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.

Textbook:

1. Srinivasa K G, “Internet of Things”, CENGAGE Leaning India, 2017.

5.1. IOT PHYSICAL DEVICES AND ENDPOINTS - ARDUINO UNO

- Arduino is an open source advancement prototyping platform which depends on simple to utilize equipment and programming.
- Arduino is a basic single board microcontroller designed to make applications, interactive controls, or environments easily adaptive.
- 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.
- The hardware consists of a board designed around an 8-bit microcontroller, or a 32-bit ARM (Advanced RISC Machines).
- Arduino Uno is a microcontroller board based on the ATmega328P.
- It has 14 digital input/output pins (of which 6 can be used as Pulse Width Modulation (PWM) outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an In Circuit Serial Programming (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.

5.1.1. Why Arduino?

- ❖ Inexpensive:
  - o Arduino boards are relatively inexpensive compared to other microcontroller platforms.
  - o The least expensive version of the Arduino module can be assembled by hand.
- ❖ Cross-platform:
  - o The Arduino software runs on Windows, Macintosh OS and Linux operating systems.
- ❖ Simple, clear programming environment:
  - o The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well.
- ❖ Open source and extensible software:
  - o The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries.
  - o The Arduino is based on Atmel's ATMEGA microcontrollers.
  - o Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

- ❖ Entry Level
    - Easy to use and ready to first creative projects.
    - These boards and modules are the best to start learning and tinkering with electronics and coding.
- ❖ Enhanced Features
    - Experience the excitement of more complex projects, with advanced functionalities, or faster performances.
- ❖ Internet of Things
    - Make connected devices easily with IoT and the world wide web.
- ❖ Wearable
    - Add smartness to projects and sewing the power of electronics directly to textiles.

5.1.2. Which Arduino?

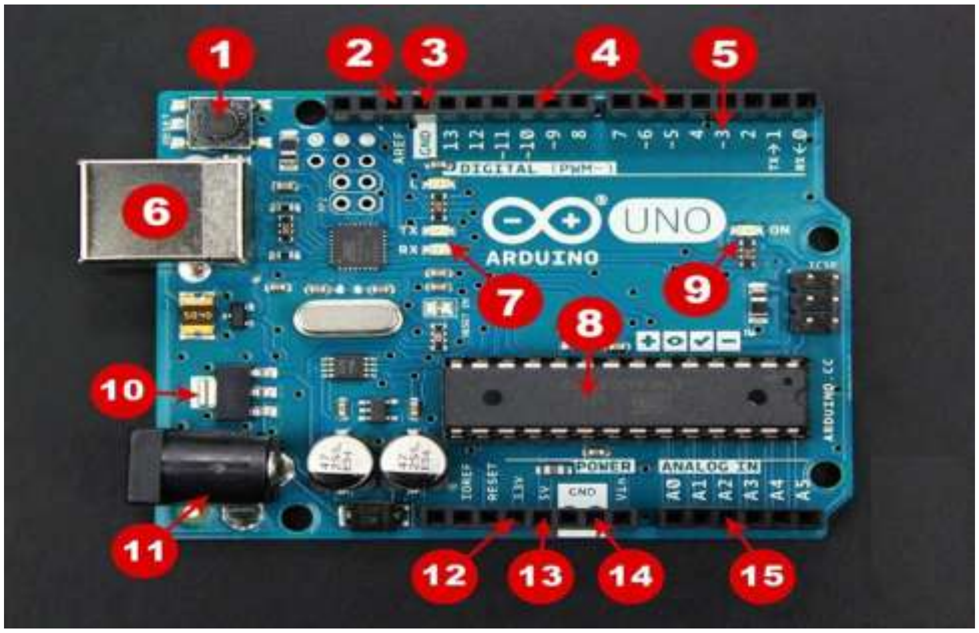
- Over ten years, since Arduino released, hundreds of Arduino boards are available in the market.
- Out of which Arduino UNO which is used in almost 99% of project use.
- Some of the boards from Arduino family are
    - ARDUINO UNO
        - A microcontroller board based on the ATmega328P.
        - 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.
        - Connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
    - ARDUINO MEGA 2560
        - A microcontroller board based on the ATmega2560.
        - It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.
        - It is the recommended board for 3D printers and robotics projects.
    - ARDUINO MICRO
        - A microcontroller board based on the ATmega32U4, featuring a built-in USB which makes the Micro recognizable as a mouse or keyboard.

- It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, an ICSP header, and a reset button.

- ARDUINO MKR1000
  - It is based on the Atmel ATSAMW25 ARM SoC (System on Chip), that is part of the Smart Connect family of Atmel Wireless devices, specifically designed for IoT projects and devices.
  - The ATSAMW25 is composed of three main blocks:
    - ✓ SAMD21 Cortex-M0+ 32bit low power ARM MCU
    - ✓ WINC1500 low power 2.4GHz IEEE® 802.11 b/g/n Wi-Fi
    - ✓ ECC508 Crypto Authentication
    - ✓ PCB Antenna.

5.2. EXPLORING ARDUINO UNO LEARNING BOARD

- Figure 5.1 shows an Arduino board



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

5.2.1. Things that Arduino can do

- can control with Arduino is an LED.
- Display a message in LCD Display
- Control DC or Servo Motors
- Read Data from outside world
- Motion sensor allows us to detect movement

- Light sensor allows to measure the quantity of light outside world
- Humidity and Temperature sensor used to measure humidity and temperature.
- Ultrasonic sensor allows to determine the distance to an object through sound
- Shields are an extension of the Arduino

5.3. Installing the Software (Arduino IDE)

- The Arduino Software (IDE) allows you to write programs and upload them to board.
- In the Arduino Software page, you will find two options:
  - Online IDE (Arduino Web Editor). It will allow to save sketches in the cloud, having them available from any device and backed up.
  - Offline, should use the latest version of the desktop IDE.

- Install the Arduino Desktop IDE accordingly to operating system.
  - Windows
  - Mac OS X
  - Linux
  - Portable IDE (Windows and Linux)

- Choose board in the list here on the right to learn how to get started with it and how to use it on the Desktop IDE.

Fundamentals of Arduino Programming:

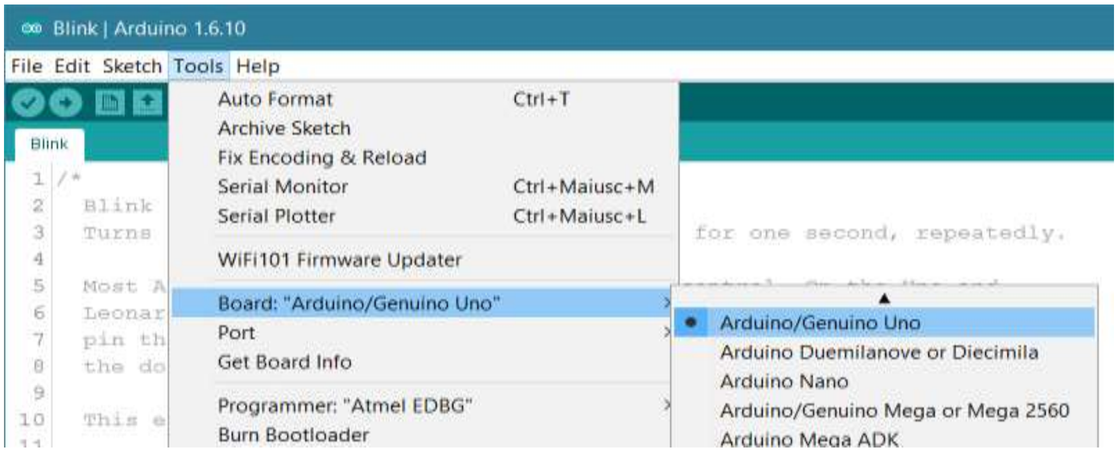
- Two required functions / methods / routines:

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

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

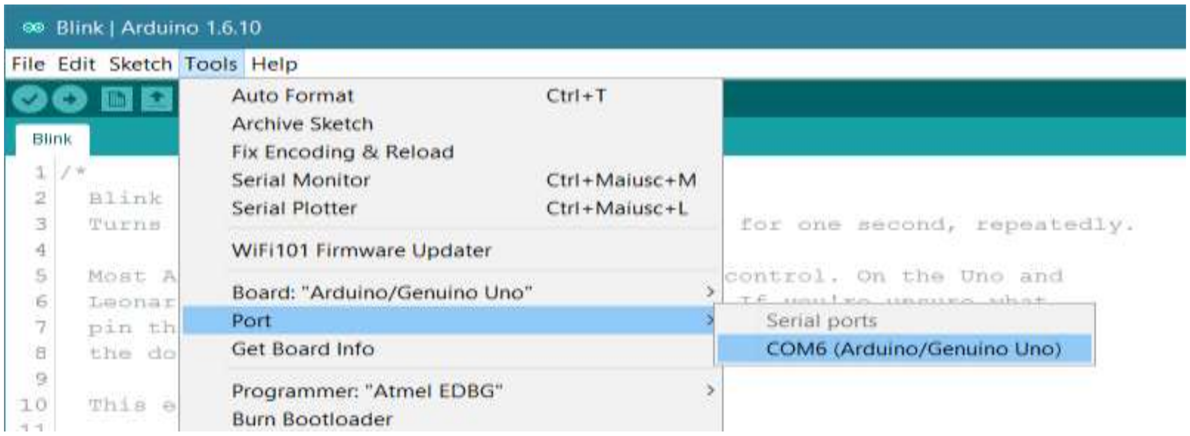
5.3.1. Connecting Arduino UNO Learning Board

- If you want to program your Arduino Uno while offline, you need to install the Arduino Desktop IDE.
- Connect your Uno board with a USB cable; sometimes this cable is called a USB printer cable.
- If you used the Installer, Windows - from XP up to 10 - will install drivers automatically as soon as you connect your board.
- You need to select the entry in the Tools > Board menu that corresponds to your Arduino or Genuino board. (Figure shows layout of Arduino IDE)



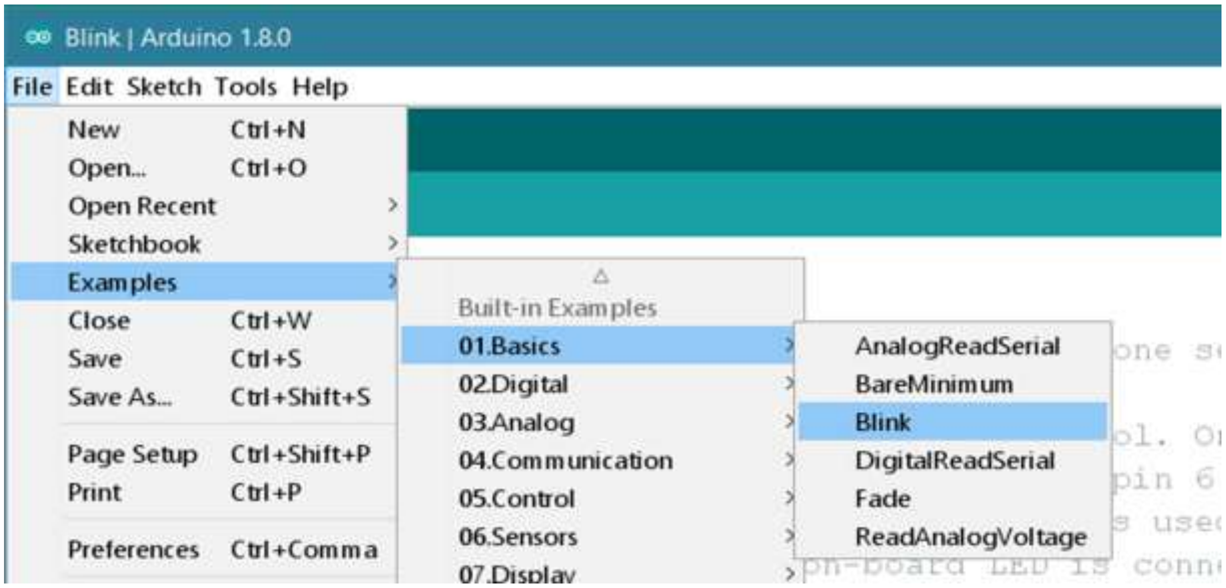
- Select the serial device of the board from the Tools | Serial Port menu.
- This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports).
- To find out, you can disconnect your board and re-open the menu; the entry that disappears should be the Arduino or Genuino board.
- Reconnect the board and select that serial port.

- **Figure shows selecting the port**

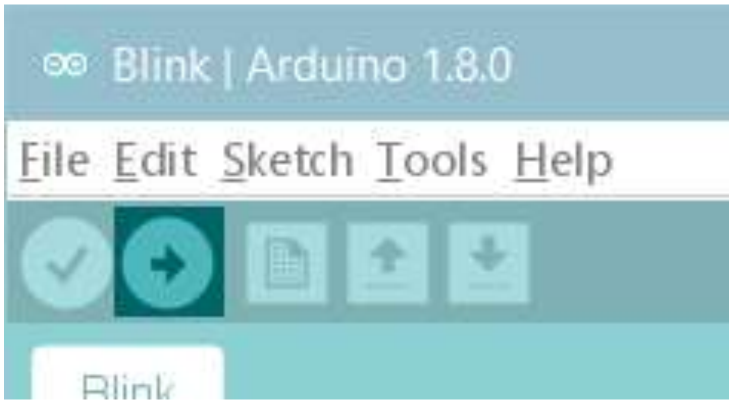


Open your first sketch.

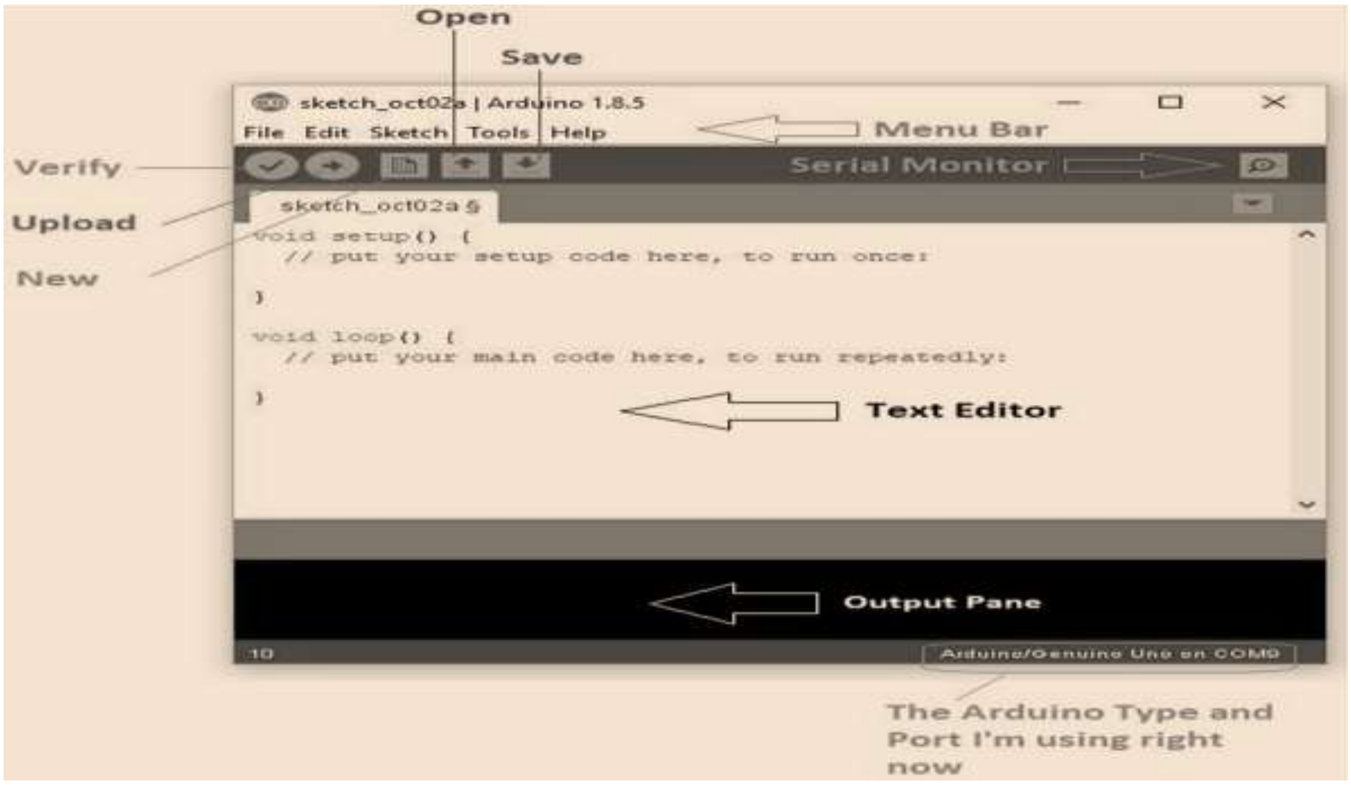
- Open the LED blink example sketch: File > Examples >01.Basics > Blink.



- **Upload the program**
- Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.



- A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in orange).
- **Layout of Arduino Uno IDE**


Open
Save
sketch_oct02a | Arduino 1.8.5
File Edit Sketch Tools Help
Menu Bar
Verify
Upload
New
Serial Monitor
sketch_oct02a §
Text Editor
Output Pane
10
Arduino/Genuine Uno on COM9
The Arduino Type and Port I'm using right now

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

- **Toolbar buttons and Functions of each button as shown in figure**

Verify / Compile	**Checks the code for Errors**
Stop	Stop the serial monitor or un highlight other buttons
New	Creates a new blank sketch, Enter a name and a location for your sketch
Open	Shows the list of sketches in your sketch book
Upload	Uploads the current sketch to the Arduino. Need to make sure that you have current board and port selected (Tools menu) before uploading
Serial Monitor	Display serial data being sent from the Arduino
Verify/Compile	Button used to check that your code is correct, before upload it to Arduino
Stop Button	Will stop the serial monitor from operating. If you need to obtain a snapshot of the serial data so far examined.

Breadboard for prototyping Arduino Uno Circuits

- The breadboard allows you to connect components together by plugging them into the little holes.
- The key is to understand how the holes are connected.
- As shown in figure, the holes in a column (When oriented as shown in figure) are connected together.
- So to connect components together you need to plug the leads you want to connected into the same column.
- Note that columns are not connected across the "trench" in the center of the board.
- These are typically used to create "rails".
- These are typically used for ground and supply voltage that need to connect many components.
- Some rows are marked (+) and Some (-). These are just markings
- The row will set at whatever voltage that we need to connect it.
- In order to keep circuit organized you need to use a breadboard, pictures below Figure.

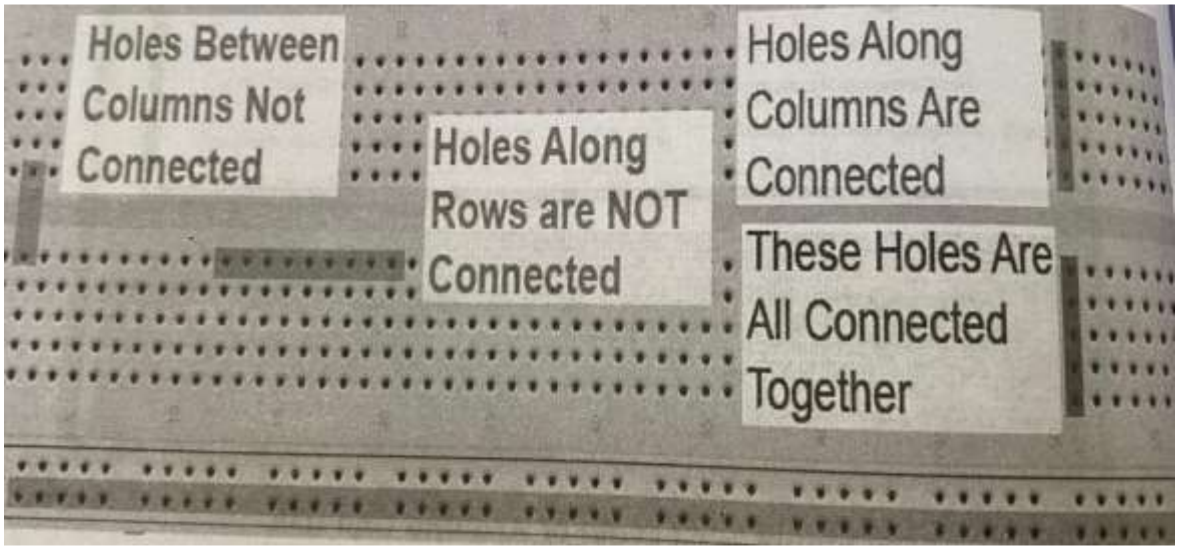

Holes Between Columns Not Connected
Holes Along Rows are NOT Connected
Holes Along Columns Are Connected
These Holes Are All Connected Together

Fig.: Breadboard for prototyping Arduino Uno circuits

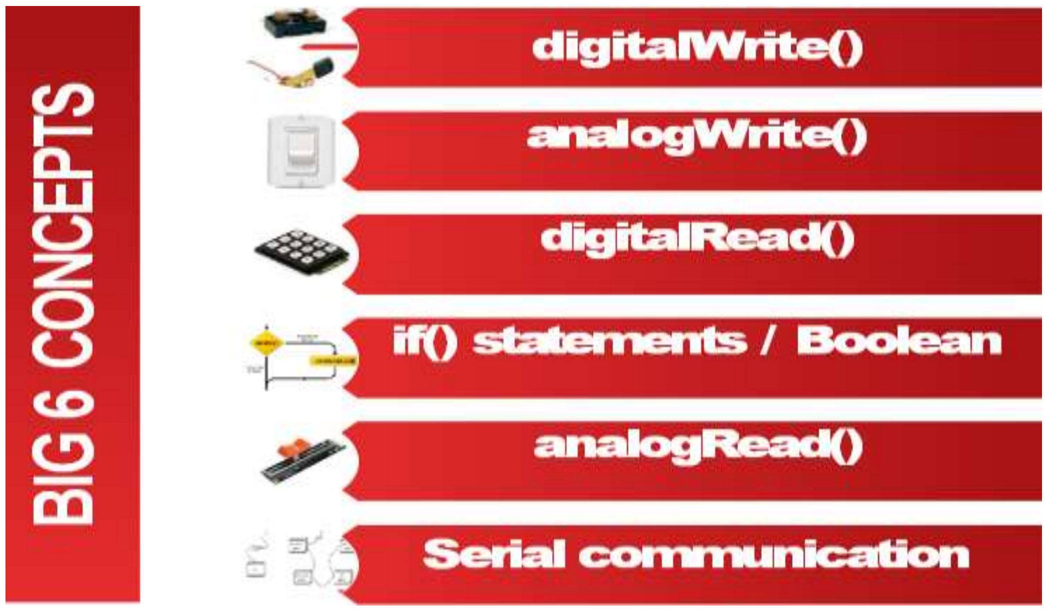
Technical Specifications of Arduino UNO

Microcontroller Arduino	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V

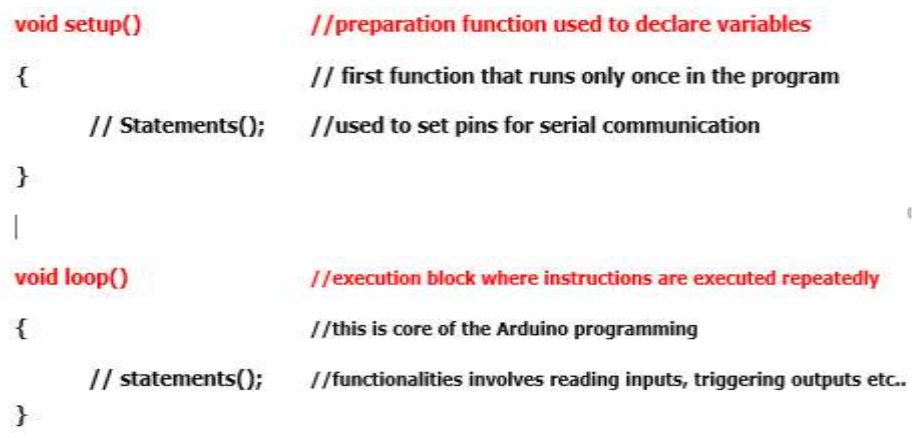
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

5.4. Fundamentals of Arduino Programming

- The Arduino IDE supports the languages C and C++ using special rules of code structuring.



- The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures.
- User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution.
- Structure: Structure of Arduino programming contains two part



```
void setup()              //preparation function used to declare variables
{                         // first function that runs only once in the program
       // Statements();   //used to set pins for serial communication
}
|
void loop()               //execution block where instructions are executed repeatedly
{                         //this is core of the Arduino programming
       // statements();   //functionalities involves reading inputs, triggering outputs etc..
}
```

Setup()

- This function is called once when a sketch starts after power-up or reset.
- It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch

```
void setup ( )   //after calling setup(), loop() function dos its task
{
  digitalWrite(pin, HIGH);          //set pin ON
  delay(10000);                     // pauses for ten thousand milliseconds
  digitalWrite(pin, LOW);           //set pin OFF
  delay(10000);                      // pauses for ten thousand milliseconds
}
```

loop():

- After setup() function exits (ends), the loop() function is executed repeatedly in the main program.
- It controls the board until the board is powered off or is reset.

```
void loop( )     //after calling setup(), loop() function does its task
  {
         digitalWrite(pin, HIGH);          //set pin ON
         delay(10000);                     // pauses for ten thousand milliseconds
```

```
            digitalWrite(pin, LOW);               //set pin OFF
            delay(10000);                          // pauses for ten thousand milliseconds
        }
```

Functions

- A function is a piece of code that has a name and set of statements executed when function is called.
- Functions are declared by its type followed with name of a function.
- Syntax:

```
type functionName (parameters)
{
        Statement(s);
}
```

- Example:

```
int delayvar( )
{
  int var;                              //create temporary variable
  var=analogRead(potent);               //read from potentiometer
  var=var/4;                            //convert the value of variable var
  return var;                           //return var
}
```

{ } curly braces

- They define beginning and end of function blocks, unbalanced braces may leads to compilation errors.

semicolon

- It is used to end a statement and separate elements of a program
- Syntax: int x=14;x

Single line comments

- Single line comment begins with // and ends with instructions.
- Syntax: // this is single line comment

```
/* …………….. */ block Multiline Comments
```

- Multiline comments begins with /* with a description of the block and ends with */
- Syntax:

```
/* this is multiline comment
```

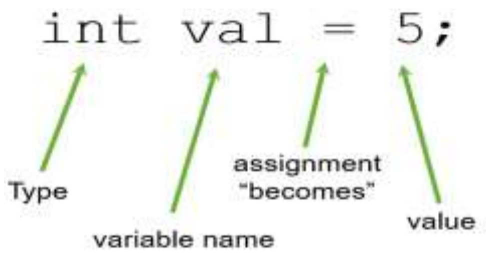
- Use the closing comments to avoid errors */

Variables

- A variable is a way of storing value for later use in a program.
- A variable is defining by its value type as an int, long, float etc by setting a specified name and optionally assigning an initial value.
- A global variable can be seen in every part of the program which is declared at the beginning of the program before setup() function
- A local variable is defined inside a function in which it was declared.
- Example

```
int var;       //global variable
void setup() {
        // nothing is required;
}

void loop() {
  for(int local=0;local<5;)
  {
        local++;
        //local variable & visibility is with in for loop
        }
        float local_f;
        //local variable & visibility is with in loop
}
```



- Example

```
int greenLED = 9;           //Declare delay time variable
void setup() {
        Declare delayTime
        pinMode(greenLED, OUTPUT);
}

void loop() {
        digitalWrite(greenLED, HIGH);
        delay(delayTime);           //Delay Time Variable
        digitalWrite(greenLED, LOW)
        delay(delayTime);
```

- Example

```
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);
        }
```

Data Types

Data Type	Syntax	Range
byte	byte x=100;	O to 255
int	int y=200;	32767 to -32768
long	long var=8000;	2147483647 to -2147483648

float	float x=3.142;	3.4028235E+38 to -3.4028235E+38
arrays	int myarray []={10,20,30,40}	Size depends on the data type associated with declaration

Operators

Operator	Syntax
Arithmetic Operators (+, -, /, *)	x=x+5; y=y-6; z=z*2; p=p/q;
Assignment Operators (=, ++, --, +=, -=, *=, /=)	x++; x+=y; x-=y; x*=y; x/=y
Comparison Operators (==, !=, <, >, <=, >=)	x==y; x!=y; x<y; x>y;

Logical Operators (&& , ǁ, !)	x>2 && x<5; //returns Boolean value
	x>2 ǁ x<5; //returns Boolean value
	!x>2;

Constants

Constants	Usage

TRUE / FALSE	Boolean constants true=2 and false=0 defined in logic levels
	If(b==true)
	{
	// do somethings
	}

INPUT / OUTPUT	Used with pinMode() function to define levels pinMode(13, OUTPUT)
HIGH / LOW	Used to define pin levels HIGH – 1, ON, 5 Volts LOW – 0, OFF, 0 Volts DigitalWrite (13, HIGH)

Flow Control Statements

Operator	Syntax

if	if(some_variable == value)
	{
	   //statement(s);
	}
	else {
	  //statement(s);
	}

for	for(initialization; condition; Expression) {
	   //DoSomethings;
	}
	for(int p=0;p<5;p++){
	  digitalWrite(13,HIGH); //sets pin 13 ON
	  delay(250); //pauses 250 ms
	  digitalWrite(13,LOW); //sets pin 13 OFF
	  delay(250); //pauses 250 ms
	}

while	While loop executes until the expression inside parenthesis becomes false.
	while(condition)
	{
	   //Statement(s); //evaluates till comparison results in a false value
	}

do while	Bottom evaluated loop, works same way as while loop but condition is tested at the end of loop.
	do
	{
	   //Statement(s); //evaluates till comparison results in a false value
	} while(condition)

Digital & Analog input output pins and their usage

Methods	Usage

pinMode(pin, mode)	Before using a pin as a digital input or output, must first configure the pin, which is done with pinMode().
	Used in setup() method to configure pin to behave as Input/Output
	pinMode(pin, INPUT) //pin set to INPUT
	pinMode(pin, OUTPUT) //pin set to OUTPUT

DigitalRead(pin)	Read value from a specified pin with result being HIGH/LOW Val=DigitalRead(pin); //val will be equal to input pin
DigitalWrite(pin, value)	Outputs to HIGH/LOW on a specified pin. DigitalWrite(pin, HIGH); ////pin set to HIGH

Example	int x=13; //connect x to pin 13
	int p=7 //connect push button to pin 7
	int val=0; //variable to store the read value
	void setup() {
	pin MODE(x, OUTPUT); //sets ‘x’ as OUTPUT
	}
	void loop() {
	val=digitalRead (p); //sets ‘value’ to 0
	digitalWrite(x, val); //sets ‘x’ to button value
	}

analogRead(pin)	Read value from a specified analog pin works on pins 0 -5.
	val = analogRead(pin); //value equal to pin
	analogRead() //reads the voltage value on a pin and returns int value
	The pin argument denotes the analog pin you want to read from.
	When referring to an analog pin, call them as A0, A1, A2,…A6.
	This function takes approximately 100 microseconds to perform.

analogWritanalogRead(pin,value)	Writes an analog value using pulse width modulation (PWM) to a pin marked PWM works on pins 3,5,6,9,10.

	It uses a simple technique to “emulate” an analog output. It relies on two things: a pulse width and a duty cycle. It is a way of simulating any value within a range by rapidly switching between 0 volts and 5 volts.

Analog I/O

Methods	Usage

Example	int x=10; //connect ‘x’ to pin 13
	int p=0; //connect potentiometer to analog pin 7
	int val;
	void setup( )
	{ }
	void loop()
	{
	val=analog Read(p);
	val /=4;
	analog Write(x, val); //outputs PWM signal to ‘x’
	}

Time

Methods	Usage
delay(ms)	Pauses for amount of time specified in miliseconds.
millis()	Returns the number of milliseconds since Arduino is running val = millis(); //val will be equal to millis()

Mathematical Functions

Methods	Usage
min(x, y)	Calculates minimum of two numbers val=min(val, 10) // sets ‘val’ to smaller than 10 or equal to 10.
max(x,y)	Calculates maximum of two numbers val=max(val, 10) // sets ‘val’ to larger than 10 or equal to 10.

Random: Arduinos are capable of generating pseudo-random numbers using the random() function:

Methods	Usage
randomSeed(value)	Sets a value/seed as starting point

random(min, max)
random(max)
	Allows to return numbers within a range of specified by min and max values
	Val=random(100, 200) //sets 'val' to random number between 100 to 200.

Example	int num;
	int x=10;
	void setup()
	{
	randomSeed(millis()); //set millis() as seed
	num = random(200); //random number 0 – 200
	analogWrite(x, number); //outputs PWM signal
	delay(500);
	}

Serial

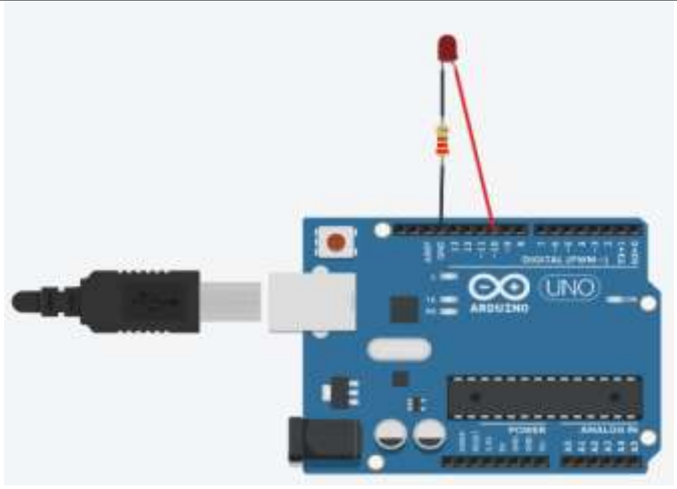
Methods	Usage

Serial.begin(rate)	Opens serial port and sets the baud rate for serial data transmission
	void setup()
	{
	Serial.begin(9600); //sets default rate to 9600 bps
	}

Serial.println(data);	Prints data to the serial port Serial.println(value); //sends value to serial monitor

5.5. Examples

5.5.1. Blink LED

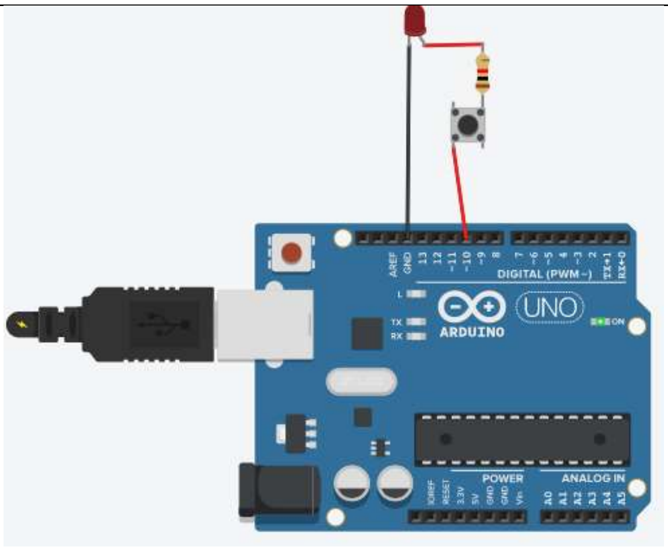


```
void setup(){
  pinMode(10, OUTPUT);
}

void loop(){
  digitalWrite(10, HIGH);
  delay(500); // Wait for 1000 millisecond(s)
  digitalWrite(10, LOW);
  delay(500); // Wait for 1000 millisecond(s)
  //exit(0);    //to exit from simulation (only once LED will blink
}
```

https://www.tinkercad.com/things/53AHrLloDKL-frantic-jaiks-lahdi/editel?tenant=circuits

5.5.2. Toggle the LED using switch

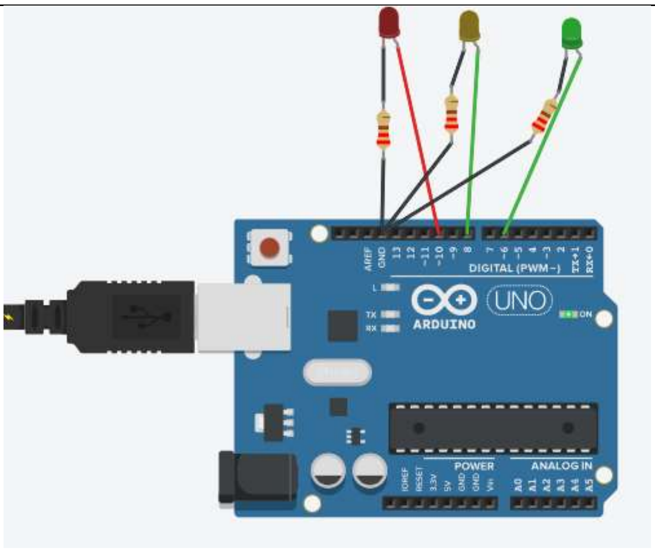


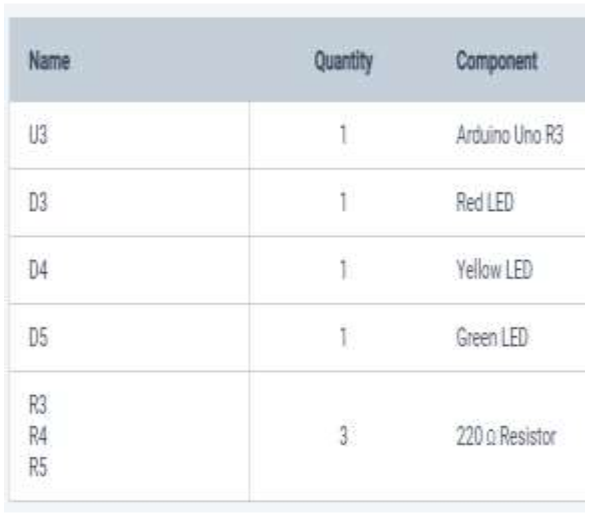
```
void setup(){
  pinMode(10, OUTPUT);
}

void loop(){
  digitalWrite(10, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

https://www.tinkercad.com/things/53AHrLloDKL-toggleledswitch/editel

5.5.3. Traffic Light Simulation





Name	Quantity	Component
U3	1	Arduino Uno R3
D3	1	Red LED
D4	1	Yellow LED
D5	1	Green LED
R3 R4 R5	3	220 Ω Resistor

```
void setup() {
  pinMode(10, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(6, OUTPUT);
}
void loop(){
digitalWrite(10, HIGH);
  delay(500); // Wait for 1000 millisecond(s)

  digitalWrite(10, LOW);
  delay(500); // Wait for 1000 millisecond(s)
digitalWrite(8, HIGH);
  delay(500); // Wait for 1000 millisecond(s)
  digitalWrite(8, LOW);
  delay(500); // Wait for 1000 millisecond(s)
  digitalWrite(6, HIGH);
  delay(500); // Wait for 1000 millisecond(s)
  digitalWrite(6, LOW);
  delay(500); // Wait for 1000 millisecond(s)
}
```

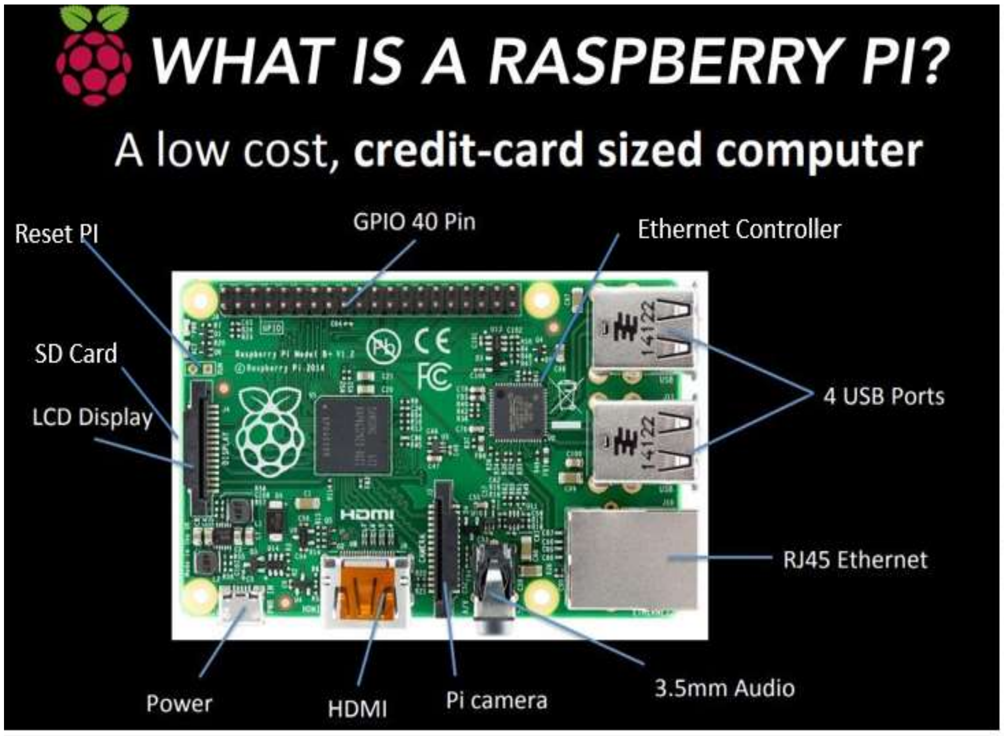
https://www.tinkercad.com/things/53AHrLloDKL-frantic-jaiks-lahdi/editel?tenant=circuits

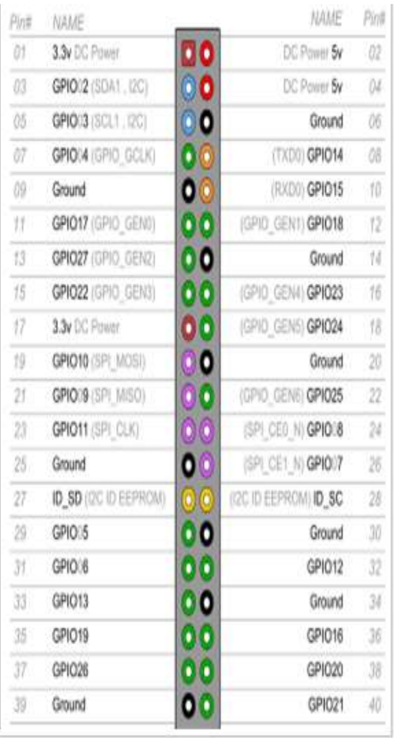
5.5.4. Important Functions

Function Name	Description
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

5.6. Raspberrypi

- Raspberry Pi is a series of credit card sized board computers developed in UK by Raspberypi Foundation to promote the teaching of basic computer science in schools and developing countries.
- Generations:
  - First generation: RaspberryPi 1 Model B) was released in February 2012 followed by simpler and inexpensive model A.
  - In 2014, the foundation released a board with improved design in Paspberry 1 Model B+.
- Raspberry Pi is the name of a series of single-board computers made by the Raspberrypi 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.





Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	DC Power 5v	04
05	GPIO03 (SCL1 , I2C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Processor

- The Broadcom BCM2835 SoC used in first generation Raspberrypi used in first generation smartphones which includes 700MHz ARM 1176JZF-S processor, Video Core IV graphics Processing Unit (GPU) and RAM.
- Raspberrypi 2 uses BCM2836 SoC with 900MHz 32-bit quad core ARM Cortex A7 processor with 256KB shared L2 cache.
- Raspberrypi 3 uses BCM2837 SoC with 1.2GHz 64-bit quad core ARM Cortex A53 processor with 512KB shared L2 cache

Power Source

- Power to Raspberrypi is via micro USB port on the side of the unit.
- Recommended voltage is +5V and recommended input current 2A.
- Raspberrypi can function on lower current power (5V @1A)

SD Card

- Raspberrypi does not have locally available storage.
- Framework is stacked on SD card which is embedded on SD card space on Raspberrypi

GPIO (General Purpose Input and Output)

- GPIO is a non specific pins on a coordinated circuit to know is an input or output pin which can be controlled by a client at run time.
- GPIO Capabilities
  - GPIO pins can be designated to be input or output
  - GPIO pins can be empowered / crippled
  - Input values are LOW, HIGH
  - Yield values are writable / meaningful
  - Input values are frequently utilized as IRQs

DSI Display X

- The Rasberrypi Connector S2 is a display serial interface (DSI) for connecting a liquid crystal display (LCD) panel using a 15-pin ribbon cable.
- The mobile industry processor interface (MIPI) inside the Broadcom BCM2835 IC feeds graphics data display panel through this connector.

Audio Jack

- A standard 3.5 mm TRS connector is accessible on the RPi for stereo sound yield

Status LEDs

OK	– SDCard Access (GPIO16)
Power	- 3.3V Power "PWR"
FDX	- Full Duplex (LAN) (Model B)
LNK	- Link/Activity (LAN) (Model B)
10M/100	- 10/100Mbit (LAN) (Model B)

Ethernet Port

- Ethernet port is accessible on model B and Model B+ (LAN9512 LAN)

CSI Connector

- Camera Serial Interface (CSI) is a serial interface connector outlines by MIPI (Mobile industry Processor Interface) for interfacing computerized cameras with portable processor.

HDMI

- High Definition multimedia interface (HDMI) to give both video and Sound yield.

JTAG Headers

- JTAG is acronym for Joint Test Action Group used for accessing to gadget pins by means of a serial port.

Useful commands to rum from terminal or command line

Command	Description
rasp-config	Change your pi configuration settings
ssh	Connect your pi to other computers
startx	Start GUI (Graphical User Interface)
sudo	Run commands as super user

ifconfig	Get details of your Ethernet or wireless network adapter
shutdown	This will shut down your pi.
rpi-update	Updates your Raspberrypi firmware
nano	Text editor for changing or adding files, save, edit or create
ls	List out the files of current working directory
touch	Create a blank file
cd	Go to directory or folder
mkdir	Create directory
find	Searches whole file system for files or directories
ping	Test connectivity between two devices
clear	Clears a terminal screen
df-h	Shows disk space

rasp-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.

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

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

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

Why RaspberryPi?

- Inexpensive
- Cross Platform
- Simple
- Clear Programming Environment
- Open Source
- Extensible Hardware and Software

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.
- 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.

Wireless Temperature Monitoring System Using Pi:

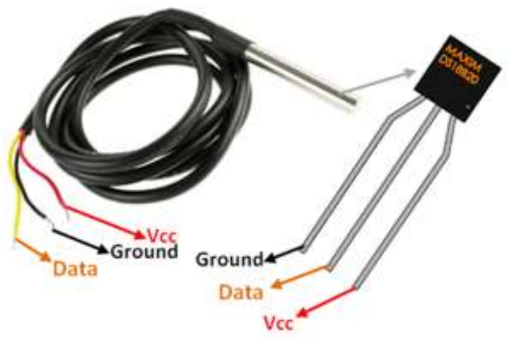
- Raspberry Pi which having inbuilt wi-fi, which makes Raspberry Pi to suitable for IoT applications, so that by using IoT technology this monitoring system works by uploading the temperature value to the Thingspeak cloud by this project you can able to learn to how to handle cloud-based application 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.
- 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.
- 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)
- 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

- 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.
- 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

- **DS18B20 Temperature Sensor Pinout** & **DS18B20 Temperature Sensor**



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.

- Set up your local network and wireless connectivity
- Enable SSH
- Enable SSH on a headless Raspberry Pi (add file to SD card on another machine)
- Set up your client

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.

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.

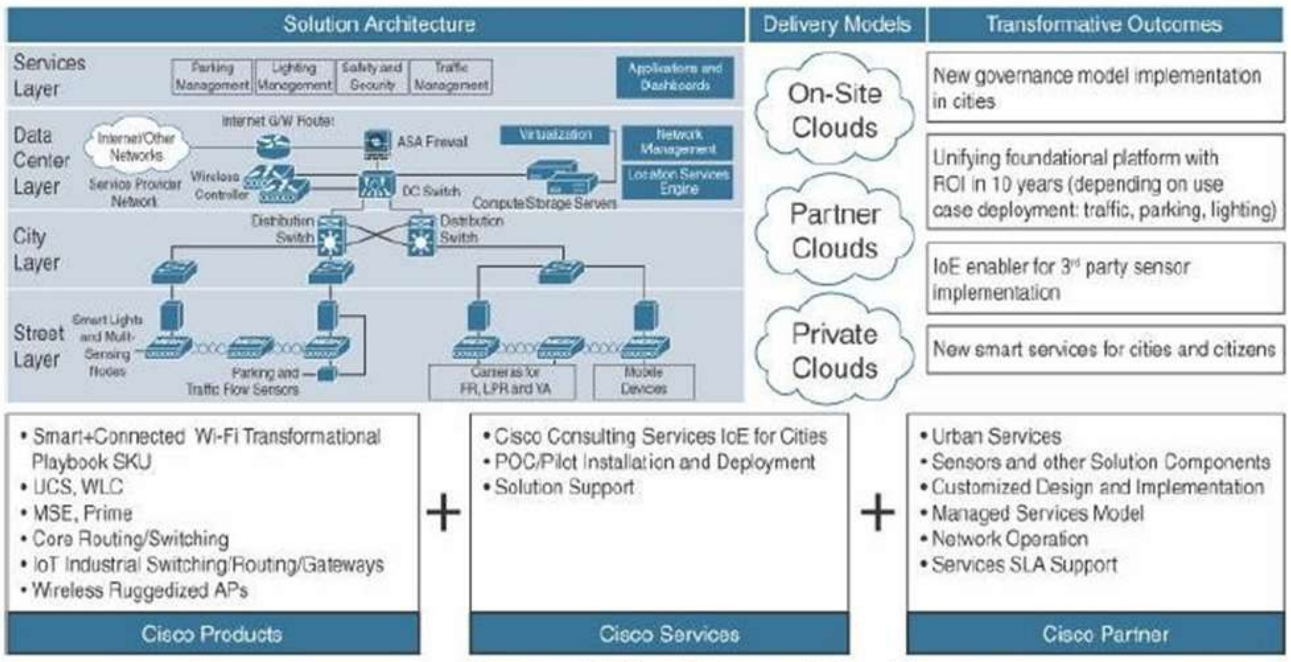
- **Serial**: The serial interface on Raspberry Pi has receive (rx) and transmit (tx) pins for communication with serial peripherals.
- **SPI**: Serial Peripheral Interface (SPI) is a synchronous serial data protocol for communicating with one or more peripheral devices. In an SPI connection, there is 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 points on Raspberry Pi allows you to connect hardware modules. I2C interface allows synchronous data transfer with just two pins-SDA (data line) and SCI (Clock line).

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.

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



Smart Cities Layered Architecture

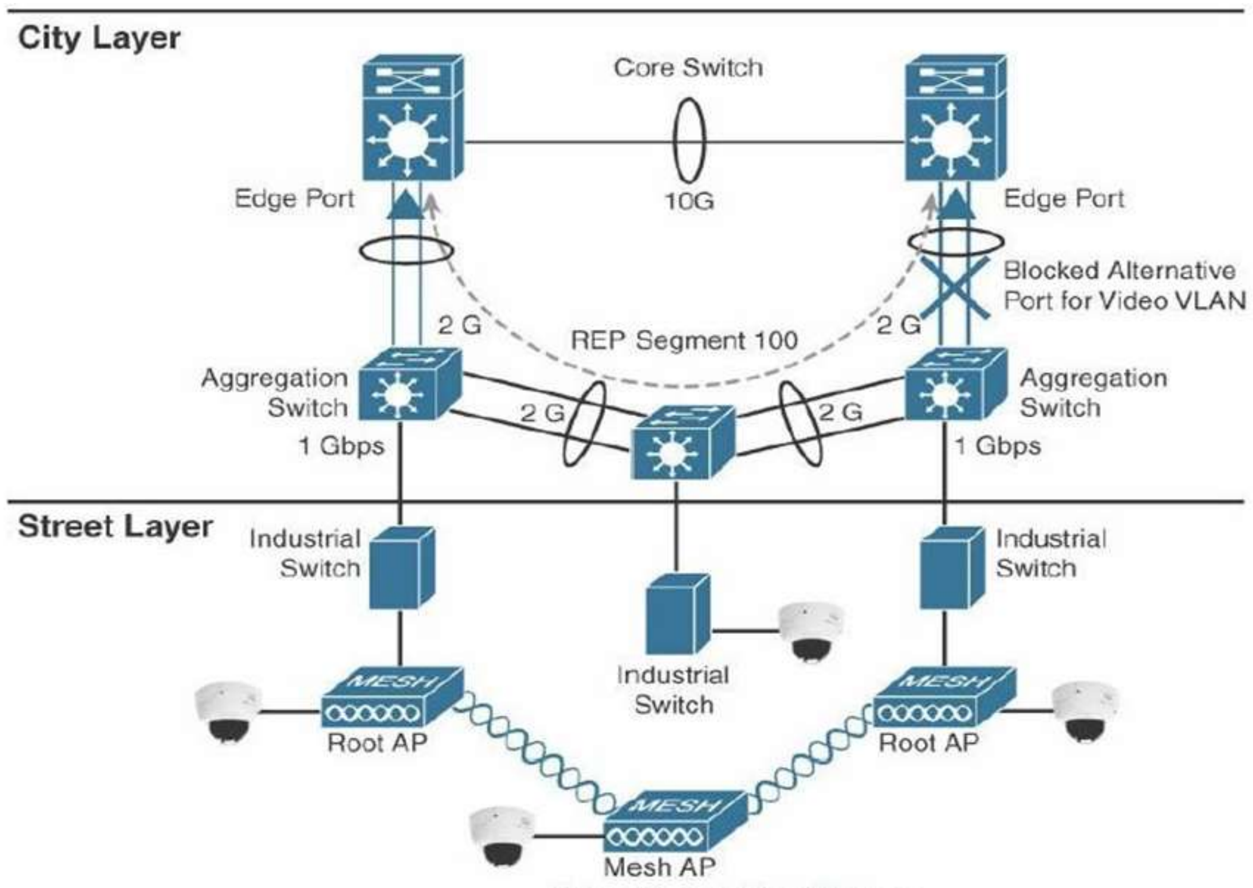
Street Layer:

- 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.

- A variety of sensors are used at the street layer for 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 vehicle 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 public parking area of pedestrians on a sidewalk, or even of birds in public parks or on public monuments.

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.
- 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.
- 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).



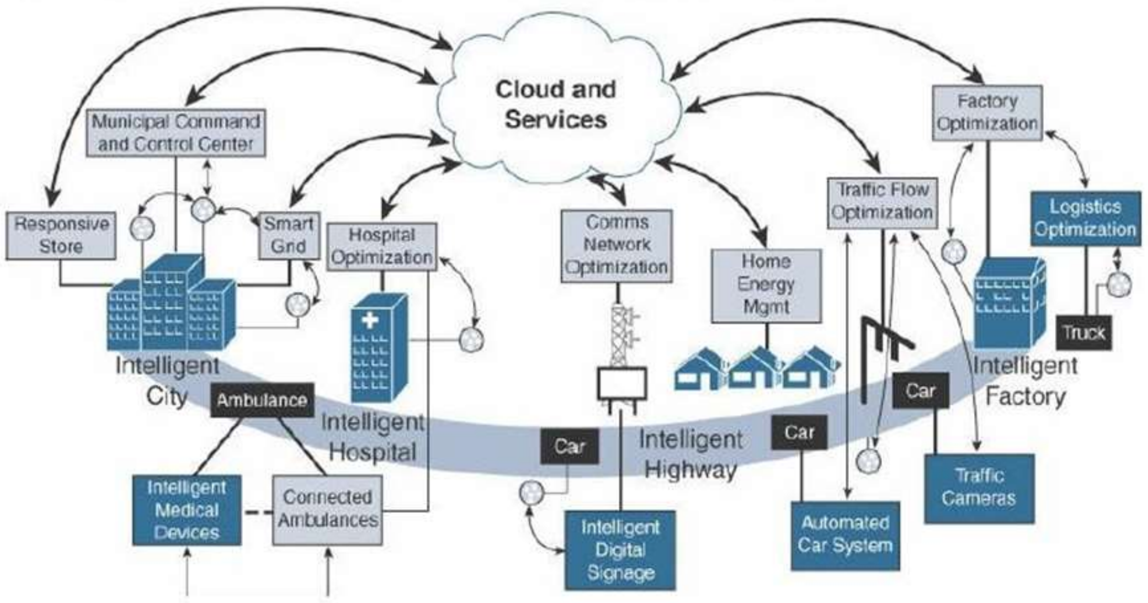
Street Layer Resiliency

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.
- 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.

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.

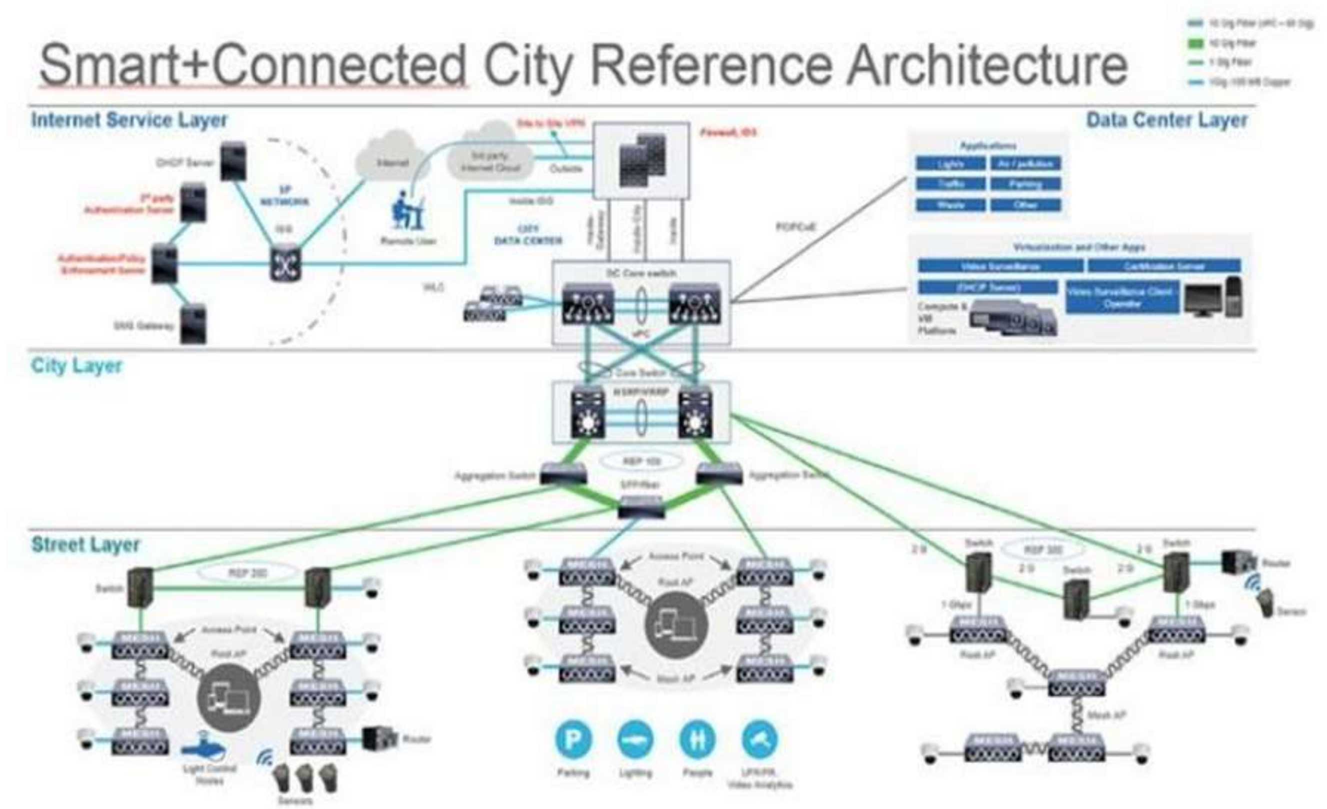


Service Layer:

- 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.

- 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.

- 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.



Key Smart and Connected Cities Reference 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.
- 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.

- The city layer transport 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 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 gateways link uses on type of encryption and the gateway to application connection used another encryption (Ex: VPN).