IOT

What is Raspberry pi

The Raspberry Pi is a series of small, affordable, single-board computers developed by the Raspberry Pi Foundation. These devices are designed to promote computer science education and facilitate experimentation with hardware and software in a compact and low-cost form factor. Here are some key theoretical aspects and concepts related to the Raspberry Pi:

1. **Single-Board Computer (SBC):** The Raspberry Pi is a type of single-board computer, meaning that the entire computer is built on a single circuit board. It includes a processor (CPU), memory (RAM), input/output (I/O) ports, and other essential components.
2. **ARM Architecture:** Most Raspberry Pi models use ARM architecture processors. ARM processors are widely used in mobile devices and embedded systems due to their energy efficiency and performance. The ARM architecture has become popular in the world of small, low-power computing.
3. **Operating Systems:** The Raspberry Pi can run various operating systems, with Raspbian (now called Raspberry Pi OS), a Linux distribution, being the official and widely used OS. Other operating systems, including various flavors of Linux and even Windows 10 IoT Core, can be installed on a Raspberry Pi.
4. **GPIO (General Purpose Input/Output):** One of the unique features of the Raspberry Pi is its GPIO pins. These pins allow the Raspberry Pi to interface with the physical world by connecting to sensors, LEDs, motors, and other hardware components. This capability makes the Raspberry Pi a versatile platform for learning about electronics and programming.
5. **Programming Languages:** The Raspberry Pi supports various programming languages, including Python, C, C++, Java, and more. Python is particularly popular due to its readability and ease of use, making it a great choice for beginners.
6. The Raspberry Pi contains a microprocessor, not a microcontroller. Specifically, it uses an ARM-based microprocessor as its central processing unit (CPU). The microprocessor in the Raspberry Pi provides general-purpose computing capabilities, similar to those found in personal computers.

Raspberry pi is not a microcontroller it is a SBC

What are microcontrollers

A microcontroller is a compact integrated circuit (IC) that contains a processor core, memory, and programmable input/output peripherals. It is designed to execute specific tasks and control a range of devices, systems, or products. Microcontrollers are widely used in embedded systems where they provide dedicated functionality within a larger electronic system.

Here are some key characteristics and features of microcontrollers:

1. **Processing Unit:** Microcontrollers have a central processing unit (CPU) that performs the computations and executes instructions. The CPU is often a microprocessor optimized for embedded applications.
2. **Memory:** Microcontrollers typically include both program memory (where the software or firmware is stored) and data memory (for temporary storage during program execution). The memory is usually integrated on the same chip.
3. **Input/Output (I/O) Peripherals:** Microcontrollers have built-in peripherals to interact with the external world. These can include digital and analog input pins, output pins, timers, communication ports (such as UART, SPI, I2C), and more. The specific peripherals vary depending on the microcontroller model and application.
4. **Clock Source:** Microcontrollers operate based on an internal or external clock source. The clock determines the speed at which the microcontroller processes instructions and interacts with peripherals.
5. **Low Power Consumption:** Many microcontrollers are designed for low power consumption, making them suitable for battery-powered or energy-efficient applications.
6. **Embedded Applications:** Microcontrollers are commonly used in embedded systems, which are dedicated computing devices designed for specific tasks. Examples include household appliances, automotive control systems, medical devices, industrial automation, and consumer electronics.
7. **Programmability:** Microcontrollers are programmable, allowing developers to load software or firmware onto the chip to define the device's behavior. Programming is typically done in languages like C, C++, or assembly language.
8. **Real-time Operation:** Microcontrollers are often used in real-time applications where tasks must be performed within specific time constraints. Real-time capabilities make them suitable for applications like control systems and robotics.

Different Raspberry Pi Models

Model	Release Date	CPU	RAM	USB Ports	Ethernet	Wi-Fi	Bluetooth	GPIO Pins	HDMI Ports	Power Supply
Raspberry Pi 1	February 2012	ARM1176JZF-S	256MB	2	Yes	No	No	26	1	Micro USB
Raspberry Pi 2	February 2015	Quad-core ARM Cortex-A7	1GB	4	Yes	No	No	40	1	Micro USB
Raspberry Pi 3	February 2016	Quad-core ARM Cortex-A53	1GB	4	Yes	Yes	Yes (4.2)	40	1	Micro USB
Raspberry Pi 3B+	March 2018	Quad-core ARM Cortex-A53	1GB	4	Yes	Yes	Yes (4.2)	40	1	Micro USB
Raspberry Pi 4	June 2019	Quad-core Cortex-A72	2GB/ 4GB/ 8GB	2/4	Yes	Yes	Yes (5.0)	40	2 (micro & mini)	USB-C
Raspberry Pi Zero	November 2015	Single-core ARM1176JZF-S	512MB	1	No	No	No	40	Mini HDMI	Micro USB
Raspberry Pi Zero W	February 2017	Single-core ARM1176JZF-S	512MB	1	No	Yes	Yes (4.1)	40	Mini HDMI	Micro USB
Raspberry Pi Pico	January 2021	Dual-core ARM Cortex-M0+	264KB RAM	N/A	No	No	No	26	N/A	Micro USB

What is arduino

Arduino boards are microcontroller-based platforms that serve as the brain of electronic projects. They come in various shapes and sizes, but they all share common features like digital and analog input/output pins, USB connectivity for programming and communication, and a microcontroller unit (MCU) at their core. Arduino boards are often based on Atmel AVR or ARM processors.

Common Arduino boards include:

1. **Arduino Uno:** A popular entry-level board with a straightforward design and a good balance of features.
2. **Arduino Nano:** A smaller, compact version of the Uno, suitable for projects with space constraints.
3. **Arduino Mega:** A larger board with more input/output pins, suitable for more complex projects.
4. **Arduino Due:** Featuring a more powerful ARM-based processor, the Due is designed for more demanding applications.

5. **Arduino Leonardo:** This board has the ability to emulate a computer mouse or keyboard, making it suitable for projects involving human-computer interaction.

Arduino IDE (Software): The Arduino IDE is a software development environment used for writing, compiling, and uploading code to Arduino boards. It provides a simple and beginner-friendly interface, making it accessible to people with varying levels of programming experience.

Key features of the Arduino IDE include:

1. **Code Editor:** A text editor for writing and editing Arduino code. The language is a simplified version of C/C++.
2. **Compiler:** The IDE includes a compiler that translates the written code into machine-readable instructions for the Arduino board.
3. **Uploader:** The IDE allows users to upload their compiled code to the Arduino board via a USB connection.
4. **Library Support:** Arduino has a vast library of pre-written code snippets and libraries that simplify programming tasks. Users can easily integrate these into their projects.

Arduino is widely used for a variety of applications, including robotics, home automation, electronic art, prototyping, and education. Its open-source nature encourages a collaborative community, and many people share their projects, code, and ideas, contributing to the growth and diversity of the Arduino ecosystem.

Is Ardunio a microcontroller?

Arduino is not a microcontroller itself; it is a platform that includes both hardware and software components. The Arduino platform comprises:

1. **Arduino Boards (Hardware):** These are microcontroller-based boards that serve as the core of Arduino projects. The most commonly used microcontroller on Arduino boards is the Atmel AVR series, such as the ATmega328 on the Arduino Uno.
2. **Arduino Integrated Development Environment (IDE):** This is the software component of the Arduino platform used for writing, compiling, and uploading code to the Arduino boards. The Arduino IDE simplifies the programming process for users, providing a user-friendly interface for writing code in a language based on C/C++.

Difference between microprocessor and microcontroller

Characteristic	Microprocessor	Microcontroller
Function	Central processing unit (CPU) for general-purpose computing.	CPU, memory, and peripherals integrated on a single chip, designed for specific applications in embedded systems.
Applications	Personal computers, servers, general-purpose computing devices.	Embedded systems, control systems, robotics, electronic devices, etc.
Components on Chip	Typically includes only the CPU. External components added as needed (e.g., memory, peripherals).	Integrates CPU, memory (program and data), and various peripherals on a single chip.
Complexity and Specialization	More complex and versatile. Not highly specialized for specific applications.	Less complex but highly specialized for specific tasks. Optimized for efficiency in particular applications.
Power Consumption	May consume more power due to higher processing capabilities.	Often designed for low-power applications, suitable for battery-powered devices.
Cost	Generally more expensive due to versatility and higher processing power.	Typically more cost-effective, designed for specific applications, and omitting unnecessary components.
Examples	Intel i7, AMD Ryzen	Atmel, ARM,8085,80386

Problem Statement No 13

Understanding the connectivity of Raspberry-Pi / Adriano with IR sensor. Write an Application to detect obstacle and notify user using LEDs.

IR Sensor

An IR (infrared) sensor is a device that can detect and measure infrared radiation in its surroundings. Infrared radiation is a type of electromagnetic radiation with wavelengths longer than those of visible light but shorter than microwaves. IR sensors are commonly used in a variety of applications, ranging from remote controls to proximity sensors and even in industrial automation.

Working

Working Principle: Infrared proximity sensors emit infrared light and measure the reflection off an object to determine its proximity. The basic principle involves an emitter and a detector. The emitter sends out infrared light, and the detector measures the amount of reflected light. Components: The emitter is typically an infrared LED, and the detector is a photodiode or phototransistor. The amount of reflected light changes based on the distance and characteristics of the reflecting surface. Operation: The sensor measures the intensity of the reflected light, and this information is used to determine the proximity of an object. When an object comes closer, the amount of reflected light increases, and vice versa.

```
int SensorPin = 13;

int LedPin = 2;

int BuzzerPin = 3;


void setup() {

pinMode(LedPin, OUTPUT);

pinMode(BuzzerPin, OUTPUT);

pinMode(SensorPin, INPUT);

Serial.begin(9600);

}


void loop() {

int SensorValue = digitalRead(SensorPin);

Serial.print("SensorPin: ");

Serial.println(SensorValue);
```

```
delay(100);
if (SensorValue == 1) {
digitalWrite(LedPin, HIGH);
digitalWrite(BuzzerPin, LOW);
} else {
digitalWrite(LedPin, LOW);
digitalWrite(BuzzerPin, HIGH);
}
}
```

Problem Statement No 14

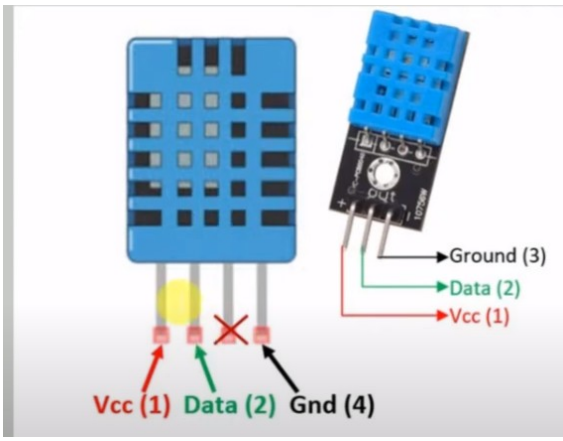
Understanding the connectivity of Raspberry-Pi /Beagle board circuit with temperature sensor. Write an application to read the environment temperature. If temperature crosses a threshold value, generate alerts using LEDs.

What is temperature sensor?

The Raspberry Pi itself does not have an onboard temperature sensor, but it can interface with external temperature sensors. One commonly used temperature sensor with the Raspberry Pi is the DHT series, specifically the DHT11 and DHT22.

For DHT11 Sensor		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	NC	No Connection and hence not used
4	Ground	Connected to the ground of the circuit
For DHT11 Sensor module		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

1. **DHT11:**
    - The DHT11 is a low-cost digital temperature and humidity sensor. It provides temperature readings in Celsius and humidity readings in percentage. The sensor has a single-wire digital interface, making it relatively simple to connect to the GPIO (General Purpose Input/Output) pins on the Raspberry Pi.
2. **DHT22 (or AM2302):**
    - The DHT22 is an advanced version of the DHT11, offering a higher level of accuracy and a wider measurement range. Like the DHT11, it provides both temperature and humidity readings.



To interface with these sensors, you may use Python libraries like Adafruit_DHT, which provides an easy-to-use interface for reading data from DHT sensors.

```
import time
import Adafruit_DHT
import RPi.GPIO as GPIO

# Set up GPIO for LEDs
GPIO.setmode(GPIO.BCM)
led_pin = 18  # Example GPIO pin for the LED
GPIO.setup(led_pin, GPIO.OUT)

# Initialize DHT11 sensor
dht_pin = 17  # GPIO pin for DHT11 data
sensor = Adafruit_DHT.DHT11

# Threshold temperature in Celsius
threshold_temp = 25.0

try:
    while True:
        # Read temperature and humidity from DHT11 sensor
        humidity, temperature = Adafruit_DHT.read_retry(sensor, dht_pin)

        if humidity is not None and temperature is not None:
            # Print temperature and humidity
            print(f"Temperature: {temperature:.2f}°C, Humidity: {humidity:.2f}%")


            # Check if temperature crosses the threshold
            if temperature > threshold_temp:
                GPIO.output(led_pin, GPIO.HIGH)  # Turn on the LED
            else:
                GPIO.output(led_pin, GPIO.LOW)  # Turn off the LED
        else:
            print("Failed to retrieve data from DHT11 sensor")

        time.sleep(2)  # Wait for 2 seconds before reading again

except KeyboardInterrupt:
    GPIO.cleanup()  # Clean up GPIO on program exit
```

Problem Statement No 15

Understanding and connectivity of Raspberry-Pi /Beagle board with camera. Write an Application to capture and store the image.

Raspberry Pi Camera Module

The Raspberry Pi Camera Module is a compact and versatile camera accessory designed specifically for use with the Raspberry Pi single-board computers. It allows users to capture still images and record video using their Raspberry Pi, providing a wide range of applications, from simple photography to advanced computer vision projects. As of my last knowledge update in January 2022, the Raspberry Pi Camera Module has evolved, and there are primarily two versions:

1. **Raspberry Pi Camera Module V1:**
    - This was the first camera module released by the Raspberry Pi Foundation.
    - It featured a 5-megapixel fixed-focus camera.
    - It supported still image capture and video recording in 1080p at 30 frames per second (fps).
    - The module connected to the Raspberry Pi via a ribbon cable and a dedicated camera port.
2. **Raspberry Pi Camera Module V2:**
    - The V2 module is an improved version of the original Camera Module.
    - It features an 8-megapixel Sony IMX219 sensor.
    - It supports still image capture up to 3280 x 2464 pixels and video recording in 1080p at 30fps, 720p at 60fps, and 640 x 480 pixels at 90fps.
    - The V2 module also connects to the Raspberry Pi via a ribbon cable and a dedicated camera port.

Video Recording

```
from picamera import PiCamera
from time import sleep
camera=PiCamera()
camera.vflip=True
camera.start_preview()
camera.start_recording('/home/pi/d_b.h264')
sleep(5)
camera.resolution=(600,900)
camera.stop_recording()
camera.stop_preview()
```

Run h264 files

omxplayer your_video_file.h264

Picture Screenshot

```
from picamera import PiCamera
from time import sleep
camera=PiCamera()
camera.start_preview()
camera.vflip=True
sleep(10)
camera.capture('/home/pi/Desktop/test_d.jpg')
camera.stop_preview()
```

Problem Statement No 16

Create a small dashboard application to be deployed on cloud. Different publisher devices can publish their information and interested application can subscribe.

What is ThingSpeak cloud

ThingSpeak is an Internet of Things (IoT) platform that enables the collection, storage, analysis, and visualization of data from various connected devices. It provides a cloud-based infrastructure to support IoT applications and projects. ThingSpeak is particularly popular for its ease of use and integration with various hardware platforms, making it accessible for both beginners and experienced developers.

```
import Adafruit_DHT as dht
from urllib.request import urlopen

myAPI = '*****DJABRJ*****'
ThingsURL = 'https://api.thingspeak.com/update?api_key=%s' % myAPI

def DHT11_data():
    hum1, temp1 = dht.read_retry(dht.DHT11, 23)
    return hum1, temp1

while True:
    hum, temp = DHT11_data()

    if isinstance(hum, float) and isinstance(temp, float):
        hum = '%.2f' % hum
        temp = '%.2f' % temp
        print(hum, temp)

        coms = urlopen(ThingsURL + '&field1=%s&field2=%s' % (temp, hum))
        print(coms.read())
```

Problem Statement No 17

Understanding the connectivity of Raspberry-Pi / Arduino with Ultra Sonic sensor. Write an application to detect Distance of an object and notify the distance to user.

What is ultra sonic sensor

An ultrasonic sensor is a device that uses ultrasonic waves, or sound waves with frequencies higher than the upper audible limit of human hearing, for various applications.

Ultrasonic sensors are commonly used for distance measurement, object detection, and proximity sensing. They work on the principle of sending out ultrasonic pulses and measuring the time it takes for the echoes to return after bouncing off an object.

Here's a basic overview of how ultrasonic sensors work:

1. **Transmitter:** The ultrasonic sensor has a component that acts as a transmitter. This component emits ultrasonic waves, typically in the range of 40 kHz to 200 kHz, depending on the sensor.
2. **Propagation of Ultrasonic Waves:** The emitted ultrasonic waves travel through the air in a directional manner. The sensor is designed to focus the waves in a specific direction.
3. **Reflection:** When the ultrasonic waves encounter an object in their path, they get reflected back towards the sensor.
4. **Receiver:** The sensor also contains a receiver that detects the reflected ultrasonic waves.
5. **Time Measurement:** The sensor measures the time it takes for the ultrasonic waves to travel to the object and back. Using the speed of sound in air, the sensor can calculate the distance to the object based on the time-of-flight of the ultrasonic waves.
6. **Distance Calculation:** The distance to the object is calculated using the formula:

   $$\text{Distance} = (\text{Speed of Sound} \times \text{Time}) / 2$$

7. **Output:** The sensor provides a distance measurement as its output, which can be used for various applications such as robotics, automation, or any scenario where non-contact distance sensing is required.

Ultrasonic sensors have several advantages, including:

- **Non-contact Operation:** Ultrasonic sensors do not require physical contact with the object being measured, making them suitable for applications where contact is impractical or undesirable.
- **Versatility:** They can be used for measuring a wide range of distances, from a few centimeters to several meters, depending on the specific sensor model.
- **Reliability:** Ultrasonic sensors are generally reliable in various environmental conditions, such as lighting variations, temperature changes, and atmospheric conditions.

```
const int trigPin = 9;
const int LED =11;
const int echoPin = 10;
long duration;
int distance;
void setup() {
  Serial.println("Started");
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(LED, OUTPUT);
  Serial.begin(4800); // Starts the serial communication
}
void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

```
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.println(distance);
  if(distance<100 && distance>20)
  {
    digitalWrite(LED, HIGH);
    delay(200);
    digitalWrite(LED, LOW);
    delay(200);
  }
  if(distance<=20)
  {
    digitalWrite(LED, HIGH);
    delay(50);
    digitalWrite(LED, LOW);
    delay(50);
  }
}
```