IOT LAB MANUAL

EXPERIMENT NO: 1
1. Using raspberry pi
a. Calculate the distance using a distance sensor.
b. Basic LED functionality.

AIM:
Using raspberry pi
a. Calculate the distance using a distance sensor.
b. Basic LED functionality.

PROGRAM:

A. Use a Raspberry Pi to calculate the distance using a distance sensor:

- Connect the distance sensor to the Raspberry Pi's GPIO pins.
- Install the required libraries and packages (e.g., RPi.GPIO, pigpio, time, etc.).
- Write a script in Python to read the sensor data and calculate the distance using a formula based on the sensor's specifications.
- Use the formula to convert the sensor data into a distance value and display the output on the screen or send it to a remote server for storage and processing.
- Run the script on the Raspberry Pi to get the distance measurements.

Here's an example code that demonstrates this process:

```
import RPi.GPIO as GPIO
import time
# Set the GPIO mode
GPIO.setmode(GPIO.BCM)
# Define the GPIO pin for the sensor
TRIG = 23
ECHO = 24
# Set up the GPIO pins for the sensor
GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
# Set the trigger pin to low
GPIO.output(TRIG, False)
# Wait for the sensor to settle
```

```
time.sleep(2)
# Send a 10uS pulse to trigger the sensor
GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)
# Measure the pulse duration
while GPIO.input(ECHO)==0:
    pulse_start = time.time()
while GPIO.input(ECHO)==1:
    pulse_end = time.time()
# Calculate the distance
pulse_duration = pulse_end - pulse_start
distance = pulse_duration * 17150
distance = round(distance, 2)
# Display the distance
print("Distance:",distance,"cm")
# Clean up the GPIO pins
GPIO.cleanup()
```

OUTPUT:

B. Use a Raspberry Pi to implement basic LED functionality:

- Connect the LED to the Raspberry Pi's GPIO pins, with a resistor in series to protect the LED from excessive current.
- Install the required libraries and packages (e.g. RPi.GPIO, time, etc.).
- Write a script in Python to control the GPIO pin connected to the LED and turn it on and off.
- Run the script on the Raspberry Pi to control the LED.

Here's an example code that demonstrates this process:

```
import RPi.GPIO as GPIO
import time
# Set the GPIO mode
GPIO.setmode(GPIO.BCM)
# Define the GPIO pin for the LED
LED = 18
# Set up the GPIO pin for the LED
GPIO.setup(LED, GPIO.OUT)
# Turn on the LED
GPIO.output(LED, True)
time.sleep(1)
# Turn off the LED
GPIO.output(LED, False)
# Clean up the GPIO pins
GPIO.cleanup()
```

This code sets up the GPIO pin 18 as an output and turns the LED on for 1 second and then off. You can modify the code to implement different LED functionality, such as blinking, fading, etc.

OUTPUT:

VIVA QUESTIONS

- What is a raspberry pi?

- What is a LED functionality?

EXPERIMENT NO: 2
2. Using Arduino
a. Calculate the distance using a distance sensor.
b. Basic LED functionality.
c. Calculate temperature using a temperature sensor.

AIM:
Using Arduino
a. Calculate the distance using a distance sensor.
b. Basic LED functionality.
c. Calculate temperature using a temperature sensor.

PROGRAM:

A. Use Arduino to Calculate the distance using a distance sensor:

- Connect the distance sensor to the Arduino's digital pins.
- Install the required libraries and packages (e.g. NewPing library).
- Write a sketch in the Arduino IDE to read the sensor data and calculate the distance using the NewPing library.
- Use the library to convert the sensor data into a distance value and display the output on the serial monitor or send it to a remote server for storage and processing.
- Upload the sketch to the Arduino to get the distance measurements.

Here's an example code that demonstrates this process:

```
#include <NewPing.h>
#define TRIGGER_PIN  12
#define ECHO_PIN     11
#define MAX_DISTANCE 200
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
void setup() {
  Serial.begin(9600);
}
void loop() {
  delay(50);
  int distance = sonar.ping_cm();
  Serial.print("Distance: ");
  Serial.print(distance);
```

```
  Serial.println(" cm");
}
```

This code uses the NewPing library to calculate the distance using the sensor connected to digital pins 12 and 11. The distance is displayed on the serial monitor and updated every 50 milliseconds.

OUTPUT:

B. Use Arduino to implement basic LED functionality:

Connect the LED to the Arduino's digital pins, with a resistor in series to protect the LED from excessive current.

Write a sketch in the Arduino IDE to control the digital pin connected to the LED and turn it on and off.

Upload the sketch to the Arduino to control the LED.

Here's an example code that demonstrates this process:

```
const int LED = 13;
void setup() {
  pinMode(LED, OUTPUT);
}
void loop() {
  digitalWrite(LED, HIGH);
  delay(1000);
  digitalWrite(LED, LOW);
  delay(1000);
}
```

This code sets up digital pin 13 as an output and turns the LED on and off every 1 second. You can modify the code to implement different LED functionality, such as blinking, fading, etc.

OUTPUT:

C. Use Arduino to Calculate temperature using a temperature sensor:

- Connect the temperature sensor (e.g. LM35) to the Arduino's analog pins.
- Write a sketch in the Arduino IDE to read the sensor data and convert it into a temperature value.
- Display the temperature output on the serial monitor or send it to a remote server for storage and processing.
- Upload the sketch to the Arduino to get the temperature readings.

Here's an example code that demonstrates this process:

```
const int TEMP_SENSOR = A0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  float voltage = analogRead(TEMP_SENSOR) * 0.0048828125;
  float temperature = voltage * 100;
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println("°C");
  delay(1000);
}
```

This code reads the analog voltage from the temperature sensor connected to analog pin A0 and converts it into a temperature value in degrees Celsius. The temperature is displayed on the serial monitor and updated every 1 second.

OUTPUT:

VIVA QUESTIONS

Define Arduino

What is temperature sensor

EXPERIMENT NO: 3

3. Using Node MCU

a. Calculate the distance using a distance sensor.

b. Basic LED functionality.

c. Calculate temperature using a temperature sensor.

AIM:

Using Node MCU

a. Calculate the distance using a distance sensor.

b. Basic LED functionality.

c. Calculate temperature using a temperature sensor.

PROGRAMS:

A. Using Node MCU to Calculate the distance using a distance sensor:

To calculate the distance using a distance sensor in IoT with NodeMCU, you would need to connect the distance sensor to the NodeMCU board. Then you would need to write a program in the Lua programming language to read the distance from the sensor and send it to a remote server.

Here is an example of how this can be done:

- Connect the distance sensor to the NodeMCU board.
- Write a program in Lua to read the distance from the sensor and store it in a variable.
- Use the NodeMCU's Wi-Fi capability to send the distance data to a remote server.
- On the remote server, process the data and display the distance on a webpage.

Here is an example code in Lua to read the distance from a HC-SR04 distance sensor:

```
-- Trigger pin will be used to send a signal to the sensor to measure distance
local trigger_pin = 4
-- Echo pin will be used to receive the signal back from the sensor
local echo_pin = 3
-- Set trigger_pin as an output pin
gpio.mode(trigger_pin, gpio.OUTPUT)
-- Set echo_pin as an input pin
```

```
gpio.mode(echo_pin, gpio.INPUT)
-- Function to calculate distance
function get_distance()
  -- Send a 10us pulse to trigger the measurement
  gpio.write(trigger_pin, gpio.HIGH)
  tmr.delay(10)
  gpio.write(trigger_pin, gpio.LOW)
  -- Measure the duration of the pulse received back on the echo pin
  local duration = pulse_in(echo_pin, gpio.HIGH)
  -- Calculate the distance based on the duration
  local distance = duration / 58

  -- Return the calculated distance
  return distance
end
-- Call the get_distance function and print the result
print("Distance: " .. get_distance() .. " cm")
```

This is just a basic example and you may need to modify the code based on your specific requirements. You can also use a library such as the HC-SR04 library in NodeMCU to simplify the process.

OUTPUT:

B. Using Node MCU to Basic LED functionality in IOT with output:

To control a basic LED in IoT with NodeMCU, you can use the following steps:

- Connect the LED to the NodeMCU board.
- Write a program in Lua to control the state of the LED (on or off).
- Use the NodeMCU's Wi-Fi capability to send commands to the LED to turn it on or off.
- On a remote device, such as a smartphone or a computer, you can create a user interface to send the commands to the NodeMCU.

Here is an example code in Lua to control an LED connected to pin D1 on the NodeMCU board:

```
-- LED pin
local led_pin = 1
-- Set the LED pin as an output pin
gpio.mode(led_pin, gpio.OUTPUT)
-- Function to turn the LED on
function turn_on_led()
  gpio.write(led_pin, gpio.HIGH)
end
-- Function to turn the LED off
function turn_off_led()
  gpio.write(led_pin, gpio.LOW)
end
-- Call the turn_on_led function to turn the LED on
turn_on_led()
-- Wait for 2 seconds
tmr.delay(2000000)
-- Call the turn_off_led function to turn the LED off
turn_off_led()
```

This is just a basic example and you can modify it to fit your specific requirements. You can also use the NodeMCU API to create a web server that allows you to control the LED through a web browser.

OUTPUT:

C. Using Node MCU to Calculate temperature using a temperature sensor:

To calculate temperature using a temperature sensor in IoT with NodeMCU, you would need to connect the temperature sensor to the NodeMCU board. Then you would need to write a program in the Lua programming language to read the temperature from the sensor and send it to a remote server. Here is an example of how this can be done:

Connect the temperature sensor to the NodeMCU board.
Write a program in Lua to read the temperature from the sensor and store it in a variable.
Use the NodeMCU's Wi-Fi capability to send the temperature data to a remote server.
On the remote server, process the data and display the temperature on a webpage.

Here is an example code in Lua to read the temperature from a DS18B20 temperature sensor:

```
-- Pin that the DS18B20 is connected to
local pin = 2
-- Initialize the DS18B20 library
ds18b20.setup(pin)
-- Function to read the temperature
function get_temperature()
  -- Read the temperature from the DS18B20
  local temperature = ds18b20.read()
    -- Return the temperature
  return temperature
end
-- Call the get_temperature function and print the result
print("Temperature: " .. get_temperature() .. "°C")
```

This is just a basic example and you may need to modify the code based on your specific requirements. You can also use a library such as the DS18B20 library in NodeMCU to simplify the process.

OUTPUT:

BEYOND THE SYLLABUS

EXPERIMENT NO: 1. Using Micro Python Calculate the distance using a distance sensor

AIM: Using Micro Python Calculate the distance using a distance sensor

DESCRIPTION:

In this code, we first import the required libraries (time, machine, and utime). Then, we define the trigger and echo pins of the distance sensor using machine. Pin and set their modes to output (for the trigger) and input (for the echo).

The measure distance function uses the sensor to measure the distance. It first sends a trigger pulse to the sensor, then waits for the echo to be received. The time taken for the echo to be received is then used to calculate the distance.

Finally, the code enters an infinite loop that continuously calls the measure distance function, displays the distance on the output display, and waits for half a second before measuring the distance again.

PROGRAM:

```
import time
import machine
import utime
# Define the trigger and echo pins of the distance sensor
trig_pin = machine.Pin(4, machine.Pin.OUT)
echo_pin = machine.Pin(5, machine.Pin.IN)
# Function to measure the distance using the sensor
def measure_distance():
  trig_pin.value(0)
  utime.sleep_us(2)
  trig_pin.value(1)
  utime.sleep_us(10)
  trig_pin.value(0)
    while echo_pin.value() == 0:
```

```
    pass
  start = utime.ticks_us()
    while echo_pin.value() == 1:
    pass
  end = utime.ticks_us()
    # Calculate the distance
  duration = end - start
  distance = (duration / 2) / 29.1
    return distance
# Main loop to continuously measure and display the distance
while True:
  distance = measure_distance()
  print("Distance: {:.2f} cm".format(distance))
  time.sleep(0.5)
```

OUTPUT:

EXPERIMENT NO: 2. Basic LED functionality for using micro python

AIM: Basic LED functionality for using micro python

DESCRIPTION:

In this code, we first import the required libraries (machine and time). Then, we define the pin the LED is connected to using machine.Pin and set its mode to output.

The code then enters an infinite loop that alternates between turning the LED on (by setting its pin value to 1) and displaying "LED On", and turning the LED off (by setting its pin value to 0) and displaying "LED Off". The time.sleep function is used to control the delay between turning the LED on and off.

PROGRAM:

```
import machine
import time

# Define the pin the LED is connected to
led_pin = machine.Pin(2, machine.Pin.OUT)

# Main loop to turn the LED on and off
while True:
  led_pin.value(1)
  print("LED On")
  time.sleep(1)
  led_pin.value(0)
  print("LED Off")
  time.sleep(1)
```

OUTPUT: