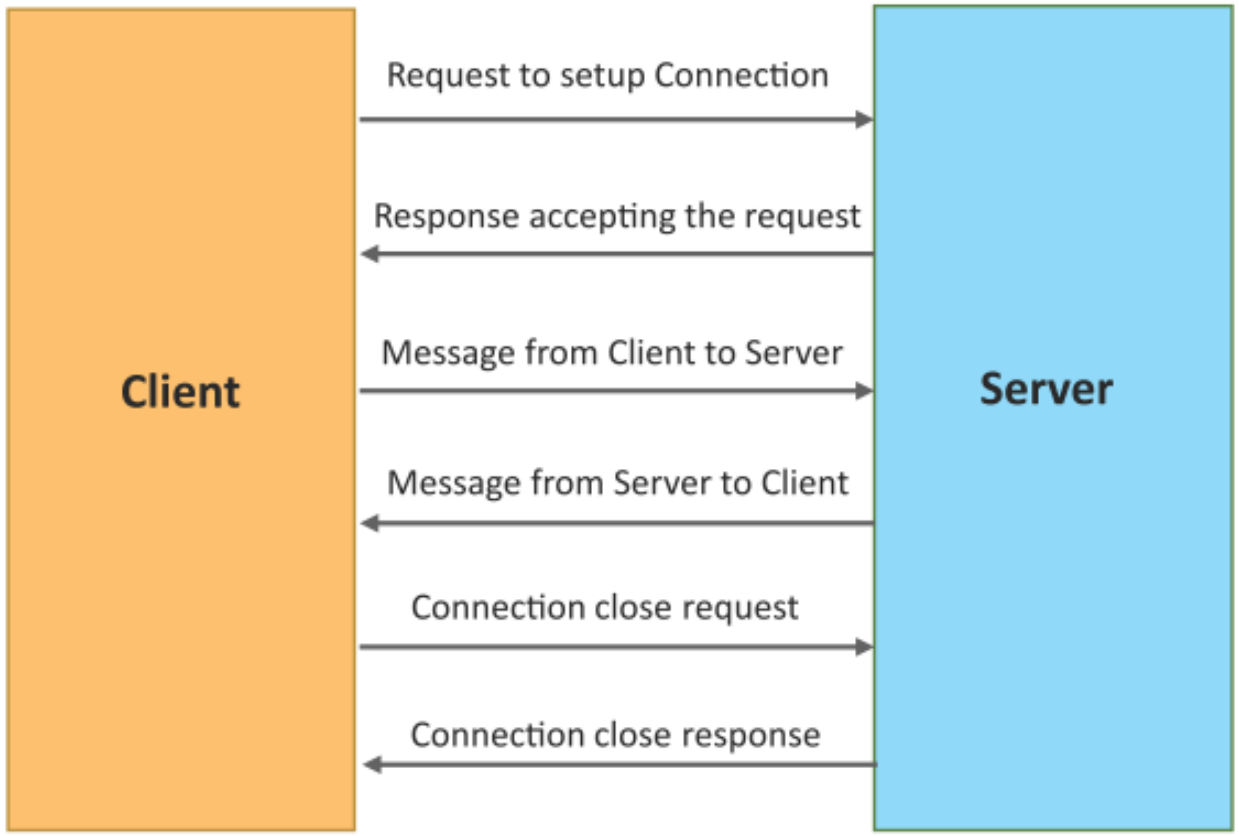
Push-Pull communication model

- Push-Pull is a communication model in which the data producers push the data to queues and the consumers pull the data from the queues. Producers do not need to be aware of the consumers.
- Queues help in decoupling the messaging between the producers and consumers.
- Queues also act as a buffer which helps in situations when there is a mismatch between the rate at which the producers push data and the rate rate at which the consumers pull data.



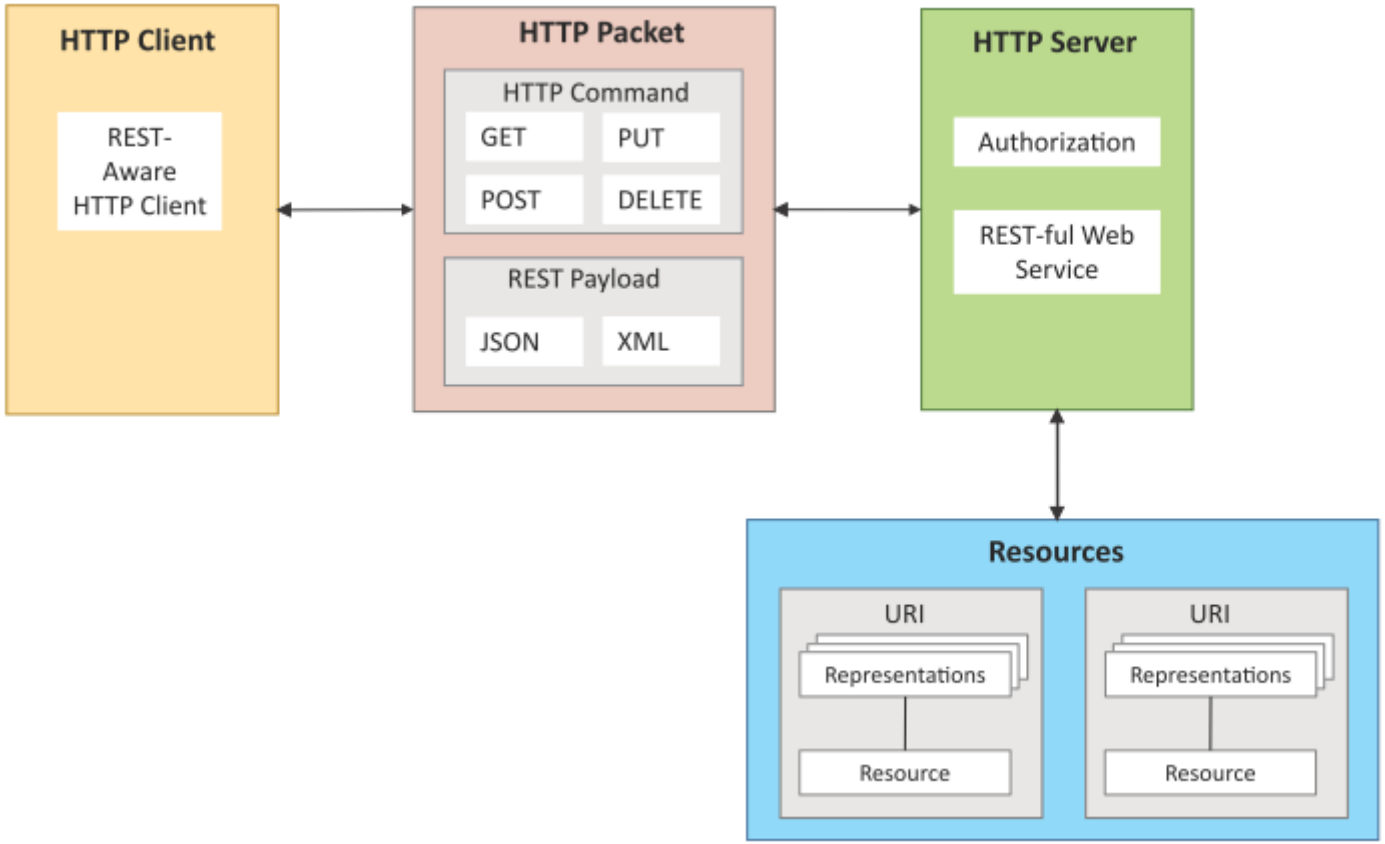
Exclusive Pair communication model

- Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and server.
- Once the connection is setup it remains open until the client sends a request to close the connection.
- Client and server can send messages to each other after connection setup.



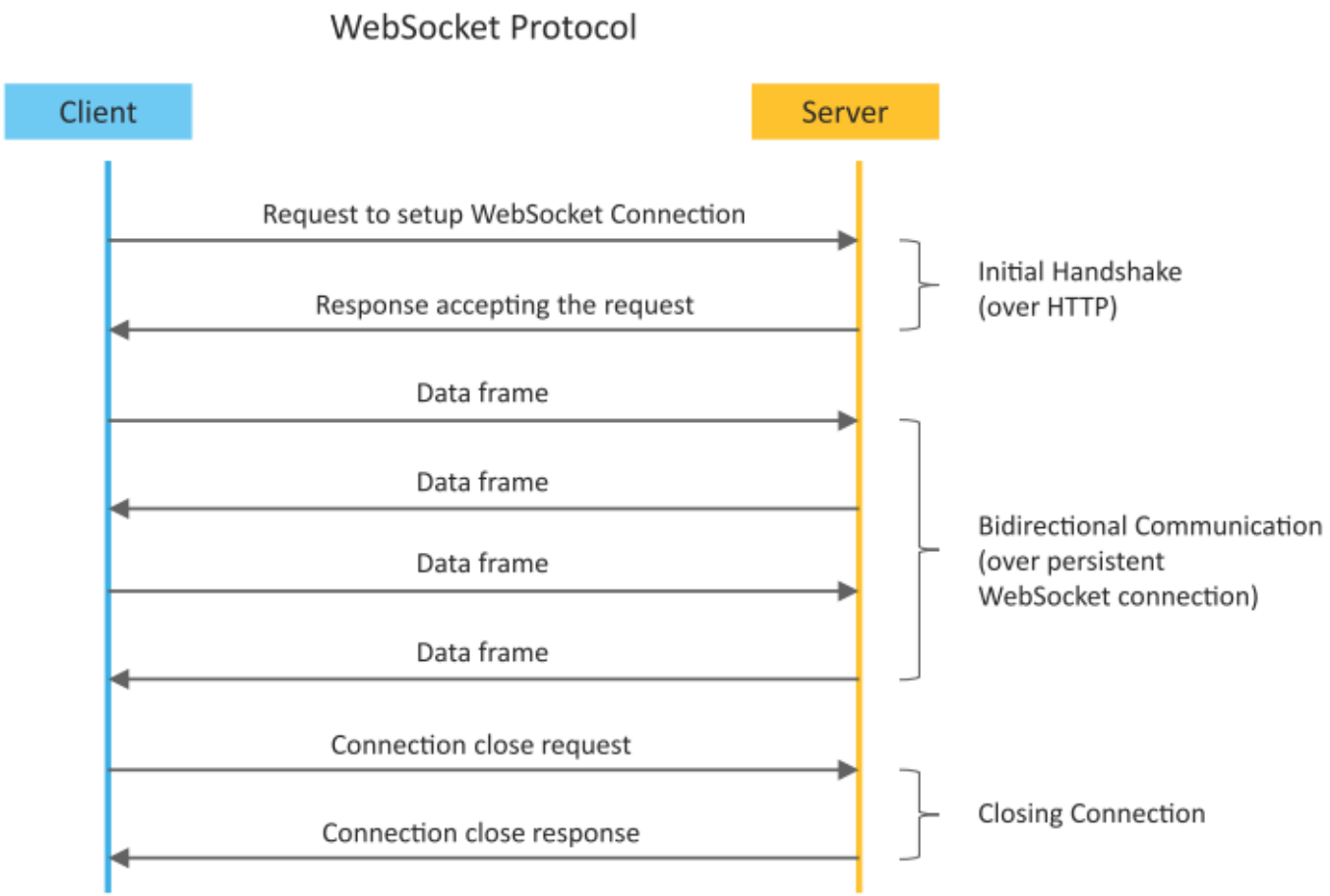
REST-based Communication APIs

- Representational State Transfer (REST) is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred.
- REST APIs follow the request-response communication model.
- The REST architectural constraints apply to the components, connectors, and data elements, within a distributed hypermedia system.



WebSocket-based Communication APIs

- WebSocket APIs allow bi-directional, full duplex communication between clients and servers.
- WebSocket APIs follow the exclusive pair communication model



IoT Levels & Deployment Templates

An IoT system comprises of the following components:

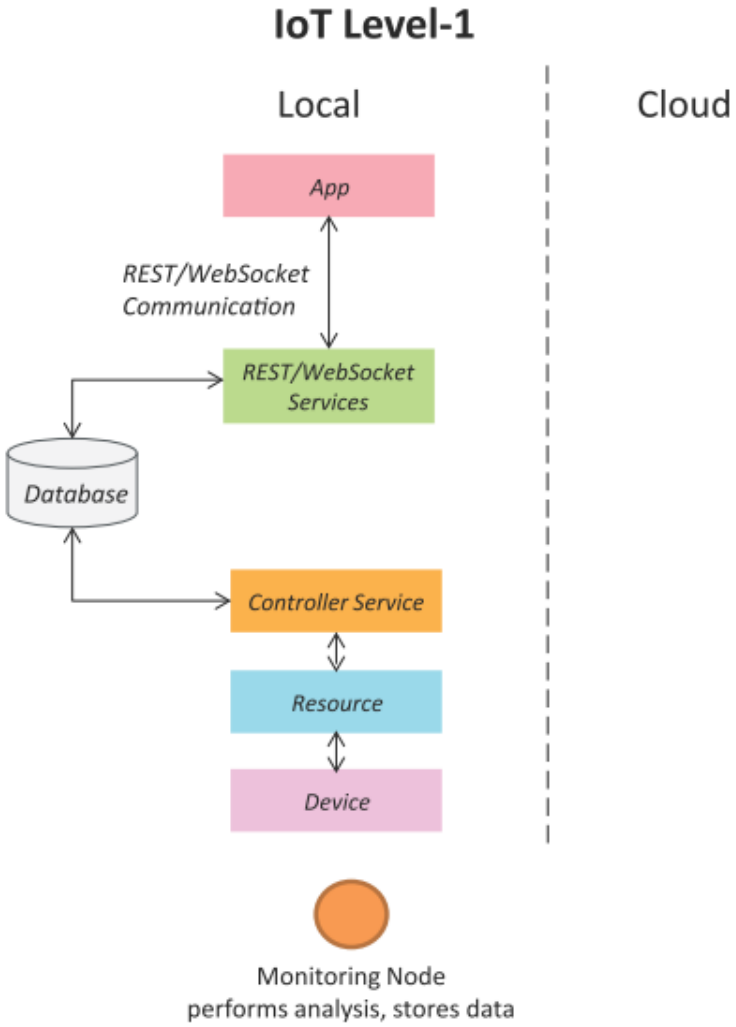
- **Device**: An IoT device allows identification, remote sensing, actuation and remote monitoring.
- **Resource**: Resources are software components on the IoT device for accessing, processing, and storing sensor information, or controlling actuators connected to the device. Resources also include the software components that enable network access for the device.
- **Controller Service**: Controller service is a native service that runs on the device and interacts with the web services. Controller service sends data from the device to the web service and receives commands from the application (via web services) for controlling the device.

IoT Levels & Deployment Templates

- **Database**: Database can be either local or in the cloud and stores the data generated by the IoT device.
- **Web Service**: Web services serve as a link between the IoT device, application, database and analysis components. Web service can be either implemented using HTTP and REST principles (REST service) or using WebSocket protocol (WebSocket service).
- **Analysis Component**: The Analysis Component is responsible for analyzing the IoT data and generate results in a form which are easy for the user to understand.
- **Application**: IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view the processed data.

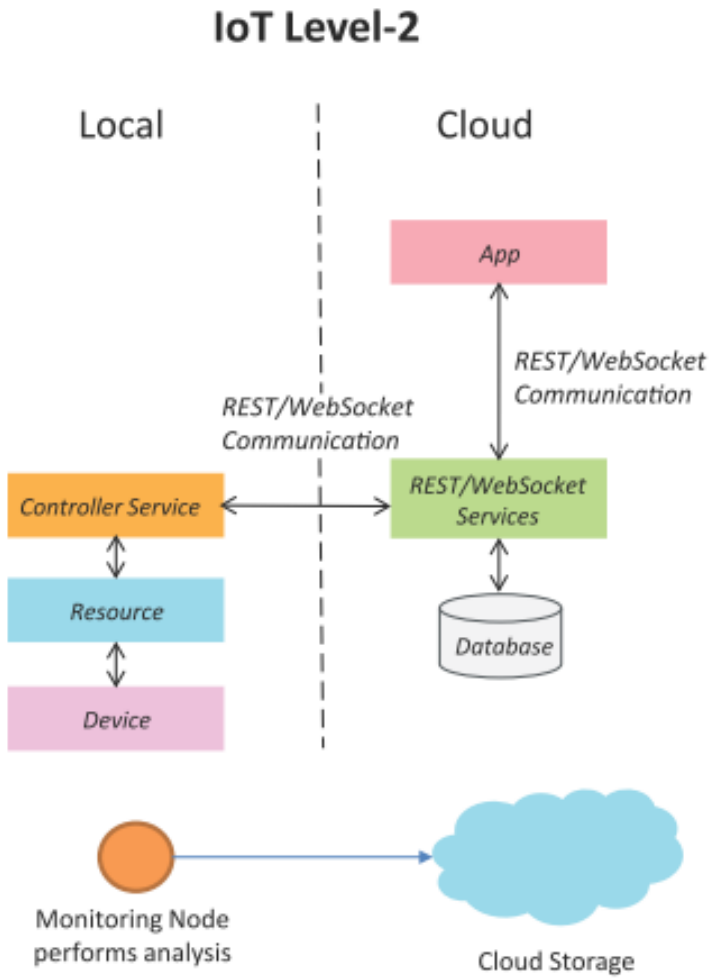
IoT Level-1

- A level-1 IoT system has a single node/device that performs sensing and/or actuation, stores data, performs analysis and hosts the application
- Level-1 IoT systems are suitable for modeling low-cost and low-complexity solutions where the data involved is not big and the analysis requirements are not computationally intensive.



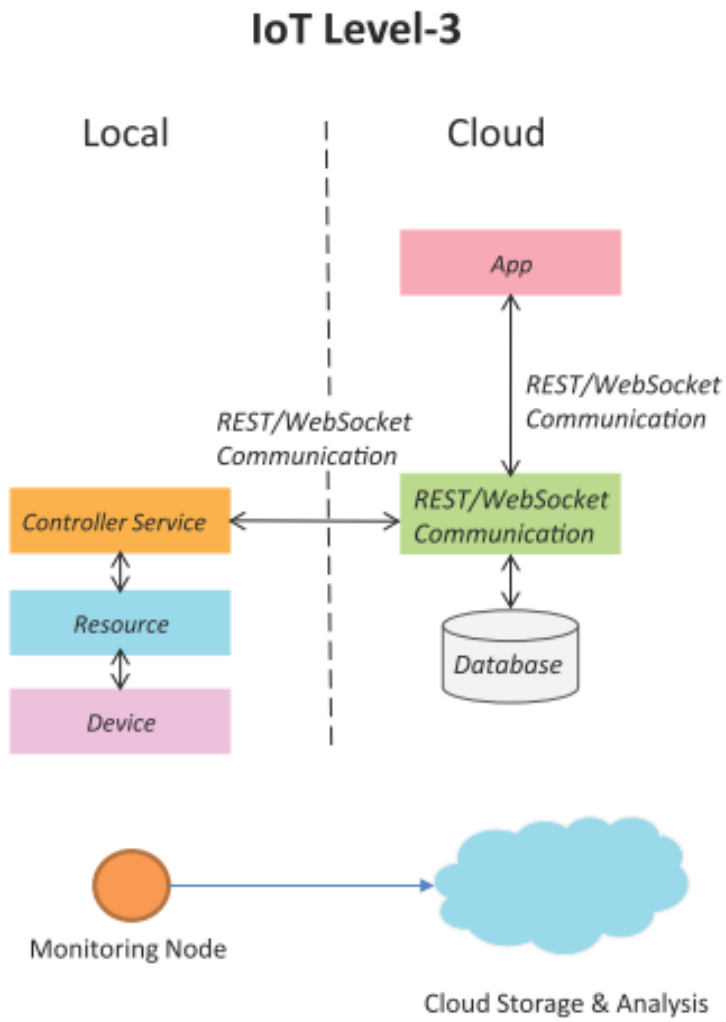
IoT Level-2

- A level-2 IoT system has a single node that performs sensing and/or actuation and local analysis.
- Data is stored in the cloud and application is usually cloud-based.
- Level-2 IoT systems are suitable for solutions where the data involved is big, however, the primary analysis requirement is not computationally intensive and can be done locally itself.



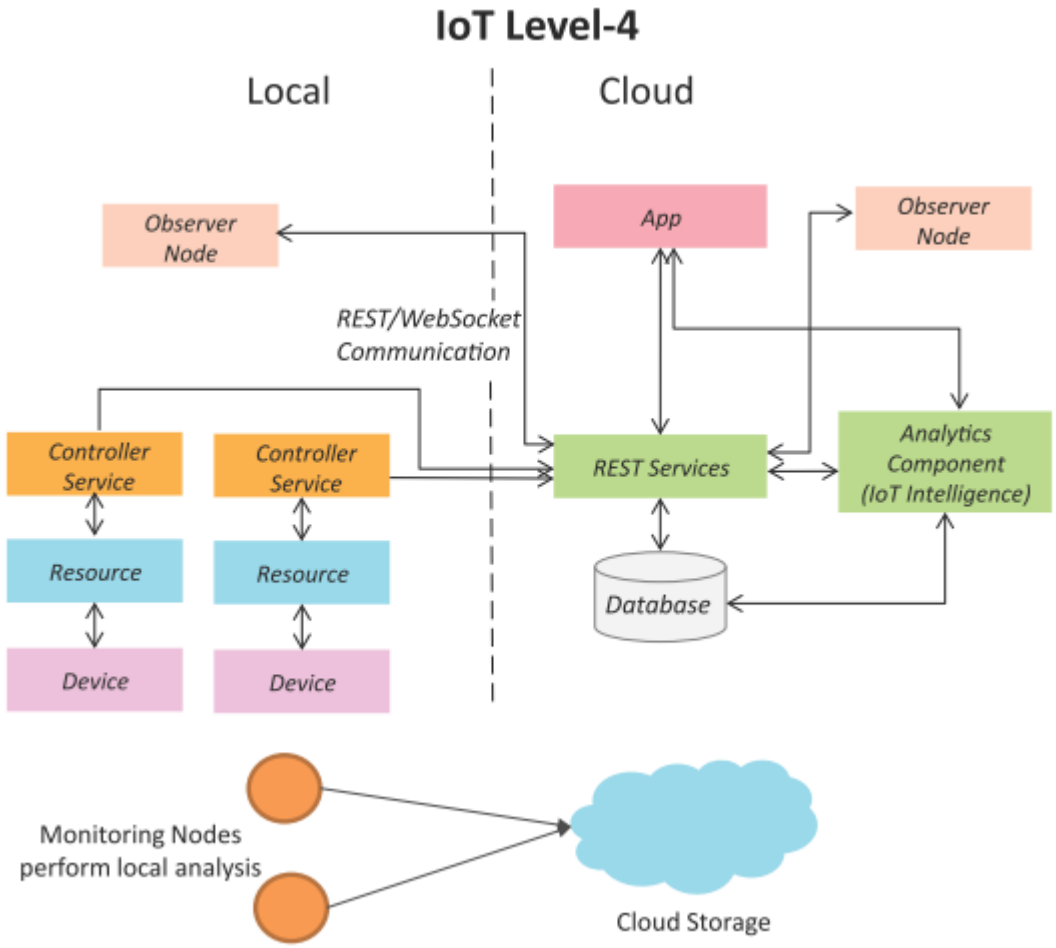
IoT Level-3

- A level-3 IoT system has a single node. Data is stored and analyzed in the cloud and application is cloud-based.
- Level-3 IoT systems are suitable for solutions where the data involved is big and the analysis requirements are computationally intensive.



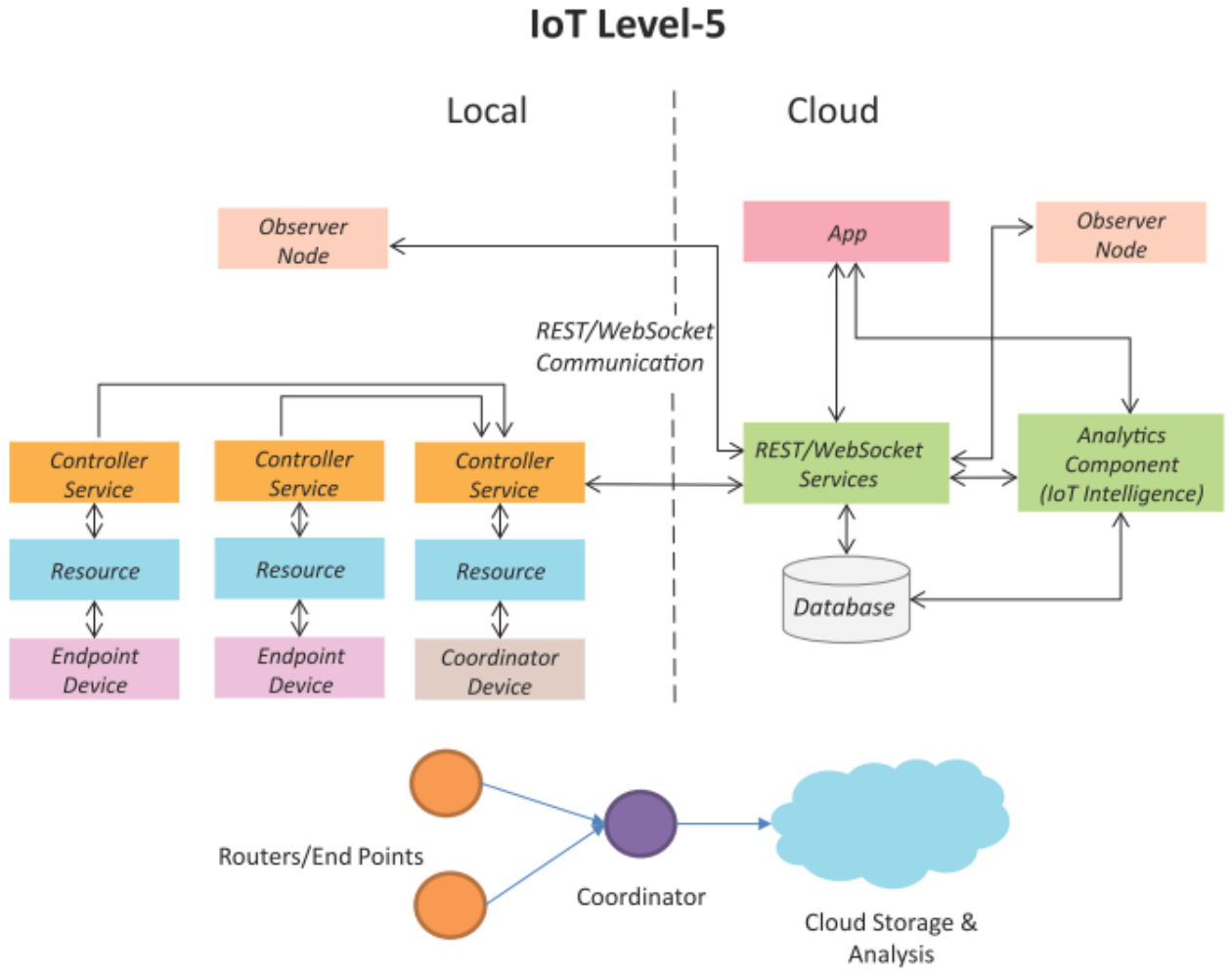
IoT Level-4

- A level-4 IoT system has multiple nodes that perform local analysis. Data is stored in the cloud and application is cloud-based.
- Level-4 contains local and cloud-based observer nodes which can subscribe to and receive information collected in the cloud from IoT devices.
- Level-4 IoT systems are suitable for solutions where multiple nodes are required, the data involved is big and the analysis requirements are computationally intensive.



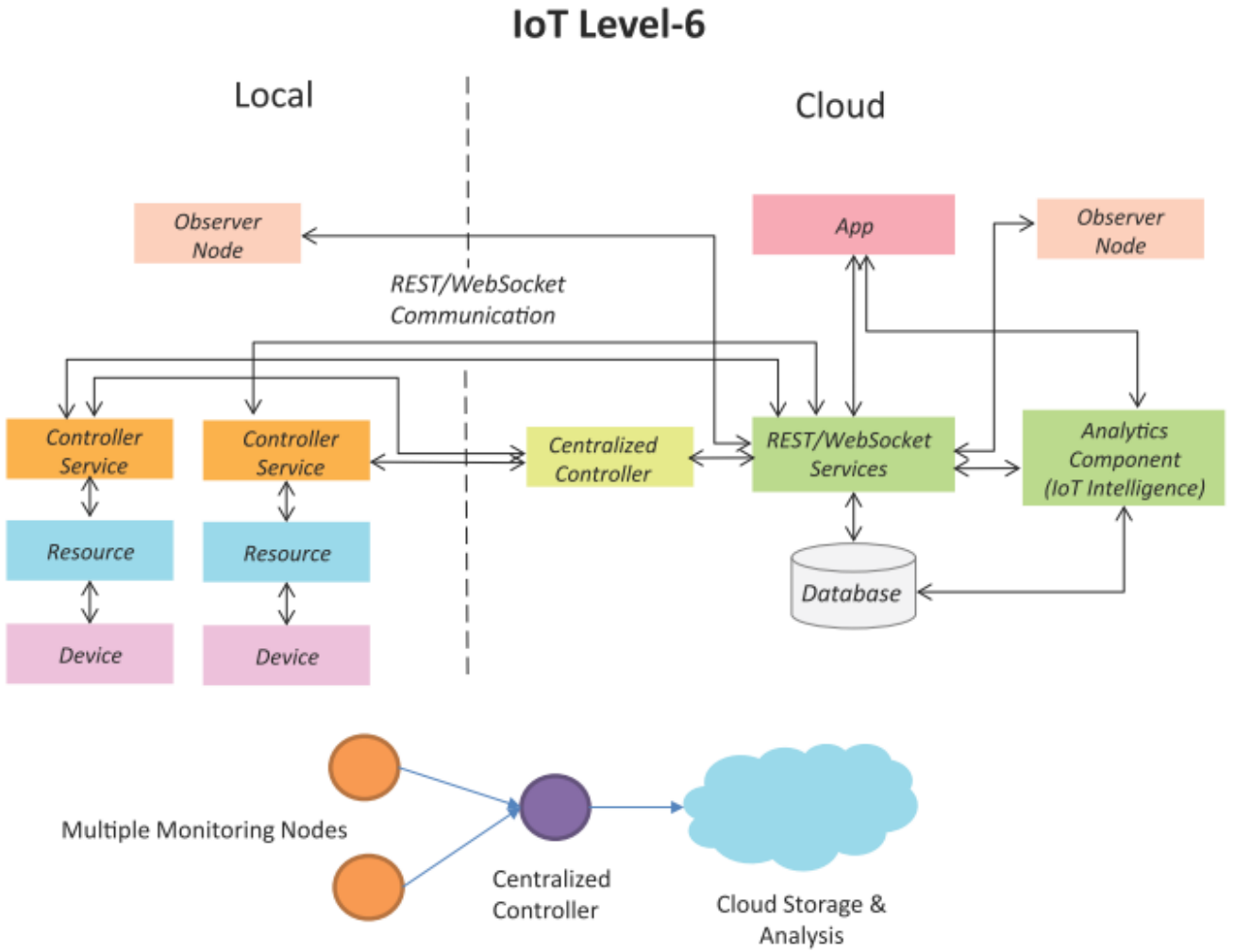
IoT Level-5

- A level-5 IoT system has multiple end nodes and one coordinator node.
- The end nodes that perform sensing and/or actuation.
- Coordinator node collects data from the end nodes and sends to the cloud.
- Data is stored and analyzed in the cloud and application is cloud-based.
- Level-5 IoT systems are suitable for solutions based on wireless sensor networks, in which the data involved is big and the analysis requirements are computationally intensive.



IoT Level-6

- A level-6 IoT system has multiple independent end nodes that perform sensing and/or actuation and send data to the cloud.
- Data is stored in the cloud and application is cloud-based.
- The analytics component analyzes the data and stores the results in the cloud database.
- The results are visualized with the cloud-based application.
- The centralized controller is aware of the status of all the end nodes and sends control commands to the nodes.



IoT Applications

- **Home Automation**
  - Smart Lighting
  - Smart Appliances
  - Intrusion Detection
  - Smoke/Gas Detectors
- **Cities**
  - Smart Parking
  - Smart Lighting
  - Smart Roads
  - Structural Health Monitoring
  - Surveillance
  - Emergency Response
- **Environment**
  - Weather Monitoring
  - Air Pollution Monitoring
  - Noise Pollution Monitoring
  - Forest Fire Detection
  - River Floods Detection
- **Energy**
  - Smart Grids
  - Renewable Energy Systems
  - Prognostics

IoT Applications

- **Retail**
  - Inventory Management
  - Smart Payments
  - Smart Vending Machines
- **Logistics**
  - Route Generation & Scheduling
  - Fleet Tracking
  - Shipment Monitoring
  - Remote Vehicle Diagnostics
- **Agriculture**
  - Smart Irrigation
  - Green House Control
- **Industry**
  - Machine Diagnosis & Prognosis
  - Indoor Air Quality Monitoring
- **Health & Lifestyle**
  - Health & Fitness Monitoring
  - Wearable Electronics

IoT Device

- A Device or "Thing" in Internet of Things (IoT) can be any object that has a unique identifier and which can send/receive data (including user data) over a network (such as a smartphone, smart TV, computer, refrigerator or car).

- IoT devices are connected to the Internet and send information about themselves or about their surroundings (e.g. information sensed by the connected sensors) over a network (to other devices or servers/storage) or allow actuation upon the physical entities/environment around them remotely.

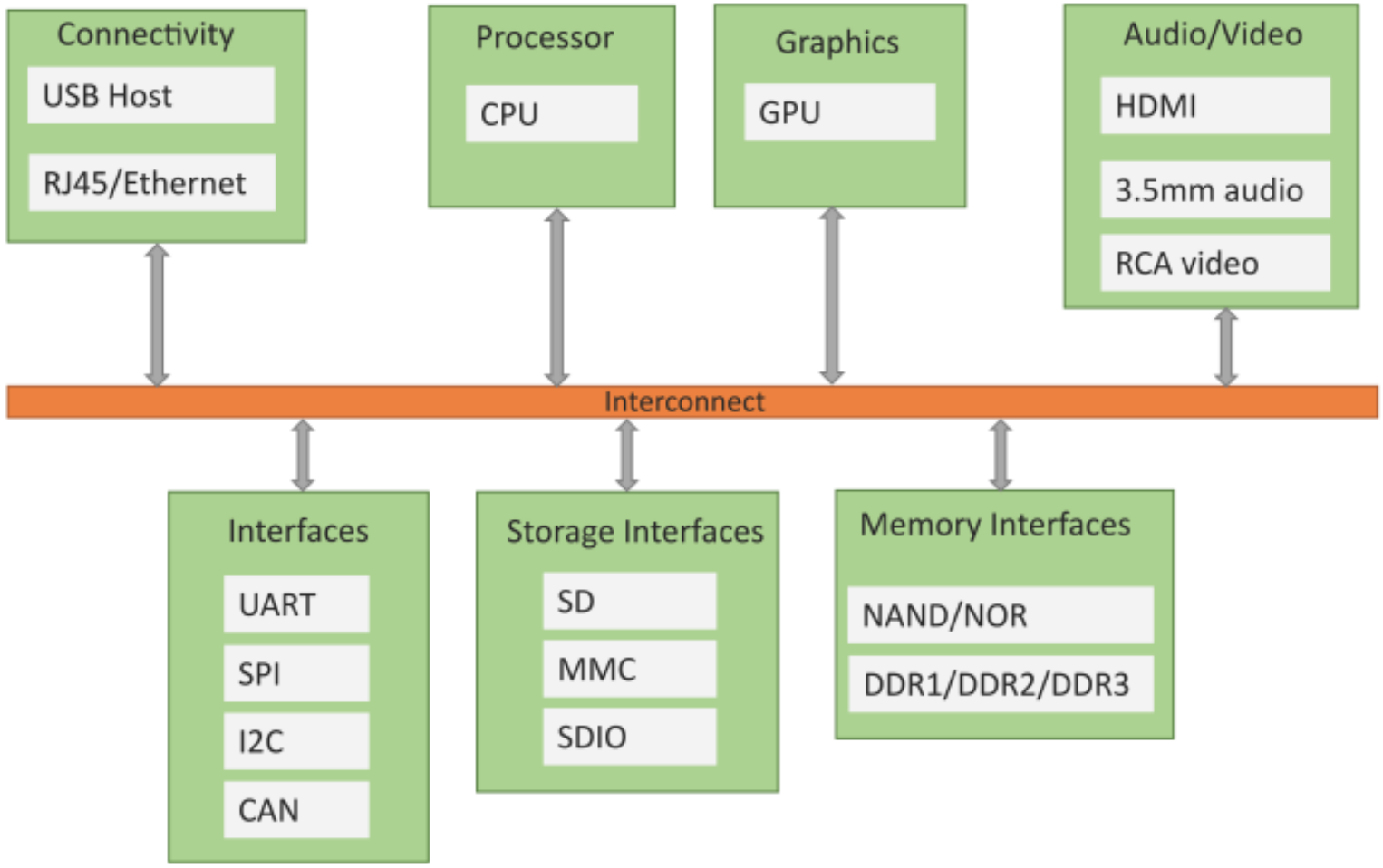
IoT Device Examples

- A home automation device that allows remotely monitoring the status of appliances and controlling the appliances.
- An industrial machine which sends information about its operation and health monitoring data to a server.
- A car which sends information about its location to a cloud-based service.
- A wireless-enabled wearable device that measures data about a person such as the number of steps walked and sends the data to a cloud-based service.

Basic building blocks of an IoT Device

- **Sensors**
  - Sensors can be either on-board the IoT device or attached to the device.
- **Actuators**
  - IoT devices can have various types of actuators attached that allow taking actions upon the physical entities in the vicinity of the device.
- **Communication**
  - Communication modules are responsible for sending collected data to other devices or cloud-based servers/storage and receiving data from other devices and commands from remote applications.
- **Analysis & Processing**
  - Analysis and processing modules are responsible for making sense of the collected data.

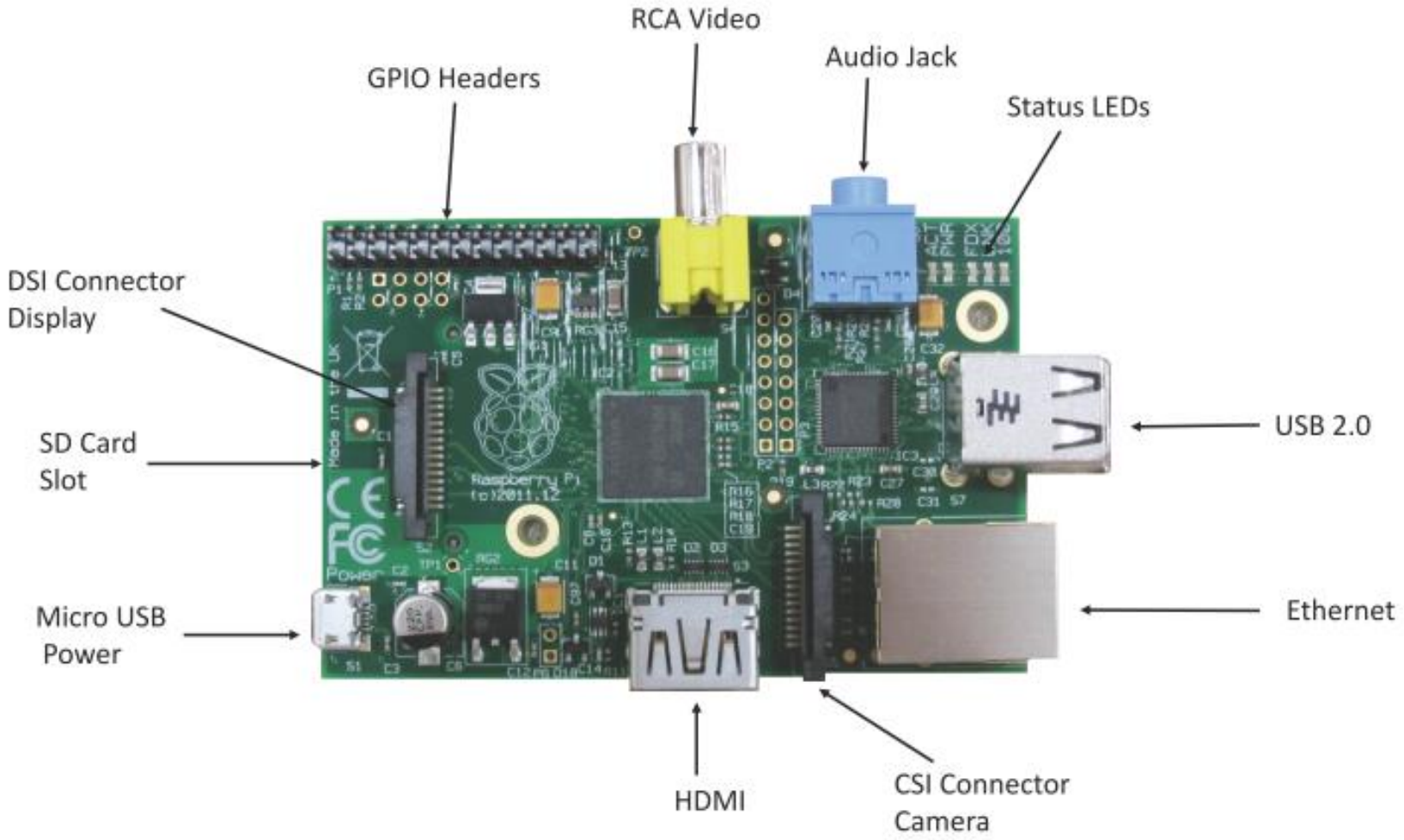
Block diagram of an IoT Device



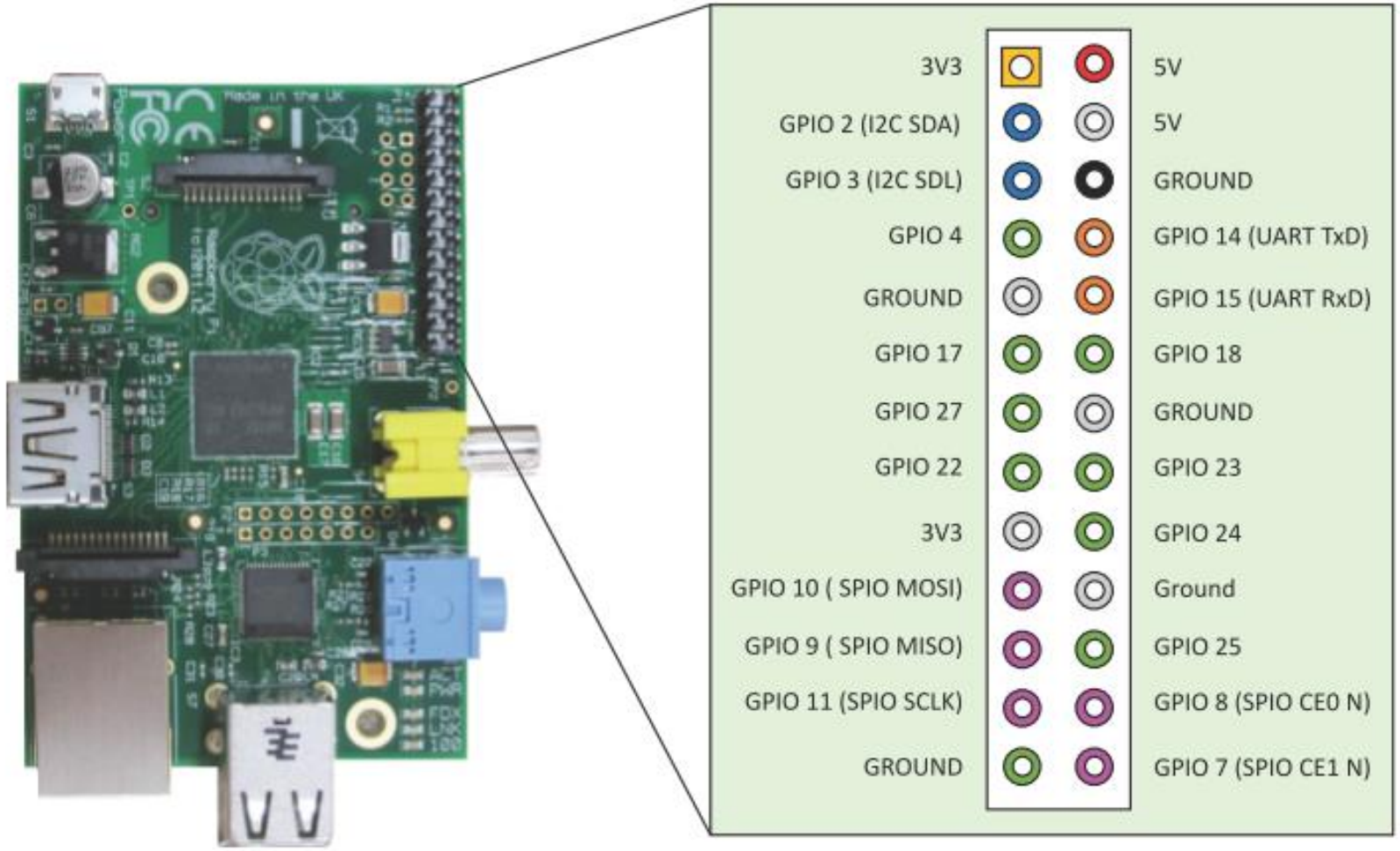
Exemplary Device: Raspberry Pi

- Raspberry Pi is a low-cost mini-computer with the physical size of a credit card.
- Raspberry Pi runs various flavors of Linux and can perform almost all tasks that a normal desktop computer can do.
- Raspberry Pi also allows interfacing sensors and actuators through the general purpose I/O pins.
- Since Raspberry Pi runs Linux operating system, it supports Python "out of the box".

Raspberry Pi



Raspberry Pi GPIO

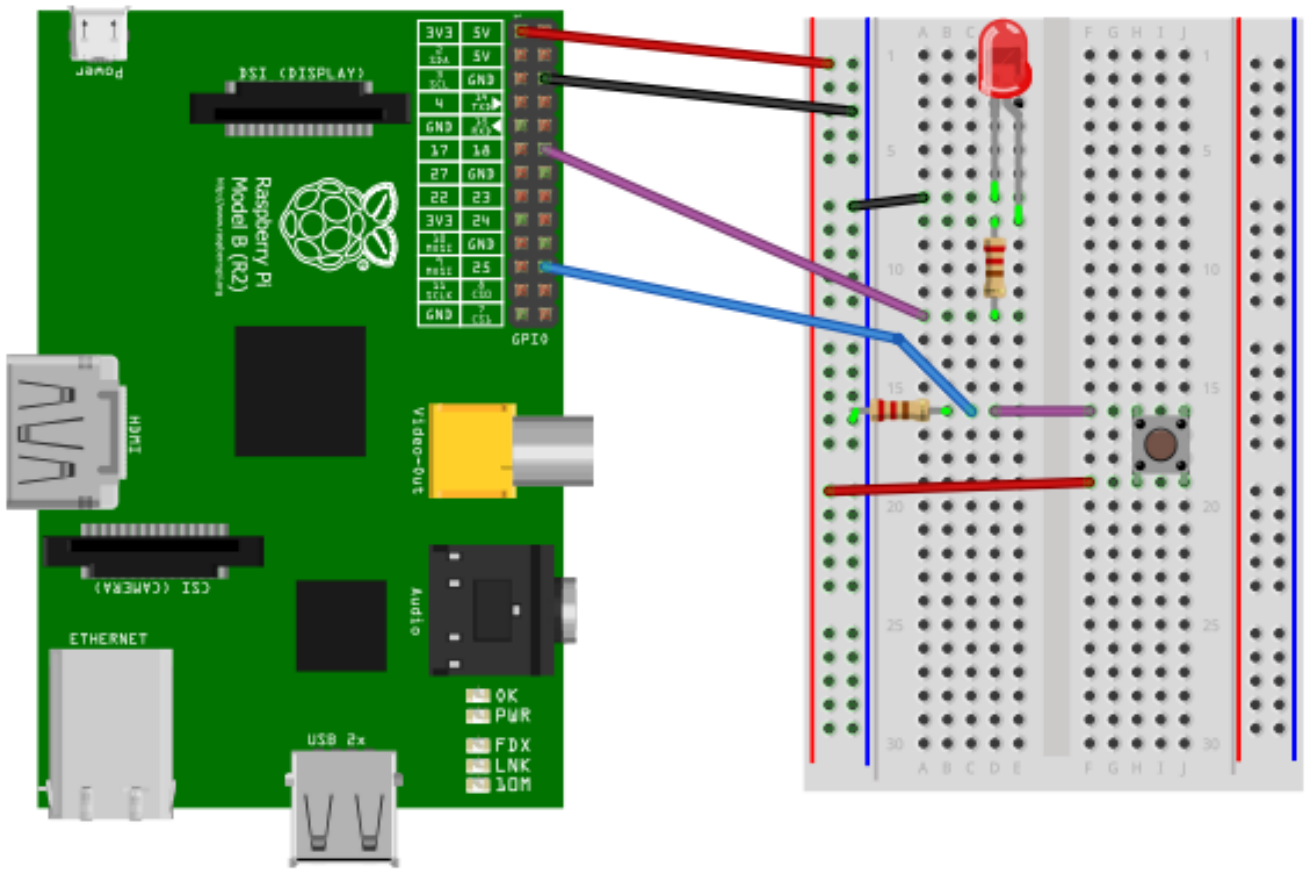


Raspberry Pi Interfaces

- Serial
  - The serial interface on Raspberry Pi has receive (Rx) and transmit (Tx) pins for communication with serial peripherals.
- SPI
  - Serial Peripheral Interface (SPI) is a synchronous serial data protocol used for communicating with one or more peripheral devices.
- I2C
  - The I2C interface pins on Raspberry Pi allow you to connect hardware modules. I2C interface allows synchronous data transfer with just two pins - SDA (data line) and SCL (clock line).

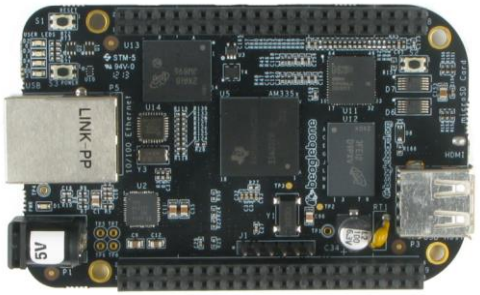
Raspberry Pi Example: Interfacing LED and switch with Raspberry Pi

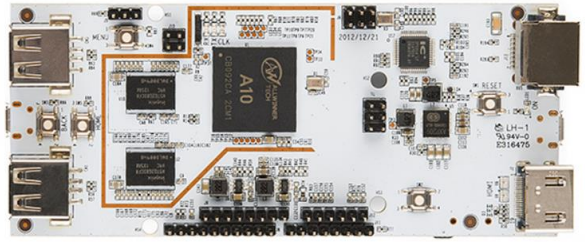
```
from  time  import  sleep
import  RPi.GPIO  as  GPIO
GPIO.setmode(GPIO.BCM)

#Switch  Pin
GPIO.setup(25,  GPIO.IN)

#LED  Pin
GPIO.setup(18,  GPIO.OUT)
state=false

def  toggleLED(pin):
    state  =  not  state
    GPIO.output(pin,  state)

while  True:
    try:
        if  (GPIO.input(25)  ==  True):
            toggleLED(pin)
        sleep(.01)
        except  KeyboardInterrupt:
            exit()
```



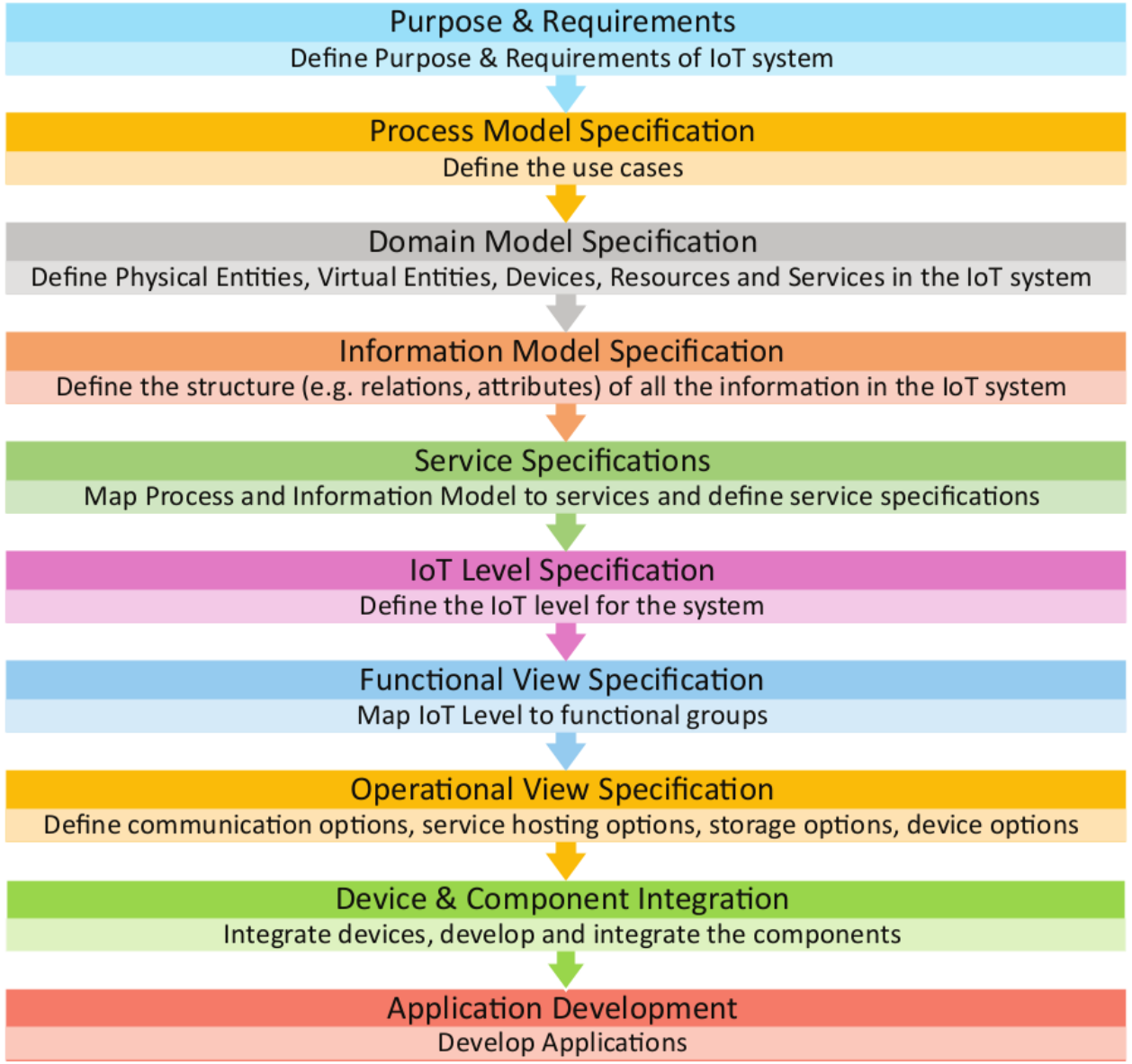
Other Devices

- BeagleBone Black
- pcDuino
- Cubieboard





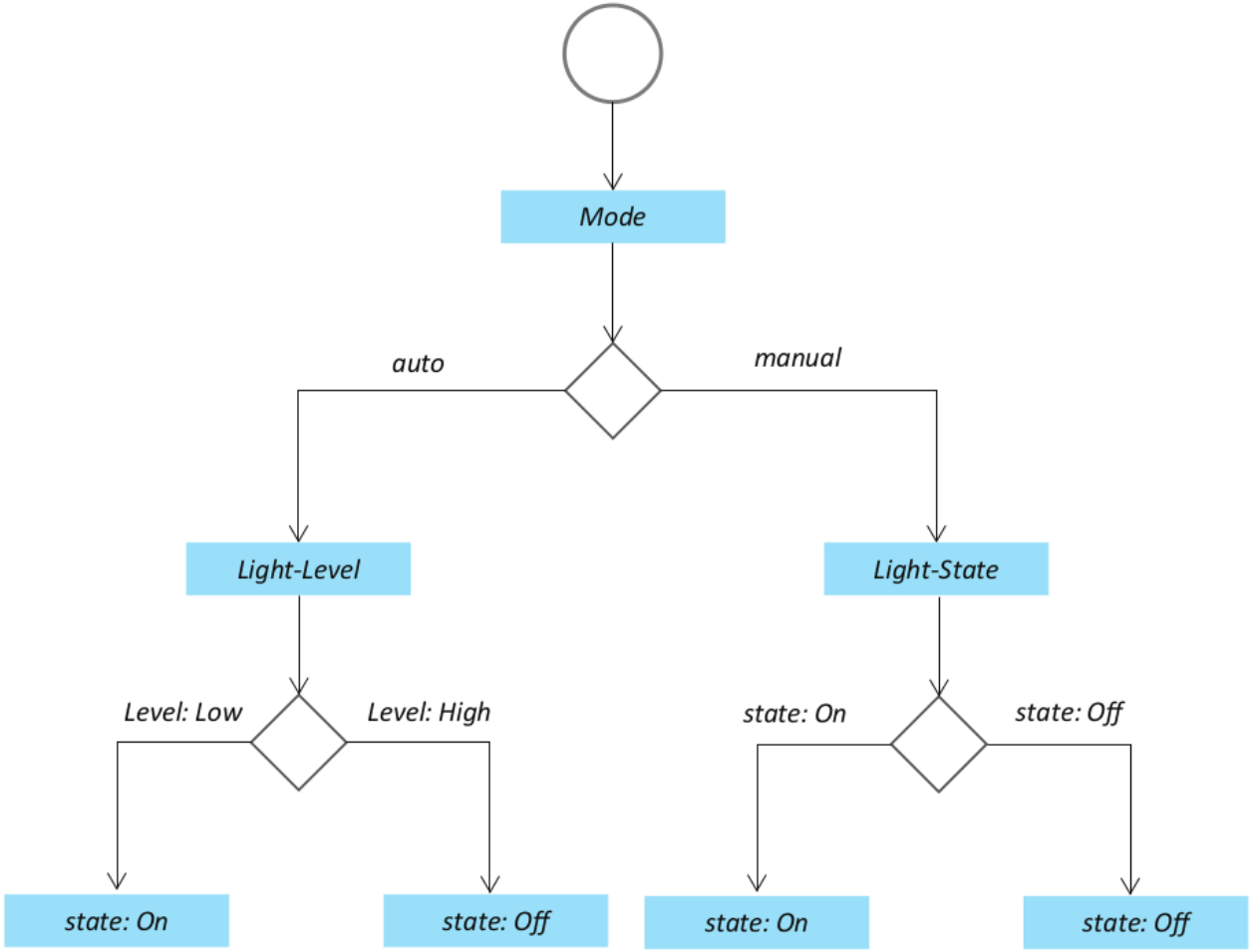
IoT Design Methodology



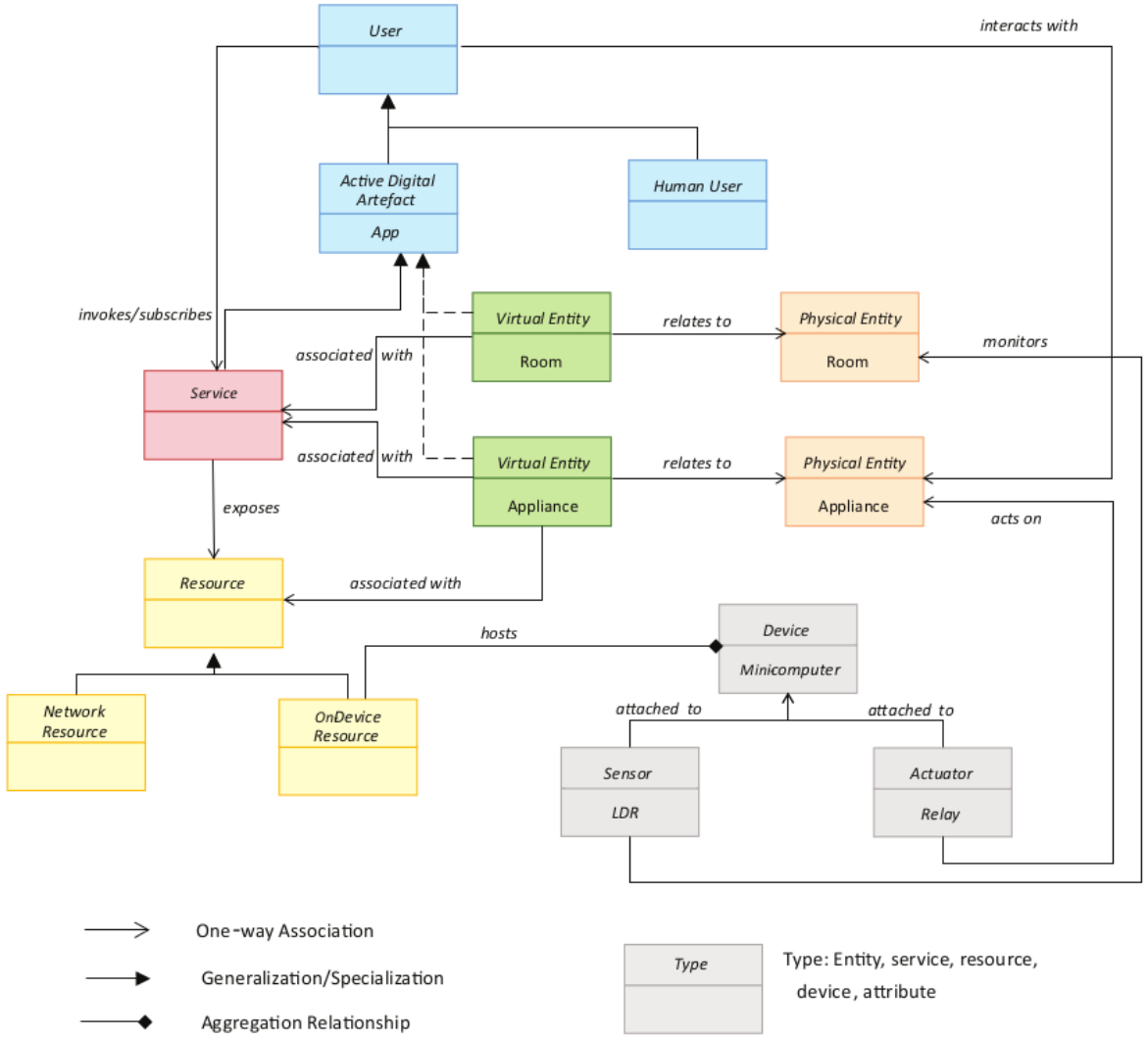
Step:1 - Purpose & Requirements

- Applying this to our example of a smart home automation system, the purpose and requirements for the system may be described as follows:
  - **Purpose** : A home automation system that allows controlling of the lights in a home remotely using a web application.
  - **Behavior** : The home automation system should have auto and manual modes. In auto mode, the system measures the light level in the room and switches on the light when it gets dark. In manual mode, the system provides the option of manually and remotely switching on/off the light.
  - **System Management Requirement** : The system should provide remote monitoring and control functions.
  - **Data Analysis Requirement** : The system should perform local analysis of the data.
  - **Application Deployment Requirement** : The application should be deployed locally on the device, but should be accessible remotely.
  - **Security Requirement** : The system should have basic user authentication capability.

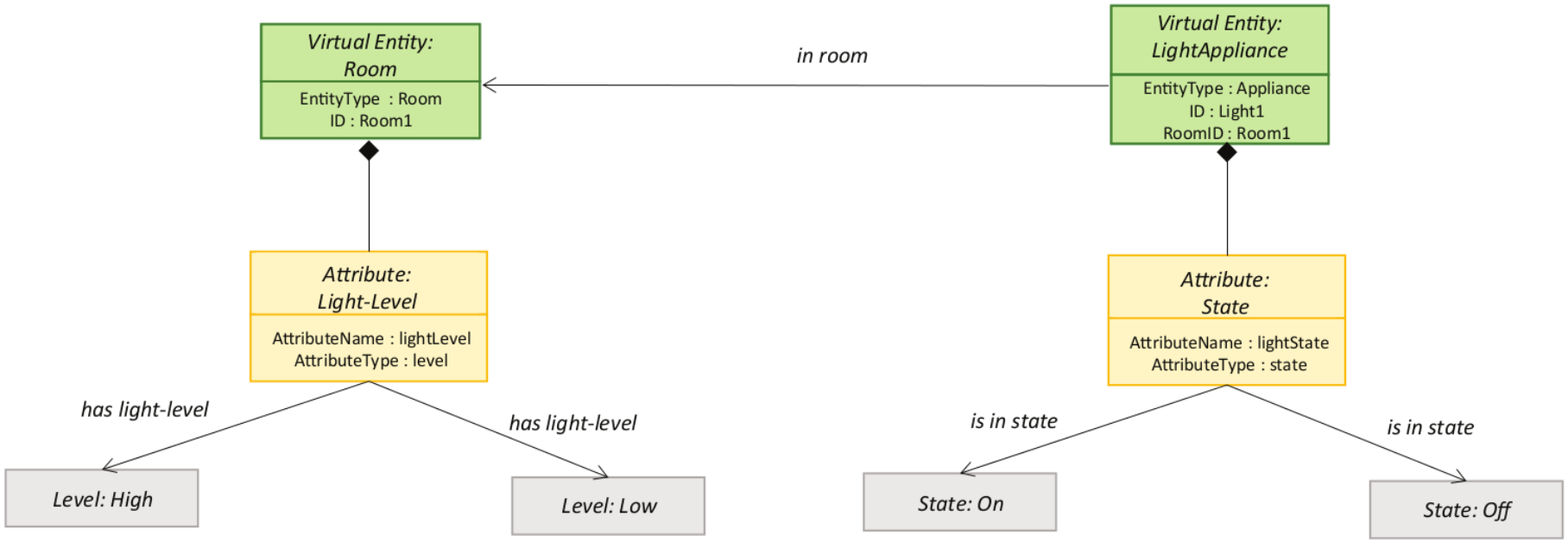
Step:2 - Process Specification



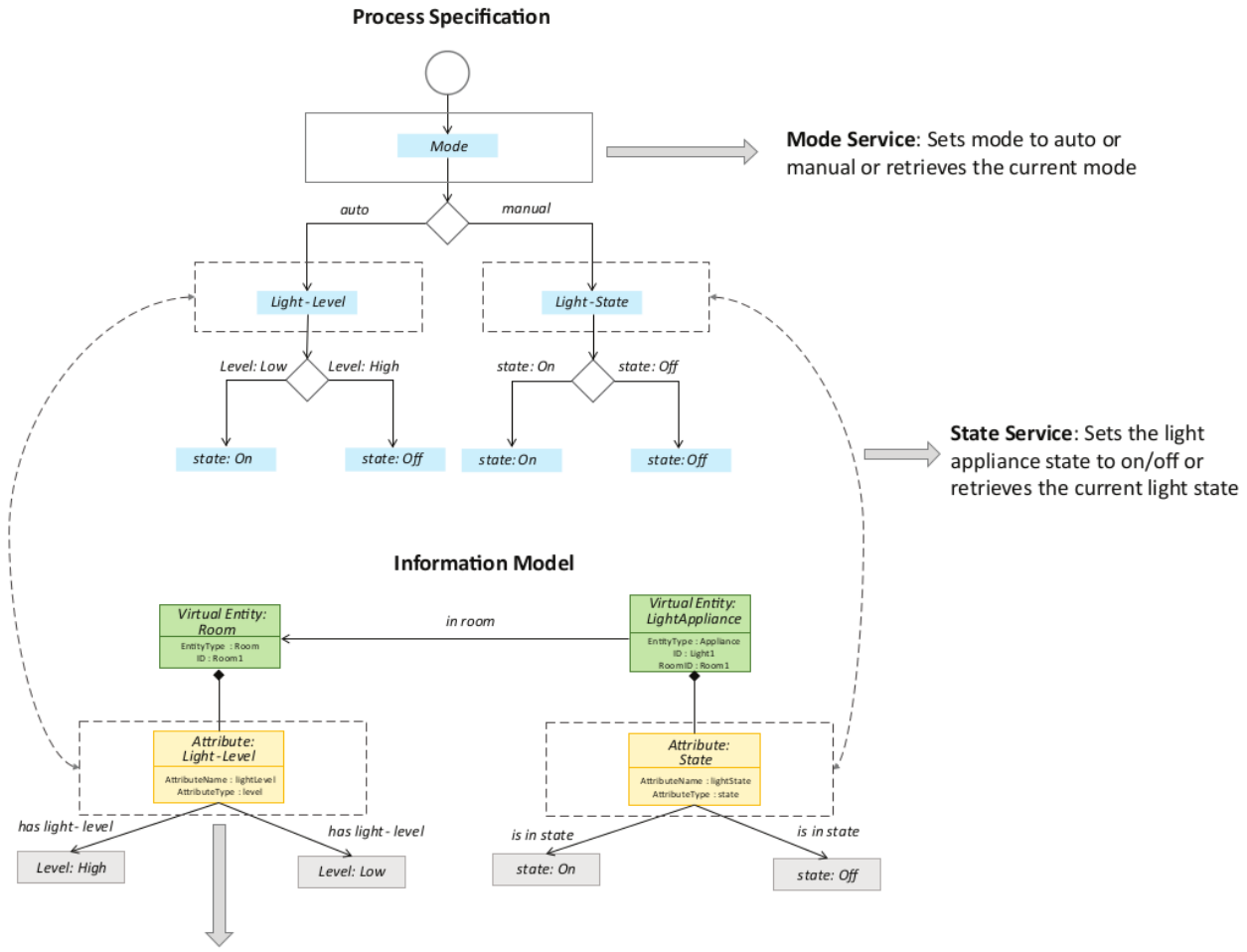
Step 3: Domain Model Specification



Step 4: Information Model Specification



Step 5: Service Specifications

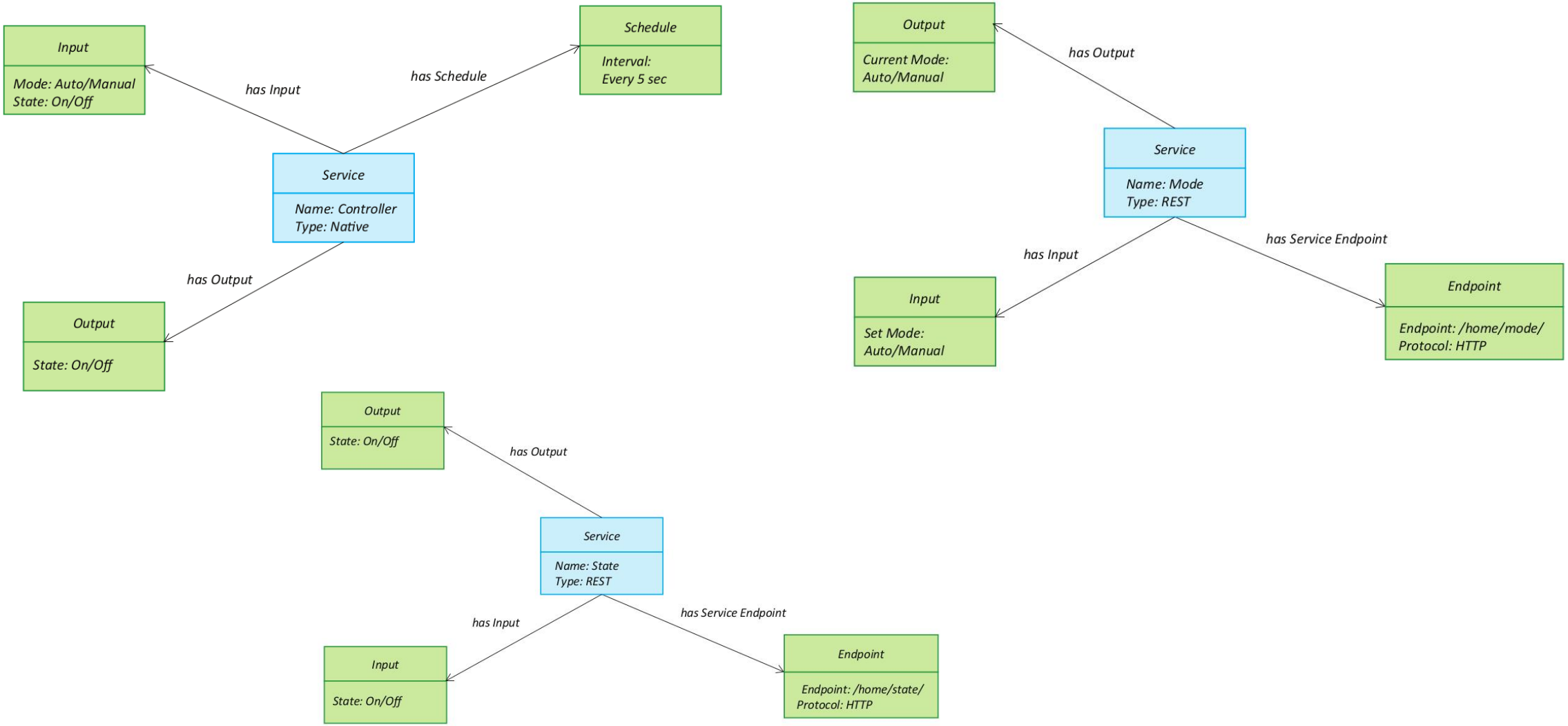


Mode Service: Sets mode to auto or manual or retrieves the current mode

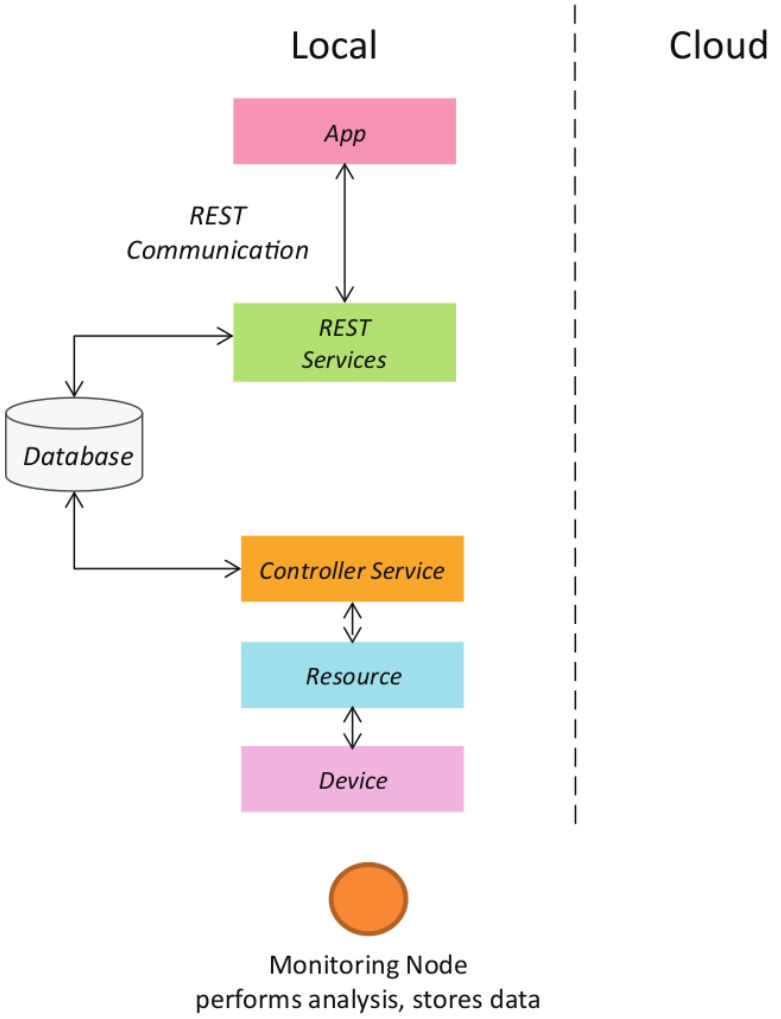
State Service: Sets the light appliance state to on/off or retrieves the current light state

Controller Service: In auto mode, the controller service monitors the light level and switches the light on/off and updates the status in the status database.
In manual mode, the controller service, retrieves the current state from the database and switches the light on/off.

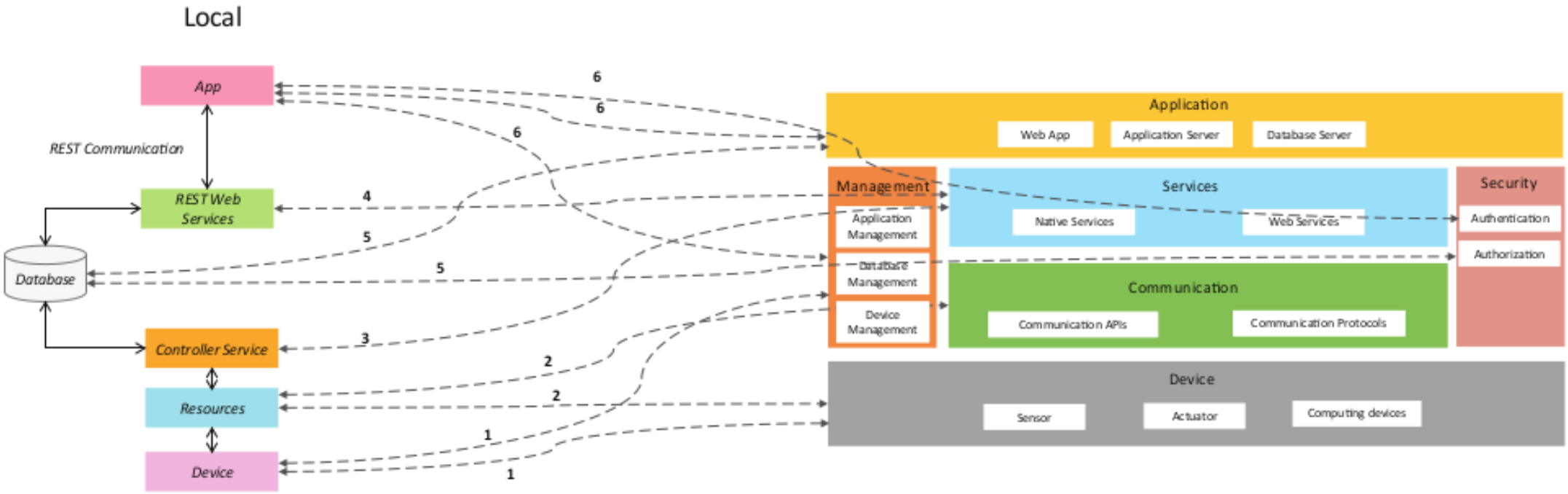
Step 5: Service Specifications



Step 6: IoT Level Specification



Step 7: Functional View Specification



1. IoT device maps to the Device FG (sensors, actuators devices, computing devices) and the Management FG (device management)

2. Resources map to the Device FG (on-device resource) and Communication FG (communication APIs and protocols)

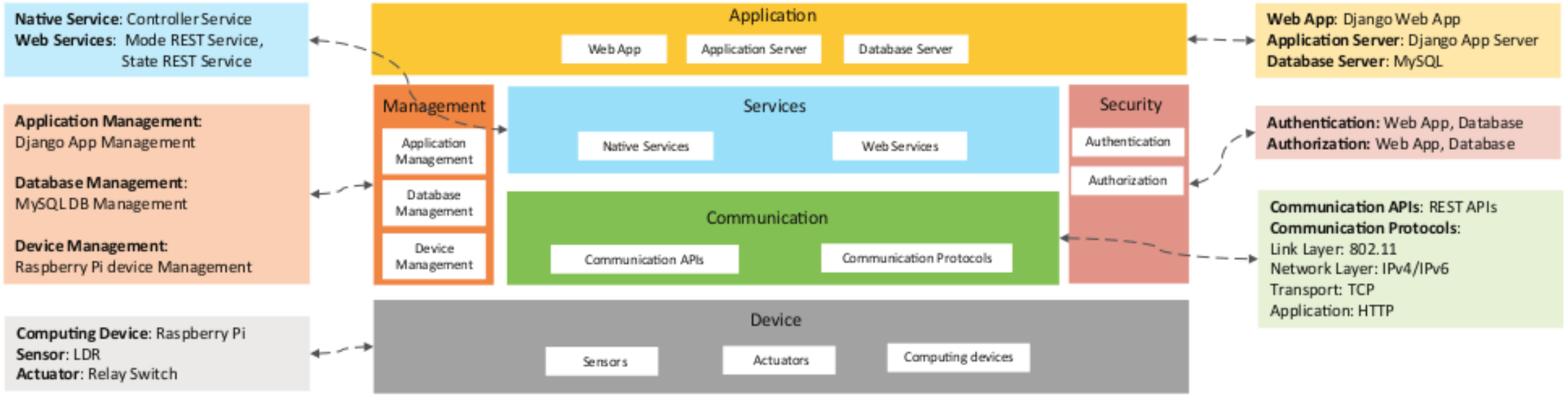
3. Controller service maps to the Services FG (native service). Web Services map to Services FG (web services)

4. Web Services map to Services FG (web services)

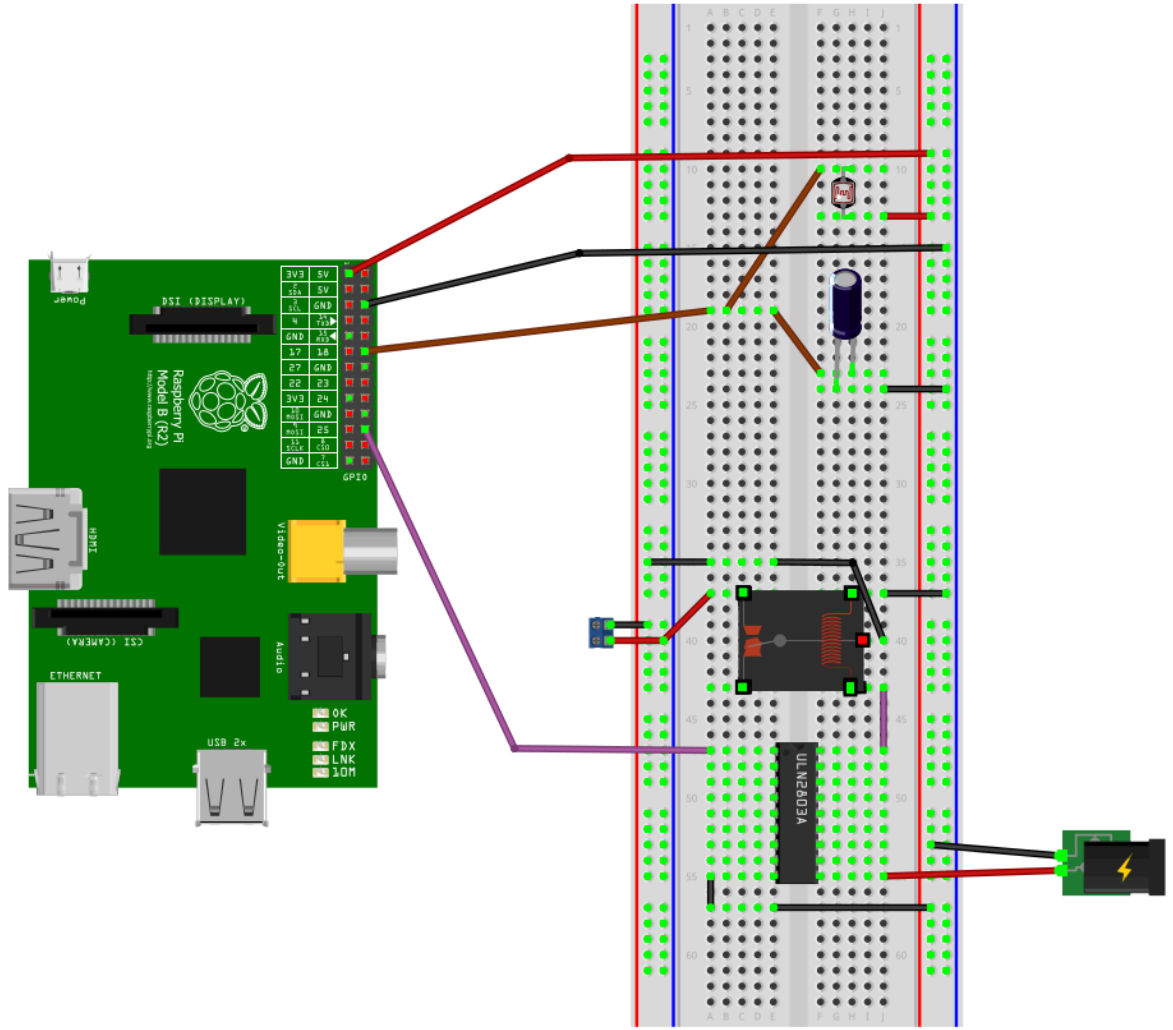
5. Database maps to the Management FG (database management) and Security FG (database security)

6. Application maps to the Application FG (web application, application and database servers), Management FG (app management) and Security FG (app security)

Step 8: Operational View Specification



Step 9: Device & Component Integration

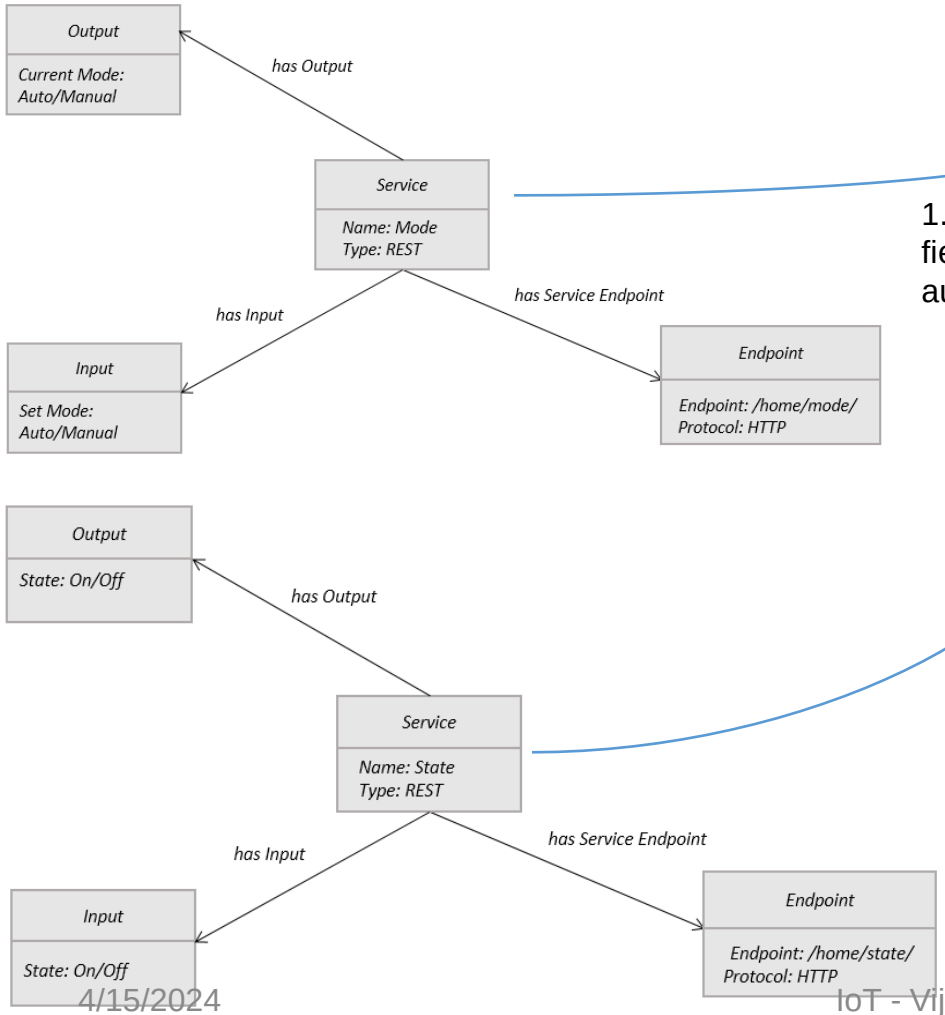


Step 10: Application Development

- Auto
  - Controls the light appliance automatically based on the lighting conditions in the room
- Light
  - When Auto mode is off, it is used for manually controlling the light appliance.
  - When Auto mode is on, it reflects the current state of the light appliance.

Implementation: RESTful Web Services

REST services implemented with Django REST Framework



1. Map services to models. Model fields store the states (on/off, auto/manual)

```
# Models – models.py
from django.db import models

class Mode(models.Model):
        name = models.CharField(max_length=50)

class State(models.Model):
        name = models.CharField(max_length=50)
```

2. Write Model serializers. Serializers allow complex data (such as model instances) to be converted to native Python datatypes that can then be easily rendered into JSON, XML or other content types.

```
# Serializers – serializers.py
from myapp.models import Mode, State
from rest_framework import serializers

class ModeSerializer(serializers.HyperlinkedModelSerializer):
        class Meta:
                model = Mode
                fields = ('url', 'name')

class StateSerializer(serializers.HyperlinkedModelSerializer):
        class Meta:
                model = State
                fields = ('url', 'name')
```

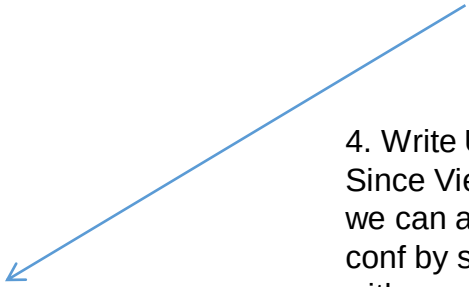
Implementation: RESTful Web Services

```
# Models – models.py
from django.db import models

class Mode(models.Model):
        name = models.CharField(max_length=50)

class State(models.Model):
        name = models.CharField(max_length=50)
```

3. Write ViewSets for the Models which combine the logic for a set of related views in a single class.

```
# Views – views.py
from myapp.models import Mode, State
from rest_framework import viewsets
from myapp.serializers import ModeSerializer,
StateSerializer

class ModeViewSet(viewsets.ModelViewSet):
        queryset = Mode.objects.all()
        serializer_class = ModeSerializer

class StateViewSet(viewsets.ModelViewSet):
        queryset = State.objects.all()
        serializer_class = StateSerializer
```

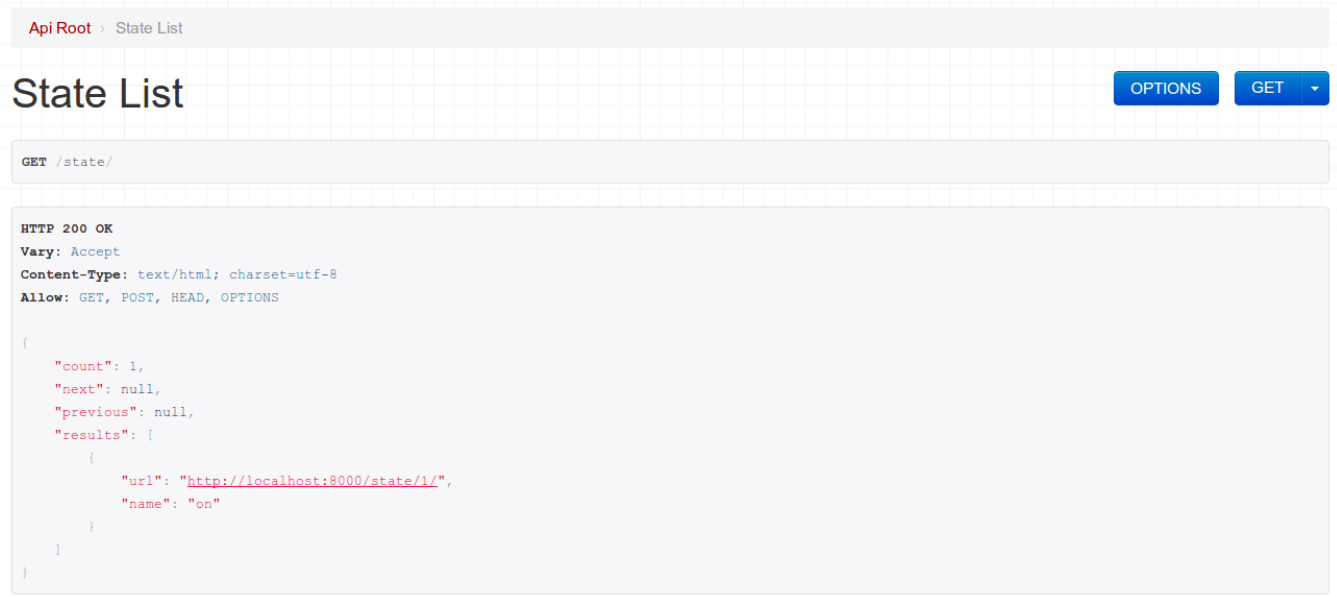
```
# URL Patterns – urls.py
from django.conf.urls import patterns, include, url
from django.contrib import admin
from rest_framework import routers
from myapp import views
admin.autodiscover()
router = routers.DefaultRouter()
router.register(r'mode', views.ModeViewSet)
router.register(r'state', views.StateViewSet)
urlpatterns = patterns('',
    url(r'^', include(router.urls)),
    url(r'^api-auth/', include('rest_framework.urls', namespace='rest_framework')),
    url(r'^admin/', include(admin.site.urls)),
    url(r'^home/', 'myapp.views.home'),
)
```



4. Write URL patterns for the services. Since ViewSets are used instead of views, we can automatically generate the URL conf 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.

Implementation: RESTful Web Services

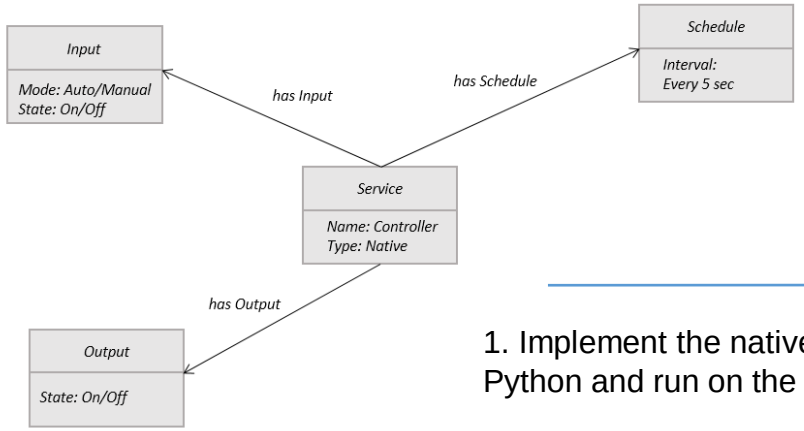
Screenshot of browsable State REST API



Screenshot of browsable Mode REST API

Implementation: Controller Native Service

Native service deployed locally



1. Implement the native service in Python and run on the device

```
#Controller service
import RPi.GPIO as GPIO
import time
import sqlite3 as lite
import sys

con = lite.connect('database.sqlite')
cur = con.cursor()

GPIO.setmode(GPIO.BCM)
threshold = 1000
LDR_PIN = 18
LIGHT_PIN = 25

def readldr(PIN):
        reading=0
        GPIO.setup(PIN, GPIO.OUT)
        GPIO.output(PIN, GPIO.LOW)
        time.sleep(0.1)
        GPIO.setup(PIN, GPIO.IN)
        while (GPIO.input(PIN)==GPIO.LOW):
                reading=reading+1
        return reading

def switchOnLight(PIN):
        GPIO.setup(PIN, GPIO.OUT)
        GPIO.output(PIN, GPIO.HIGH)

def switchOffLight(PIN):
        GPIO.setup(PIN, GPIO.OUT)
        GPIO.output(PIN, GPIO.LOW)
```

```
def runAutoMode():
        ldr_reading = readldr(LDR_PIN)
        if ldr_reading < threshold:
                switchOnLight(LIGHT_PIN)
                setCurrentState('on')
        else:
                switchOffLight(LIGHT_PIN)
                setCurrentState('off')

def runManualMode():
        state = getCurrentState()
        if state=='on':
                switchOnLight(LIGHT_PIN)
                setCurrentState('on')
        elif state=='off':
                switchOffLight(LIGHT_PIN)
                setCurrentState('off')

def getCurrentMode():
        cur.execute('SELECT * FROM myapp_mode')
        data = cur.fetchone()        #(1, u'auto')
        return data[1]

def getCurrentState():
        cur.execute('SELECT * FROM myapp_state')
        data = cur.fetchone()        #(1, u'on')
        return data[1]

def setCurrentState(val):
        query='UPDATE myapp_state set name="'+val+'"'
        cur.execute(query)

while True:
        currentMode=getCurrentMode()
        if currentMode=='auto':
                runAutoMode()
        elif currentMode=='manual':
                runManualMode()
        time.sleep(5)
```

Implementation: Application

1. Implement Django Application View

```
# Views – views.py
def home(request):
	out=''
		if 'on' in request.POST:
			values = {"name": "on"}
			r=requests.put('http://127.0.0.1:8000/state/1/', data=values, auth=('username', 'password'))
			result=r.text
			output = json.loads(result)
			out=output['name']
		if 'off' in request.POST:
			values = {"name": "off"}
			r=requests.put('http://127.0.0.1:8000/state/1/', data=values, auth=('username', 'password'))
			result=r.text
			output = json.loads(result)
			out=output['name']
		if 'auto' in request.POST:
			values = {"name": "auto"}
			r=requests.put('http://127.0.0.1:8000/mode/1/', data=values, auth=('username', 'password'))
			result=r.text
			output = json.loads(result)
			out=output['name']
		if 'manual' in request.POST:
			values = {"name": "manual"}
			r=requests.put('http://127.0.0.1:8000/mode/1/', data=values, auth=('username', 'password'))
			result=r.text
			output = json.loads(result)
			out=output['name']

		r=requests.get('http://127.0.0.1:8000/mode/1/', auth=('username', 'password'))
		result=r.text
		output = json.loads(result)
		currentmode=output['name']
		r=requests.get('http://127.0.0.1:8000/state/1/', auth=('username', 'password'))
		result=r.text
		output = json.loads(result)
		currentstate=output['name']
		return render_to_response('lights.html',{'r':out, 'currentmode':currentmode, 'currentstate':currentstate},
context_instance=RequestContext(request))
```

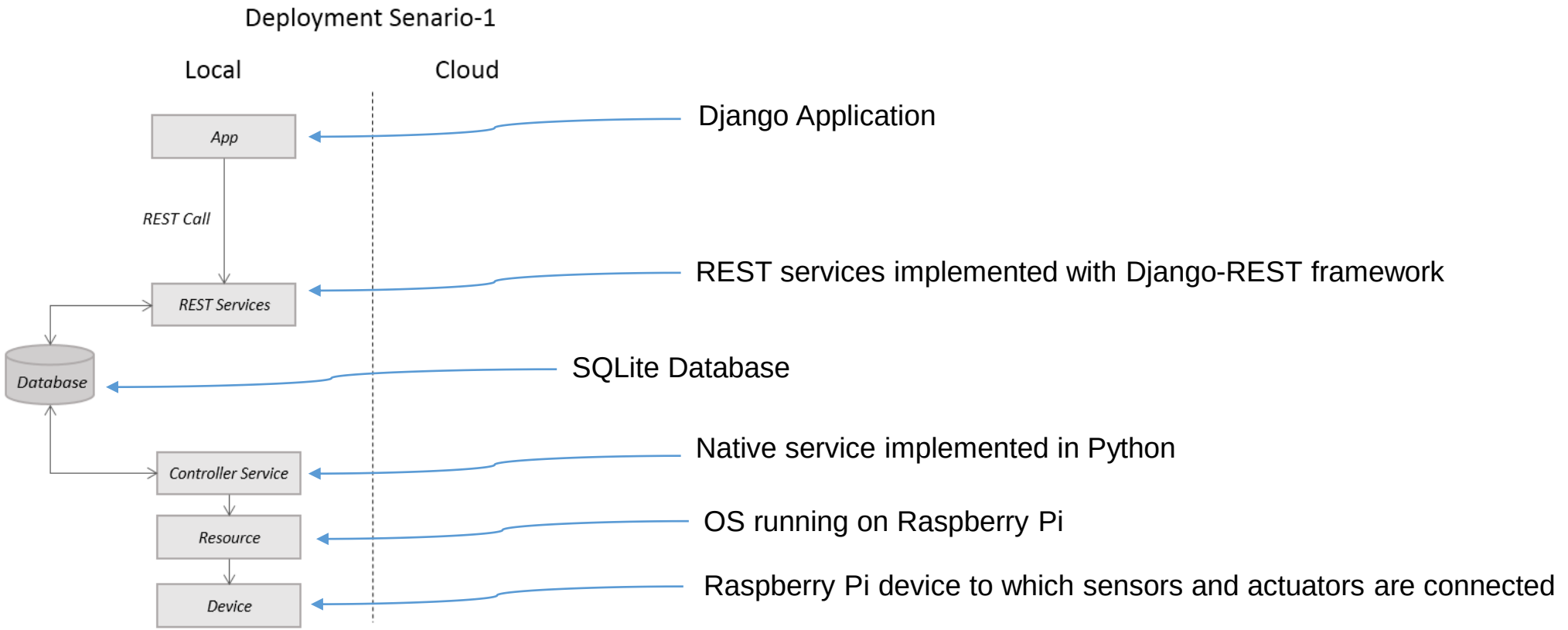
Implementation: Application

2. Implement Django Application Template

```
<div class="app-content-inner">
<fieldset>
<div class="field clearfix">
<label class="input-label icon-lamp" for="lamp-state">Auto</label>
<input id="lamp-state" class="input js-lamp-state hidden" type="checkbox">
{% if currentmode == 'auto' %}
<div class="js-lamp-state-toggle ui-toggle " data-toggle=".js-lamp-state">
{% else %}
<div class="js-lamp-state-toggle ui-toggle js-toggle-off" data-toggle=".js-lamp-state">
{% endif %}
<span class="ui-toggle-slide clearfix">
<form id="my_form11" action="" method="post">{% csrf_token %}
<input name="auto" value="auto" type="hidden" />
<a href="#" onclick="$(this).closest('form').submit()"><strong class="ui-toggle-off">OFF</strong></a>
</form>
<strong class="ui-toggle-handle brushed-metal"></strong>
<form id="my_form13" action="" method="post">{% csrf_token %}
<input name="manual" value="manual" type="hidden" />
<a href="#" onclick="$(this).closest('form').submit()"><strong class="ui-toggle-on">ON</strong></a>
</form></span>
</div></div>
<div class="field clearfix">
<label class="input-label icon-lamp" for="tv-state">Light</label>
<input id="tv-state" class="input js-tv-state hidden" type="checkbox">
{% if currentstate == 'on' %}
<div class="js-tv-state-toggle ui-toggle " data-toggle=".js-tv-state">
{% else %}
<div class="js-tv-state-toggle ui-toggle js-toggle-off" data-toggle=".js-tv-state">
{% endif %}
{% if currentmode == 'manual' %}
<span class="ui-toggle-slide clearfix">
<form id="my_form2" action="" method="post">{% csrf_token %}
<input name="on" value="on" type="hidden" />
<a href="#" onclick="$(this).closest('form').submit()"><strong class="ui-toggle-off">OFF</strong></a>
</form>
<strong class="ui-toggle-handle brushed-metal"></strong>
<form id="my_form3" action="" method="post">{% csrf_token %}
<input name="off" value="off" type="hidden" />
<a href="#" onclick="$(this).closest('form').submit()"><strong class="ui-toggle-on">ON</strong></a>
</form>
</span>
{% endif %}
{% if currentmode == 'auto' %}
{% if currentstate == 'on' %}
<strong class="ui-toggle-on">&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;ON</strong>
{% else %}
<strong class="ui-toggle-on">&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;OFF</strong>
{% endif %}{% endif %}
</div>
</div>
</fieldset></div></div></div>
```

Finally - Integrate the System

- Setup the device
- Deploy and run the REST and Native services
- Deploy and run the Application
- Setup the database



Back to the Fourth Industrial Revolution

Unlike the first three revolutions……..

the fourth revolution is different in:

- **VELOCITY** – exponential pace of adoption (NOT LINEAR)
- **SCOPE** – DISRUPTIONS in ALMOST EVERY INDUSTRY ON EARTH
- **SYSTEMS IMPACT** – Transformation of entire systems of production, management and governance

Dr. Klaus Schwab, Chairman WEF 2016

Big Effect on Our "World"

EFFECT ON BUSINESS

- Customer expectations at affordable cost
- Innovative product enhancements
- Collaborative approach to innovation – include customers, partners & universities
- ORGANIZATION FORMS – OLD HIERARCHICAL FORM was suitable for MASS PRODUCTION, new forms must emerge for the UBER-Like world

EFFECT ON GOVERNMENT

- NEW POLICIES FOR PRIVACY RIGHTS
- NEW SECURITY POLICIES
- FAIR DATA SHARING AND COLLABORATION contracts
- NEW REGULATIONS COVERING INTELLIGENT MACHINES & NETWORKS
- REGULATION OF CORPORATE ORGANIZATIONAL FORMS
- JOB CREATION FOR THE NEW ECONOMY
- ADOPTION OF "AGILE GOVERNANCE"

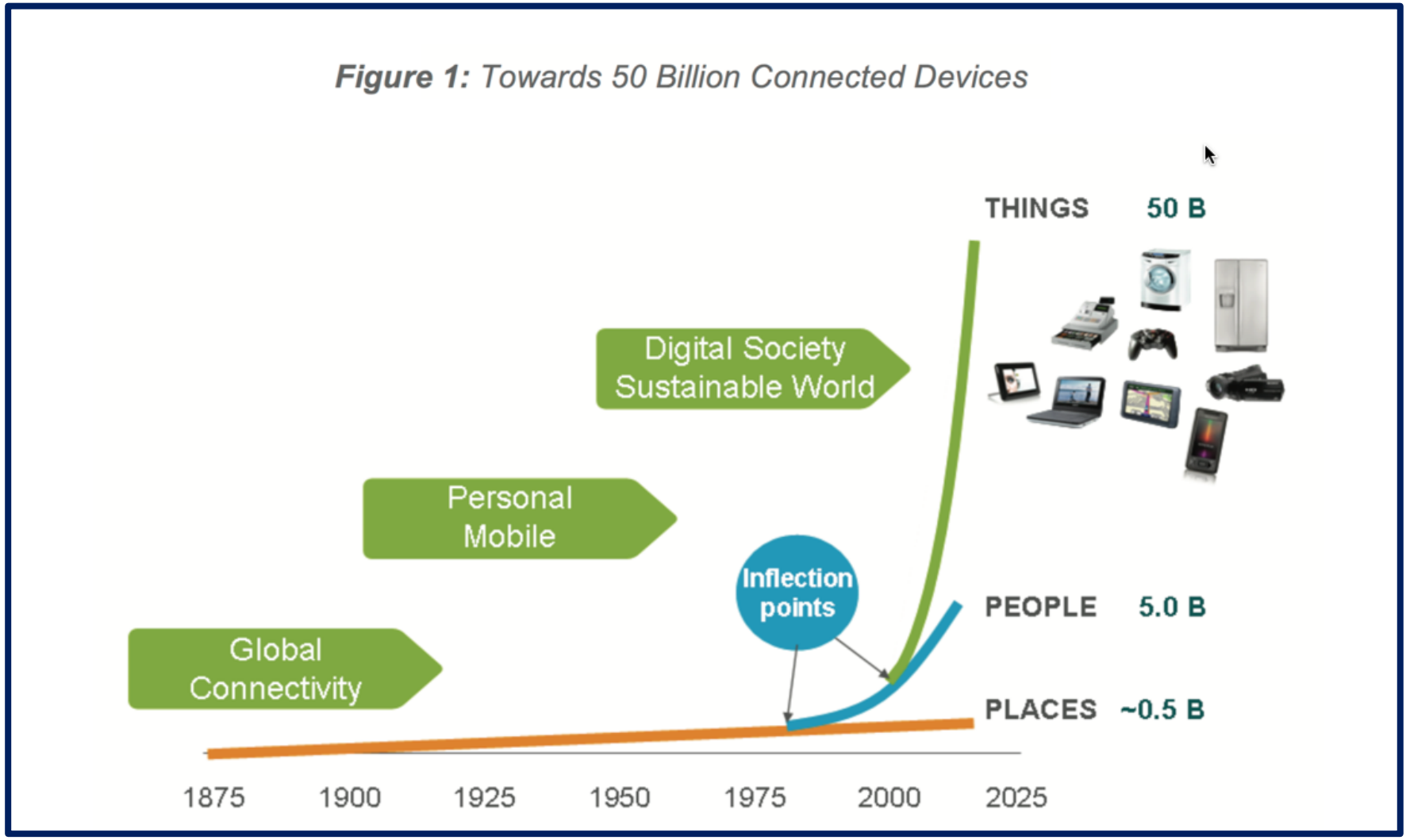
Adapted from
Dr. Klaus Schwab, WEF
www.weforum.org

Impact of IoT on People

- CHANGE IN PERCEPTIONS OF INDIVIDUAL identity & Privacy
- NEW Notions of ownership
- Consumption patterns – A Balance between A material world and spending on “experience”
- Work & Leisure – A balance between LOYALTY-to-SELF versus corporation
- CONCEPT of WORK, Career development & preparation – COMPETENCY-based LIFE-LONG LEARNING
- Relationships with people - A BALANCE between Exchanging status (i.e., “Sharing”) versus meaningful discourse in a uber-connected world
- AUGMENTATION OF COGNITIVE, health & WORK capabilities

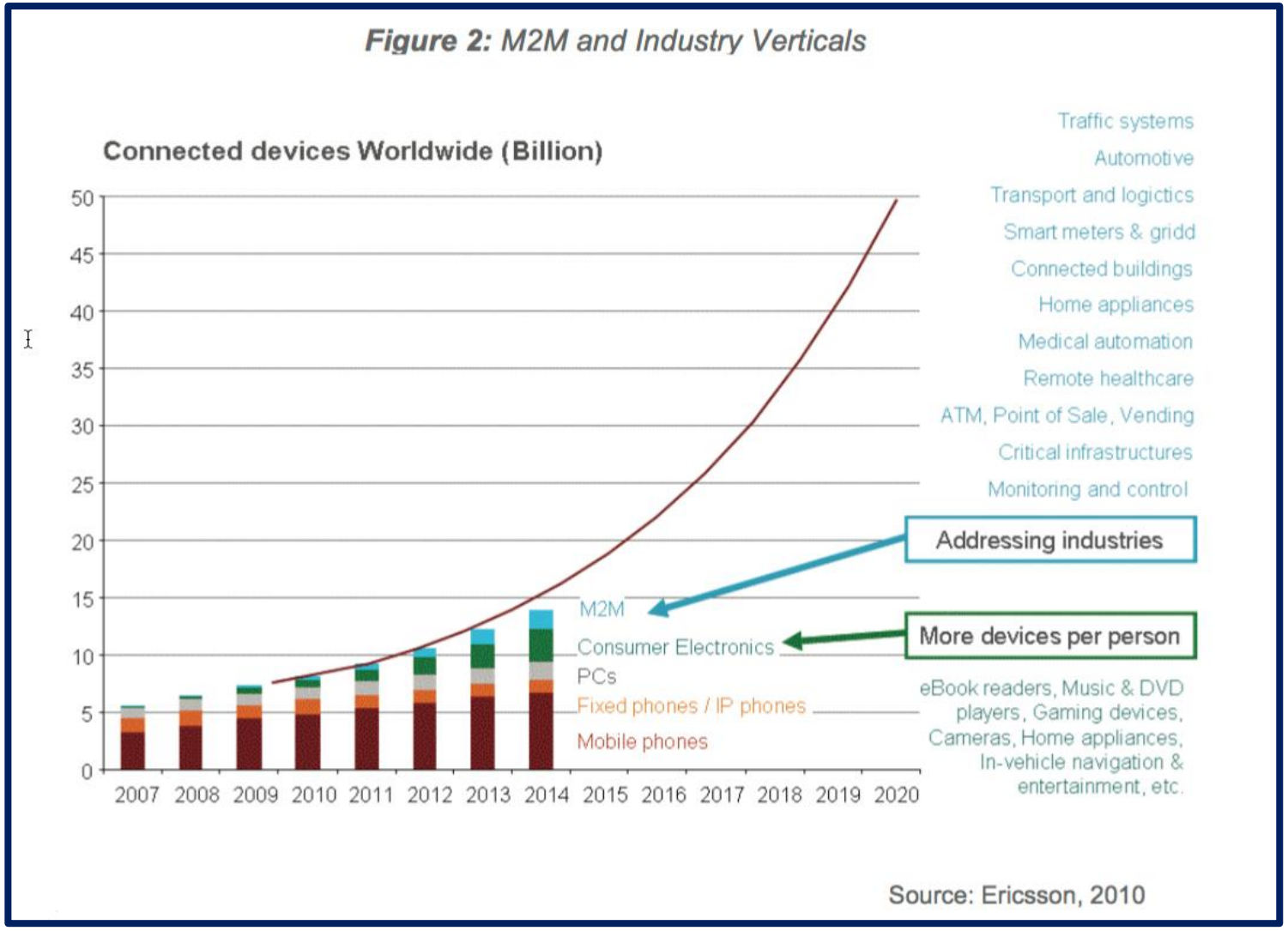
Adapted from
Dr. Klaus Schwab, WEF
www.weforum.org

Predictions Continue for Explosive Growth



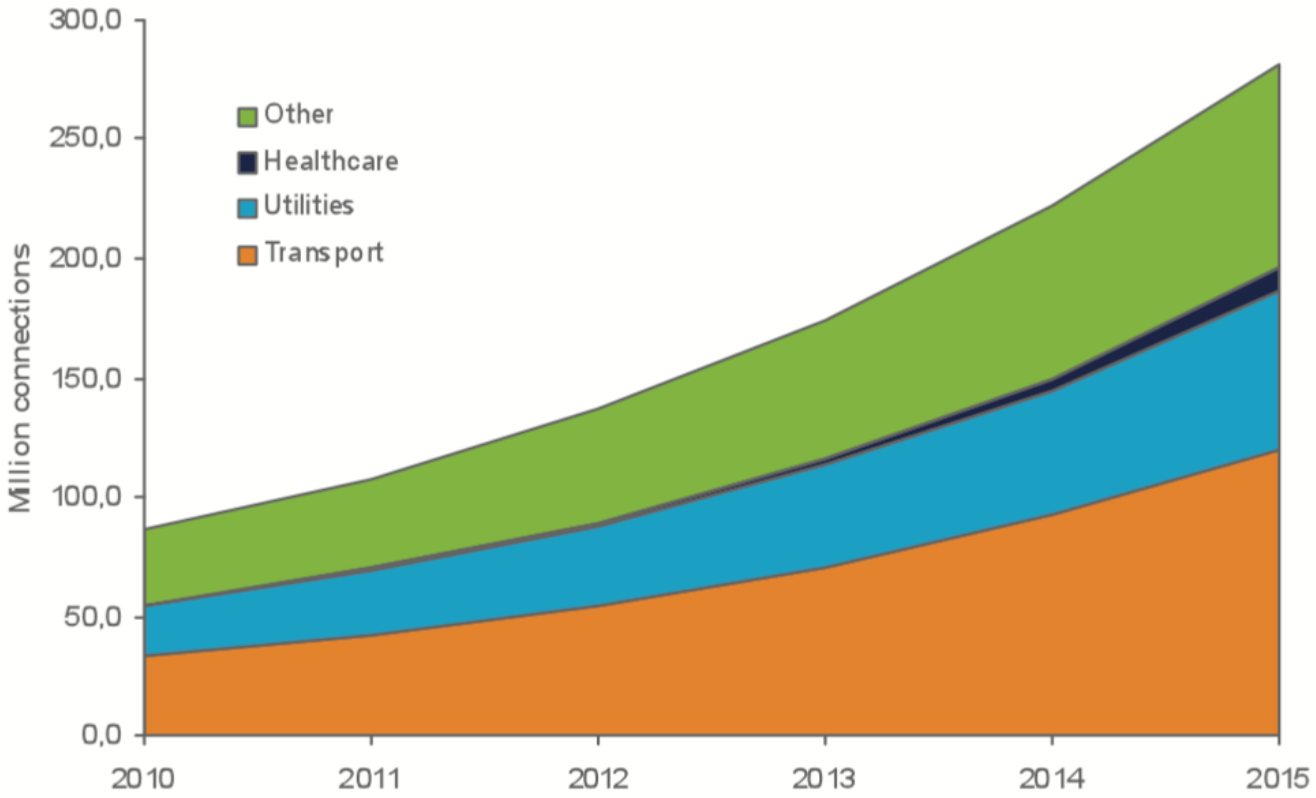
Source: Ericsson 2010

IoT and Business Areas



Major Industries Impacted by IOT

Figure 3: *Global M2M Connections by Industry, 2010-2015*

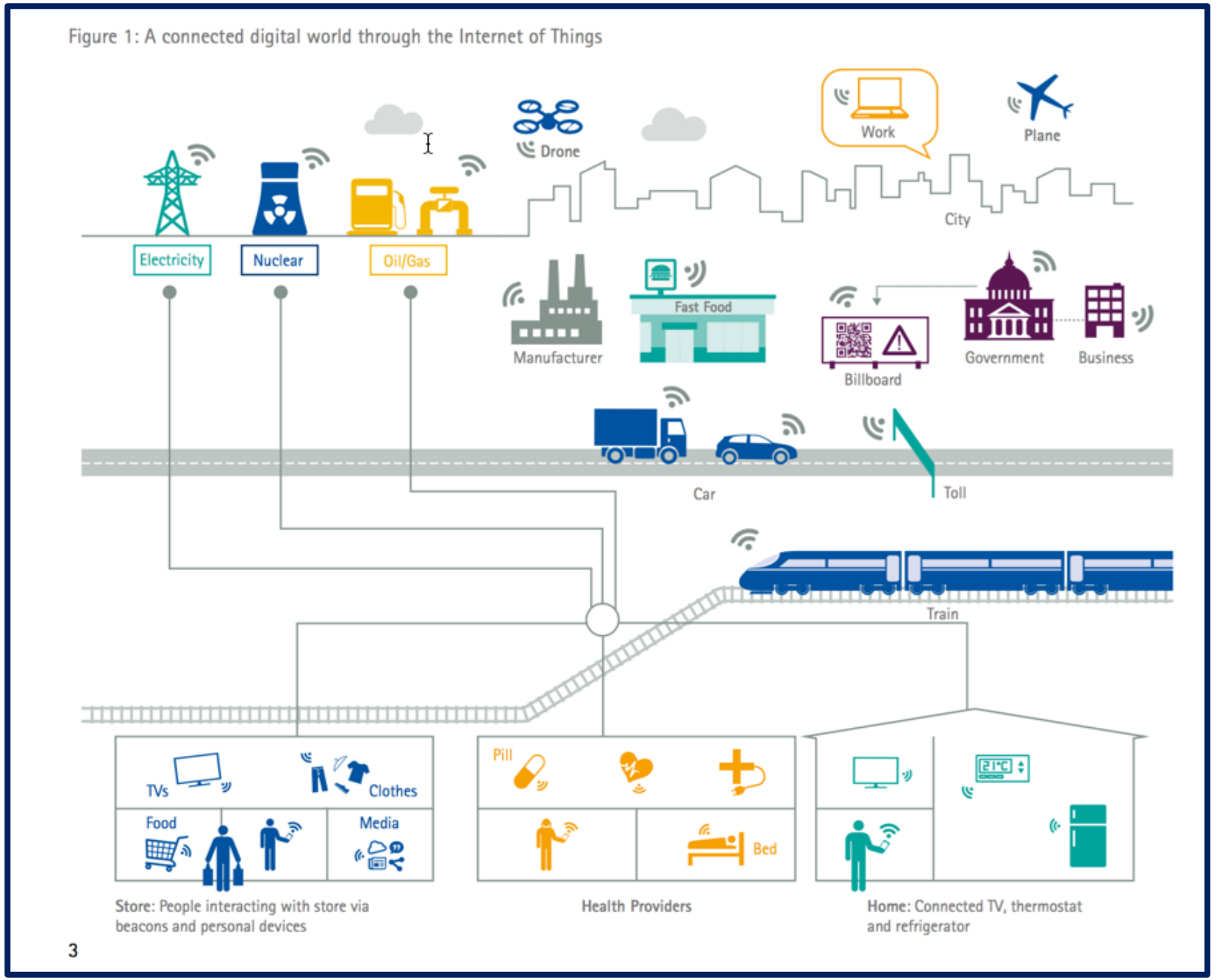


Source: ABI Research, 2009

Figure 4: *Ericsson's "Doctor In A Box" Application Concept*

Source: Ericsson, 2009

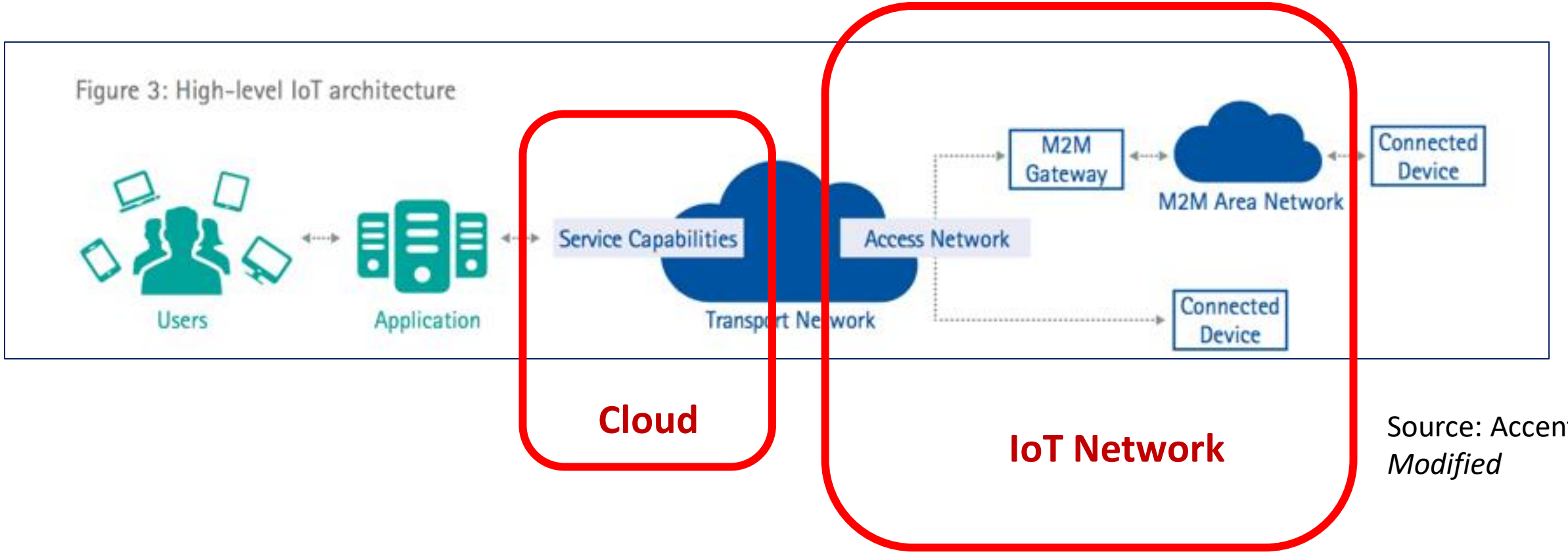
Connected Cyber-Physical World



Source: Accenture

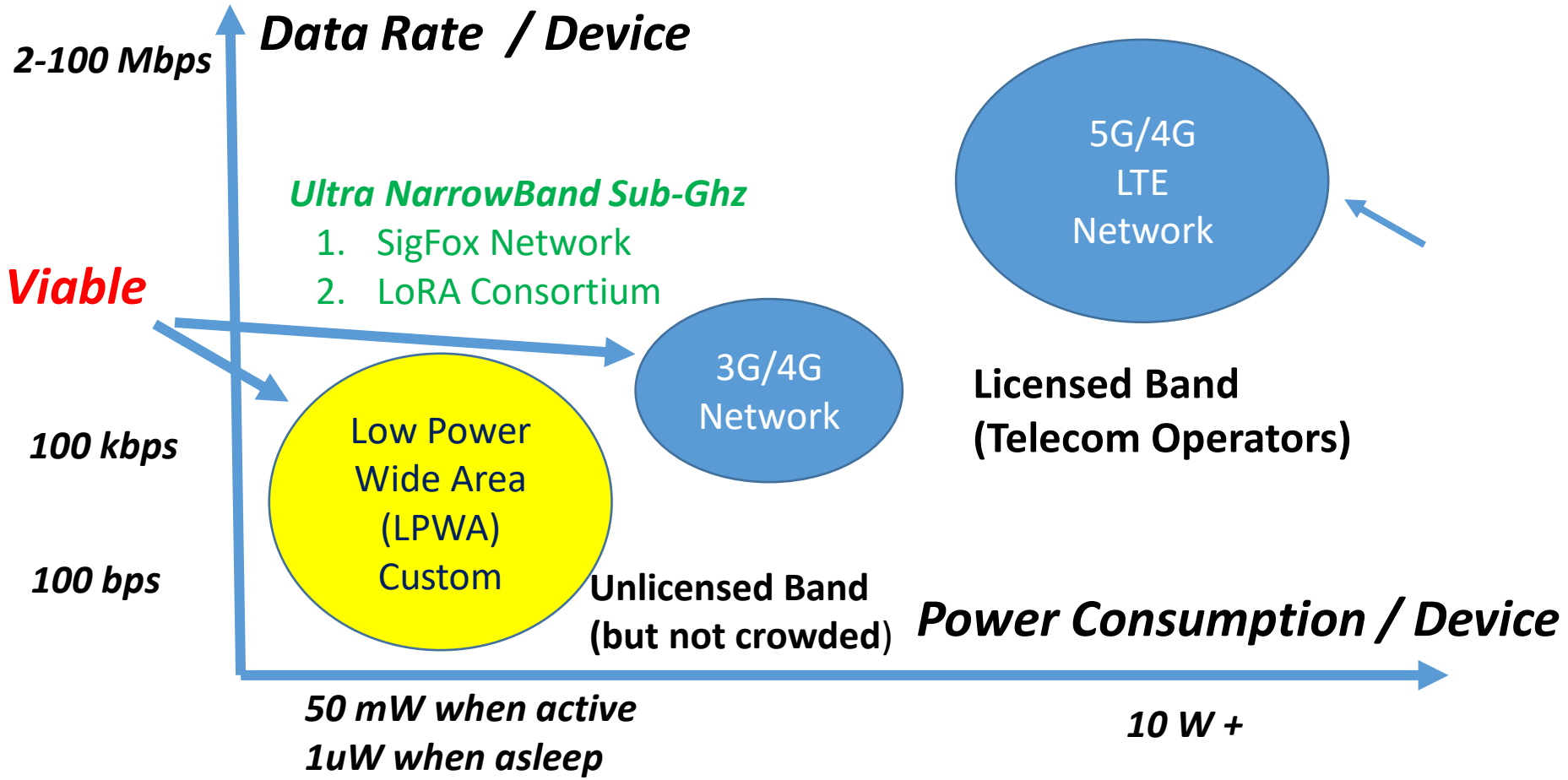
IoT Deployment Network System Architectures

Generic Architecture for an IoT Ecosystem

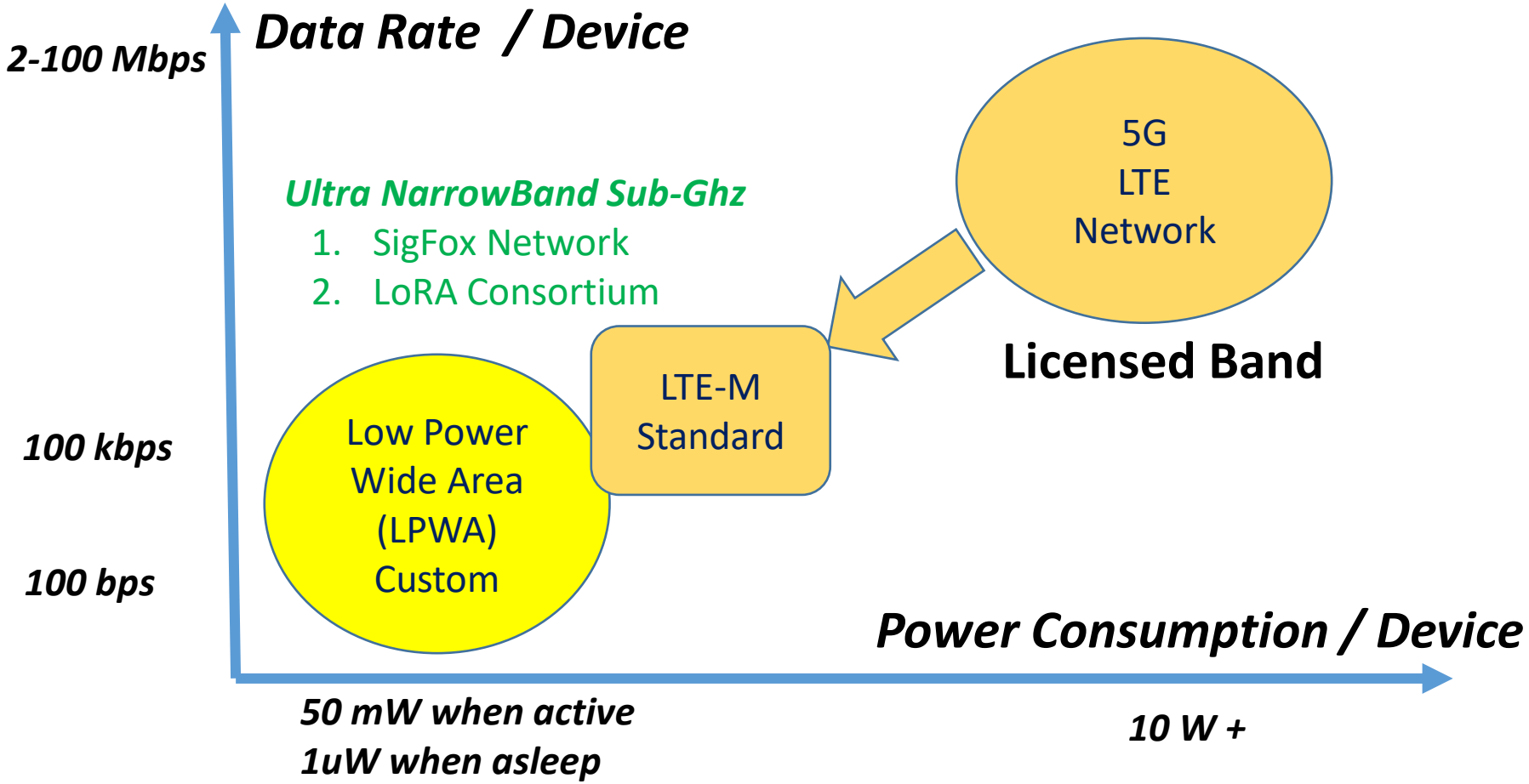


Source: Accenture
Modified

Options for the IOT Network TODAY



Options for the IOT Network in 3 years



Three Candidate Architectures for IoT Networks

- **Telecom Network Operator** – 4G/5G Network Operator that provides both LTE Advanced Release 15/16 and LTE-M.
- **LPWA Network Operator** – Low Power Wide Area Network Operator (e.g., SigFox) to provide a dedicated IoT network
- **Hybrid Model** – Combination of Low Power Sensor-Centric Network for collecting and pooling sensor data, combined with a Telecom/5G backhaul network for aggregating data into the cloud (e.g., LoRA standard)

It is unclear as to which business model/architecture will prevail & how revenues will be shared.

Other options for IoT Networks

- WiFI is power-hungry and has limited range and lot of interference.
- Bluetooth has limited range, consume power, and lot of interference
- IOT networks have a range of 20-40 km, and each cell may have to support hundreds of thousands of active devices, as opposed to cellular (4G) base stations that can support around hundred ACTIVE devices per cell that is a around KM in size.

Business Models for IoT Network Operator

- The 5G/4G Cellular operator bills based on data consumption per device. Typical cost is $5 per GB of data per month. The cost of of a 5G smartphone is expected between $300-$500.
- the IOT network operator is likely to bill at a one-time cost upfront for each device supported for its lifetime - $1-$5 per device total over its lifetime. The value is in receiving the low-rate data reliably and not in the amount of data received. Cost of electronics for the IoT device is $1-$5.
- New Service level Agreement (SLA) models to emerge ?

Key factors for a seamless IoT experience

A seamless Internet of Things (IoT) experience will consist of six components that span enterprise and consumer use cases:

— *Hyperconnected.* Connectivity through multiple standards will be pervasive, connecting a vast number of devices and sensors that seamlessly share data.

— *Integrated.* Integration within and across tech stacks of devices will be effortless (including minimized sign-in effort, self-managed devices, and over-the-air patch updates), with simultaneous use of multiple connectivity standards, platforms, and back-end systems.

— *Secure and trusted.* Dynamic cybersecurity will enable a high degree of trust in handling the multilayered complexity of legacy systems and new solutions, with security enabled through AI-based threat protection at all layers.

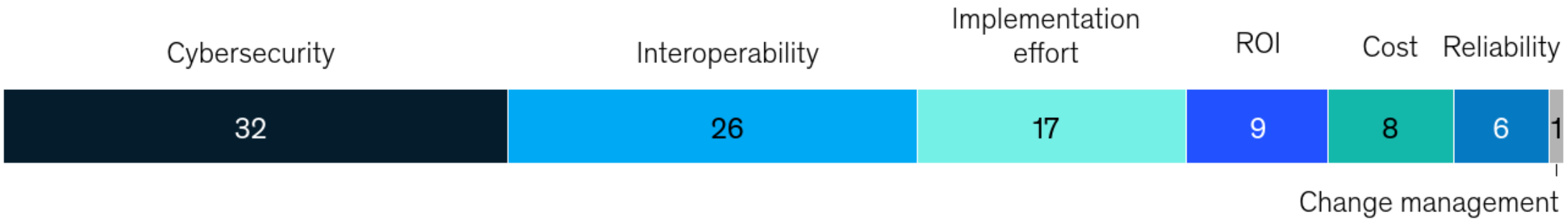
— *Intelligent.* Devices and systems will have the intelligence (enabled by AI and machine learning) to draw insights from data and make real-time decisions, allowing the leap from monitored to automated implementation.

— *Mobile.* Devices and networks will require minimal maintenance, be battery efficient, and have a persona (corporate or personal identity) to allow for futuristic experiences.

— *Hyperpersonalized.* There will be personalized experiences across different platforms and scenarios (from home to office and everywhere in between), enabled by the other factors.

Enterprise buyers rate cybersecurity as the biggest obstacle to B2B Internet of Things adoption and spending.

Top impediment to Internet of Things adoption, % of respondents



Note: Figures do not sum to 100%, because of rounding.
Source: McKinsey B2B Internet of Things Survey, 117 buyers, Q3 2022

McKinsey & Company

Confidentiality, integrity, and availability make up a well-known enterprise framework for assessing cyber-risk impact:

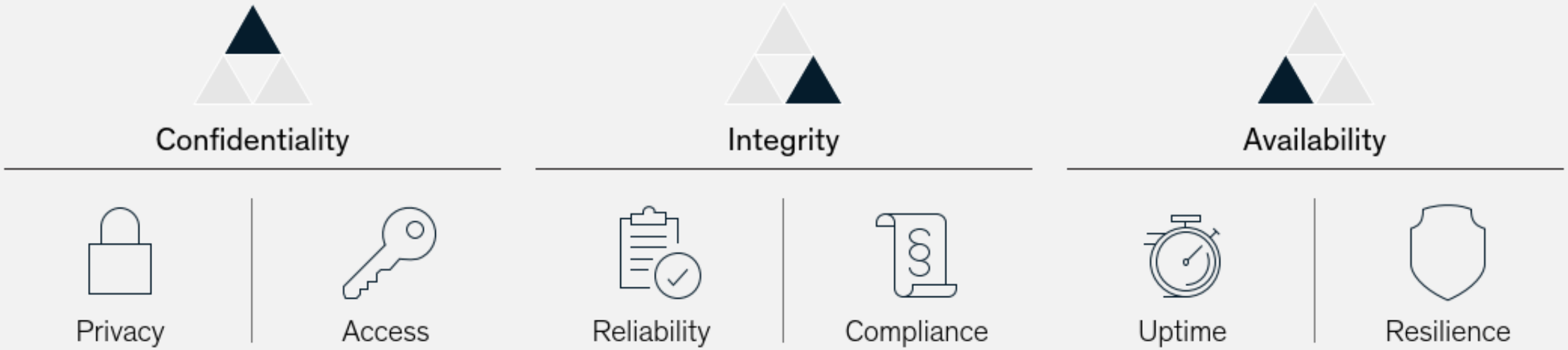
- *Confidentiality.* Only authorized endpoints or users have access.
- *Integrity.* Data are transferred as expected—complete and unaltered.
- *Availability.* Data and system functionality meet user demand and expectations.

The framework can be expanded to six dimensions in Internet of Things (IoT) environments, capturing the unique risks and concerns to ensure IoT security—especially those around safety in operational environments (exhibit):

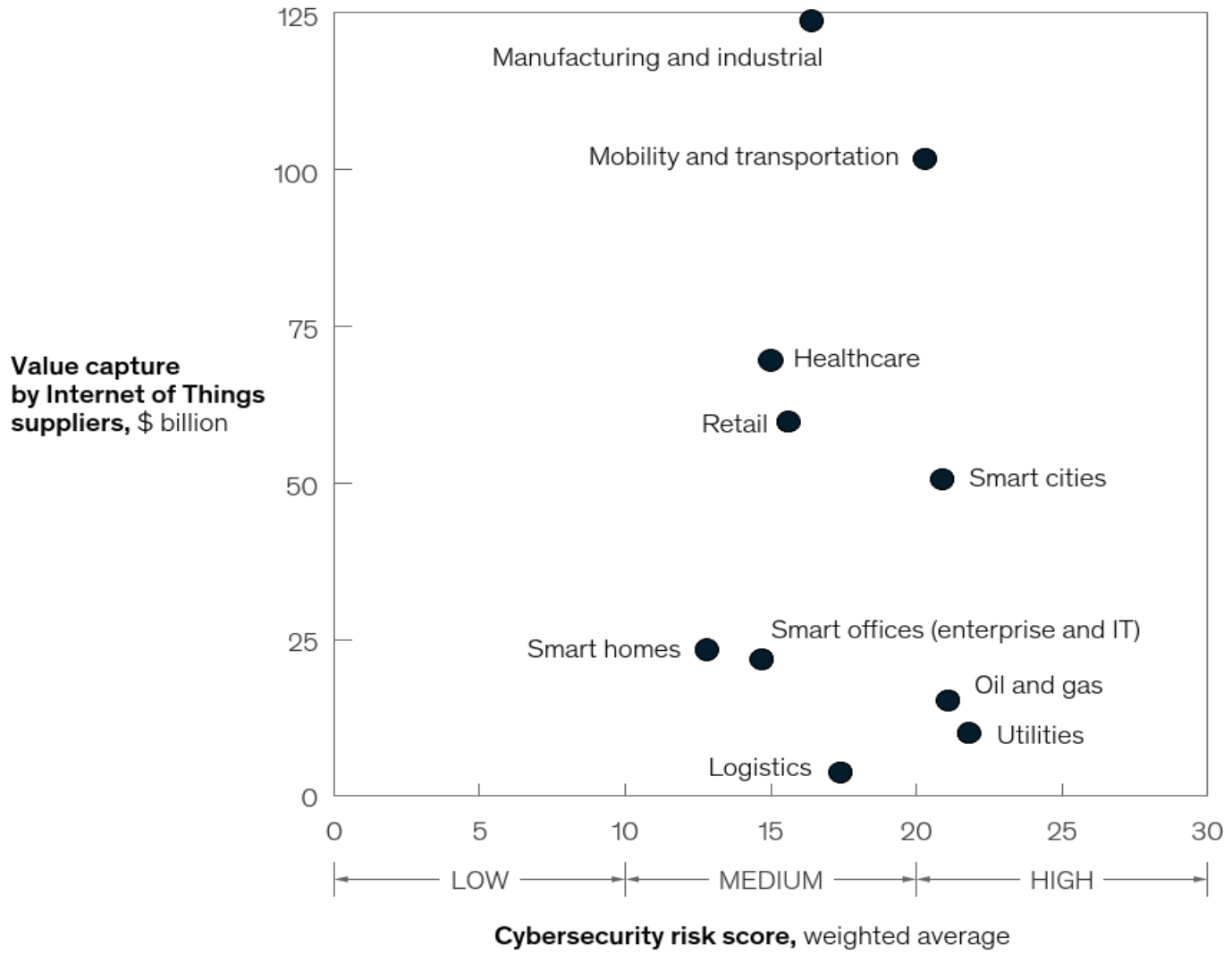
- *Confidentiality: privacy and access.* As the IoT is adopted in highly regulated industries (such as healthcare), the data flowing through the IoT tech stack should be handled with the highest degree of protection, with uninterrupted access for authorized users.
- *Integrity: reliability and compliance.* As the IoT optimizes processes—and eventually, autonomously controls them—users should trust that the data flowing throughout the IoT tech stack are accurate, relevant, and unadulterated. As cybersecurity regulation for the IoT evolves, users should trust that IoT systems are compliant with and have automatically adapted to the latest regulations.
- *Availability: uptime and resilience.* For the IoT to make the transition from being used for open-loop systems to being used for closed-loop systems, users shouldn't need to worry about any operational disruptions. Also, in IoT use cases for which humans are in the loop, cybersecurity should be capable of guaranteeing fail-safe mechanisms.

Expanding an enterprise cyber-risk framework to six dimensions can offer a more holistic security approach in Internet of Things environments.

Confidentiality, integrity, and availability framework

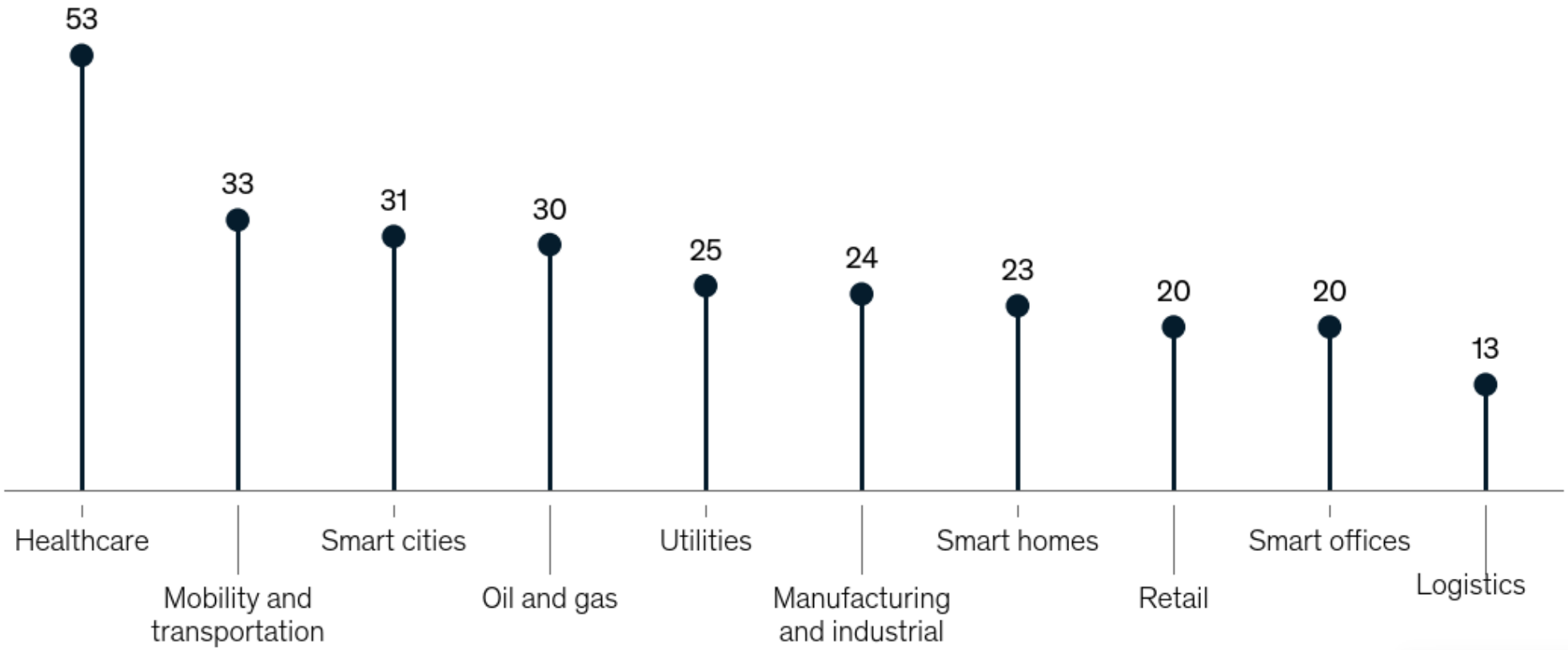


Baseline 2030 Internet of Things value capture and cybersecurity risk score, by use case



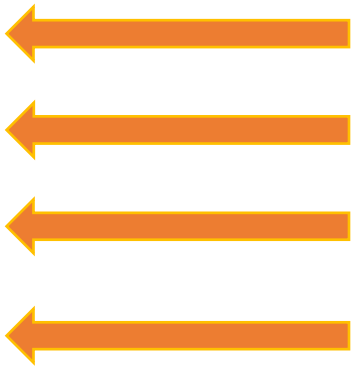
Increased cybersecurity can help all industries grow, but certain industries are poised to unlock the most Internet of Things value.

Average increase in Internet of Things spending if cybersecurity risk is managed, by use case, %



The Seven Tenets of Succesful IoT DEPLOYMENT

- Confidentiality: Data is not available to unauthorized parties
- Integrity: Data or Code cannot be changed or damaged or erased by unauthorized parties (data at rest or in motion)
- Availability: Network and data is responsive and available to authorized parties
- Controllability
- Visibility
- Safety
- Standardization



New Tenets

Controllability

- Operators should be able to control their IoT networks and devices remotely, to:
  - Upgrade or update their devices & networks – devices have an identity and state
  - Provide automated facilities for configuring their devices
  - Ability to contain or isolate threats (at device, network and cloud-level) into pre-defined isolated regions of containment, e.g., *zones & conduits* in ISA/IEC 62443. Identify zones that can be controlled, that can be observed but not controlled, and zones that are uncontrollable.
  - Ability to configure their networks for time-based work-flows and guarantees of deterministic and predictable operation
  - Provide robust and/or redundant pathways for control and configuration, including poison pills for rogue devices
  - Prevent data loss and support backup, reset and and remote wipes

Visibility

- IoT Networks operators should be able to view & maintain a “current” state of their network and its devices in the cloud:
  - From a network health perspective
  - From a risk perspective –
    - Zones functioning normally
    - Nodes/Zones partitioned into control, supervisor, safety, vendor, …, external zones (e.g., ISA 62443)
    - Relative importance and sensitivity of data and operations in different partitions/zones of the network, and their region of influence.

Safety

- IoT Operators must implement policies to ensure that their networks and endpoint devices cannot affect safety of the users and the environment
  - Frequent network-based checks to ensure IoT devices and components are operating as programmed. No backdoors to compromise safety, specially in industrial control systems and smart grids
  - Checks to detect compromise, including voting based self-checking protocols
  - Tests to identify timing issues that can affect safety (congestion, denial of service, jamming, ..)
  - Self-reporting by devices and networks based on early detection of anomalies (loss of power, network congestion & outages, intrusions, etc.)

Standardization

- Follow standardized protocols to limit risk and obtain benefit of industry knowledge of weaknesses of various components
- Expand & Improve Existing Standards
  - NIST Special Publication 800-82 Revision 2 Final Public Draft: **Guide to Industrial Control System (ICS) Security.** February 2015
  - ISA/EIC 62443 **Security for Industrial Automation and Control Systems – Security Risk Assessment and System Design**, May 2014

Zones & Conduits to Secure & Isolate Risk in an Oil Refinery Information Network

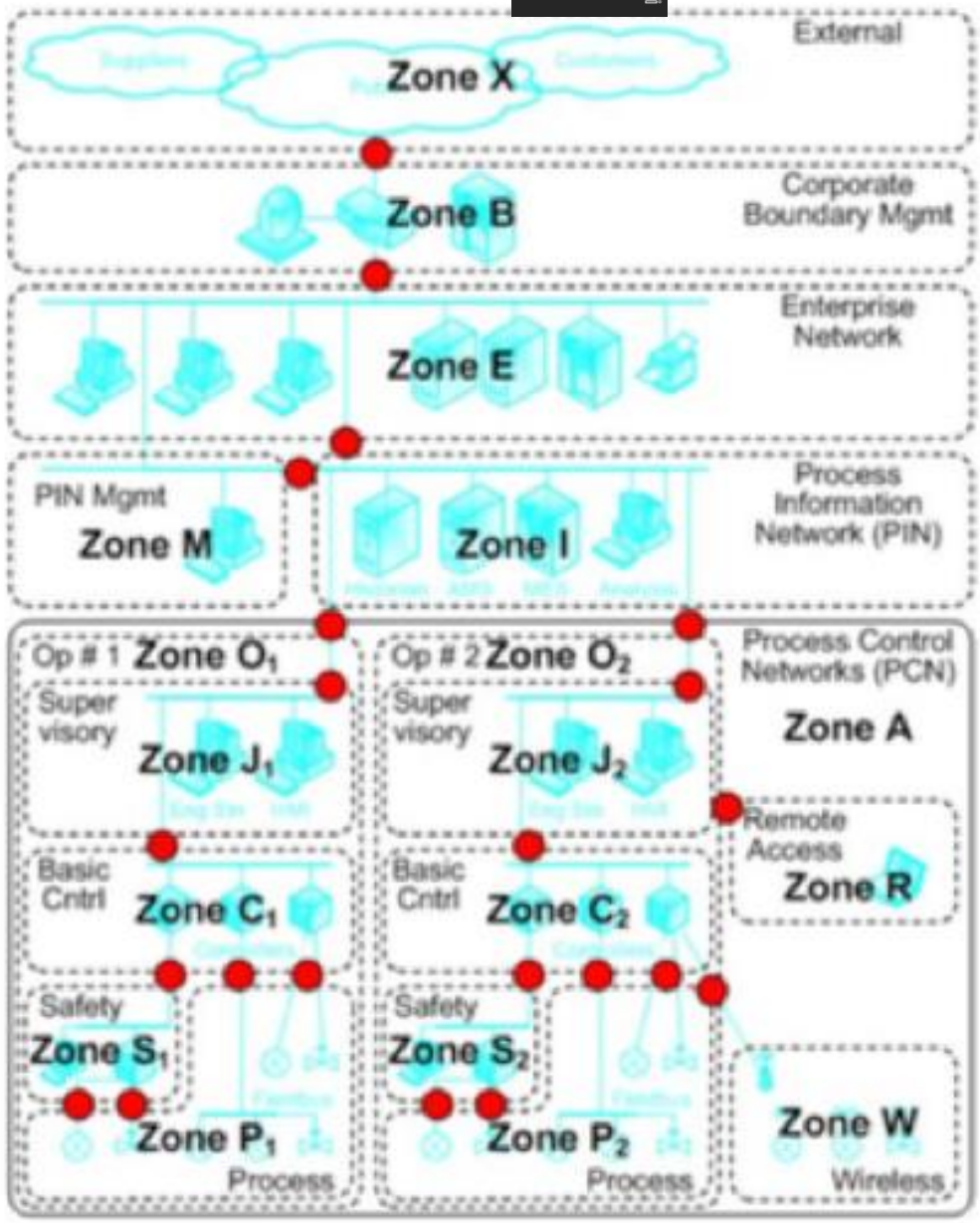


Figure 9: Defined Conduits in Refinery

How Confidentiality, Integrity and Availability solved ?

- Several types of threats
  - Rogue nations & entities
  - Hackers and Theft
  - Mischief Mongers
  - Political Activists
  - INSIDER THREATS

- **Best Current Practice:**
  - Authentication, Cryptography & Encryption (both for data at rest and in motion)
  - Strong passwords for all accounts
  - TLS Certificates & Validation
  - Device to Cloud, and Device to Device authentication
  - Protect against man-in-the-middle attacks (by someone who placed himself in the communications path)
  - Foreign code restrictions, trust zones & debug modes restricted
  - Detection and prevention of attacks that compromise availability, through multiple/redundant network connections

Source: Veracode

Summary & TAKEAWAYS

- THE FOURTH INDUSTRIAL REVOLUTION is here – at the heart of this revolution is **IOT**
- IOT ECOSYSTEM MODELS are still evolving – with TELECOM OPERATORS on ONE SIDE and DEDICATED IOT LPWA OPERATORS EYEING The prize
- BUSINESS MODELS FOR IOT NETWORK ARE LIKELY TO BE DIFFERENT FROM existing licensed and unlicensed networks

- Some suggestions are offered to Corporations – *the SEVEN TENETS of SUCCESSFUL IOT DEPLOYMENT*

IoT Security

- **Authentication**
  - Authentication refers to digitally confirming the identity of the entity requesting access to some protected information.
  - In IoT & Cloud, where applications and data are accessed over the internet, the complexity of digital authentication mechanisms increases rapidly.
- **Authorization**
  - Authorization refers to digitally specifying the access rights to the protected resources using access policies.
  - Authorization in a IoT & Cloud requires the use of the cloud service providers services for specifying the access policies.
- **Security of data at rest**
  - Due to the multi-tenant environments used in the cloud, the application and database servers of different applications belonging to different organizations can be provisioned side-by-side increasing the complexity of securing the data.
  - Appropriate separation mechanisms are required to ensure the isolation between applications and data from different organizations.

IoT Security

- **Security of data in motion**
  - In IoT, the applications and the data are moved out of the in-house IT infrastructure to the cloud provider.
  - Therefore, appropriate security mechanisms are required to ensure the security of data in motion.
- **Data Integrity**
  - Data integrity ensures that the data is not altered in an unauthorized manner after it is created, transmitted or stored. Due to the outsourcing of data storage in IoT & Cloud computing environments, ensuring integrity of data is important.
- **Auditing**
  - Auditing mechanisms are required to get visibility into the application, data accesses and actions performed by the application users, including mobile users and devices such as wireless laptops and smartphones.

Vulnerable IoT Perimeters: When IoT networks are designed, there is lack of planning of good security implementation which can allow an intruder to easily gain access to the network. Let's take an example of Smart Meter. If a cyber criminal compromised this device, he is able to access a domestic network and also can monitor the connections between objects in IoT.

Increase in Data Breaches: Data breaches are one of the biggest threats in IoT devices. Cyber attackers can try to spy on the communications between devices in IoT network. Devices accessed through Internet of Things may be used for cyber espionage purposes by an intelligence agency or by some companies for commercial purposes. The FBI's chief information security officer warned the impact of IoT data breaches could be much worse for end users than previous enterprise data breaches

Malware and Botnet Attacks: Malicious users designed the code for attempting to attack against IOT networks. Cyber criminals can exploit vulnerabilities in firmware running on the devices and run their arbitrary code, turning IoT components to unplanned use. Some of the Malware used in IOT is Linux worm, Linux.Darlloz. Graphics processing units-based malware and ransomware attacks are growing rapidly, due to the increase in data, bigger networks, and the Internet of Things (IoT), according to Intel Security's five-year retrospective threat report. The analysis found that ransomware continued to grow rapidly, with the number of new ransomware samples rising 58 percent in Q2. According to Intel Security, the total number of ransomware samples also grew by 127 percent year-on-year, with the company attributing the increase to fast-growing new families, such as CTB-Locker and CryptoWall. The release of the report marks the five-year anniversary since Intel Security purchased McAfee for $7.7 billion.

OWASP Introduces Vulnerabilities in IoT

- The Open Web Application Security Project (OWASP) comes with best practices to improve the security of IoT. It is natural that the project also analyzed the top 10 security issues related to the popular paradigm:
- **Insecure Web Interface Insecure** Web Interface is a common vulnerability found in IoT. OWASP Zap and shodan tools are available and with them we can access these devices. The most famous example of this to date is the case of the web application on TrendNet cameras that exposed a full video feed to anyone who accessed it.

OWASP Introduces Vulnerabilities in IoT (contd…)

- **Insufficient Authentication/Authorization** Most IoT devices are protected with a weak password and it is easily exploited through a brute force attack. The attack could come from external or  internal users. Some devices in IoT are configured with a base64 password encoding mechanism and sent between devices in plain text so attacker can use an online website through which they try to convert base64 code to simple text. Many IoT devices are secured with “Spaceballs quality” passwords like “1234”, put their password checks in client-side Java code, send credentials without using HTTPS or other encrypted transports, or require no passwords at all.

OWASP Introduces Vulnerabilities in IoT (contd…)

- **Insecure Network Services** Insecure network services may be vulnerable to buffer overflow attacks. Some other attacks can also be done, like DOS and DDOS attacks, which leave systems inaccessible to clients or users. In order to find insecure network services, we use several tools, like Nmap and other fuzzers. Examples of these types of services abound in IoT documentation and are regularly lit up by security researchers. In August 2014, a sweep of more than 32,000 devices found "at least 2000 devices with hard-coded Telnet logins."

OWASP Introduces Vulnerabilities in IoT (contd…)

- **Lack of Transport Encryption** IoT devices have a lack of transport encryption which are exploited by an attacker who is trying to intercept the information exchanged between IoT devices. This attack can be done from internal and external users.
- **Privacy Concerns** An attacker uses a different path, like lack of authentication, lack of strong transport encryption or other ports and network services through which they gain access to personal data. One of the biggest vulnerabilities, as per OWASP Standard, is that home users may not understand computer security, but they do understand physical security (“is my door locked?”) and privacy (“is that camera watching me?”). Furthermore, their fears are widespread.

OWASP Introduces Vulnerabilities in IoT (contd…)

- **Insecure Cloud Interface** We can identify an insecure cloud interface vulnerability through reviewing the connections to the cloud interface and analyzing if SSL is secure. We also attempt a password reset on the portal to find a live user, which can lead to user enumeration. Since most security professionals already know how to evaluate systems for these types of vulnerabilities, we won't spend much time on it in this article, except to remind you that you should get the permission of any remote cloud service before you attempt to perform any type of penetration test against it.

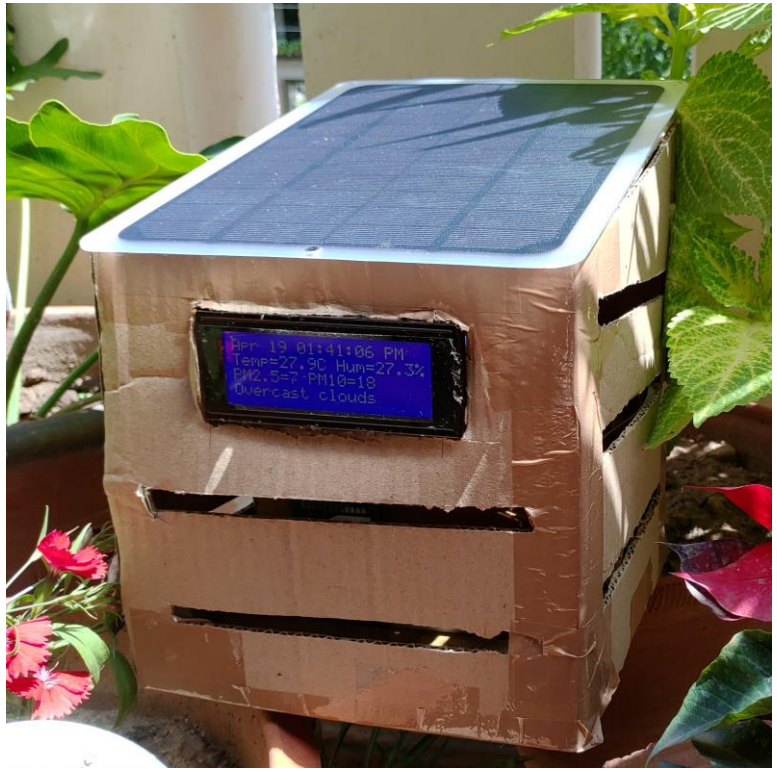
OWASP Introduces Vulnerabilities in IoT (contd…)

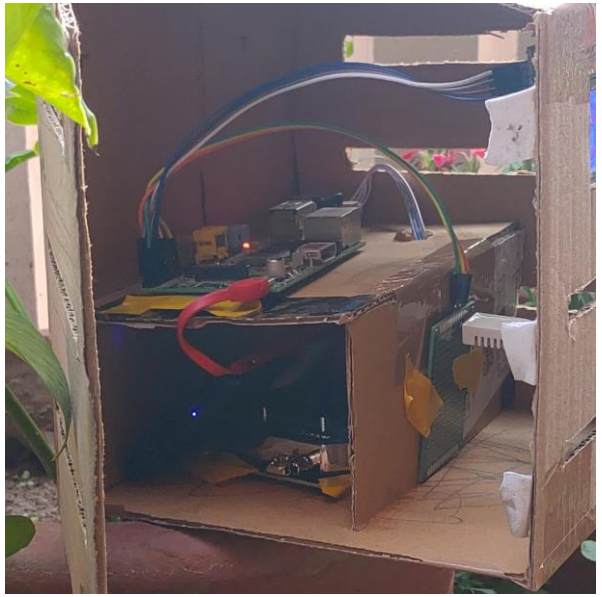
- Insecure Mobile Interface
- Insufficient Security Configurability
- Insecure Software/Firmware
- Poor Physical Security

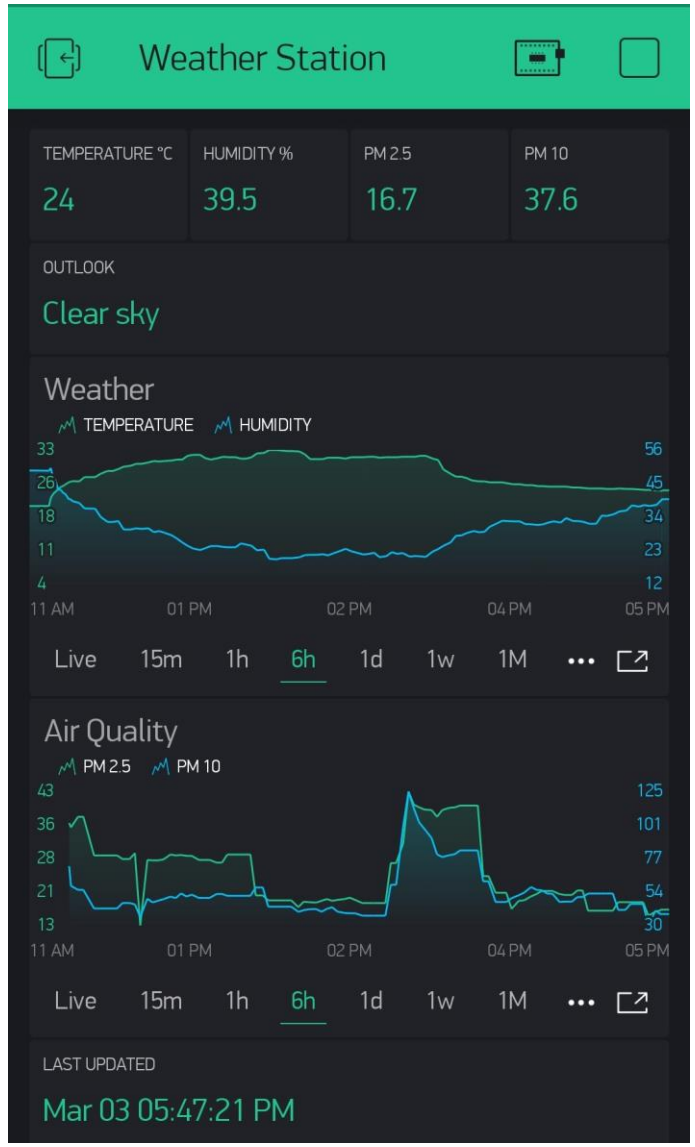
OWASP Introduces Vulnerabilities in IoT (contd…)

- Insecure Mobile Interface
- Insufficient Security Configurability
- Insecure Software/Firmware
- Poor Physical Security

Solar-Powered Weather & Air Pollution Monitoring Station

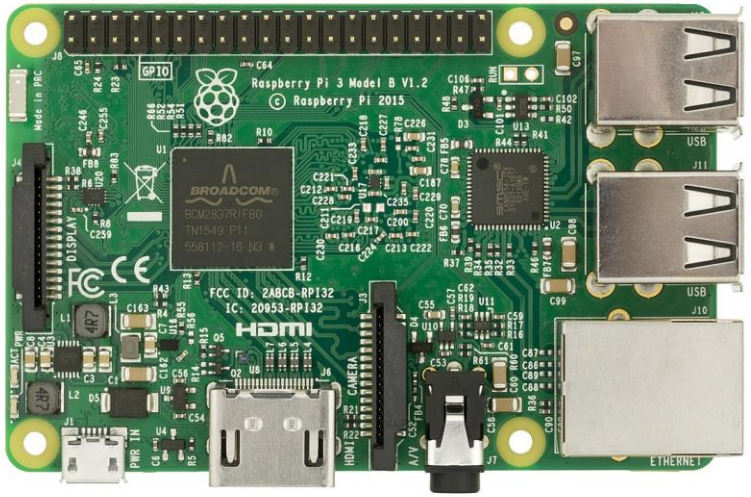






Components

- Raspberry Pi
- Nova SDS011 Air Quality Sensor
- AM2302/DHT22 Temperature & Humidity Sensor
- 20W Solar Panel with 5V USB output



Implementation

- **Fetch Temperature and Humidity data**

```
import Adafruit_DHT
pin=22
sensor = Adafruit_DHT.AM2302
humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
```

Implementation

- **Fetch PM2.5 and PM10 data**

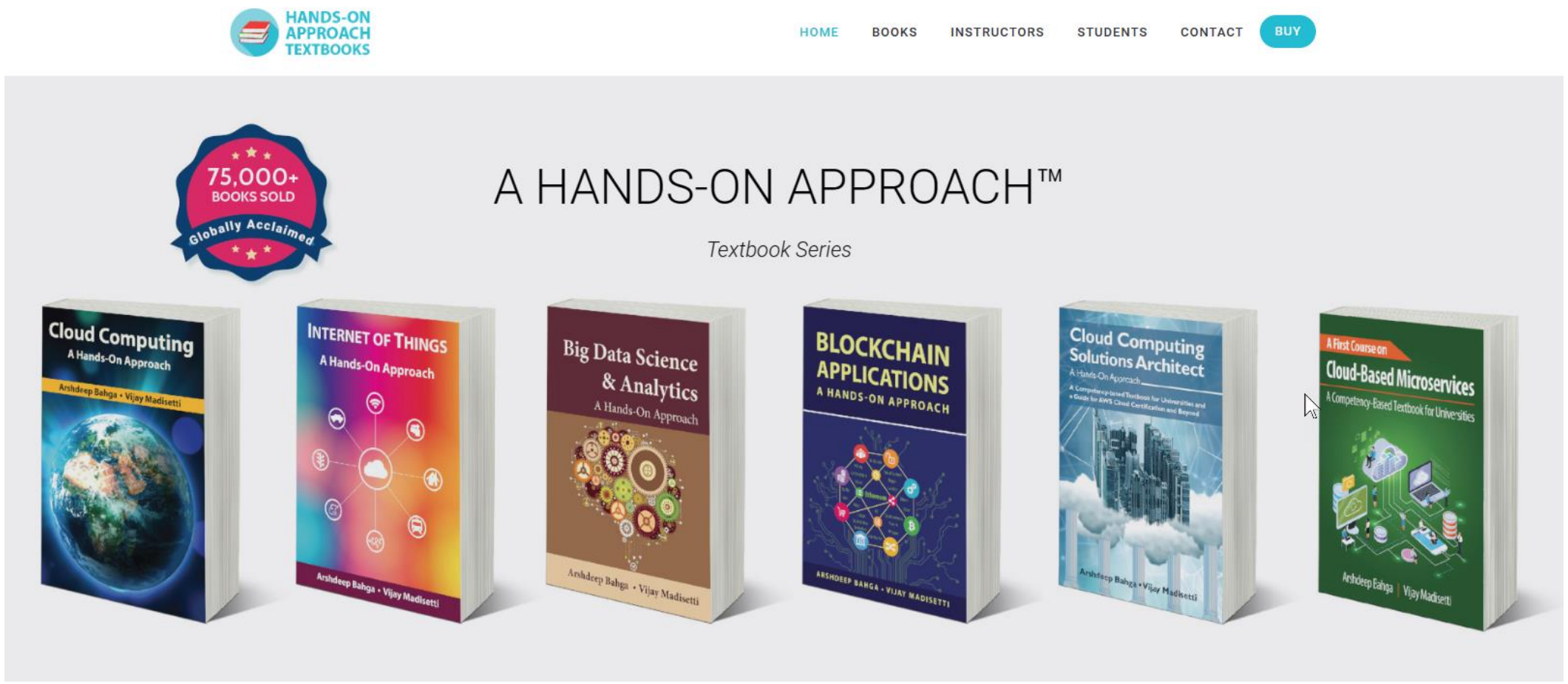
```
import serial, time, struct
ser = serial.Serial("/dev/ttyUSB0", baudrate=9600,
                    stopbits=1, parity="N", timeout=2)
ser.flushInput()
byte, lastbyte = "\x00", "\x00"
while True:
    lastbyte = byte
    byte = ser.read(size=1)
    print byte.encode('hex')
    if lastbyte == "\xAA" and byte == "\xC0":
        sentence = ser.read(size=8) # Read 8 more bytes
        print str(sentence.encode('hex'))
        readings = struct.unpack('<hhxxcc',sentence)
        pm_25 = readings[0]/10.0
        pm_10 = readings[1]/10.0
```

Implementation

- **Publish data to Blynk**

```
import requests, datetime
def blynkEventUpdate(pin, val):
    put_header={"Content-Type": "application/json"}
    put_body = json.dumps([val])
    r = requests.put('http://blynk-cloud.com/'+BLYNK_AUTH+'/update/'+pin,
                     data=put_body, headers=put_header)
    print r.status_code
    return 1

timestampstr=str(datetime.datetime.now().strftime("%b %d %I:%M:%S %p"))
blynkEventUpdate('V1', timestampstr)
blynkEventUpdate('V2', temperature)
blynkEventUpdate('V3', humidity)
blynkEventUpdate('V4', pm_25)
blynkEventUpdate('V5', pm_10)
```

Further Reading



Textbook Series on Advanced and Emerging Technologies

Thank You