INTERNET OF THINGS TECHNOLOGY

[As per Choice Based Credit System (CBCS) scheme]

(Effective from the academic year 2016 -2017)

SEMESTER – VIII

Subject Code	**15CS81**	IA Marks	20
Number of Lecture Hours/Week	04	Exam Marks	80
Total Number of Lecture Hours	50	Exam Hours	03

CREDITS – 04

HKBK COLLEGE OF ENGINEERING, BENGALURU

Syllabus- Internet Of Things Technology

Course Objectives: This course will enable students to
• Assess the genesis and impact of IoT applications, architectures in real world. • Illustrate diverse methods of deploying smart objects and connect them to network. • Compare different Application protocols for IoT. • Infer the role of Data Analytics and Security in IoT. • Identifysensor technologies for sensing real world entities and understand the role of IoT in various domains of Industry.

Course Outcomes: After studying this course, students will be able to
• Interpret the impact and challenges posed by IoT networks leading to new architectural models. • Compare and contrast the deployment of smart objects and the technologies to connect them to network. • Appraise the role of IoT protocols for efficient network communication. • Elaborate the need for Data Analytics and Security in IoT. • Illustrate different sensor technologies for sensing real world entities and identify the applications of IoT in Industry.

Syllabus- Internet Of Things Technology

Module – 1	Teaching Hours
What is IoT, Genesis of IoT, IoT and Digitization, IoT Impact, Convergence of IT and IoT, IoT Challenges, IoT Network Architecture and Design, Drivers Behind New Network Architectures, Comparing IoT Architectures, A Simplified IoT Architecture, The Core IoT Functional Stack, IoT Data Management and Compute Stack.	10 Hours
Module – 2	
Smart Objects: The "Things" in IoT, Sensors, Actuators, and Smart Objects, Sensor Networks, Connecting Smart Objects, Communications Criteria, IoT Access Technologies.	10 Hours
Module – 3	
IP as the IoT Network Layer, The Business Case for IP, The need for Optimization, Optimizing IP for IoT, Profiles and Compliances, Application Protocols for IoT, The Transport Layer, IoT Application Transport Methods.	10 Hours

Syllabus- Internet Of Things Technology

Module – 4	
Data and Analytics for IoT, An Introduction to Data Analytics for IoT, Machine Learning, Big Data Analytics Tools and Technology, Edge Streaming Analytics, Network Analytics, Securing IoT, A Brief History of OT Security, Common Challenges in OT Security, How IT and OT Security Practices and Systems Vary, Formal Risk Analysis Structures: OCTAVE and FAIR, The Phased Application of Security in an Operational Environment	**10 Hours**
Module – 5	
IoT Physical Devices and Endpoints - Arduino UNO: Introduction to Arduino, Arduino UNO, Installing the Software, Fundamentals of Arduino Programming. IoT Physical Devices and Endpoints - RaspberryPi: Introduction to RaspberryPi, About the RaspberryPi Board: Hardware Layout, Operating Systems on RaspberryPi, Configuring RaspberryPi, Programming RaspberryPi with Python, Wireless Temperature Monitoring System Using Pi, DS18B20 Temperature Sensor, Connecting Raspberry Pi via SSH, Accessing Temperature from DS18B20 sensors, Remote access to RaspberryPi, Smart and Connected Cities, An IoT Strategy for Smarter Cities, Smart City IoT Architecture, Smart City Security Architecture, Smart City Use-Case Examples.	**10 Hours**

Syllabus- Internet Of Things Technology

Question paper pattern:

The question paper will have ten questions.
There will be 2 questions from each module.
Each question will have questions covering all the topics under a module.
The students will have to answer 5 full questions, selecting one full question from each module.

Text Books:

1. David Hanes, Gonzalo Salgueiro, Patrick Grossetete, Robert Barton, Jerome Henry,"**IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things**", 1stEdition, Pearson Education (Cisco Press Indian Reprint). (**ISBN:** 978-9386873743)
2. Srinivasa K G, "**Internet of Things**",CENGAGE Leaning India, 2017

Reference Books:

1. Vijay Madisetti and ArshdeepBahga, "**Internet of Things (A Hands-on-Approach)**", 1stEdition, VPT, 2014. (**ISBN:** 978-8173719547)
2. Raj Kamal, "**Internet of Things: Architecture and Design Principles**", 1st Edition, McGraw Hill Education, 2017. (**ISBN:** 978-9352605224)

Module – 5 IoT Physical Devices and End Point-Aurdino Uno

Arduino:

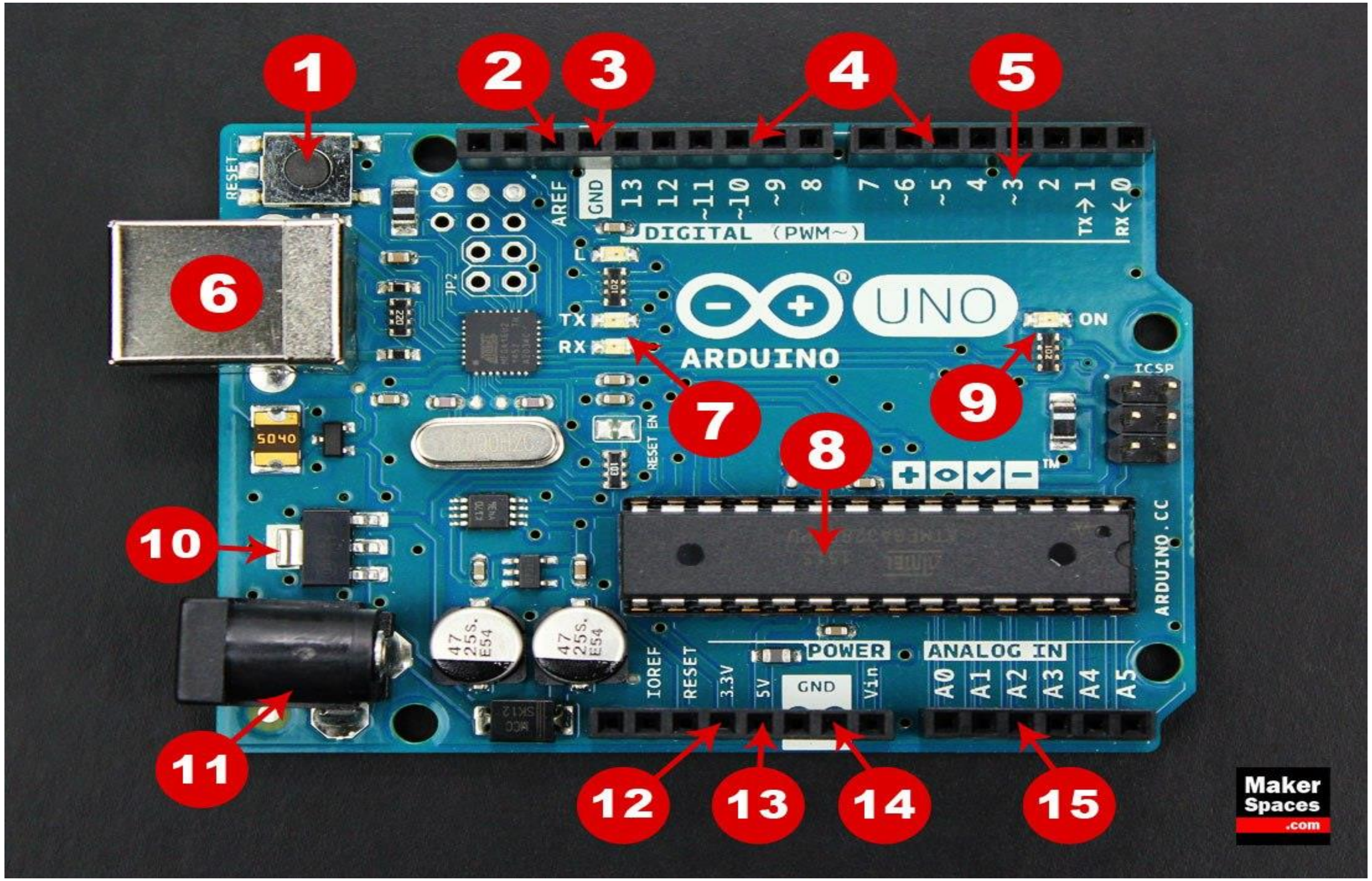
- **Arduino is an open-source electronics platform based on easy-to-use hardware and software.**

- **Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.**

Arduino UNO:

- **Arduino Uno is a microcontroller board based on the ATmega328P.**
- **It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.**
- **"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases.**

Module – 5 IoT Physical Devices and End Point- Aurdino Uno



Module – 5 IoT Physical Devices and End Point- Aurdino Uno

1. **Reset Button** – This will restart any code that is loaded to the Arduino board

2. **AREF** – Stands for "Analog Reference" and is used to set an external reference voltage

3. **Ground Pin** – There are a few ground pins on the Arduino and they all work the same

4. **Digital Input/Output** – Pins 0-13 can be used for digital input or output

5. **PWM** – The pins marked with the (~) symbol can simulate analog output

6. **USB Connection** – Used for powering up your Arduino and uploading sketches

7. **TX/RX** – Transmit and receive data indication LEDs

8. **ATmega Microcontroller** – This is the brains and is where the programs are stored

9. **Power LED Indicator** – This LED lights up anytime the board is plugged in a power source

10. **Voltage Regulator** – This controls the amount of voltage going into the Arduino board

11. **DC Power Barrel Jack** – This is used for powering your Arduino with a power supply

12. **3.3V Pin** – This pin supplies 3.3 volts of power to your projects

13. **5V Pin** – This pin supplies 5 volts of power to your projects

14. **Ground Pins** – There are a few ground pins on the Arduino and they all work the same

15. **Analog Pins** – These pins can read the signal from an analog sensor and convert it to digital

Fundamentals of Arduino Programming:

Two required functions / methods / routines:

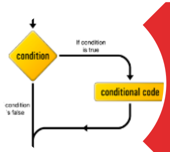
```
void setup()
{
	// runs once
}

void loop()
{
	// repeats
}
```

BIG 6 CONCEPTS

digitalWrite()

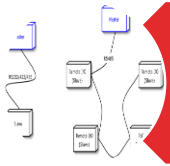
analogWrite()

digitalRead()



if() statements / Boolean

analogRead()

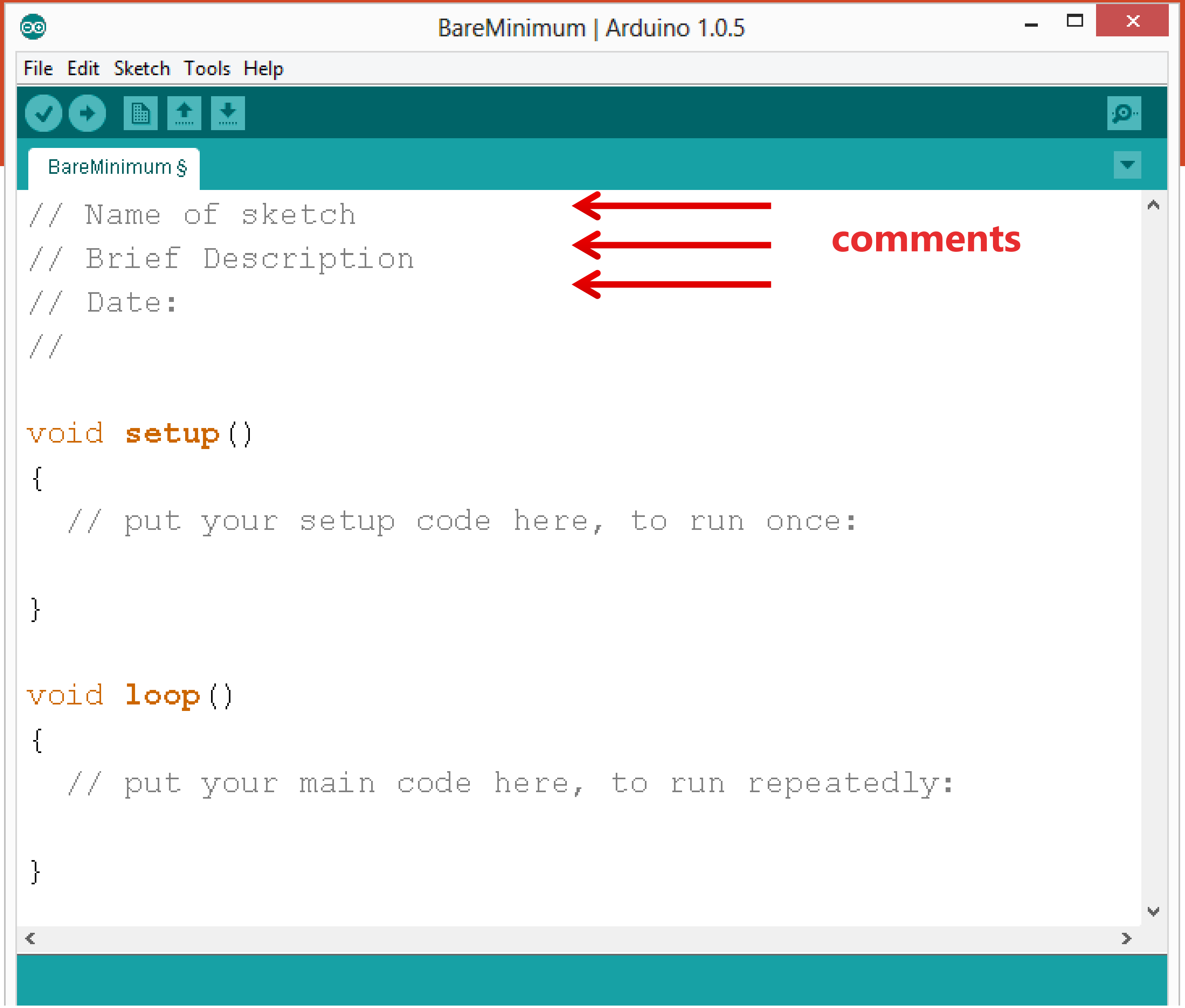


Serial communication

Comments, Comments, Comments

Comments

```
// this is for single line comments

// it's good to put at the top and before anything 'tricky'

/* this is for multi-line comments

   Like this…

   And this….

*/
```


BareMinimum | Arduino 1.0.5
File Edit Sketch Tools Help
BareMinimum §
// Name of sketch
// Brief Description
// Date:
//
void setup()
{
// put your setup code here, to run once:
}
void loop()
{
// put your main code here, to run repeatedly:
}
comments

Three commands to know...

```
pinMode(pin, INPUT/OUTPUT);

      ex: pinMode(13, OUTPUT);

digitalWrite(pin, HIGH/LOW);

      ex: digitalWrite(13, HIGH);

delay(time_ms);

      ex: delay(2500); // delay of 2.5 sec.
```

```
// NOTE: -> commands are CASE-sensitive
```

Arduino Code Basics

Arduino programs run on two basic sections:

```
void setup() {

     //setup motors, sensors etc

}
void loop() {

     // get information from sensors
     // send commands to motors

}
```

SETUP

The setup section is used for assigning input and outputs (Examples: motors, LED’s, sensors etc) to ports on the Arduino

It also specifies whether the device is OUTPUT or INPUT

To do this we use the command “pinMode”

16

SETUP

```
void setup() {

    pinMode(9, OUTPUT);

}
```

port #

Input or Output

LOOP

```
void loop() {

    digitalWrite(9, HIGH);
    delay(1000);
    digitalWrite(9, LOW);
    delay(1000);

}
```

Port # from setup

Turn the LED on
or off

Wait for 1 second
or 1000 milliseconds

DECLARING A VARIABLE

```
int val = 5;
```

Type

variable name

assignment
"becomes"

value

USING VARIABLES

```
int delayTime = 2000;
int greenLED = 9;
void setup() {

      pinMode(greenLED, OUTPUT);

}

void loop() {

      digitalWrite(greenLED, HIGH);
      delay(delayTime);
      digitalWrite(greenLED, LOW);
      delay(delayTime);

}
```

Declare delayTime Variable

Use delayTime Variable

Using Variables

```
int delayTime = 2000;
int greenLED = 9;

void setup() {
      pinMode(greenLED, OUTPUT);
}
void loop() {
      digitalWrite(greenLED, HIGH);
      delay(delayTime);
      digitalWrite(greenLED, LOW);
 delayTime = delayTime - 100;
      delay(delayTime);
}
```

subtract 100 from delayTime to gradually increase LED's blinking speed

Conditions

To make decisions in Arduino code we use an ‘if’ statement

‘If’ statements are based on a TRUE or FALSE question

VALUE COMPARISONS

GREATER THAN

a > b

GREATER THAN OR EQUAL

a >= b

LESS

a < b

LESS THAN OR EQUAL

a <= b

EQUAL

a == b

NOT EQUAL

a != b

IF Condition

```
if(true)
{
    "perform some action"
}
```

IF Example

```
int counter = 0;

void setup() {
    Serial.begin(9600);
}
void loop() {

    if(counter < 10)
    {
        Serial.println(counter);
    }
    counter = counter + 1;


}
```

Input & Output

Transferring data from the computer to an Arduino is done using Serial Transmission

To setup Serial communication we use the following

```
void setup() {
    Serial.begin(9600);
}
```


26

Writing to the Console

```
void setup() {

     Serial.begin(9600);
 Serial.println("Hello World!");

}

void loop() {}
```

IF - ELSE Condition

```
if( “answer is true”)
{
    “perform some action”
}
else
{
    “perform some other action”
}
```

IF - ELSE Example

```
int counter = 0;
void setup() {
    Serial.begin(9600);
}
void loop() {
    if(counter < 10)
    {
        Serial.println("less than 10");
    }
    else
    {
        Serial.println("greater than or equal to 10");
        Serial.end();
    }
    counter = counter + 1;}
```

IF - ELSE IF Condition

```
if( “answer is true”)
{
    “perform some action”
}
else if( “answer is true”)
{
    “perform some other action”
}
```

IF - ELSE Example

```
int counter = 0;
void setup() {
	Serial.begin(9600);
}
void loop() {

	if(counter < 10)
	{
		Serial.println("less than 10");
	}
	else if (counter == 10)
	{
		Serial.println("equal to 10");
	}
	else
	{
		Serial.println("greater than 10");
		Serial.end();
	}
	counter = counter + 1;
}
```

BOOLEAN OPERATORS - AND

If we want all of the conditions to be true we need to use ‘AND’ logic (AND gate)

We use the symbols &&

- **Example**

```
if ( val > 10 && val < 20)
```


32

BOOLEAN OPERATORS - OR

If we want either of the conditions to be true we need to use ‘OR’ logic (OR gate)

We use the symbols ||

- **Example**

```
if ( val < 10 || val > 20)
```


33

BOOLEAN VARIABLES

```
boolean done = true;
```

```
boolean done = false;
void setup() {
Serial.begin(9600);}
void loop() {
  if(!done)  {
Serial.println("HELLOWORLD");
    done = true;  }
}
```

Important functions

- Serial.println(value);
  - Prints the value to the Serial Monitor on your computer
- pinMode(pin, mode);
  - Configures a digital pin to read (input) or write (output) a digital value
- digitalRead(pin);
  - Reads a digital value (HIGH or LOW) on a pin set for input
- digitalWrite(pin, value);
  - Writes the digital value (HIGH or LOW) to a pin set for output

OUTLINE

- Essential Programming Concepts
  - Delay
  - Infinite Loop
- General Input/Output
  - Polling or Busy/Wait I/O
  - Interrupt Processing
- Timers and Internal Inteerrupts
- High-Level Language Extensions
- Code Transformations for Embedded Computing
  - Loop Unrolling
  - Loop Merging
  - Loop Peeling
  - Loop Tiling

DELAY (1/3)

- Delays are essential in embedded systems, unlike high-performance systems where we want the program to execute as fast as possible
- Delays are used to synchronize events, or read inputs with a specific sampling freqency (more on Bus/Wait I/O)

DELAY (3/3)

- Okay, so how do we build a delay function?
- Our reference is the system clock frequency
- We use a register or a timer to measure ticks
- Each tick is 1/frequency
- Example: Assuming a 16-bit processor, an increment and a jump instruction is 1-cycle each and a 10 MHz system clock, build a 1-sec delay:
- T = 1/10 MHz = 100 ns
- 1 s/100 ns = 10,000,000

```
     int i=5000000; //2 ticks per iteration
BACK: i--;
     if (i!=0) goto BACK;
```

Infinite Loop (1/2)

- Embedded Systems are mostly single-functioned
- Their core application never terminates
- Infinite loops are not forbidden as long as they are done correctly

Infinite Loop (2/2)

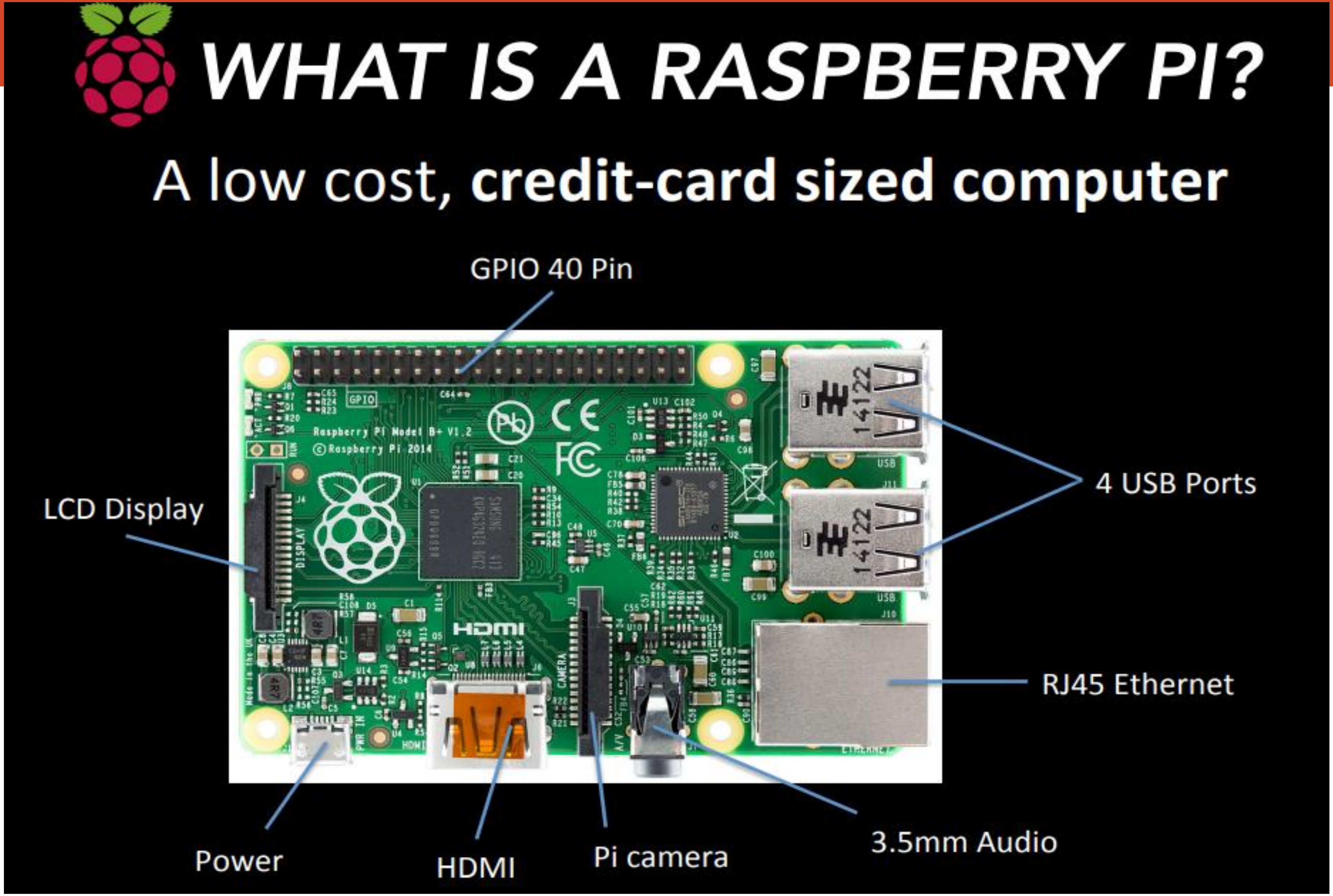
```
void main()
{
  light enum {RED, ORANGE, GREEN};
  loop: light = RED;  //no exit from loop!
      delay(20000); //20-sec red
      light = ORANGE;
      delay(2000);  //2-sed orange
      light = GREEN;
      delay(20000); //20-sec green
      goto loop;    //just repeat sequence
}
```

RaspberryPi:

- **Raspberry Pi is the name of a series of single-board computers made by the Raspberry Pi Foundation, a UK charity that aims to educate people in computing and create easier access to computing education.**

- **The Raspberry Pi launched in 2012, and there have been several iterations and variations released since then. The original Pi had a single-core 700MHz CPU and just 256MB RAM, and the latest model has a quad-core 1.4GHz CPU with 1GB RAM. The main price point for Raspberry Pi has always been $35 and all models have been $35 or less, including the Pi Zero, which costs just $5.**

Module – 5 IoT Physical Devices and End



Module – 5 IoT Physical Devices and End

BASIC PI COMMAND LINE - 1

Useful commands to run from a terminal or command line.

rasp-config
Change your pi configuration settings.

startx
Start the GUI (Graphical User Interface)

ifconfig
Get the details of your Ethernet or wireless network adapter.

rpi-update
Updates your Raspberry Pi firmware.

ssh
Connect your pi to other computers.

sudo
Run commands as super user.

shutdown
This will shutdown your pi.

nano
This is your text editor for changing or adding files. Save, edit, create.

Module – 5 IoT Physical Devices and End

BASIC PI COMMAND LINE - 2

Useful commands to run from a terminal or command line

ls
List out the current directory files.

cd
Go to directory or folder.

find
Searches whole system for files or directories.

clear
Clears the terminal screen.

mv
Move files or folders.

touch
Create a blank file.

mkdir
Create a directory.

ping
Test connectivity between two devices.

df -h
Shows disk space.

iwconfig
Wireless configuration tool.

Module – 5 IoT Physical Devices and End

Hardware Layout:

raspi-config

- The Raspberry Pi configuration tool in Raspbian, allowing you to easily enable features such as the camera, and to change your specific settings such as keyboard layout

config.txt

- The Raspberry Pi configuration file

Wireless networking

- Configuring your Pi to connect to a wireless network using the Raspberry Pi 3's or Pi Zero W's inbuilt wireless connectivity, or a USB wireless dongle

Wireless access point

- Configuring your Pi as a wireless access point using the Raspberry Pi 3 and Pi Zero W's inbuilt wireless connectivity, or a USB wireless dongle

Module – 5 IoT Physical Devices and End Point-Aurdino Uno

Using a proxy

- **Setting up your Pi to access the internet via a proxy server**

HDMI Config

- **Guide to setting up your HDMI device, including custom settings**

Audio config

- **Switch your audio output between HDMI and the 3.5mm jack**

Camera config

- **Installing and setting up the Raspberry Pi camera board**

External storage config

- **Mounting and setting up external storage on a Raspberry Pi**

Module – 5 IoT Physical Devices and End Point-Aurdino Uno

Localisation

- **Setting up your Pi to work in your local language/time zone**

Default pin configuration

- **Changing the default pin states.**

Device Trees config

- **Device Trees, overlays, and parameters**

Kernel command line

- **How to set options in the kernel command line**

UART configuration

- **How to set up the on-board UARTS.**

Firmware warning icons

- **Description of warning icons displayed if the firmware detects issues**

Module – 5 IoT Physical Devices and End

LED warning flash codes

- Description of the meaning of LED warning flashes that are shown if a Pi fails to boot or has to shut down

Securing your Raspberry Pi

- Some basic advice for making your Raspberry Pi more secure

Screensaver

- How to configure screen blanking/screen saver

The boot folder

- What it's for and what's in it

Operating Systems on RaspberryPi:

You have a few options when it comes to interacting with the Raspberry Pi. The first and most common is to use it like you would a full desktop computer (just smaller). This involves connecting a keyboard, mouse, and monitor. With this setup, you are likely best served by installing Raspbian with Desktop, which gives you a full graphical user interface(GUI) to work with. This is the best option if you want an experience similar to working with other operating systems (OS), such as Windows, macOS, or other popular Linux flavors, like Ubuntu.

Programming RaspberryPi with Python:

The Raspberry Pi is an amazing single board computer (SBC) capable of running Linux and a whole host of applications. Python is a beginner-friendly programming language that is used in schools, web development, scientific research, and in many other industries.

Module – 5 IoT Physical Devices and End

Wireless Temperature Monitoring System Using Pi:

Raspberry Pi which having inbuilt wi-fi, which makes Raspberry Pi to suitable for IoT applications, so that by using IoT technology this monitoring system works by uploading the temperature value to the Thingspeak cloud by this project you can able to learn to how to handle cloud-based application using API keys. In this monitoring system, we used Thingspeak cloud, the cloud which is suitable to view the sensor logs in the form of graph plots. Here we created one field to monitor the temperature value, that can be reconfigurable to monitor a number of sensor values in various fields. This basic will teach you to how to work with a cloud by using LM35 as a temperature sensor, to detect the temperature and to upload those values into the cloud.

Module – 5 IoT Physical Devices and End

HARDWARE REQUIRED

- Raspberry Pi
- SD card
- Power supply
- VGA to HDMI converter (Optional)
- MCP3008 (ADC IC)
- A temperature sensor(LM35)

SOFTWARE REQUIRED

- Raspbian Stretch OS
- SD card Formatter
- Win32DiskImager (or) Etcher

PYTHON LIBRARIES USED

- RPi.GPIO as GPIO (To access the GPIO Pins of Raspberry Pi)
- Time library (For Time delay)
- Urllib2 to handle URL using Python programming

DS18B20 Temperature Sensor

The DS18B20 is a 1-wire programmable Temperature sensor from maxim integrated. It is widely used to measure temperature in hard environments like in chemical solutions, mines or soil etc. The constriction of the sensor is rugged and also can be purchased with a waterproof option making the mounting process easy. It can measure a wide range of temperature from -55°C to +125° with a decent accuracy of ±5°C. Each sensor has a unique address and requires only one pin of the MCU to transfer data so it a very good choice for measuring temperature at multiple points without compromising much of your digital pins on the microcontroller.

Module – 5 IoT Physical Devices and End

DS18B20 Temperature Sensor

Applications:

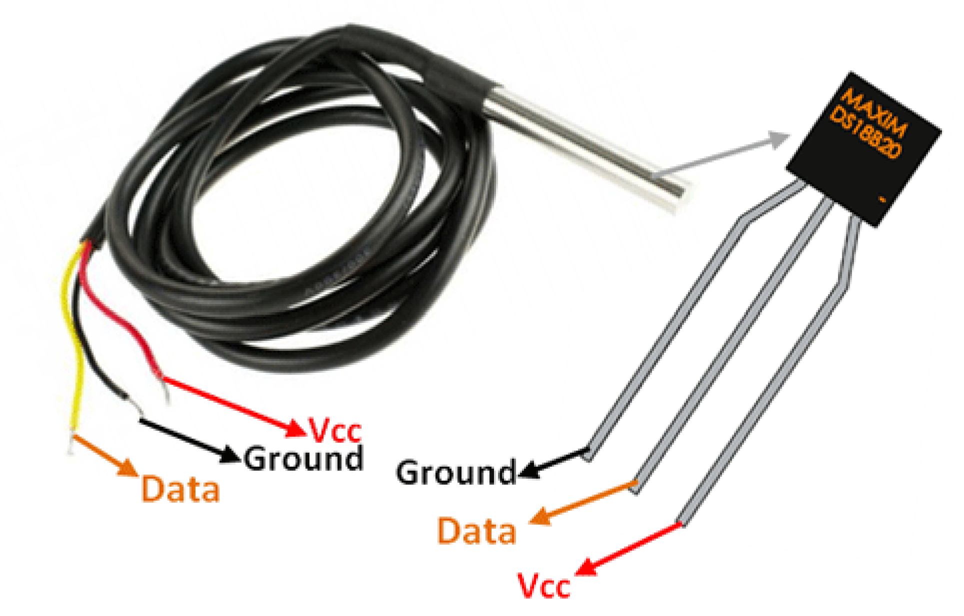
- **Measuring temperature at hard environments**
- **Liquid temperature measurement**
- **Applications where temperature has to be measured at multiple points**

Pin Configuration:

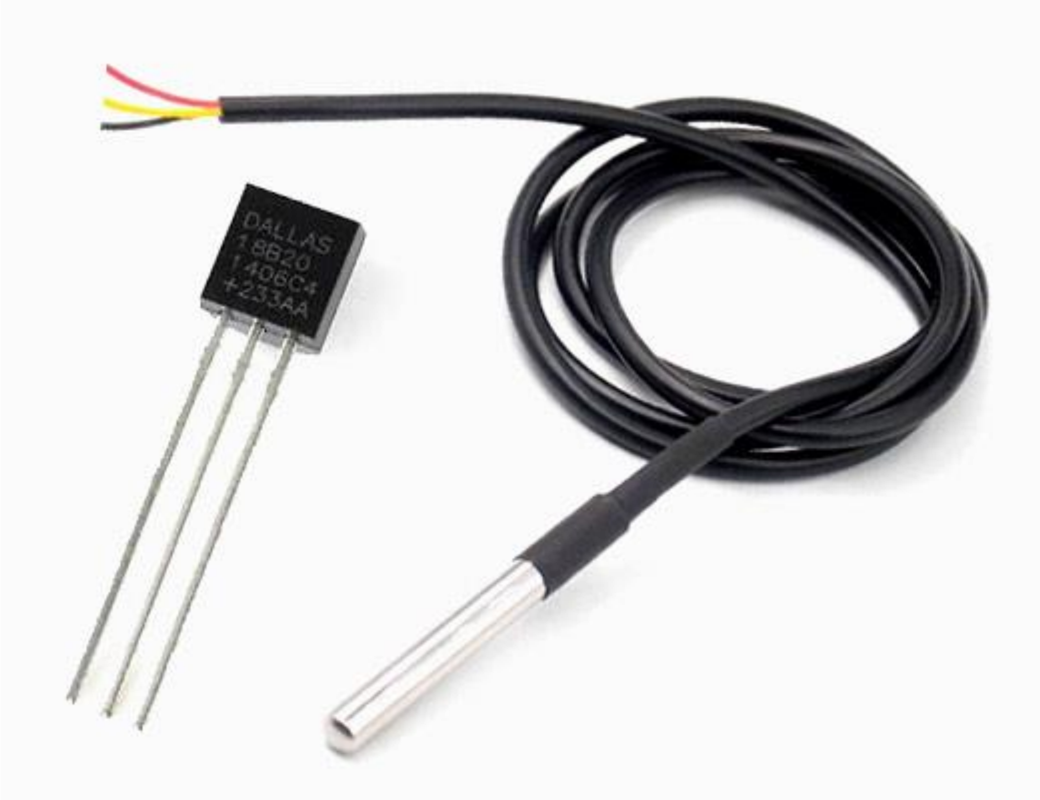
No	Pin Name	Description
1	Ground	Connect to the ground of the circuit
2	Vcc	Powers the Sensor, can be 3.3V or 5V
3	Data	This pin gives output the temperature value which can be read using 1-wire method

Module – 5 IoT Physical Devices and End

DS18B20 Temperature Sensor Pinout



DS18B20 Temperature Sensor



Module – 5 IoT Physical Devices and End

Connecting Raspberry Pi via SSH:

You can access the command line of a Raspberry Pi remotely from another computer or device on the same network using SSH.The Raspberry Pi will act as a remote device: you can connect to it using a client on another machine.

1. Set up your local network and wireless connectivity

2. Enable SSH

3. Enable SSH on a headless Raspberry Pi (add file to SD card on another machine)

4. Set up your client

Module – 5 IoT Physical Devices and End

Accessing Temperature from DS18B20 sensors:

The DS18B20 is a digital thermometer that allows to get 9-bit to 12-bit Celsius temperature measurements (programmable resolution). The temperature conversion time depends on the resolution used. For a 9-bit resolution it takes at most 93.75 ms and for a 12-bit resolution it takes at most 750 ms.The device is able to measure temperatures from -55°C to +125°C and has a ±0.5°C accuracy in the range from -10°C to +85°C.

Additionally, it has an alarm functionality with programmable upper and lower temperature trigger points. These thresholds are stored internally in non-volatile memory, which means they are kept even if the device is powered off .

The sensor communicates using the OneWire protocol, which means it only requires a pin from a microcontroller to be connected to it. Furthermore, each sensor has a unique 64-bit serial code, allowing multiple DS18B20 devices to function on the same OneWire bus.In terms of power supply, the device can operate with a voltage between 3.0 V and 5.5 V, which means it can operate with the same voltage of the ESP32 without the need for level conversion.

Module – 5 IoT Physical Devices and End

Remote access to RaspberryPi:

To access a Raspberry Pi (or any home computer for that matter) from outside your home network, you'd usually need to jump through a lot of hoops, get an IP address, and tweak a few settings on your home router. If you just need to control a few simple things on your Raspberry Pi, that's overkill. We're going to outline two methods that skip all of that.
The first thing you need to do is get your Raspberry Pi set up and connected to your home network. Since you're exposing your Raspberry Pi to the internet, be sure you change your default password during the set up process. Once that's done, come back here to set up everything else.

Remote Log Into Your Raspberry Pi's Full Operating System Using VNC Connect:

VNC has long been the best way to access any computer remotely on the same network. Recently, VNC Connect came out to make it easy to access your Raspberry Pi from anywhere using a cloud connection. Once it's set up, you can access your Raspberry Pi's graphic interface from any other computer or smartphone using the VNC Viewer app.

Module – 5 IoT Physical Devices and End

RaspberryPi Interface:

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. In an SPI connection, there is one master device and one or more peripheral devices. There are five pins on Raspberry Pi for SPI interface:

- ✓ MISO(Master In Slave Out): Master line for sending data to the peripherals.
- ✓ MOSI(Master out Slave In): Slave line for sending data to the master.
- ✓ SCK(Serial Clock): Clock generated by master to synchronize data transmissions.
- ✓ CE0(Chip Enable 0): To enable or disable devices.
- ✓ CE0(Chip Enable 1): To enable or disable 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).

Module – 5 IoT Physical Devices and End

RaspberryPi OS: (Not linux)

1. RISC OS Pi
2. Free BSD
3. NetBSD
4. Plan 9
5. Haiku

RaspberryPi OS: (Linux based)

1. Xbean
2. Open SUSE
3. Arc OS
4. Kano OS
5. Nard SDX

RaspberryPi OS: (Media center based)

1. OSMC
2. OpenELEC
3. LitreELEC
4. Xbian
5. Rasplex

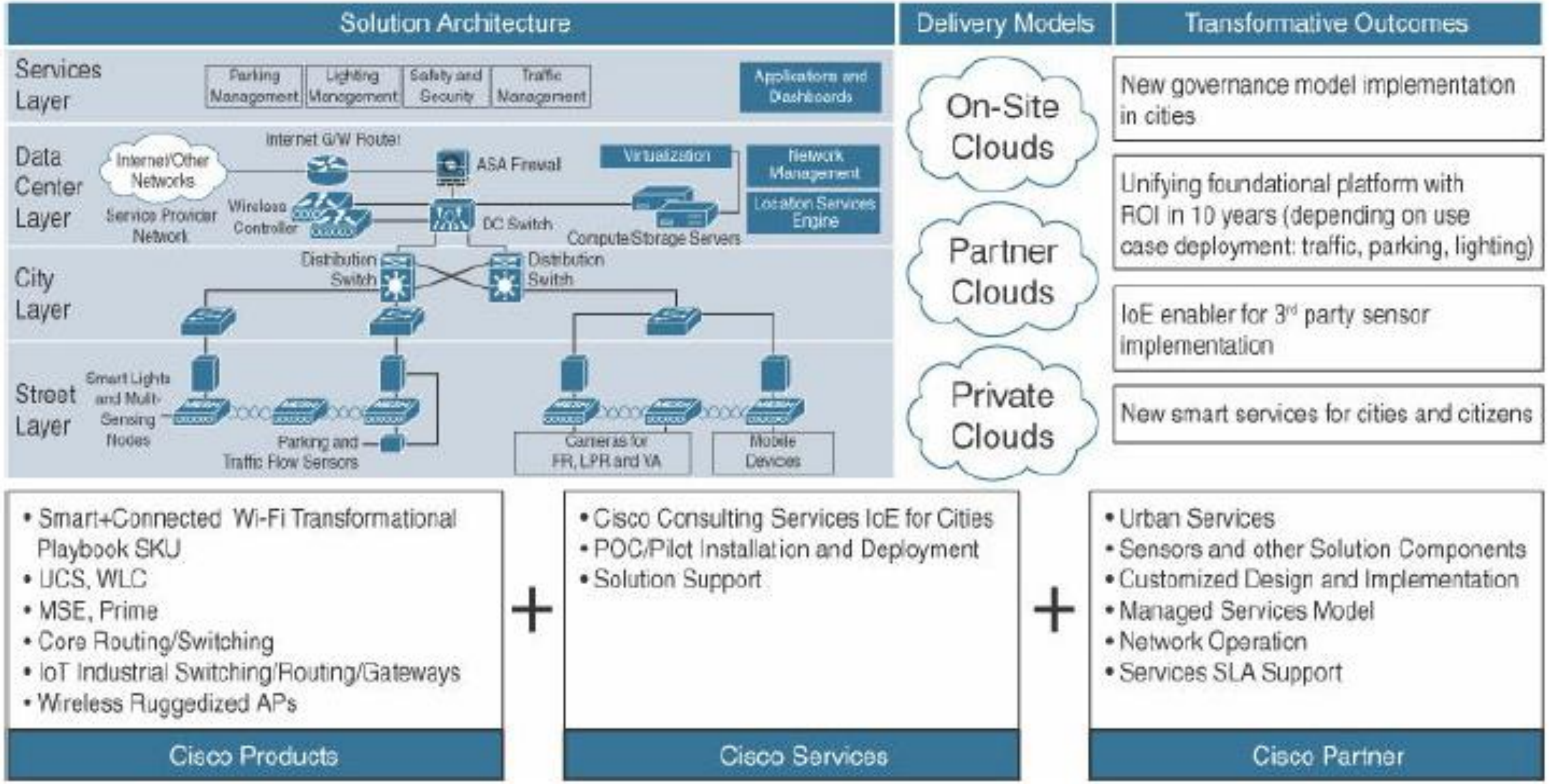
RaspberryPi OS: (Audio based)

1. Volumio
2. Pimusixbox
3. Runeaudio

Module – 5 Smart City IoT Architecture:

- **A smart city IoT infrastructure is a four-layered architecture, as shown in Figure**
- **Data flows from devices at the street layer to the city network layer and connect to the data center layer, where the data is aggregated, normalized, and virtualized.**
- **The data center layer provides information to the services layer, which consists of the applications that provide services to the city.**
- **In smart cities, multiple services may use IoT solutions for many different purposes. These services may use different IoT solutions, with different protocols and different application languages**

Module – 5 Smart City IoT Architecture:



Smart Cities Layered Architecture

Module – 5 Smart City IoT Architecture:

Street Layer:

- **The street layer is composed of devices and sensors that collect data and take action based on instructions from the overall solution, as well as the networking components needed to aggregate and collect data.**
- **A sensor is a data source that generates data required to understand the physical world. Sensor devices are able to detect and measure events in the physical world.**
- **ICT connectivity solutions rely on sensors to collect the data from the world around them so that it can be analyzed and used to operationalize use cases for cities.**

Module – 5 Smart City IoT Architecture:

Street Layer:

A variety of sensors are used at the street layer for a variety of smart city use cases. Here is a short representative list:

- A magnetic sensor can detect a parking event by analyzing changes in the surrounding magnetic field when a heavy metal object, such as a car or a truck, comes close to it (or on top of it).
- A lighting controller can dim and brighten a light based on a combination of time-based and ambient conditions.
- Video cameras combined with video analytics can detect vehicles, faces, and traffic conditions for various traffic and security use cases.
- An air quality sensor can detect and measure gas and particulate matter concentrations to give a hyper-localized perspective on pollution in a given area.
- Device counters give an estimate of the number of devices in the area, which provides a rough idea of the number of vehicles moving or parked in a street or a public parking area, of pedestrians on a sidewalk, or even of birds in public parks or on public monuments—for cities where bird control has become an issue.

Module – 5 Smart City IoT Architecture:

City Layer:

- **At the city layer, which is above the street layer, network routers and switches must be deployed to match the size of city data that needs to be transported.**
- **This layer aggregates all data collected by sensors and the end-node network into a single transport network.**
- **The city layer may appear to be a simple transport layer between the edge devices and the data center or the Internet.**

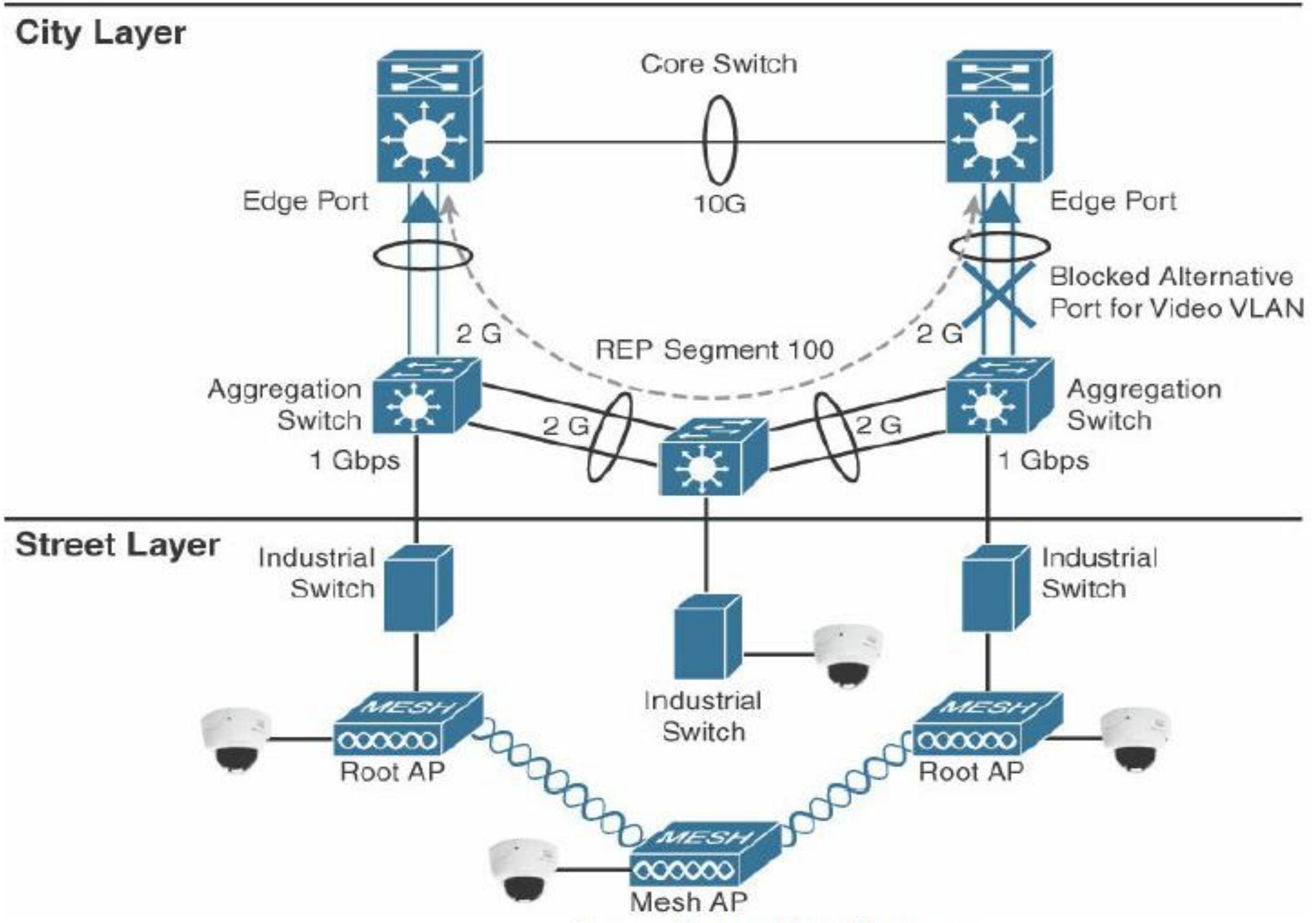
Module – 5 Smart City IoT Architecture:

City Layer:

- **However, one key consideration of the city layer is that it needs to transport multiple types of protocols, for multiple types of IoT applications. Some applications are delay- and jitter sensitive, and some other applications require a deterministic approach to frame delivery.**

- **A missed packet may generate an alarm or result in an invalid status report. As a result, the city layer must be built around resiliency, to ensure that a packet coming from a sensor or a gateway will always be forwarded successfully to the headend station.**

Module – 5 Smart City IoT Architecture:



Street Layer Resiliency

Module – 5 Smart City IoT Architecture:

City Layer:

- In this model, at least two paths exist from any aggregation switch to the data center layer. A common protocol used to ensure this resiliency is Resilient Ethernet Protocol (REP).

Module – 5 Smart City IoT Architecture:

Data Center Layer:

- **Ultimately, data collected from the sensors is sent to a data center, where it can be processed and correlated.**
- **Based on this processing of data, meaningful information and trends can be derived, and information can be provided back.**
- **For example, an application in a data center can provide a global view of the city traffic and help authorities decide on the need for more or less common transport vehicles. At the same time, an automated response can be generated**

Module – 5 Smart City IoT Architecture:

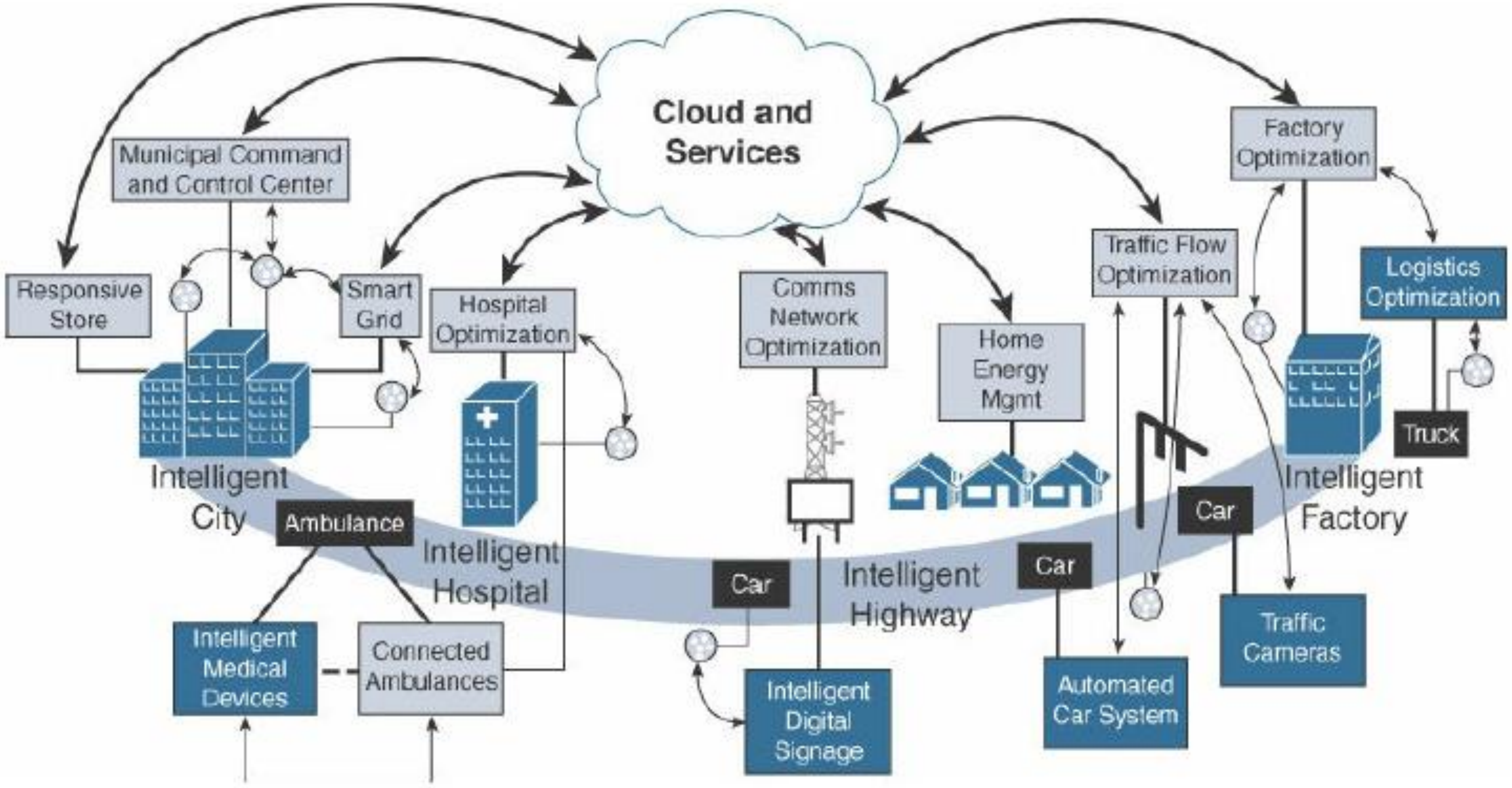
Data Center Layer:

- **The cloud model is the chief means of delivering storage, virtualization, adaptability, and the analytics know-how that city governments require for the technological mashup and synergy of information embodied in a smart city.**
- **Traditional city networks simply cannot keep up with the real-time data needs of smart cities; they are encumbered by their physical limitations.**
- **The cloud enables data analytics to be taken to server farms with large and extensible processing capabilities.**

Module – 5 Smart City IoT Architecture:

Data Center Layer:

Figure shows the vision of utilizing the cloud in smart solutions for cities. The cloud provides a scalable, secure, and reliable data processing engine that can handle the immense amount of data passing through it.



Module – 5 Smart City IoT Architecture:

Service Layer:

- **Ultimately, the true value of ICT connectivity comes from the services that the measured data can provide to different users operating within a city.**
- **Smart city applications can provide value to and visibility for a variety of user types, including city operators, citizens, and law enforcement.**
- **The collected data should be visualized according to the specific needs of each consumer of that data and the particular user experience requirements and individual use cases.**

Smart City Security Architecture:

- **A serious concern of most smart cities and their citizens is data security.**
- **Vast quantities of sensitive information are being shared at all times in a layered, real-time architecture, and cities have a duty to protect their citizens' data from unauthorized access, collection, and tampering.**
- **In general, citizens feel better about data security when the city itself, and not a private entity, owns public or city-relevant data.**
- **It is up to the city and the officials who run it to determine how to utilize this data.**

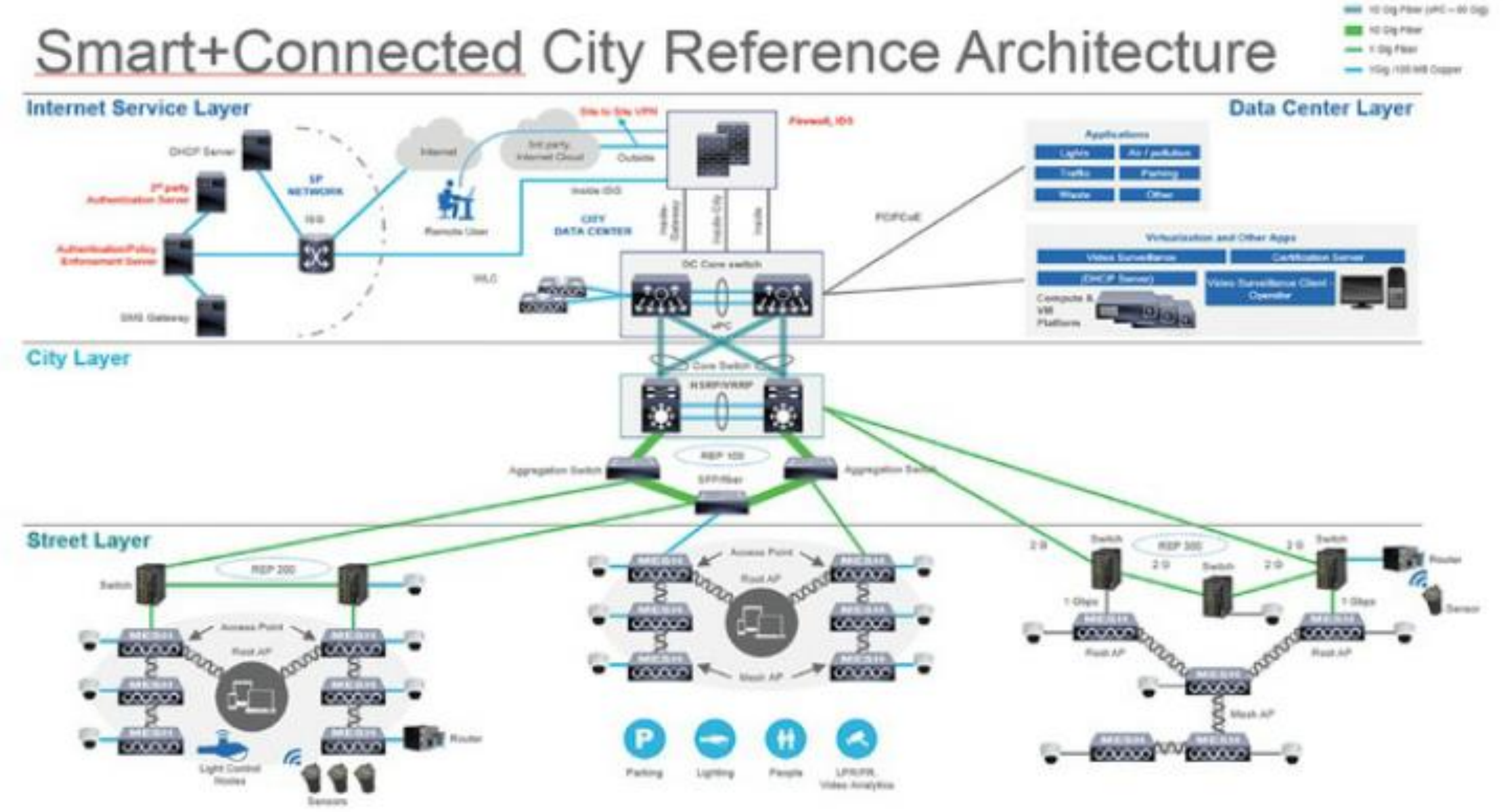
Module – 5 Smart City Security Architecture

Smart City Security Architecture:

- A security architecture for smart cities must utilize security protocols to fortify each layer of the architecture and protect city data.
- Figure shows a reference architecture, with specific security elements highlighted.
- Security protocols should authenticate the various components and protect data transport throughout.

Module – 5 Smart City Security Architecture

Smart City Security Architecture:



Key Smart and Connected Cities Reference Architecture

Module – 5 Smart City Security Architecture

Smart City Security Architecture:

- Starting from the street level, sensors should have their own security protocols.
- Some industry-standard security features include device/sensor identification and authorization; device/sensor data encryption; Trusted Platform Module, which enables self-destruction when the sensor is physically handled; and user ID authentication and authorization.

Module – 5 Smart City Security Architecture

Smart City Security Architecture:

- **Sensor identification and authorization typically requires a pre-installed factory X.509 certificate and public key infrastructure (PKI) at the organization level, where a new certificate is installed through a zero-touch deployment process.**

- **This additional processing may slow the deployment but ensures the security of the exchanges.**

- **Another consideration may be the type of data that the sensor is able to collect and process. For example, a roadside car counter may include a Bluetooth sensor that uniquely identifies each driver or pedestrian**

Module – 5 Smart City Security Architecture

Smart City Security Architecture:

The city layer transports data between the street layer and the data center layer. It acts as the network layer. The following are common industry elements for security on the network layer:

- **Firewall:** A firewall is located at the edge, and it should be IPsec- and VPN-ready, and include user- and role-based access control. It should also be integrated with the architecture to give city operators remote access to the city data center.
- **VLAN:** A VLAN provides end-to-end segmentation of data transmission, further protecting data from rogue intervention. Each service/domain has a dedicated VLAN for data transmission.
- **Encryption:** Protecting the traffic from the sensor to the application is a common requirement to avoid data tampering and eavesdropping. In most cases, encryption starts at the sensor level. In some cases, the sensor-to-gateway link uses one type of encryption, and the gateway-to-application connection uses another encryption (for example, a VPN).