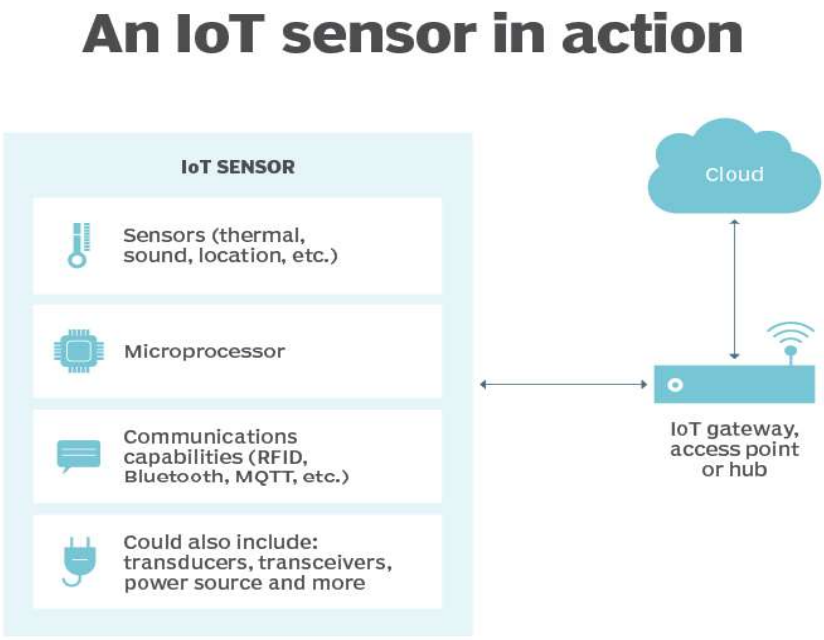
UNIT – IV

IoT System Design

What is a sensor?

A sensor is a device that detects and responds to some type of input from the physical environment. The input can be light, heat, motion, moisture, pressure or any number of other environmental phenomena. The output is generally a signal that is converted to a human-readable display at the sensor location or transmitted electronically over a network for reading or further processing.

Sensors play a pivotal role in the internet of things (IoT). They make it possible to create an ecosystem for collecting and processing data about a specific environment so it can be monitored, managed and controlled more easily and efficiently. IoT sensors are used in homes, out in the field, in automobiles, on airplanes, in industrial settings and in other environments. Sensors bridge the gap between the physical world and logical world, acting as the eyes and ears for a computing infrastructure that analyzes and acts upon the data collected from the sensors.



What are the types of sensors?

Sensors can be categorized in multiple ways. One common approach is to classify them as either active or passive. An active sensor is one that requires an external power source to be able to respond to environmental input and generate output. For example, sensors used in weather satellites often require some source of energy to provide meteorological data about the Earth's atmosphere.

A passive sensor, on the other hand, doesn't require an external power source to detect environmental input. It relies on the environment itself for its power, using sources such as light or thermal energy. A good example is the mercury-based glass thermometer. The mercury expands and contracts in response to fluctuating temperatures, causing the level to be higher or lower in the glass tube. External markings provide a human-readable gauge for viewing the temperature.

Some types of sensors, such as seismic and infrared light sensors, are available in both active and passive forms. The environment in which the sensor is deployed typically determines which type is best suited for the application.

Another way in which sensors can be classified is by whether they're analog or digital, based on the type of output the sensors produce. Analog sensors convert the environmental input into output analog signals, which are continuous and varying. Thermocouples that are used in gas hot water heaters offer a good example of analog sensors. The water heater's pilot light continuously heats the thermocouple. If the pilot light goes out, the thermocouple cools, sending a different analog signal that indicates the gas should be shut off.

In contrast to analog sensors, digital sensors convert the environmental input into discrete digital signals that are transmitted in a binary format (1s and 0s). Digital sensors have become quite common across all industries, replacing analog sensors in many situations. For example, digital sensors

are now used to measure humidity, temperature, atmospheric pressure, air quality and many other types of environmental phenomena.

As with active and passive sensors, some types of sensors -- such as thermal or pressure sensors -- are available in both analog and digital forms. In this case, too, the environment in which the sensor will operate typically determines which is the best option.

Sensors are also commonly categorized by the type of environmental factors they monitor. Here are some common examples:

- **Accelerometer.** This type of sensor detects changes in gravitational acceleration, making it possible to measure tilt, vibration and, of course, acceleration. Accelerometer sensors are used in a wide range of industries, from consumer electronics to professional sports to aerospace and aviation.
- **Chemical.** Chemical sensors detect a specific chemical substance within a medium (gas, liquid or solid). A chemical sensor can be used to detect soil nutrient levels in a crop field, smoke or carbon monoxide in a room, pH levels in a body of water, the amount of alcohol on someone's breath or in any number of other scenarios. For example, an oxygen sensor in a car's emission control system will monitor the gasoline-to-oxygen ratio, usually through a chemical reaction that generates voltage. A computer in the engine compartment reads the voltage and, if the mixture is not optimal, readjusts the ratio.
- **Humidity.** These sensors can detect the level of water vapors in the air to determine the relative humidity. Humidity sensors often include temperature readings because relative humidity is dependent on the air temperature. The sensors are used in a wide range of industries and settings, including agriculture, manufacturing, data centers, meteorology, and heating, ventilation and air conditioning (HVAC).

- **Level.** A level sensor can determine the level of a physical substance such as water, fuel, coolant, grain, fertilizer or waste. Motorists, for example, rely on their gas level sensors to ensure they don't end up stranded on the side of the road. Level sensors are also used in tsunami warning systems.
- **Motion.** Motion detectors can sense physical movement in a defined space (the field of detection) and can be used to control lights, cameras, parking gates, water faucets, security systems, automatic door openers and numerous other systems. The sensors typically send out some type of energy -- such as microwaves, ultrasonic waves or light beams -- and can detect when the flow of energy is interrupted by something entering its path.
- **Optical.** Optical sensors, also called photosensors, can detect light waves at different points in the light spectrum, including ultraviolet light, visible light and infrared light. Optical sensors are used extensively in smartphones, robotics, Blu-ray players, home security systems, medical devices and a wide range of other systems.
- **Pressure.** These sensors detect the pressure of a liquid or gas, and are used extensively in machinery, automobiles, aircraft, HVAC systems and other environments. They also play an important role in meteorology by measuring the atmospheric pressure. In addition, pressure sensors can be used to monitor the flow of gases or liquids, often so that the flow can be regulated.
- **Proximity.** Proximity sensors detect the presence of an object or determine the distance between objects. Proximity monitors are used in elevators, assembly lines, parking lots, retail stores, automobiles, robotics and numerous other environments.
- **Temperature.** These sensors can identify the temperature of a target medium, whether gas, liquid or air. Temperature sensors

are used across a wide range of devices and environments, such as appliances, machinery, aircraft, automobiles, computers, greenhouses, farms, thermostats and many other devices.

- **Touch.** Touch sensing devices detect physical contact on a monitored surface. Touch sensors are used extensively in electronic devices to
  support trackpad and touchscreen technologies. They're also used in many other systems, such as elevators, robotics and soap dispensers.

DHT11–Temperature and Humidity Sensor

What is a DHT11 sensor

DHT11 is a low-cost digital sensor, used to measure temperature and humidity in the surroundings. DHT sensor has three main components i.e.

1. A resistive-type sensor (used to measure humidity)
2. NTC Temperature Sensor (used to measure temperature)
3. 8-Bit Microcontroller (to calibrate & serially transmit the values as a digital signal)

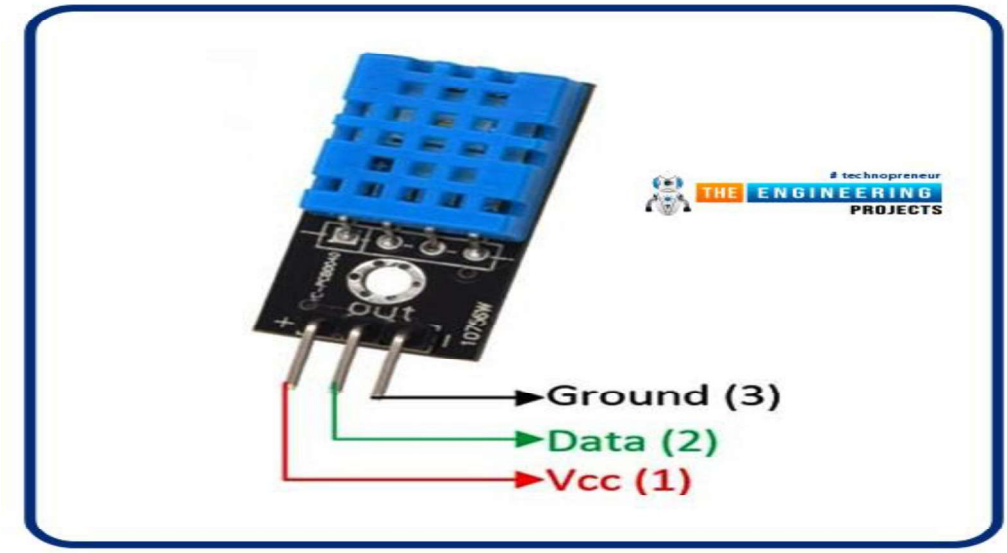
Pinout of DHT11 sensor:

DHT11 has 4 pins in total, which are:

1. Vcc: We need to provide +5V at this pin.
2. Data Pin: We need to connect it to the Digital Pin of the Microcontroller to get humidity & temperature data.
3. NC: Not Connected.
4. GND: We need to connect it to the Ground Pin.

Pin`s Number	Pin`s Name	Function
1	Vcc	Input power for the power supply pin ranges from 3.5 to 5.5 volts.
2	Data	Serial data can be sent to the output pins, which can then be used to calculate humidity and temperature.
3	NC	There isn't a pin for connection. Using this pin will not work.
4	Ground	The sensor is connected to the ground of the system.

Pinout of the DHT11 Module:



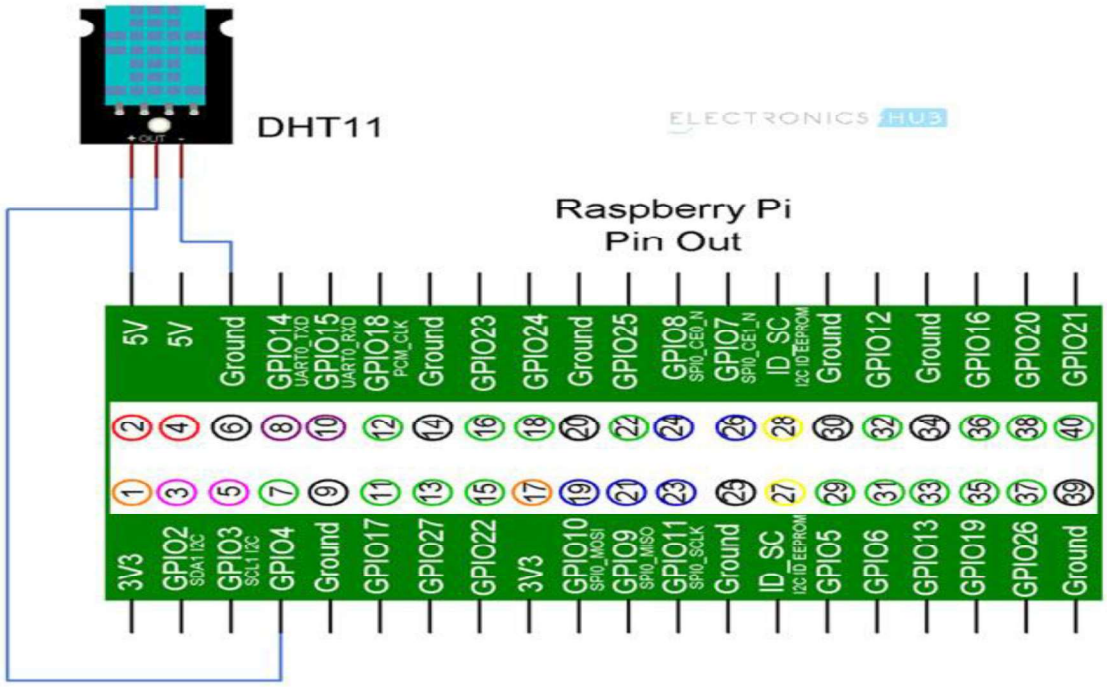
DHT11 Working Principle

This device uses a thermistor to monitor temperature and a capacitive humidity sensor to measure relative humidity. A humidity-detecting capacitor's electrodes are separated by a dielectric substrate that retains moisture. When the humidity level fluctuates, the capacitance value changes as well. Analog resistance values are measured, processed, and stored by the IC, which then translates them into digital values.

The thermistor works on the simple principle of change in resistance due to a change in temperature. When the ambient temperature changes the

thermistor starts self-heating its elements. its resistance value is changed with respect to this change in temperature. This change depends on the type of thermistor used. The resistance value of this sensor is monitored, as it warms up using a negative temperature coefficient thermometer (NTC). This sensor is commonly made of semiconductor ceramics in order to achieve a higher resistance value even at the smallest temperature change.

Circuit Diagram of DHT11 with Raspberry Pi



Ultrasonic Sensor

An Ultrasonic Sensor consists of a transmitter and a receiver, the transmitter emits the ultrasonic wave, which after hitting some object bounces back and receives by the ultrasonic receiver. If the Ultrasonic sensor is operated at 5V, it normally measures a distance of up to 450 centimeters.

Ultrasonic Sensor Pinout

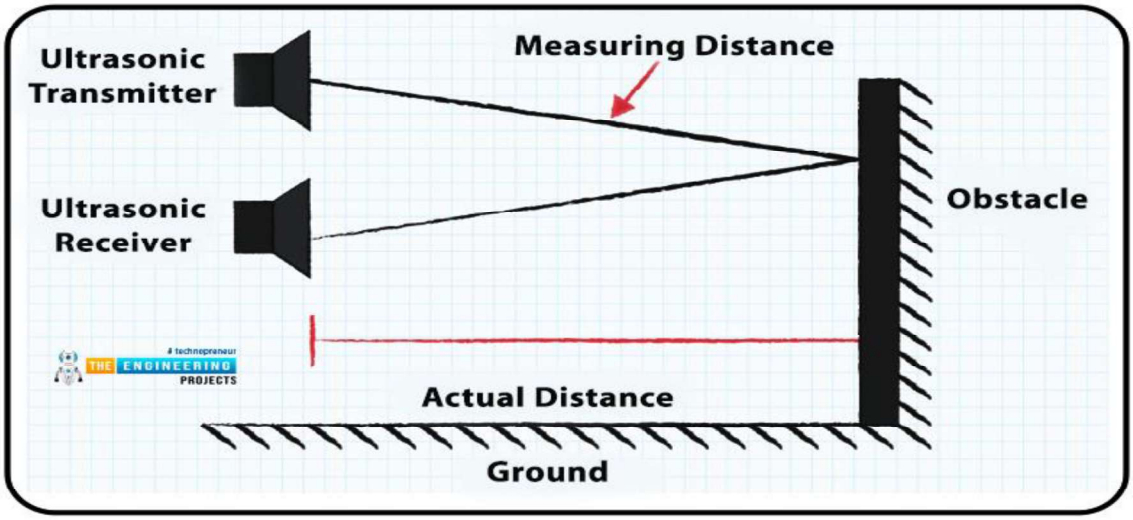
The Ultrasonic Sensor has four connections:

- Vcc: We need to provide +5V here.
- Trig: Trigger Pin(Connected to Microcontroller)
- Echo: The sensor's response (Connected to Microcontroller)
- GND: We need to provide Gound here.

Ultrasonic Sensor Working Principle

An ultrasonic sensor is made up of two parts: a transmitter and a receiver, arranged side by side as close as possible. Ultrasonic sensors measure distance by sending and receiving the ultrasonic wave. The ultrasonic sensor has a sender to emit the ultrasonic waves and a receiver to receive the ultrasonic waves. The transmitted ultrasonic wave travels through the air and is reflected by hitting the Object. Raspberry Pi calculates the time taken by the ultrasonic pulse wave to reach the receiver from the sender.

How Is the Distance Calculated?



We know that the speed of sound in air is nearly 344 m/s,
So, the known parameters are time and speed (constant). Using these parameters, we can calculate the distance travelled by the sound wave.

Formula:

Distance = Speed * Time/2
Time taken by pulse is for to and from travel of ultrasonic signals, while we need only half of this. Therefore, time is taken as time/2.

Speed of sound at sea level = 343 m/s or 34300 cm/s

Thus, Distance = 17150 * Time (unit cm)

Interface Ultrasonic Sensor with Raspberry Pi

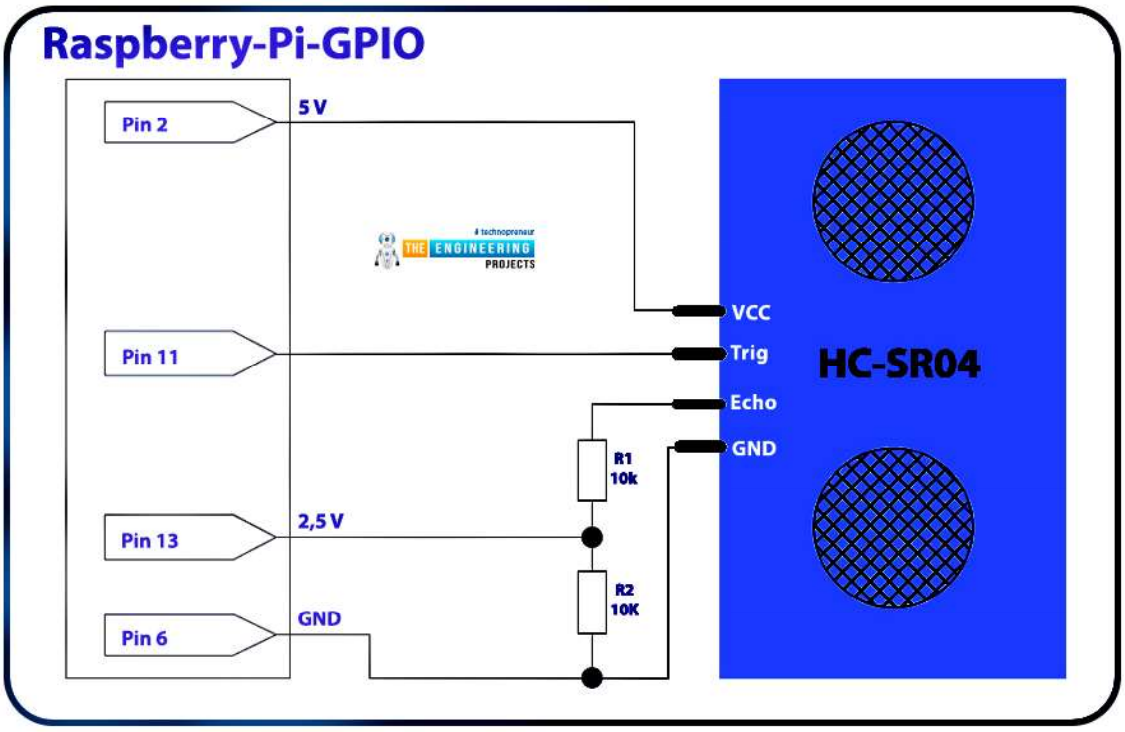
As we discussed in the Pinout section, Ultrasonic Sensor has 4 pins in total.

- The sensor emits an ultrasonic signal if a high pulse is detected on the Trigger pin.
- An echo will return a HIGH signal to the Microcontroller, almost at the same time.
- Until the reflected sound is detected, the Echo Pin keeps on sending a HIGH signal.
- When the sensor's receiver gets a sound back, the Echo output gets LOW.
- With the time difference between the rising and falling edges of your Echo pin, we can easily get the distance between the sensor and the obstacle in the software.

Circuit Diagram

Here's the circuit Diagram of the Ultrasonic sensor (HC-SR04) with Raspberry

Pi



- As you can see in the above figure, we have placed two resisters at the Echo Pin, to divide the voltage, as we need 3.3V max at RPi4 GPIO.
- Any spare GPIO pin can be used, instead of Pin11 and Pin13.

Motion Detection with PIR Sensor & Raspberry Pi

What is a PIR sensor?

- Infrared motion detectors, also known as passive infrared (PIR) sensors, are a type of motion sensor that makes use of IR light to identify and locate the source of motion.

- The sensor can detect human movement within an 8-meter radius around it.
    - Anything, alive or otherwise, having a temperature greater than zero degrees Celsius emits infrared radiation.
    - The wavelength of infrared radiation is far greater than that of visible light, making it invisible to the human eye.
    - Passive infrared (PIR) sensors are specifically designed to detect these heat signatures.

As their name implies, passive motion sensors don't put out any rays of their own but instead pick up the infrared radiations emitted by other objects, making them ideal for use in intruder alarm devices. However, active detectors may produce and detect infrared light at the same time.

PIR Sensor Pinout

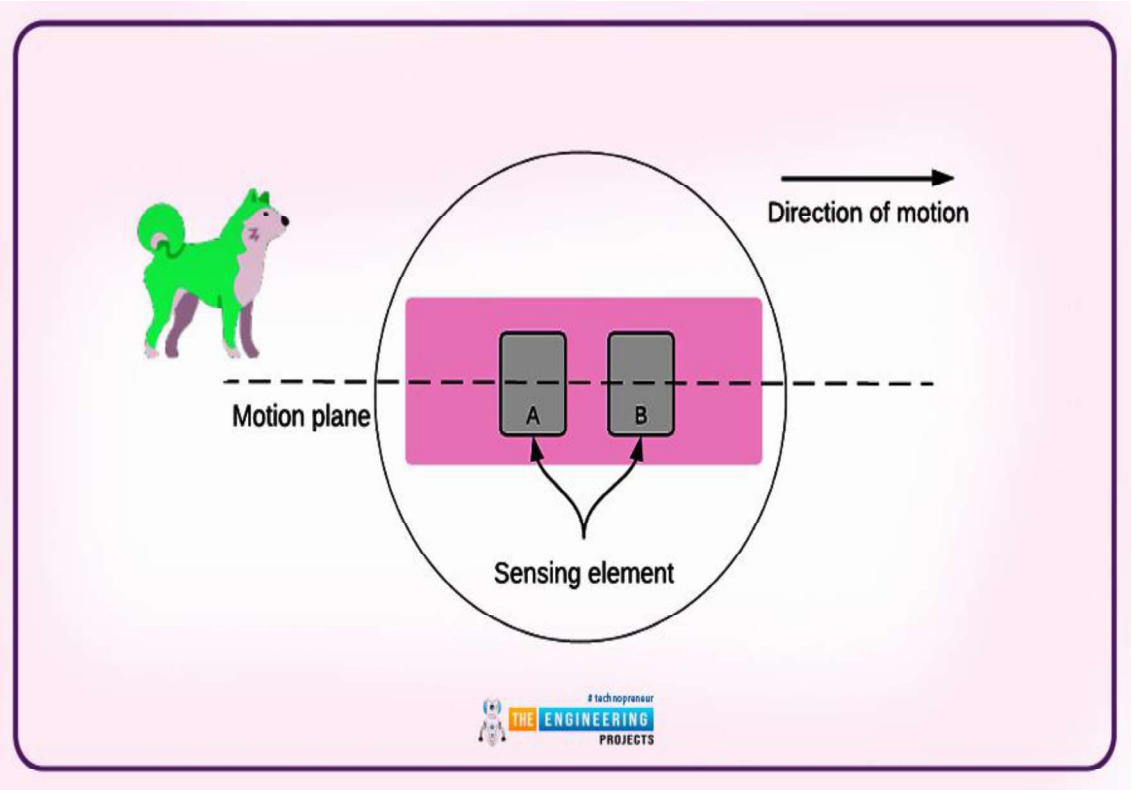
The PIR motion sensor has three pins:

- Pin1 is Vcc: We need to provide +5V to this pin.
- Pin2 is the data Pin: It's a digital Pin and sends sensors' data.
- Pin3 is GND: We need to connect it to the ground.

PIR Sensor Working Principle

In PIR Sensor, crystals sensitive to infrared light are used as sensors. As it’s a passive IR sensor, the sensor doesn't emit any IR waves, instead, it waits for the infrared-emitting object.

The IR sensing component consists of two subassemblies, A and B.



When there is no motion, the two detectors pick up identical infrared readings, which cancel out one another. Sensing element A will pick up the presence of infrared light, when an infrared-emitting object, such as a dog, enters the sensor's field of vision. Since the intensity of the infrared light striking sensing element B is still relatively low, the resulting differential change is positive.

As the object moves past the sensor, the intensity of the infrared light falling on sensing element B will be greater than that falling on sensing element A, resulting in a negative differential change. The BISS0001 logic chip onboard detects and amplifies this potential difference before outputting it as a digital signal.

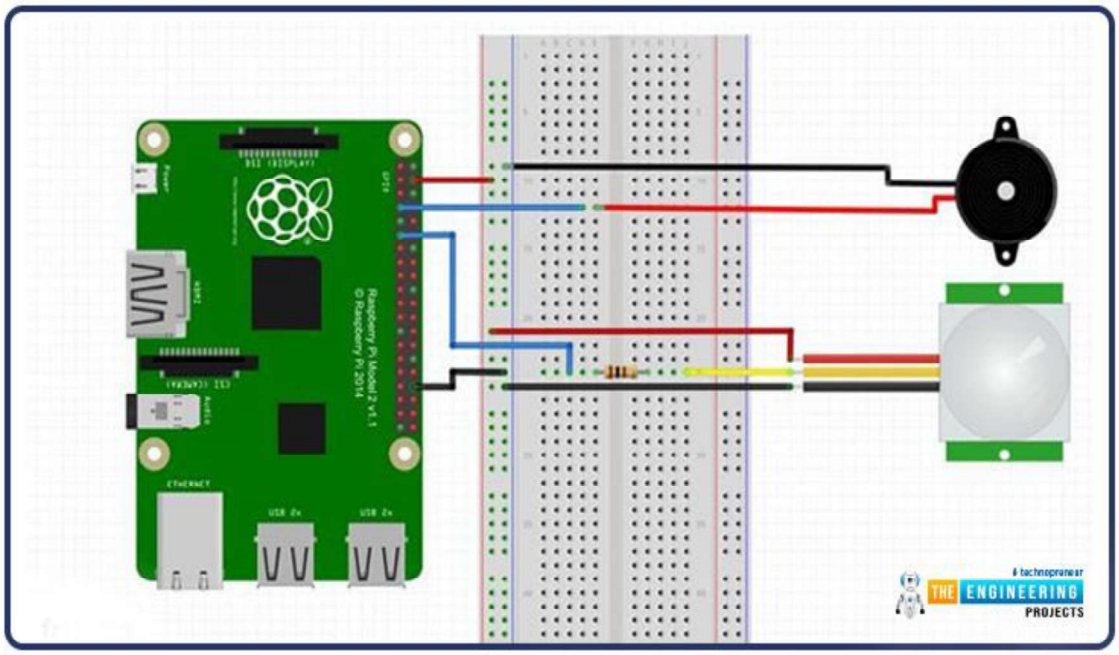
When the infrared detector detects movement, it sends a signal to the microcontroller through the data input, which goes HIGH.

Circuit Diagram of PIR Sensor with Raspberry Pi

We will design a simple Motion Detection Project using PIR Sensor & Piezo Speaker with Raspberry Pi 4. It's a simple security system where the PIR sensor will detect motion and Piezo Speaker will trigger the alarm.

A piezo buzzer is an easy-to-use speaker that makes noise whenever an electric current passes through it. The buzzer will sound an audible alert when the motion is detected.

Here's the circuit diagram of PIR Sensor with RPi4:



Just follow these steps to build the circuit.

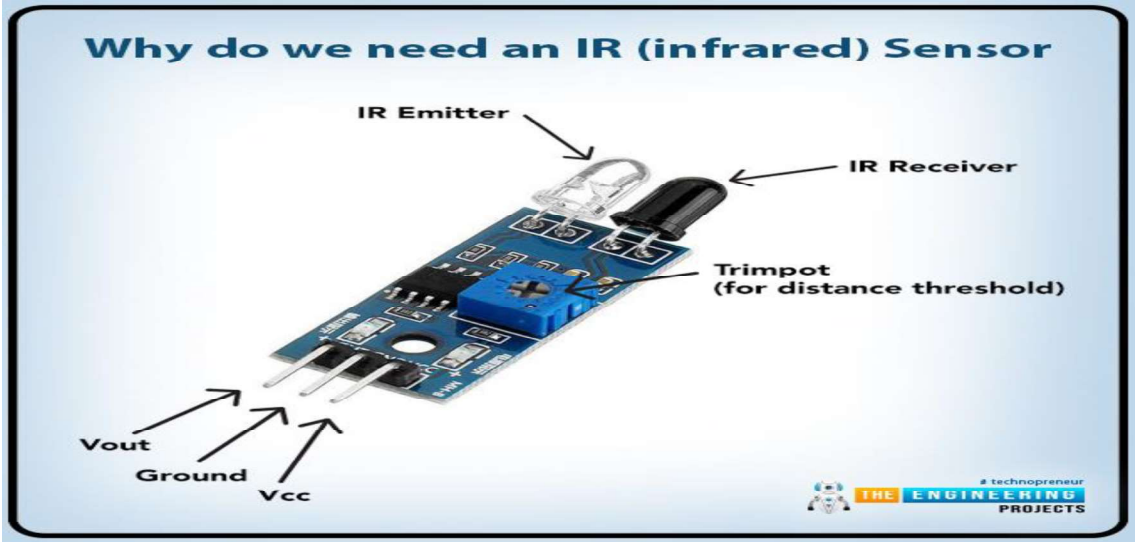
1. Connect the Ground Pin to the breadboard's ground rail.
2. Connect the breadboard's positive rail with the +5v Pin.
3. One wire of the Piezo buzzer should be connected to Pin7 of RPi and the second wire to the ground.
4. Connect Pin11 of RPi4 to the breadboard with a wire. Connect a resistor of 100 ohms to the wire's terminal. The PIR sensor's yellow wire should be connected to the other end of the resistor.
5. The PIR sensor's red wire must be connected to the breadboard's 5V line, while the black wire must be connected to the ground rail.

Infrared (IR) Sensor:

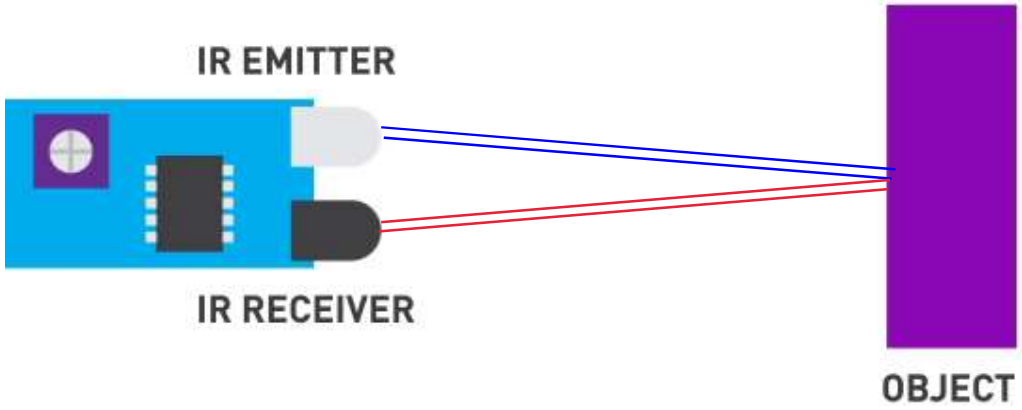
What is IR Sensor?

- The IR sensor (Infrared sensor) consists of a transmitter and a receiver; the transmitter sends out infrared radiations, which are reflected by the barriers/receiver and get back to the receiver of the sensor.
- The sensor computes the incoming signal and sets the module's output.

- With an IR sensor and a Raspberry Pi, we can see what's around us, how fast things are moving, and how far away things are from the sensor.
- There are three connections on this IR sensor: Vcc, Ground, and output.
- Vcc is linked to 5V, Ground is grounded, and output is the pin, from where we read the sensor's data.



Working Principle



An infrared sensor incorporates an infrared LED (light-emitting diode) and a photodiode, which, when joined together, can function as a photo-coupler or optocoupler.

The infrared LED is a transmitter that produces IR radiations. Visually, the IR LED is identical to a regular LED, although the IR radiation is invisible to the naked eye. Radiation sent out by an infrared source is primarily detected by an infrared receiver. Photodiodes are one physical shape that these infrared detectors can take. Unlike regular photodiodes, IR photodiodes only respond to infrared light. Variations in voltage, wavelength, package size, etc., are the primary categories separating the many types of infrared receivers.

The wavelength of the IR receiver must be the same as that of the IR transmitter whenever the two are utilized together. In this case, an infrared LED acts as the sender, and an infrared photodiode acts as the receiver. An infrared LED produces infrared light, which an infrared photodiode can detect. The amount of IR light collected correlates with the photo-resistance diodes and the resulting shift in output voltage. It is on this concept that the IR detector operates.

Some of the infrared radiation sent out by the transmitter will be reflected to the receiver. The IR receiver can calibrate the sensor output to the level of the response.

Infrared Detector Varieties

There are two types of IR sensors are available and they are,

- Active Infrared Sensor
- Passive Infrared Sensor

Active Infrared Sensor

Active infrared sensors consist of two elements: infrared source and infrared detector. Infrared sources include the LED or infrared laser diode. Infrared detectors include photodiodes or phototransistors. The energy emitted by the infrared source is reflected by an object and falls on the infrared detector.

Passive Infrared Sensor

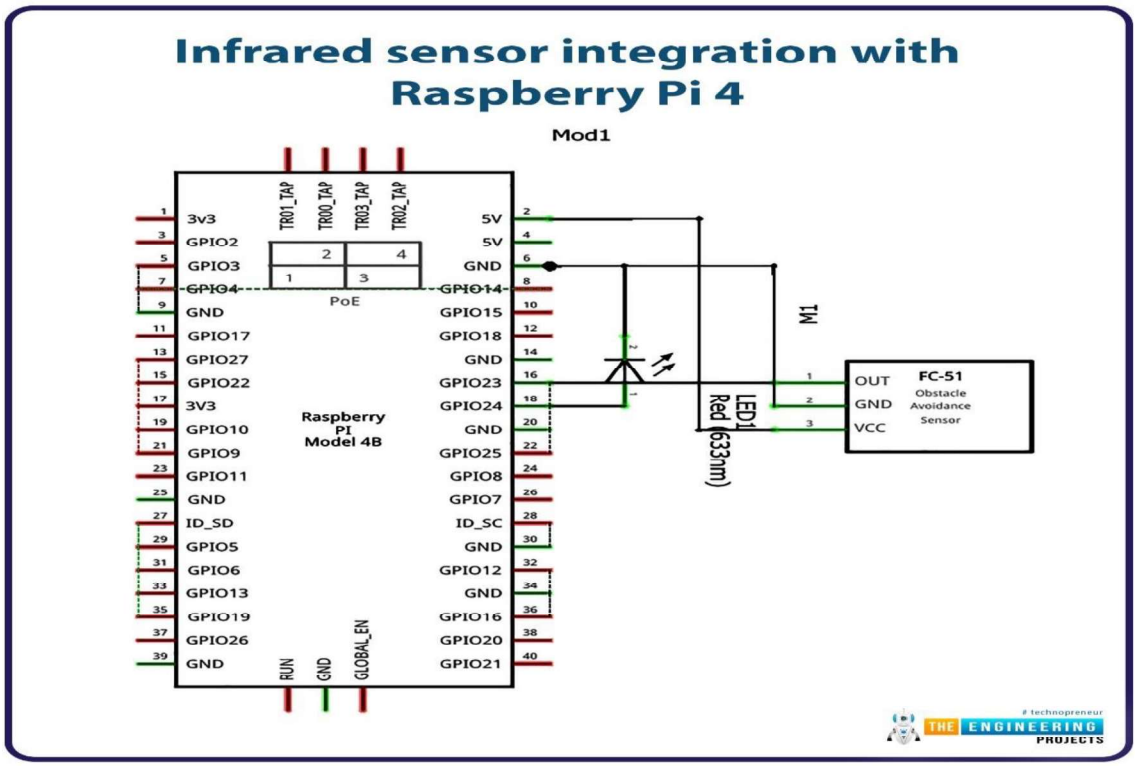
Passive infrared sensors are basically Infrared detectors. Passive infrared sensors do not use any infrared source and detector.

They are of two types: quantum and thermal.

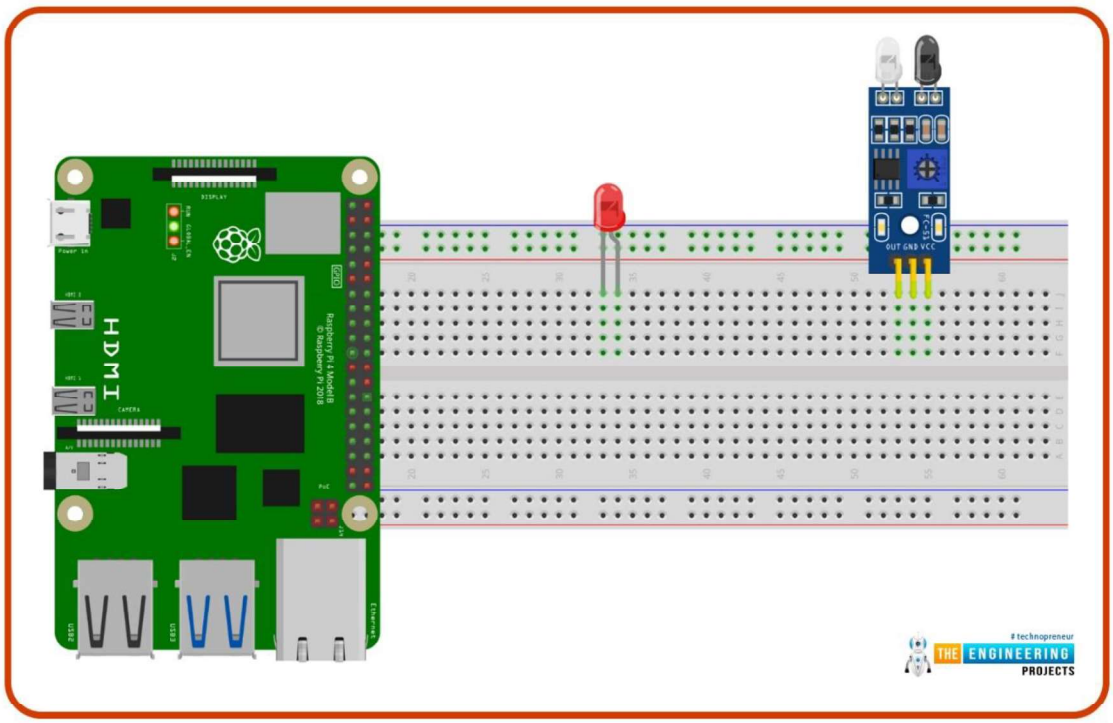
Thermal infrared sensors use infrared energy as the source of heat. Thermocouples, pyroelectric detectors, and bolometers are the common types of thermal infrared detectors. Quantum type infrared sensors offer higher detection performance. It is faster than thermal type infrared detectors. The photo sensitivity of quantum type detectors is wavelength dependent.

IR Sensor with Raspberry Pi

- The presence-detection circuit is depicted as follows in the schematic:

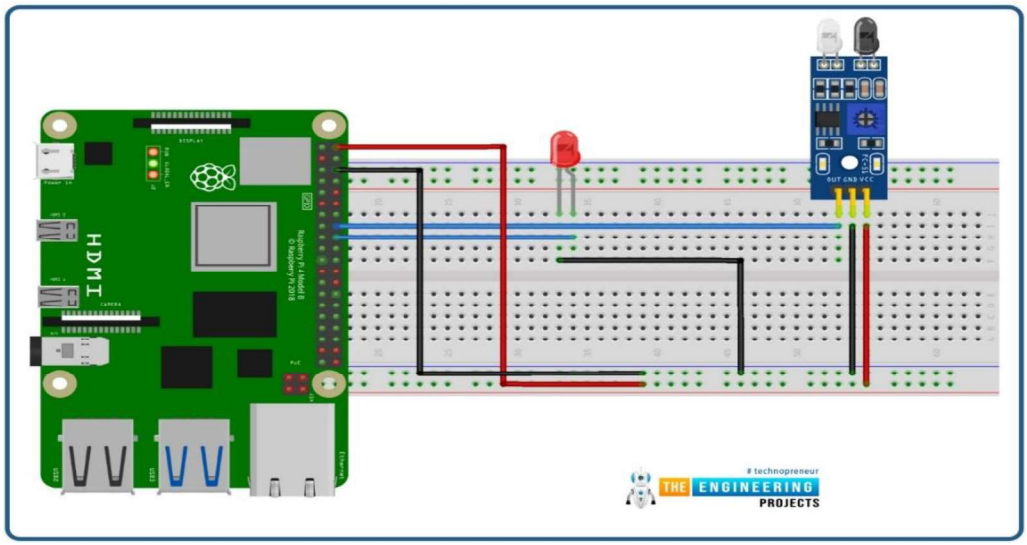


- To construct the circuit depicted in the preceding diagram, we will first attach the Pi 4, the IR sensor, and the LED to the breadboard.



- We'll hook up the Infrared sensor and LED to the Raspberry Pi using the table as a guide.

Cathode of the LED	Connect it with the ground of Raspberry Pi 4
Anode of the LED	Connect it with the GPIO 24 (BCM 18)
Vcc of the IR sensor	Connect it with the 5 volts of Raspberry Pi 4
Ground of the IR sensor	Connect it with the ground of Raspberry Pi 4
Out of the IR sensor	Connect it with the GPIO 23 (BCM 16)



Interfacing a Light Sensor (LDR) with Raspberry Pi

What is a photoresist?

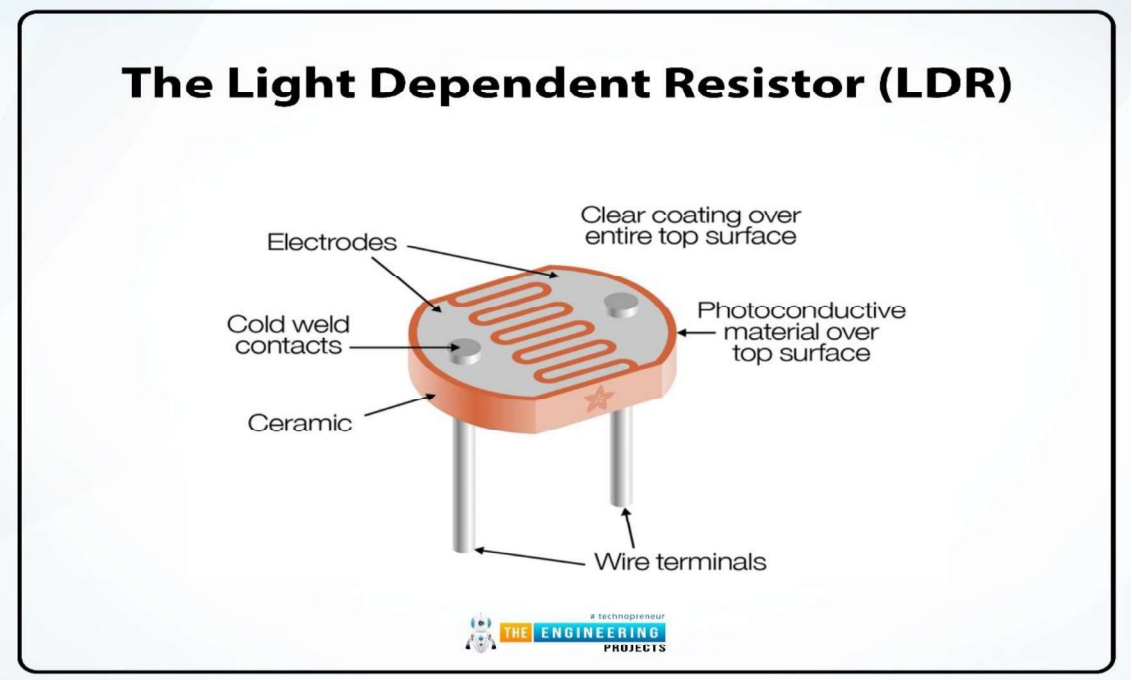
It is a common practice to employ photoresistors to determine the presence or absence of visible light or to quantify the amount of light hitting a particular surface. Their resistance is exceptionally high in the dark, reaching up to 1M ohm, but when subjected to light, the LDR sensor's resistance reduces rapidly, often to only a few ohms. Light-dependent resistors (LDRs) are nonlinear devices whose sensitivity shifts depending on the incident wavelength of light. To protect their ecosystems, some nations have outlawed the use of lead and cadmium in LDRs.

By analyzing the electromagnetic radiation in the "Infrared", "Visible" and "Ultraviolet" regions of the electromagnetic spectrum, Light Sensors can produce an output signal indicative of the brightness of the surrounding light. A passive device called a light sensor transforms this "light energy," which can come from either the visible or infrared regions of the spectrum, into an electrical signal. Because they convert the energy of light (photons) into a usable form of electricity, light sensors are also referred to as photoelectric devices or photo sensors (electrons).

There are two primary types of photoelectric devices: those that produce electricity when exposed to light (photovoltaics, photoemissive, etc.) and those that modify their electrical properties when exposed to light (photoresistors, photoconductors, etc.)

Light Dependent Resistor (LDR)

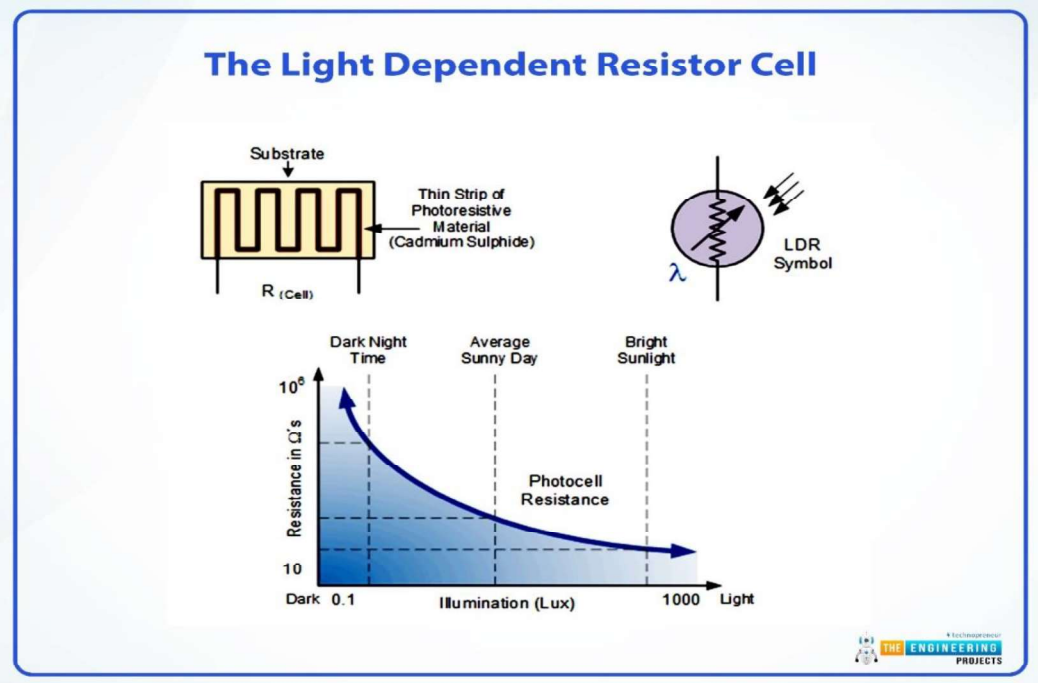
The light-dependent resistor (LDR) sensor is used to detect the intensity of light in the surroundings. The LDR is a device constructed from a sensitive semiconductor material i.e., cadmium sulfide, which undergoes a dramatic shift in electrical resistance when exposed to light, going from several 1000 Ohms in the dark to just a few Ohms, when illuminated.



Most photoresistive light sensors employ cadmium sulfide(CdS). However, other semiconductor substrate materials like lead sulfide (PbS), lead selenide (PbSe), and indium antimony (InSb) can detect light intensity as well. Since cadmium sulfide has a spectral response curve similar to the

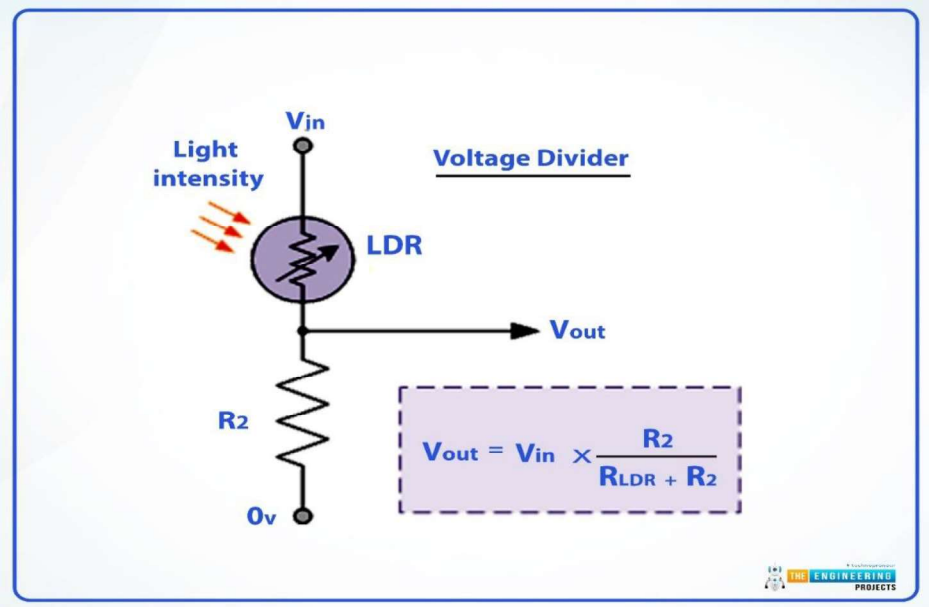
human eye's and can be modulated with a handheld torch, it is utilized to create photoconductive cells. The peak wavelength at which it is most sensitive is typically between 560-600nm (nanometers), making it part of the visible spectrum.

The Light Dependent Resistor Cell

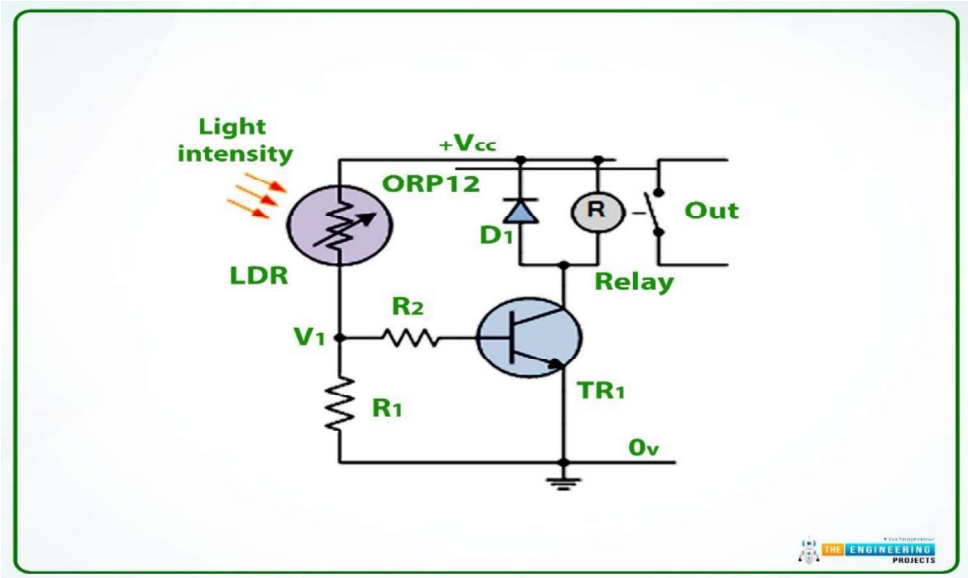


The ORP12 cadmium sulfide photoconductive cell is the most widely used photoresistive light sensor. This photosensitive resistor's spectral response is concentrated around 610 nm in the yellow-to-orange part of the spectrum. When the cell is in the dark, its resistance is extremely high at around 10M's, but it drops to about 100's when illuminated (lit resistance). As the resistive path zigzags across the ceramic substrate, the dark resistance increases and the dark current drops. Because of its low price and wide range of possible applications, the CdS photocell is frequently used in auto-dimming systems,

light- and dark-sensing controls for streetlights, and photographic exposure meters.



Below is an illustration of how a light-dependent resistor can be used as a light-sensitive switch.

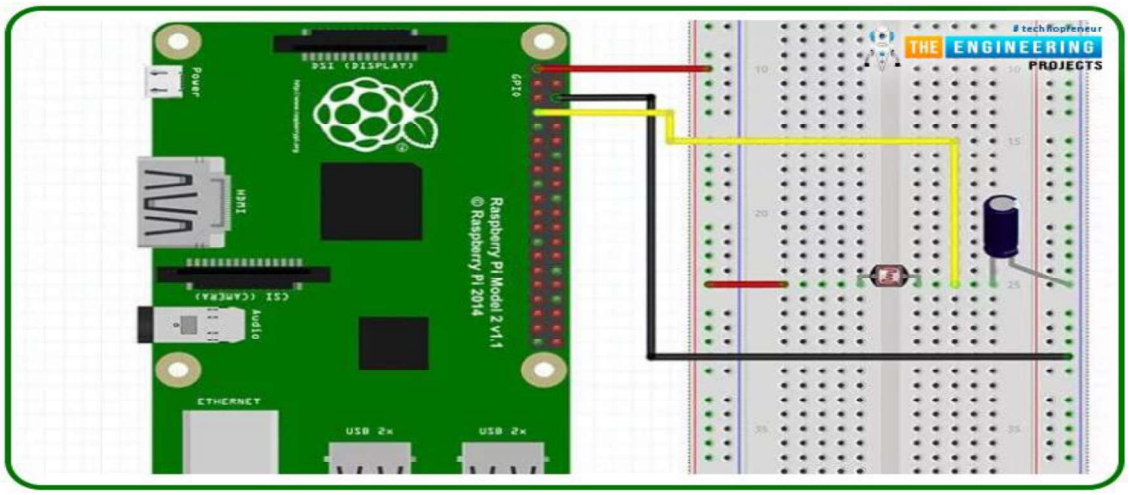


This simple circuit for detecting light consists of a relay activated by exposure to sunlight. The photoresistor LDR and the resistor R1 make up a potential divider circuit. In the absence of light, the LDR's resistance rises into the Megaohm (M) range, and as a result, the transistor TR1 receives zero base bias, turning the relay off. The LDR's resistance drops in response to more light, elevating the base bias voltage at V1. When the base bias voltage of transistor TR1 reaches a certain threshold, as defined by the resistance R1 in a potential divider network, the transistor turns "ON," activating the relay, which controls some external circuitry. With a return to darkness, the LDR's resistance rises, reducing the transistor's base voltage and turning "OFF" the transistor and relay at a predetermined level of illumination established by the potentiometer circuit.

Changing the relay's "ON" or "OFF" point to a custom brightness is as simple as swapping out the fixed resistor R1 for a potentiometer VR1. The switching end of a simple circuit like the one depicted above may need to be more consistent owing to fluctuations in temperature or supply voltage. Using the LDR in a "Wheatstone Bridge" configuration and substituting an Operational Amplifier for the transistor makes it simple to construct a light-activated circuit with increased sensitivity.

Circuit Diagram of LDR with RPi

To build the circuit of the LDR sensor with RPi4, follow these instructions. You can also refer to the below circuit diagram:



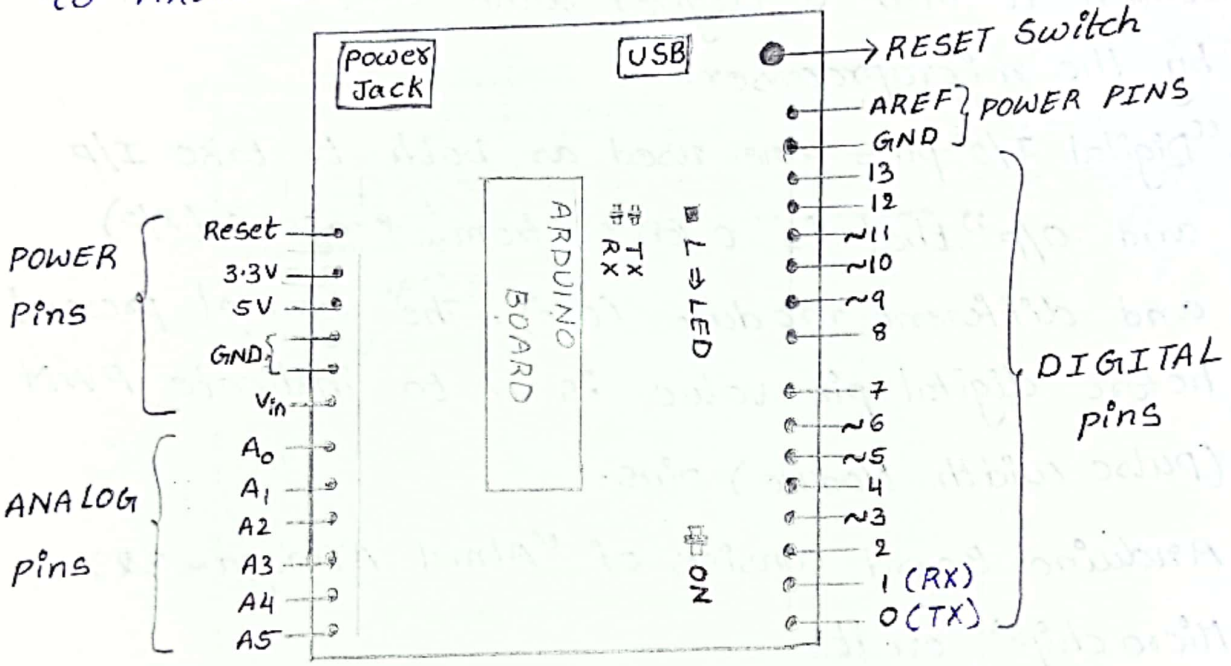
1. To begin, attach Pin1 of RPi4 to the breadboard's positive rail.
2. Then, attach pin 6 of RPi4 to the breadboard's ground rail.
3. Next, place the LDR sensor on the board with one end connected to the power supply's positive rail.
4. Connect the opposite end of the LDR sensor to the Pin4 of RPi4 using a jumper wire.

Arduino:-

→ Arduino is an "open-source microcontroller", that takes input from devices, processes the Input and produces the output based on logic we built.

→ Arduino has "14 digital Input/output pins" in which 6 pins are capable of pulse width Module output. It has "6 Analog pins".

→ The programs to activate devices that are connected to Arduino Board, are developed in "Arduino IDE".



TX ⇒ Transmitter

RX ⇒ Receiver

PIN CONFIGURATION FOR ARDUINO BOARD

→ Arduino works on "cross platform".

→ The "operating voltage of Arduino is 5V" and the Accepted input voltage is "7V to 12V"

→ Arduino Board is nothing but simply a processor along with required and peripherals to connect the devices as well as to process the data collected from devices.

Advantages:-

1) Inexpensive
2) Cross-platform
3) Simple, clear programming Environment
4) Open Source and Extensible Software & Hardware

→ "Analog pins present in Arduino Board is used to read the signal from the analog sensor and convert it into a digital value" that can be read by the microprocessor.

→ "Digital I/O pins are used as both, to take I/P and O/P" that is 0 (or) 1 from sensors (I/P) and different modules (O/P). The symbol present before digital pin value is ~ to indicate PWM (Pulse width Module) pins.

→ Arduino Board consists of "Atmel ATmega-328 Microchip" on it.

→ Arduino Board consists of "USB ports to connect Input-output devices".

→ Arduino Board has a "power Jack which is used to supply the power" for the Board.

→ Arduino Board consists of AREF (Analog Reference pin), Transmitter (TX), Receiver (RX) and RESET pin and Switch.

Steps to program in Arduino IDE:-

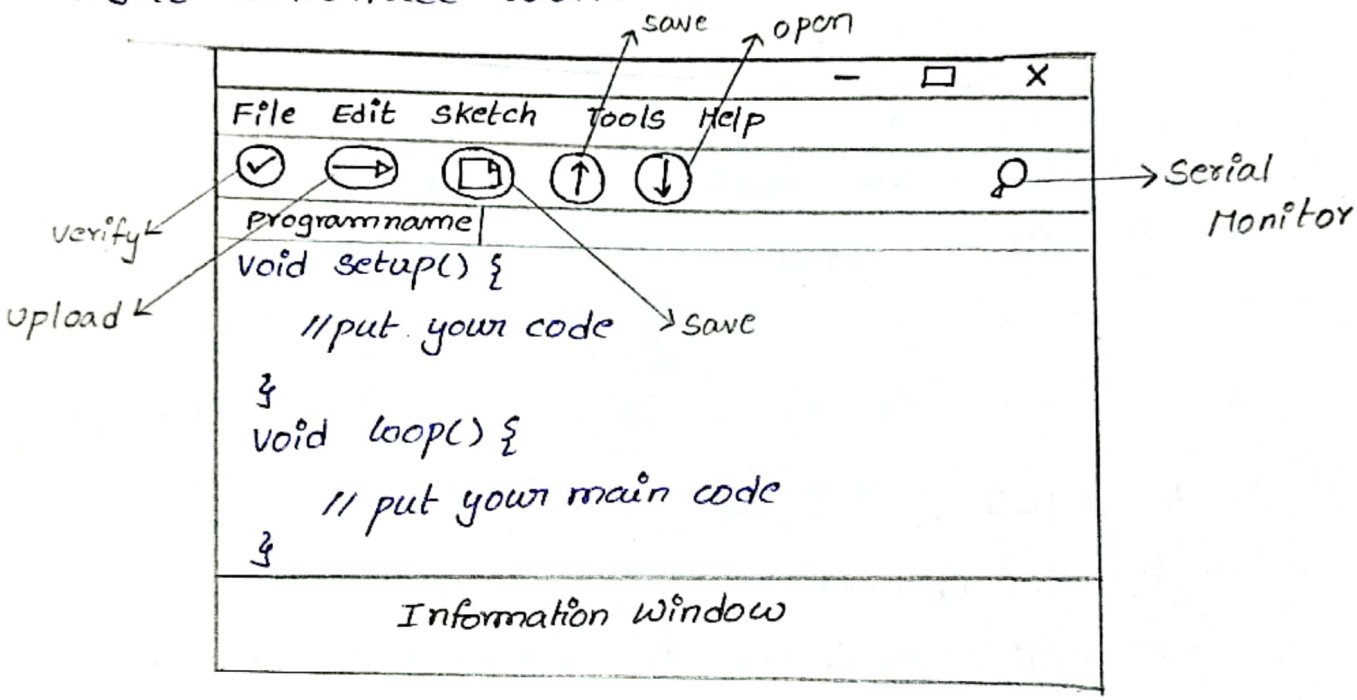
1) Download the Arduino IDE using arduino.cc/en/softwar

2) Connect the USB cable to the Arduino Board.

3) Verify if the device is detected (or) not

4) Now, open the Arduino IDE and select Menu bar in Tools then select "Arduino UNO" for the board next click on assigned CMO port.

→ The Interface looks like:-



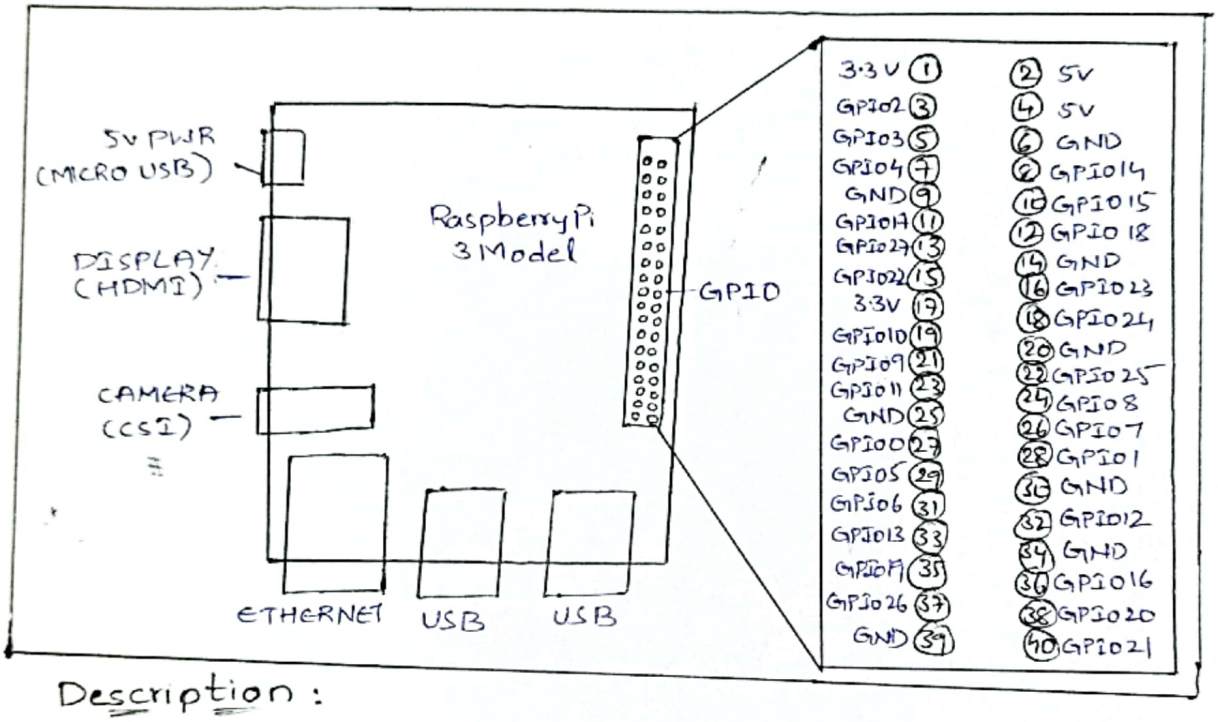
5) Compile the code

6) Upload the code

Raspberry Pi :

* It is the name of the "credit card sized computer board" developed by Raspberry foundation.
* It can be plugged into a monitor & provides fully functioning computer capability.

Pin Diagram of Raspberry Pi :



Pin Description :

A Raspberry pi 3 board has 40 pins on it.

- 40 pins
  - 4 power pins
    - 2 3.3V
    - 2 5V (used to run low power applications)
  - 8 ground pins (all are connected to each other).
  - 28 GPIO pins
    - labelled from GPIO0 to GPIO27.
    - can be used either to set the values of output pins or to read values of input pins.

Features of Raspberry Pi:

* Has quad-core ARM processor.
* Has 1 GB RAM, 1 HDMI Port, 4 USB Ports, 1 micro SD slot for storage, combination of 3.5mm audio/video port, 1 Ethernet connection, 1 Bluetooth connection.

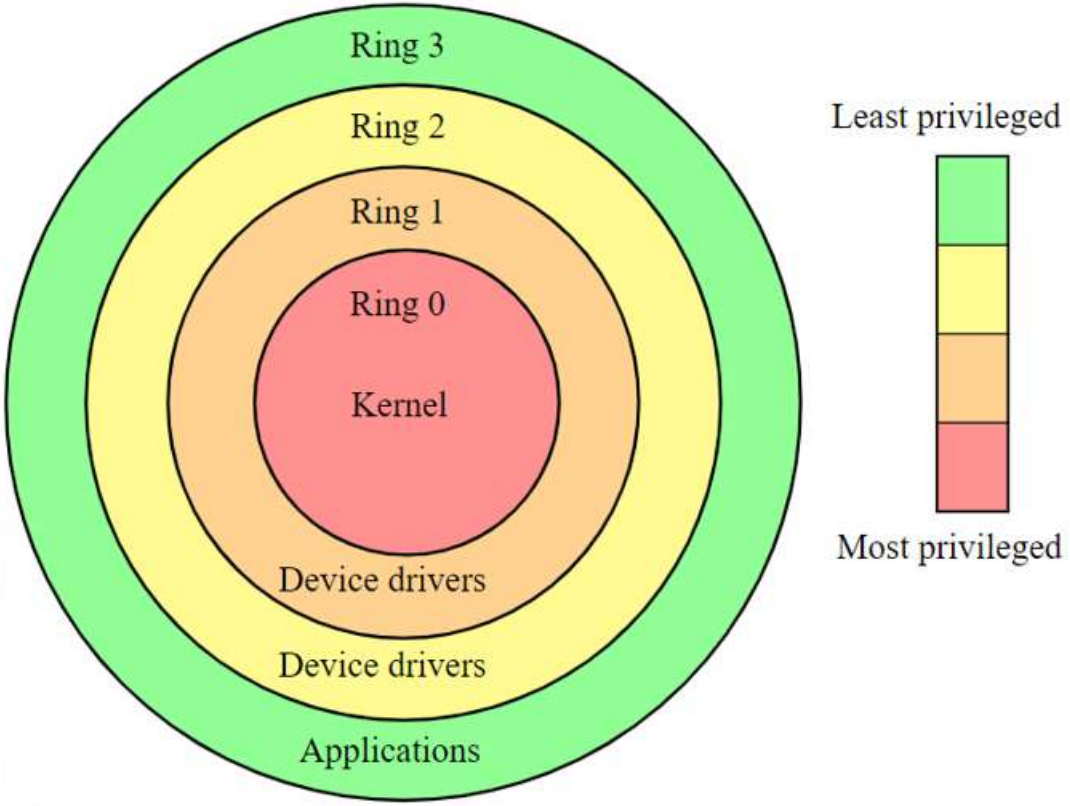
How to use RASPBERRY PI 3:

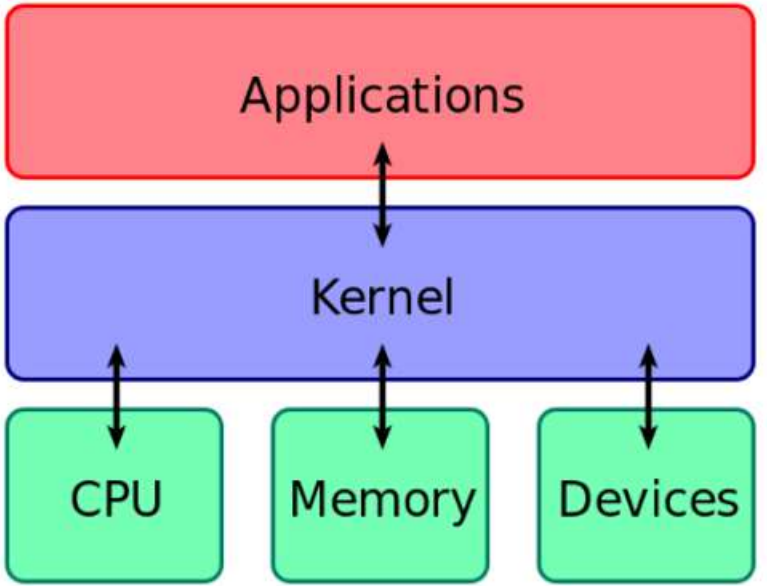
1. Take a 16 GB micro SD card & dedicate it specifically for PI OS.
2. Choose & Download OS software (Eg: Raspbian OS)
3. Format the SD card & install OS on to the SD memory card using convenient methods.
4. Take the SD card after OS installation & insert it in PI board.
5. Connect peripherals (monitor, keyboard & mouse)
6. Power the board with micro USB connector.
7. Once the power is turned ON, the PI will run on the OS installed in the memory card & will start from boot.
8. Once all drivers are checked the PI will ask for authorisation, this is set by default & can be changed.
9. After authorization you will reach desktop where all application program development starts.

Applications of Raspberry Pi:

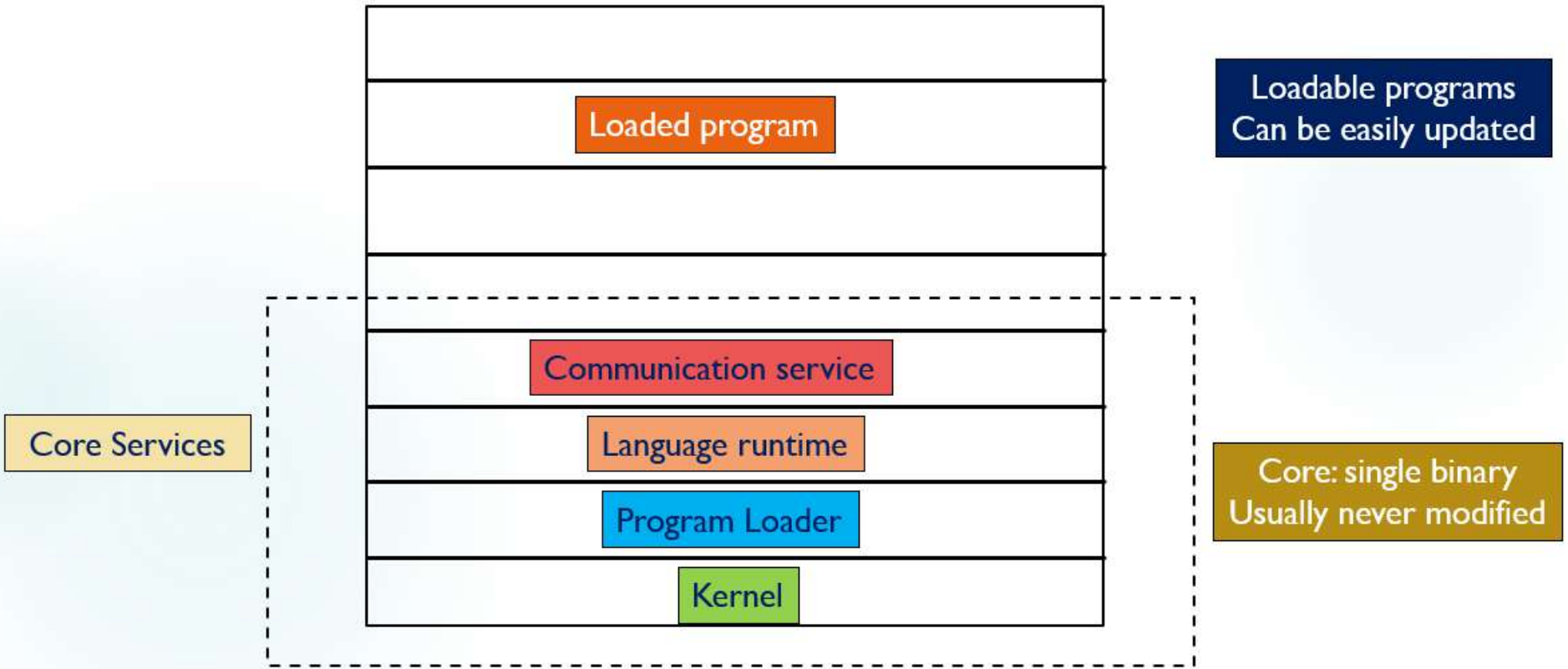
* Desktop PC
* Wireless print server
* Media Usage
* Game Servers
* Retro Gaming Machine
* Robot Controller
* Stop motion Camera

OPERATING SYSTEM OVERVIEW





THE CONTIKI OS



Contiki Operating System.

- Contiki is an operating system for IoT that specifically targets small IoT devices with limited memory, bandwith and processing power.
- It uses a minimalist design while still packing the common tools of modern operating system.
- It is popular for being very lightweight, mature and flexible, that why many researchers and professionals consider it as 'go-to' OS.
- Contiki only requires a few kilobytes to run & within a space of under 30KB, it fits its entire OS.

Features and Functions of Contiki OS:

- Contiki OS comes with a set of capabilities that includes 8051 SOC to ARM-powdered gadgets, ports are on different platforms together with Arduino and Atmel & with a lot of documentation which are device and programmer-friendly.
- It provides functionalities for management of programs, processes, resources, memory and communication.

i) Memory and Process Management: It helps in memory block allocation using 'C' i.e malloc(). The idea & implementation of protothreads have introduced the

retaining the low gadget requirements & reduces the overhead of multithreading.

(ii) Communication Management: Contiki supports both IPv4 & IPv6 which consists of TCP, UDP and HTTP protocols & additionally includes 6LOWPAN.

(iii) Document System Control: The devices that are part of IoT have the luxury of chronic garage, including Flash, Contiki OS affords guide through the "espresso Flash file system".

Contiki Communication Protocols:

Contiki supports standard protocols and recent enabling protocols for IoT:

- **uIP (for IPv4)**: This TCP/IP protocol supports 8 bit and 16 bit microcontrollers
- **uIPv6 (for IPv6)**: This is a fully compliant IPv6 extension to uIP
- **Rime**: It provides a solution when IPv4 or IPv6 prove prohibitive & offers a set of primitives for low-power systems.
- **6LoWPAN**: It provides compression technology to support low data rate wireless devices with limited sources.
- **RPL**: (Routing Protocol for Low power & Lossy Networks): This distance vector IPv6 protocol for LLN's (low power and lossy networks) allows the possible path to be found in a complex network of devices.
- **CoAP**: This protocol supports communication for simple devices, typically devices requiring heavy remote supervision.

Protothreads :

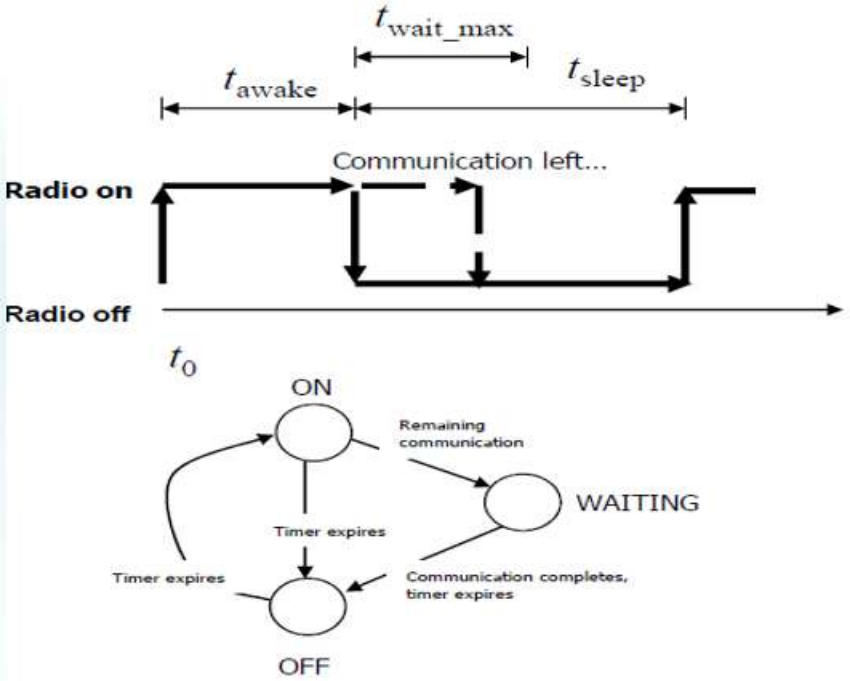
- Protothreads are extremely lightweight stackless threads designed for severely memory constrained system, such as small embedded systems or WSN nodes
- The advantage is that they do not waste memory on multiple stacks that are partially used.
- They provide sequential execution without the need for large state machines or multithreading.
- The significant advantage is that they provide conditional blocking inside a process eventhandler. without a protothread if an event handler didnot return, other processes would not be scheduled.
- It allows block operations within eventhandler.
- As a result, there is less application code, more operations.

PROTOTHREADS- EXAMPLE

- For example, consider a MAC protocol that turns off the radio subsystem on a periodic basis; but you want to make sure that the radio subsystem completes the communication before it goes to sleep state.

1. At $t=t_0$ set the radio ON
2. The radio remains on for a period of t_{awake} seconds
3. Once t_{awake} is over, the radio has to be switched off, but any on-going communication needs to be completed.
4. If there is an on-going communication, the MAC protocol will wait for a period, t_{wait_max} before switching off the radio.
5. If the communication is completed or the maximum wait time is over, then the radio will go off and will remain in the off state for a period of $t_{sleep.}$
6. The process is repeated.

RADIO SLEEP CYCLE CODE WITH EVENTS



Event driven code can be messy and complex

```
enum {ON, WAITING, OFF} state;

void eventhandler() {
  if(state == ON) {
    if(expired(timer)) {
      timer = t_sleep;
      if(!comm_complete()) {
        state = WAITING;
        wait_timer = t_wait_max;
      } else {
        radio_off();
        state = OFF;
      }
    }
  } else if(state == WAITING) {
    if(comm_complete() ||
       expired(wait_timer)) {
      state = OFF;
      radio_off();
    }
  } else if(state == OFF) {
    if(expired(timer)) {
      radio_on();
      state = ON;
      timer = t_awake;
    }
  }
}
```

CONTIKI COOJA SIMULATOR

Overview of Cooja Simulator : Cooja simulator for IoT belongs to the network simulator and it is particularly designed for the wireless sensor network. It permits the networks of Contiki motes for all the simulation processes. It is the Contiki network simulator.

Major Uses in Cooja Simulator for IOT

The following is about the uses of the cooja simulator IOT and it is useful for the research scholars to develop their research in the cooja simulator

- End to End IOT Simulation
- The cooja emulator is used to generate a scenario in which the addition of sensors and the transmission of data from the sensor to cloud services. Then the codes are used for the implementation of sensor nodes

Significant Modules in Cooja Simulator

Let us discuss the modules in the cooja simulator for IOT with their functions as follows

Modules and its Function

- 6LowPAN
    - It is a short form of IPv6 through the low power wireless personal area network. It is explained as the header compression mechanism and encapsulation to permit the IPv6 packets for the transmission of IEEE 802. 15. 4 networks
    - Powertrace
    - It is a prototype version and used for the implementation of Contiki and to attain the low power wireless systems

Vital Classes in Cooja Simulator IOT

Hereby, we have highlighted the major classes of the cooja simulator IOT below in addition the research students may use numerous classes with the support of libraries. We can provide the in-depth analysis and implementation support for the selected class to develop your research

Classes and its Purposes

- Channel Model
    - It is used to measure the impact of propagation in the packets of the radio medium
    - Multipath Ray Tracing Radio Medium
    - It represents the easy radio mediums in Cooja and by using the 2D ray tracing, it authenticates the strength of the signals among simulated radios

Essential Tools in Cooja Simulator IOT

Our research experts have lots of experience with the integrated tools and their functions in the cooja simulator IOT. You can contact us for the implementation process. Here, the tools are highlighted below

Tools and its Uses

- MSPSim and Cooja
    - In general, MSPSim is based on the Java level emulator and used for the emulation process in the sensor network areas

System Specifications in Cooja Simulator IOT

For your reference, we have listed down the important programming languages used in the cooja simulator for IOT

Programming Languages in Cooja Simulator IOT

- Java
- C

The notable operating systems in the cooja simulator for IOT are listed below. And we extend our support for your requirements in research

OS Support in Cooja Simulator IOT

- Contiki-3.x
- Ubuntu-16.04

The versions in the cooja simulator IOT are useful for the research scholars to update their knowledge. The versions such as

Versions in Cooja Simulator IOT

- Contiki OS

Substantial Protocols in Cooja Simulator for IOT

Our research professionals are well versed in all the protocols and we provide support for your selected research protocols. Here, we have listed the significant protocols in the cooja simulator IOT

- Very Simple Control Protocol
- It is a mechanical type which is apt for all the works such as building, etc. The major benefit for this protocol is the utilization of VSCP nodes
- OASIS Message Queuing Telemetry Transport
- OASIS is used to generate the stable stage for the MQTT which contains the practices, documented usage, and guidelines of the MQTT topics

FEATURES OF COOJA STIMULATOR

1) MEMORY ALLOCATION:

Contiki is designed for tiny systems, having only a few kilobytes of memory available. It is highly memory efficient. It also provides a set of mechanisms for memory allocation: memory block allocation memb, a managed memory mmem, as well as the standard C memory allocator malloc.

2) FULL IP NETWORKING

Contiki provides a full IP network stack, with standard IP protocols such as UDP, TCP and HTTP, in addition to the new low-power standards like 6lowpan, RPL and CoAP.

3) POWER AWARENESS

Contiki is designed to operate in extremely low-power systems: which may run for years on a pair of AA batteries. Contiki provides estimation for system power consumption and for understanding where the power was spent.

4) LOWPAN, RPL, CoAP

Contiki supports the recently standardized IETF protocols for low-power IVP6, networking, including the 6lowpan adaption layer, and the CoAP RESTful application-layer protocol.

5) DYNAMIC MODULE LOADING

Contiki supports dynamic loading and linking of modules at run-time. The contiki module loader

can load, relocate and link standard ELF files that can optionally be stripped off their debugging symbols to keep their size down.

6) SLEEPY ROUTERS

In wireless networks, nodes may need to relay messages from others to reach their destination. With Contiki, even relay nodes, so-called routers, can be battery-operated thanks to the Contiki MAC radio duty cycling mechanism which allows them to sleep between each relayed message.

7) HARDWARE PLATFORMS

Contiki runs on a wide range of tiny platforms, ranging from 8051-powered systems-on-a-chip through the MSP430 and the AVR to a variety of ARM devices.

8) PROTO-THREADS

To save memory but provide a nice control flow in the code, Contiki uses a mechanism called proto-threads. Proto-threads are a mixture of the event-driven and the multi-threaded programming mechanisms.

9) COFFEE FLASH FILE SYSTEM

For devices that have an external flash memory chip, Contiki provides a light weight flash file system, called Coffee. With Coffee, application programs can open, close, read from, write to, and append to files on the external flash. The performance of Coffee is within 95% of the raw throughput of the flash memory.

10) THE CONTIKI SHELL

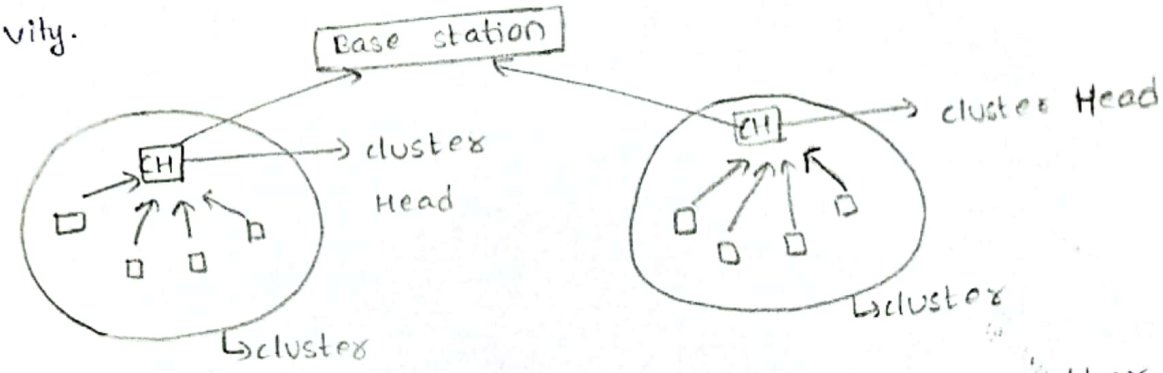
Contiki provides an optional command-line shell with a set of commands that are useful during development and debugging of contiki systems. Applications can define their own shell commands that work together with existing commands.

Clustering:

cluster Analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups.

→ Each cluster Has a cluster Head (CH) which takes care of routing of the traffic originated from the entire network.

→ clustering in WSN and Internet of Things aims at energy consumption, load balancing, reduced packet delays and increased connectivity.



→ cluster head is responsible for collecting data from the other nodes that belongs to that cluster

→ cluster heads send the aggregated data to Base station

→ cluster head selection plays significant role for energy efficiency of clustering algorithms.

→ cluster with high intra-cluster (Distance among members (or) nodes of a cluster) communication distance will consume more energy than other clusters

→ A node is selected as a cluster head which is present in collaborative and distributive mode but not in centralised mode.

centralised mode - A single node acts as cluster head for longer period of time. which leads to loss of energy.

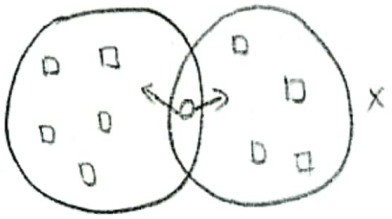
Clustering Principles

1. Active Networking

Here, multiple server nodes are running simultaneously which are working together to publish and subscribe Message.

SRC — O — O — Des. → also Referred to as multi-hop communication because packet travels from source to dest. by using several nodes.

2. Hierarchial non-overlapping clusters of sensor nodes



A single cannot be overlapped (present) is two cluster. So, A single node must point only to a single cluster

3. Not centralised mode

A single node must not be a cluster head for longer period.

4. Nodes must be present in multi-hop communication

5. There must be a dynamic interactions between objects which are connected wirelessly

6. Nodes must consist of co-operative operations and events triggering that are present inside a network

Need for Clustering

clustering is an unsupervised machine learning method of identifying and grouping similar data points into a number of groups. It will be used in market Research, pattern Recognition, data Analysis and image processing so, it helps in increasing productivity, facilitate decision-making, and generate new bussiness Opportunities.

LEACH:

→ LEACH stands for Low Energy Adaptive Clustering Hierarchy.

→ A clustering-based protocol that aims to minimize energy consumption in sensor networks

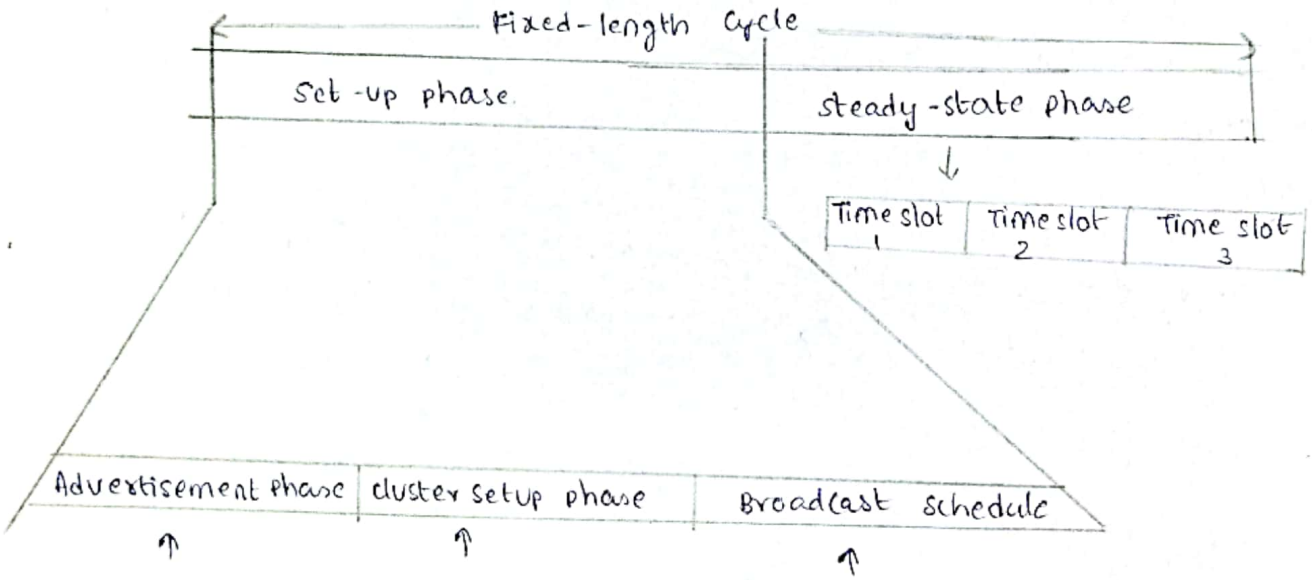
→ LEACH often perform local computation in each cluster inorder to reduce the amount of data that must transmitted to the Base station.

→ LEACH uses CDMA (Code Division multiple access), TDMA (Time Division Multiple Access) MAC to reduce inter-cluster and intra-cluster collisions.

→ LEACH Protocol includes 2-phases:

(1) Set-up phase

(2) Steady phase



- Self election of cluster heads
- cluster heads compete with CSMA

- Members compete with CSMA

- cluster head Broadcast CDMA code to members

(i) Advertisement phase

Once a node is elected as a cluster head. It sends a advertisement packet to inform the cluster nodes that it have become a cluster head. Once a node becomes cluster head it cannot become cluster head again until all the nodes in the cluster have become cluster head once.

(ii) cluster set-up

Here, non-cluster head nodes receive the cluster-head advertisement and then send join request to cluster head informing that they are members of the cluster under cluster head.

(iii) Broadcast Schedule

cluster heads creates a transmission schedule for the member nodes of their cluster. TDMA schedule is created according to the number of nodes in the cluster. Each node then transmits data in the allocated time schedule.

Steady-state phase

In steady state phase, cluster heads will deplete more energy than member nodes because they have to take the Responsibility of aggregating and relaying data to remote BS(Base station) for their member nodes.

Interfacing LED and blinking with different delays

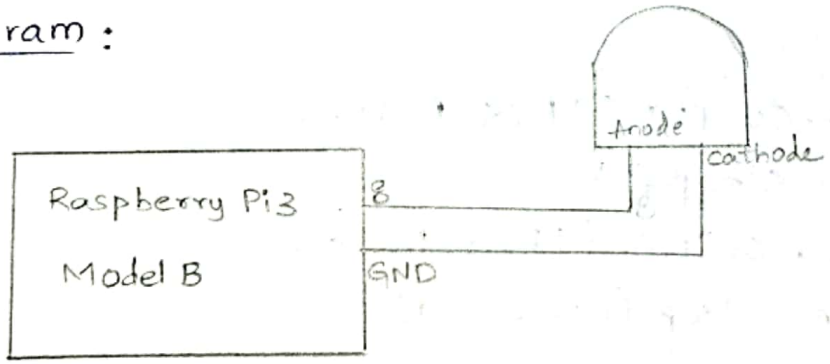
Procedure

1. connect the anode of LED to board pin no 8
2. connect cathode of LED to ground
3. open python terminal in the Raspbian OS and develop a python based API to access the Pin and control the on/off state of LED.

Python code for interfacing LED with Raspberry Pi :-

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT, initial = GPIO.LOW)
while True :
GPIO.OUTPUT(8, GPIO.HIGH)
sleep(1)
GPIO.OUTPUT(8, GPIO.LOW)
sleep(1)
```

Block diagram :



Interfacing Ultrasonic Sensor with RaspberryPi

Procedure:

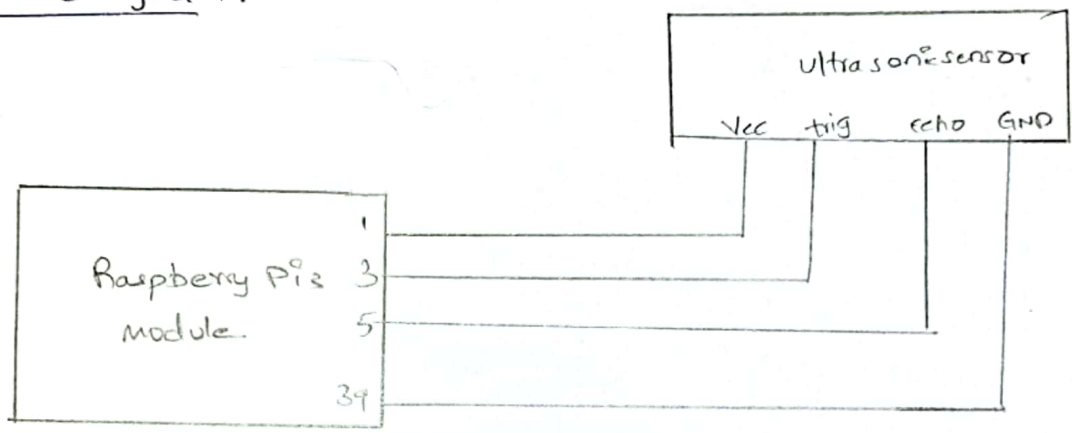
1. Connect the Trigger pin to board pin 3 and echo pin to board Pin 5
2. Connect the cathode of LED to ground.
3. Open python terminal in the Raspbian os and develop a python based API to access and configure the GPIO pins according to the working principle of Ultrasonic sensor.

Python code for interfacing HC→R04 with Raspbery Pi:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
TRIG=3
ECHO=5
print("Distance measurment in process...")
 GPIO.setup(TRIG, GPIO.OUT)
 GPIO.setup(ECHO, GPIO.IN)
 while(1):
        GPIO.OUTPUT(TRIG, False)
         time.sleep(2)
         GPIO.output(TRIG, True)
         time.sleep(0.00001)
         GPIO.output(TRIG, False)
         while GPIO.input(ECHO)==0:
                P_start = time.time()
         while GPIO.input(ECHO)==1:
                P_end = time.time()
```

```
pulse_duration = P_end - P_start
distance = pulse_duration * 17150
distance = round(distance,2)
print("distance :" +str(distance) +" cm")
```

Block diagram:



How to send data to Thingspeak cloud using Raspberry pi:

* Thingspeak is a cloud server to store the data
* Here, Raspberry pi will read its cpu temp and send it to thingspeak, and it can be monitored from anywhere in the world using internet
* This will be useful for running the pile for long time for some application at some remote place and need monitors with cpu temp.

Thingspeak:

* It is an open IOT platform for monitoring your data online
* In thingspeak channel you can set the data as private (or) public according to your choice.
* Thingspeak takes minimum of 15 sec to update your readings. it is great and very easy to use platform for building IOT projects.

Components required:-

1. Raspberry Pi
2. power cable
3. wifi (or) internet.

Steps for building Raspberry Pi data logger on cloud

1. Signup your Thingspeak

→ for creating a channel on Thingspeak your first need to signup on thingspeak

→ In case if you already have an account just sign in using your id & password

→ for creating account go to www.Thingspeak.com.

→ click on signup and if you don't have account if you already have an account click on sign in.

→ after clicking on sign up fill your details

→ After this verify our email id and click on continue

2. create a channel for your data

→ once you sign in your account create a new channel by clicking new channel button

→ fill the details, after this click on same channel button to save your details.

3. getting API Key in thingspeak.

→ for send data to thingspeak we need an unique API key which we will use later in our, python code to upload our cpu data to thingspeak. website

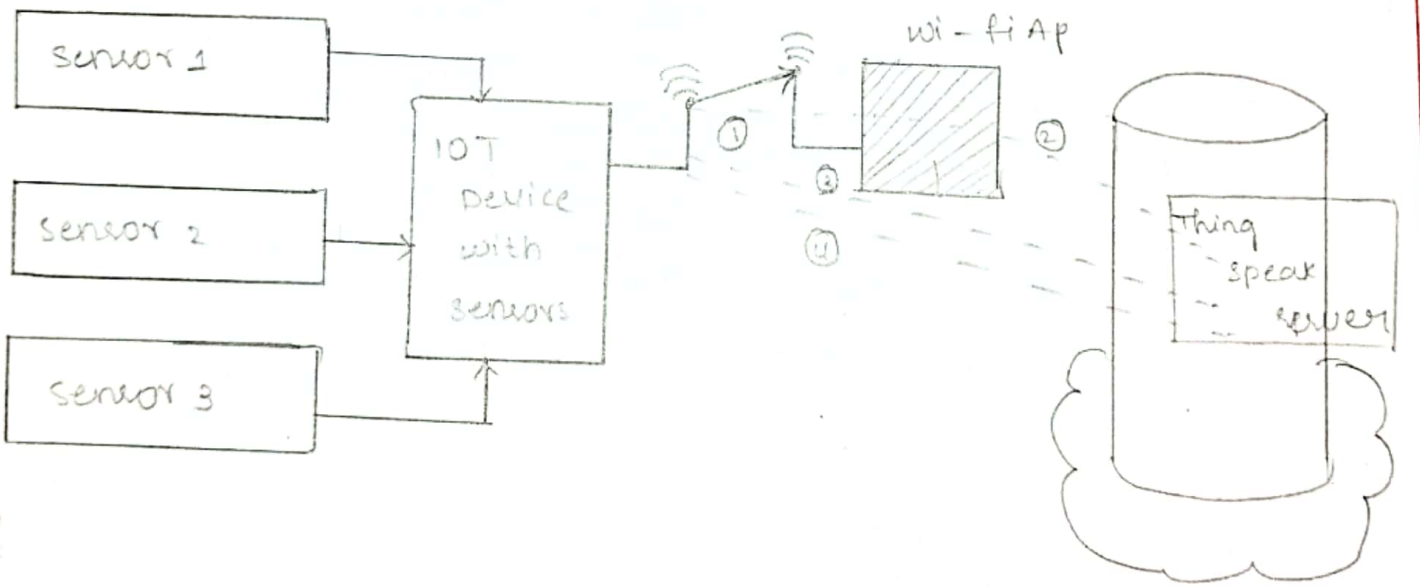
→ Click on API keys buttons to get your unique API key for uploading your cpu data

→ Now copy your "write API key" we will use this API key in our code.

4) Paste the API key ID in the variable API-key in the Python program

5) Now Run the Python program so that we can able see the o/p graph on screen of thing speak.

Setting Thingspeak cloud enviironment :-



Process:

1) connent the devices to a wi-fi Access point

2) connect the thinkspeak server using http post

3. establish a TCP connection between Thingspeak server and devices.

4. create a URL that encapsulates the sensor data and sending over the TCP link.

5. Repeat step 4 periodically.