


USN 


## Internal Assessment Test 3 – May. 2024

Sub:	Internet of Things				Sub Code:	18CS81	Branch:	CSE		
Date:	11-05-2024	Duration:	90 mins	Max Marks:	50	Sem / Sec:	VIII (A, B & C)		OBE	
<u>Answer any FIVE FULL Questions</u>								MAR KS	CO	RBT
1	Discuss different layers of Smart city IoT architecture.						10	CO1	L2	
2	i) With a neat diagram explain the wireless temperature monitoring system using Raspberry Pi. ii) Write a short note on the steps to connect Raspberry Pi via SSH.						10	CO5	L2	
3	Describe the following with respect to Arduino programming: i) if-else condition    ii) Function    iii) Variables and data types    iv) Digital I/O						10	CO5	L2	
4							10	CO5	L1	
Identify the various components shown in the above Arduino UNO board.										

PTO

USN 

## Internal Assessment Test 3 – May. 2024

Sub:	Internet of Things				Sub Code:	18CS81	Branch:	CSE		
Date:	11-05-2024	Duration:	90 mins	Max Marks:	50	Sem / Sec:	VIII (A, B & C)		OBE	
<u>Answer any FIVE FULL Questions</u>								MAR KS	CO	RBT
1	Discuss different layers of Smart city IoT architecture.						10	CO1	L2	
2	i) With a neat diagram explain the wireless temperature monitoring system using Raspberry Pi. ii) Write a short note on the steps to connect Raspberry Pi via SSH.						10	CO5	L2	
3	Describe the following with respect to Arduino programming: i) if-else condition    ii) Function    iii) Variables and data types    iv) Digital I/O						10	CO5	L2	
4							10	CO5	L1	
Identify the various components shown in the above Arduino UNO board.										

PTO

<b>5(a)</b>	Write an Arduino program to print the first 10 numbers using an if-else statement.	5	CO2	L3
<b>5(b)</b>	Explain Raspberry Pi interfaces.	5	CO2	L2
<b>6(a)</b>	Explain the basic sections present in an Arduino program with an example.	6	CO3	L2
<b>6(b)</b>	Illustrate the common industry elements for security on the network layer in Smart city architecture.	4	CO3	L3

---

**CI**

**CCI**

**HOD**

<b>5(a)</b>	Write an Arduino program to print the first 10 numbers using an if-else statement.	5	CO2	L3
<b>5(b)</b>	Explain Raspberry Pi interfaces.	5	CO2	L2
<b>6(a)</b>	Explain the basic sections present in an Arduino program with an example.	6	CO3	L2
<b>6(b)</b>	Illustrate the common industry elements for security on the network layer in Smart city architecture.	4	CO3	L3

---

**CI**

**CCI**

**HOD**

## CO PO Mapping

Course Outcomes			Modu les cover ed	P O 1	P O 2	P O 3	P O 4	P O 5	P O 6	P O 7	P O 8	P O 9	P O 1 0	P O 1 1	P O 1 2	P S O 1	P S O 2	P S O 3	P S O 4
CO1	Interpret the impact and challenges posed by IoT networks leading to new architectural models.	L2	1	3	2	2	-	-	2	-	-	-	-	-	-	-	-	-	3
CO2	Compare and contrast the deployment of smart objects and the technologies to connect them to network.	L2	2	3	2	2	-	-	2	-	-	-	-	-	-	-	-	-	3
CO3	Appraise the role of IoT protocols for efficient network communication.	L2	3	3	2	2	-	-	2	-	-	-	-	-	-	-	-	-	3
CO4	Elaborate the need for Data Analytics and Security in IoT.	L2	4	3	2	2	-	-	2	-	-	-	-	-	-	-	-	-	3
CO5	Illustrate different sensor technologies for sensing real world entities	L3	,5	3	2	2	-	-	2	-	-	-	-	-	-	-	-	-	3

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low

PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Develop applications using different stacks of web and programming technologies				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Develop intelligent applications for business and industry				

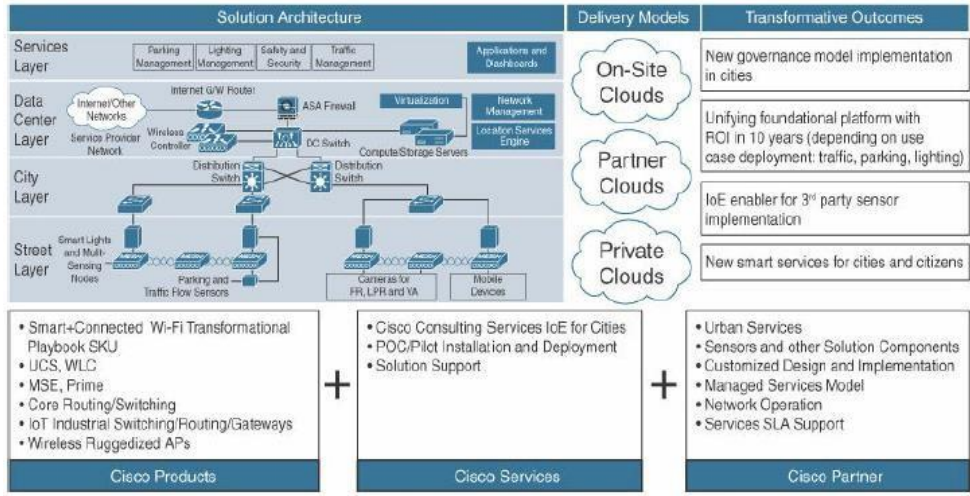
---

Internal Assessment Test 3 – May. 2024

Sub:	Internet of Things				Sub Code:	18CS81	Branch :	CSE
Date:	11-05-2024	Duration:	90 mins	Max Marks:	50	Sem / Sec:	VIII (A, B & C)	OBE

Answer any FIVE FULL Questions

MA RKS CO RBT

<b>1</b>	<p>Discuss different layers of Smart city IoT architecture.</p> <p>Solution:( 2.5 Mark for each layer)</p>  <p style="text-align: center;"><b>Smart Cities Layered Architecture</b></p>	10	CO1	L2
<p><b>Street Layer:</b> 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.</p> <p><b>City Layer:</b> 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</p>				

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.

**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

**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.

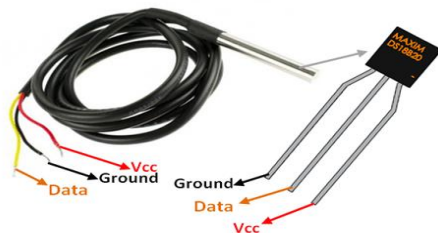
2 i) With a neat diagram explain the wireless temperature monitoring system using Raspberry Pi.

Solution:( 6 Mark)

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.

**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 construction 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.



**Applications:**

- Measuring temperature at hard environments
- Liquid temperature measurement

10

CO5

L2

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

ii) Write a short note on the steps to connect Raspberry Pi via SSH.

Solution:( 4 Mark)

One 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

3

Describe the following with respect to Arduino programming:

i) if-else condition    ii) Function    iii) Variables and data types    iv) Digital I/O

Solution:( 2.5 Mark for each)

**if-else condition:** In Arduino programming, the **if-else** condition is used for decision making. It allows you to execute certain blocks of code if a specified condition is true, and another block if the condition is false. Here's a basic syntax:

```
if (condition) {  
    // code to be executed if condition is true  
} else {  
    // code to be executed if condition is false  
}
```

For example, you might use an **if-else** condition to check if a sensor reading is above a certain threshold and then take appropriate action based on that.

ii) **Function:** Functions in Arduino programming are blocks of code that perform a specific task. They allow you to break down your code into smaller, more manageable pieces, which can be reused and called whenever needed. In Arduino, you have two types of functions: user-defined functions and built-in functions. User-defined functions are created by the programmer, while built-in functions are provided by the Arduino environment or libraries. Here's a basic syntax of a user-defined function:

```
return_type function_name(parameters) {  
    // function body  
}
```

10

CO5

L2

**Variables and data types:** Variables in Arduino programming are used to store data that can be manipulated or accessed throughout the program. Arduino supports several data types including integers, floats, characters, arrays, and more. These data types determine the size and type of data that can be stored in a variable. Here are some common data types:

**int:** Used to store integer values.

**float:** Used to store floating-point numbers.

**char:** Used to store single characters.

**bool:** Used to store boolean values (true/false).

Variables and data types are declared like this:

```
data_type variable_name = value;
```

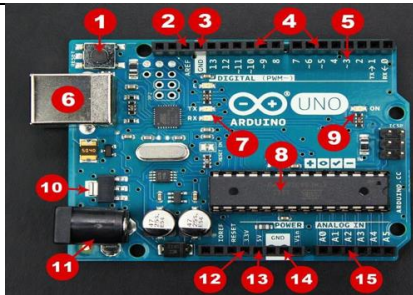
**Digital I/O:** Digital Input/Output (I/O) in Arduino refers to the process of reading digital signals from sensors or sending digital signals to control actuators such as LEDs or motors. Pins on the Arduino board can be configured as either inputs or outputs. Digital input pins are used to read the state of external devices (like switches or sensors), while digital output pins are used to send signals to control external devices (like LEDs or relays).

You can set a pin's mode as input or output using **pinMode()** function, and read or write digital values using **digitalRead()** and **digitalWrite()** functions respectively.

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);  
}
```

4



Identify the various components shown in the above Arduino UNO board.

Solution:( 10 Mark)

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

10

CO5

L1



	<p>5.PWM – The pins marked with the (~) symbol can simulate.</p> <p>6. USB Connection – Used for powering up your Arduino and uploading sketches</p> <p>7.TX/RX – Transmit and receive data indication LEDs</p> <p>8.ATmega Microcontroller – This is the brains and is where the programs are stored</p> <p>9. Power LED Indicator – This LED lights up anytime the board is plugged in a power source</p> <p>10. Voltage Regulator – This controls the amount of voltage going into the Arduino board</p> <p>11. DC Power Barrel Jack – This is used for powering your Arduino with a power supply</p> <p>12. 3.3V Pin – This pin supplies 3.3 volts of power to your projects</p> <p>13. 5V Pin – This pin supplies 5 volts of power to your projects</p> <p>14. Ground Pins – There are a few ground pins on the Arduino and they all work the same</p> <p>15. Analog Pins – These pins can read the signal from an analog sensor and convert it to digital</p>			
<b>5(a)</b>	<p>Write an Arduino program to print the first 10 numbers using an if-else statement.</p> <p>Solution:(5 Mark)</p> <pre>void setup() {   // Set up serial communication   Serial.begin(9600); }  void loop() {   // Variable to store the current number   int number = 1;    // Loop to print the first 10 numbers   while (number &lt;= 10) {     // Print the current number     Serial.println(number);      // Increment the number for the next iteration     number++;      // Delay for readability     delay(500);   }    // Endless loop to prevent the program from exiting   while (true); }</pre>	5	CO2	L3
<b>5(b)</b>	<p>Explain Raspberry Pi interfaces.</p> <p>Solution:( 5 Mark)</p> <p><b>Serial</b> (UART - Universal Asynchronous Receiver-Transmitter):</p>	5	CO2	L2

The serial interface, often referred to as UART (Universal Asynchronous Receiver-Transmitter) on the Raspberry Pi, allows for serial communication between the Raspberry Pi and other devices. UART communication uses two data lines: one for transmitting data (TX) and one for receiving data (RX). It operates asynchronously, meaning data is transmitted without the need for a separate clock signal. UART is commonly used for communication with peripherals like GPS modules, RFID readers, Bluetooth modules, and other microcontrollers. It's a straightforward and widely supported communication protocol.

**SPI (Serial Peripheral Interface):**

SPI is a synchronous serial communication interface that allows for high-speed communication between the Raspberry Pi and peripheral devices. It typically involves one master device (the Raspberry Pi) communicating with one or more slave devices using separate data lines for transmission (MOSI - Master Out Slave In) and reception (MISO - Master In Slave Out), along with a clock line (SCLK) and optionally a chip select line (CE). SPI is commonly used for connecting devices like sensors, display modules, ADCs (Analog-to-Digital Converters), DACs (Digital-to-Analog Converters), and EEPROMs (Electrically Erasable Programmable Read-Only Memory) to the Raspberry Pi.

**I2C (Inter-Integrated Circuit):**

I2C is a multi-master, multi-slave, serial communication protocol that facilitates communication between integrated circuits. It uses only two bidirectional data lines: Serial Data (SDA) and Serial Clock (SCL), which are shared between multiple devices on the bus. I2C allows for easy interfacing with various peripherals such as sensors, LCD displays, real-time clocks (RTCs), EEPROMs, and more. Each device on the bus has a unique address, allowing the Raspberry Pi to communicate with multiple devices simultaneously. Due to its simplicity and ability to connect multiple devices using a shared bus, I2C is widely used in embedded systems and IoT applications.

**6(a)** Explain the basic sections present in an Arduino program with an example. 6 CO3 L2

Solution:( 3+3 Marks)

**Setup Function:** The setup function is called once when the Arduino board is powered on or reset. It is used to initialize variables, configure pins, and perform any setup tasks necessary for the program.

```
void setup() {
//setup motors, sensors etc
}

void setup() {
    pinMode(9, OUTPUT);
}
```

**Loop Function:** The loop function is where the main logic of the program resides. It is executed repeatedly in a continuous loop after the setup function has completed. The loop function contains the code that controls the behavior of the Arduino board.

```
void loop() {
// get information from sensors
// send commands to motors
}

void loop() {
    digitalWrite(9, HIGH);
    delay(1000);
    digitalWrite(9, LOW);
    delay(1000);
}
```

6(b)	<p>Illustrate the common industry elements for security on the network layer in Smart city architecture.</p> <p>Solution:( 4 Marks)</p> <p><b>Firewall:</b> 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.</p> <p><b>VLAN:</b> 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.</p> <p><b>Encryption:</b> 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).</p>	4	CO3	L3
------	--	---	-----	----