**CODE:**

**Interfacing of Temperature Sensor (LM35):**

#include<LiquidCrystal.h>

LiquidCrystal led(7,6,5,4,3,2);

void setup()

{

Serial.begin(19200);

led.begin(16,2);

}

void loop()

{

int analogvalue = analogRead(A3);

int pecent = map(analogvalue, 0, 1023,0,600);

Serial.print("analogvalue=");

Serial.println(analogvalue);

Icd.setCursor(0,0);

lcd.print(analogvalue);

Serial.print("Pecent=");

Serial.println(pecent);

Icd.setCursor(0,1);

Icd.print(pecent);

delay(1000);

Icd.clear();

}

**Interfacing of IR Sensor:**

int ledPin-13;

int inputPin=2;

int val=0;

void setup()

{

pinMode(13,OUTPUT);

pinMode( inputPin, INPUT);

Serial.begin(9600);

}

void loop()

{

val=digitalRead(inputPin); // check the pin status (High=1/Low=0) //Active Low output

if(val==HIGH)

{

Serial.print("Object Absentin");

digitalWrite(13,LOW)

}

else

{

Serial.print("Object Present'n");

digitalWrite(13,HIGH):

}

}

**Interfacing of Ultrasonic Sensor:**

#include <LiquidCrystal.h>

LiquidCrystal Icd (12,11,5,4,3,2);

// defining the pins

const int trigPin = 10;

const int echoPin = 9;

// defining variables

long duration;

int distance;

void setup() {

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

Serial.begin(9600); // Starts the serial communication

lcd.begin(16,2);

}

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);

//Calculating the distance

distance duration 0.034/2;

lcd.setCursor (0,0);

lcd.print("Distance: ");

delay(1000);

lcd.setCursor (0,1);

lcd.print(distance);

lcd.print("cm");

}

**#Raspberry Pi Libraries**

import RPI.GPIO as GPIO #GPIO library

import time #library for sleep

import board

import digitalio

import adafruit\_character\_lcd.character\_lcd as characterlcd

#set mode as BCM

GPIO.setmode(GPIO.BCM)

#Modify this if you have a different sized character LCD

lcd\_columns = 16

lcd\_rows = 2

#Raspberry Pi Pin Config:

lcd\_rs=digitalio.DigitalInOut(board.D5)

lcd\_en=digitaho Digitalin (board.D6)

lcd\_d4=digitalio DigitalinOut(board D12)

lcd\_d5=digitalio DigitalInOut(board.D13)

led d6=digitalio. Digital InOut(board D16)

led d7=digitalio Digitalin Outboard D17)

#Initialise the lcd class

lcd =characterlcd.Character\_LCD\_Mono{

lcd\_rs, led\_en, lcd\_d4, led\_d5, led\_d6, lcd\_d7, lcd\_columns, Icd\_rows)

#set pins

IR\_OUT=21

BUZ=22

#setup pins at output

GPIO.setup(IR\_OUT, GPIO.IN)

GPIO.setup(BUZ, GPIO.OUT)

def destroy():

GPIO.output (BUZ, GPIO.LOW)

GPIO.cleanup()

if\_name\_main

try:

while True:

IR\_State GPIO.input(IR\_OUT)

if (IR State True):

print ("OBJECT DETECTED")

lcd.clear()

lcd.message "OBJECT DETECTED"

GPIO.output (BUZ, GPIO.HIGH)

time.sleep(0.5)

GPIO.output (BUZ, GPIO.LOW)

else:

lcd.clear()

lcd.message="NO OBJECT"

time.sleep(0.5)

print ("NO OBJECT")

except KeyboardInterrupt:

destroy()