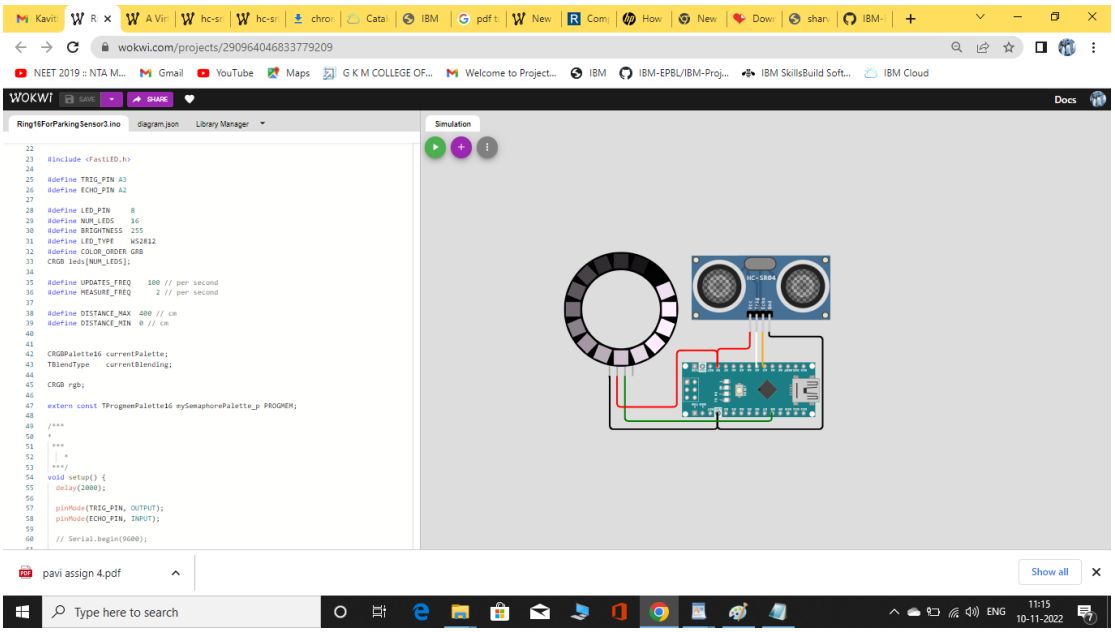
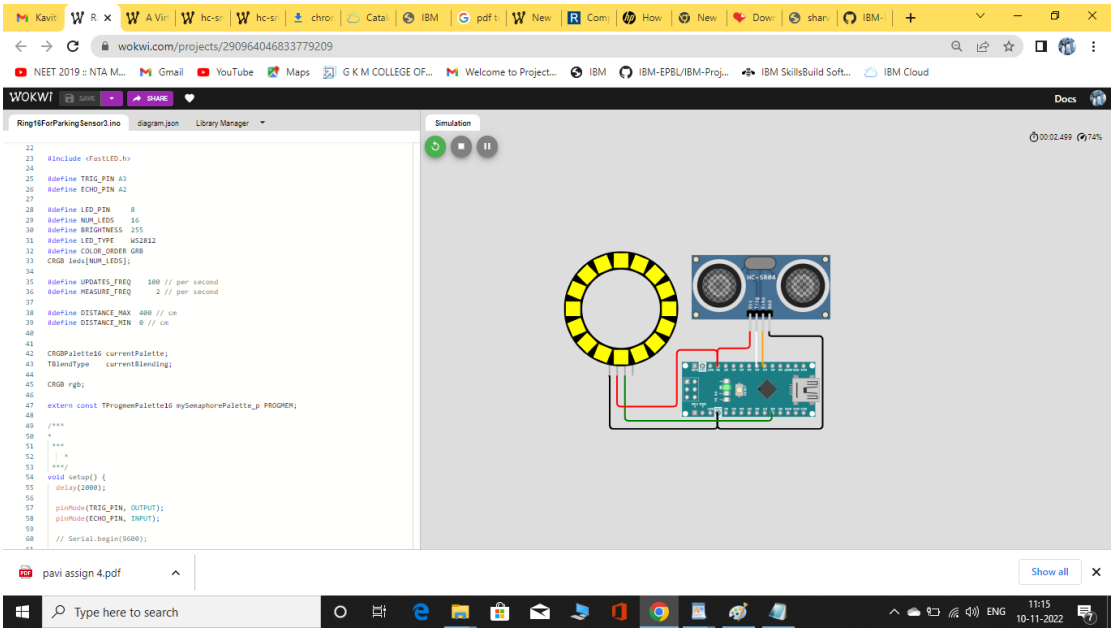
ASSIGNMENT 4

AssignmentDate	10 november2022
StudentName	KAVITHA .D
StudentRollNumber	410819106003

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100 cms send "alert" to ibm cloud and display in device recent events






```
#include <FastLED.h>

#define TRIG_PIN A3
#define ECHO_PIN A2

#define LED_PIN     8
#define NUM_LEDS    16
#define BRIGHTNESS  255
#define LED_TYPE    WS2812
#define COLOR_ORDER GRB
CRGB leds[NUM_LEDS];

#define UPDATES_FREQ    100 // per second
#define MEASURE_FREQ      2 // per second

#define DISTANCE_MAX  400 // cm
#define DISTANCE_MIN  0 // cm


CRGBPalette16 currentPalette;
TBlendType    currentBlending;

CRGB rgb;

extern const TProgmemPalette16 mySemaphorePalette_p PROGMEM;

/***
*
 ***
```

```
   *
 ***/
void setup() {
  delay(2000);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // Serial.begin(9600);

  FastLED.addLeds<LED_TYPE, LED_PIN, COLOR_ORDER>(leds, NUM_LEDS); // .setCorrection(
TypicalLEDStrip );
  FastLED.setBrightness( BRIGHTNESS );

  currentPalette = mySemaphorePalette_p;
  currentBlending = LINEARBLEND;

  // blank
  FillColorDelay( CRGB::Yellow, 1000);
  // Serial.println(" [BEGIN] ");

  // blank
  FillColorDelay( CRGB::Black, 1000);
}

/*
 *
 *
 *
 ****/
void loop() {
  static unsigned int distance = 10;
  static uint8_t colorIndex;
  static byte offset = 0;
  static unsigned int offset_delay = 100;         // rotation speed

  static unsigned long lastOffsetTime = 0;        // last rotation change
  static unsigned long lastMeasureTime = 0;       // last use of HC-SR04 sensor
  static unsigned long lastPrintTime = 0;         // last Serial.print time

  // distance is 200-1500
  // ROTATION SPEED should change 20-1 rotation per second
  offset_delay = map( distance, 20, 400, 20, 100);
  if ( millis() - lastOffsetTime >= offset_delay ) {
    offset = (offset < NUM_LEDS - 1 ? offset + 1 : 0); // increment position
    lastOffsetTime = millis();
  }

  // Measure two times per second
  if (millis() - lastMeasureTime >= 1000 / MEASURE_FREQ) {
    distance = get_distance();
    colorIndex = map(distance, DISTANCE_MIN, DISTANCE_MAX, 0, 240);
    // update palette
    newGradientPaletteByDistance(distance);
```

```
    lastMeasureTime = millis();
  }

  if ( distance > 30 ) {
    FillLEDsFromPalette( offset );
  } else if ( offset % 2 == 0 ) {
    FillColorDelay(CRGB::Blue,200);
    // fill_solid( leds, NUM_LEDS, CRGB::Blue);
  } else {
    FillColorDelay(CRGB::Red,200);
    // fill_solid( leds, NUM_LEDS, CRGB::Red);
  }

  // Print only for debugging
  // if (millis() - lastPrintTime >= 2000) {
  //   Serial.print(" [ distance: ");           Serial.print(distance, DEC);
  //   Serial.print("mm   o_delay: ");          Serial.print(offset_delay, DEC);
  //   Serial.print("    index: ");             Serial.print(colorIndex, DEC);
  //   Serial.println(" ] ");
  //   lastPrintTime = millis();
  // }

  FastLED.show();
  FastLED.delay(1000 / UPDATES_FREQ);
}


// Get color mapped by distance and create gradient palette based by this color
void newGradientPaletteByDistance(int distance) {
  uint8_t xyz[12];  // Needs to be 4 times however many colors are being used.
  // 3 colors = 12, 4 colors = 16, etc.
  CRGB rgb;

  rgb = ColorFromPalette( mySemaphorePalette_p, map(distance, DISTANCE_MIN, DISTANCE_MAX, 0,
240), BRIGHTNESS, currentBlending);

  // index
  xyz[0] = 0;     xyz[1] = rgb.r;   xyz[2] = rgb.g;   xyz[3] = rgb.b;  // anchor of first color - must be zero
  xyz[4] = 40;    xyz[5] = rgb.r;   xyz[6] = rgb.g;   xyz[7] = rgb.b;
  xyz[8] = 255;   xyz[9] = 0;       xyz[10] = 0;      xyz[11] = 0;      // anchor of last color - must be 255

  currentPalette.loadDynamicGradientPalette(xyz);
}



// fill FastLED leds[] strip/ring with colors
void FillLEDsFromPalette( byte offset) {
  //fill_solid( leds, NUM_LEDS, 0);

  // One gradient
  for ( int i = 0; i < NUM_LEDS; i++) {
    leds[ i ] = ColorFromPalette( currentPalette, (i+offset) * 255 / (NUM_LEDS - 1), BRIGHTNESS,
currentBlending);
```

```
  }
  // Two gradients
  // for ( int i = 0; i < NUM_LEDS/2; i++) {
  //   leds[ i ] = ColorFromPalette( currentPalette, (i+offset) * 255 / (NUM_LEDS/2 - 1), BRIGHTNESS,
currentBlending);
  //   leds[ i+(NUM_LEDS/2) ] = leds[i];
  // }
}

// Fill all leds with one color and wait duration ms
void FillColorDelay ( CRGB color, unsigned long duration ) {
  fill_solid( leds, NUM_LEDS, color);
  FastLED.show();
  FastLED.delay(duration);
}


// This example shows how to set up a static color palette
// which is stored in PROGMEM (flash), which is almost always more
// plentiful than RAM.  A static PROGMEM palette like this
// takes up 64 bytes of flash.
const TProgmemPalette16 mySemaphorePalette_p PROGMEM = {
  CRGB::Magenta, CRGB::Red,     CRGB::Red,     CRGB::Red,
  CRGB::Orange,  CRGB::Orange,  CRGB::Orange,  CRGB::Orange,
  CRGB::Yellow,  CRGB::Yellow,  CRGB::Yellow,  CRGB::Green,
  CRGB::Green,   CRGB::Green,   CRGB::Green,   CRGB::White
};



// get distance number 0-400 cm
// use TRIG_PIN and ECHO_PIN of HC-SR04 sensor
int get_distance() {
  static int distance;
  uint16_t duration = 0;
  uint32_t interval = 0;

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(5);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Read time of the trig and echo pins
  duration = pulseIn(ECHO_PIN, HIGH);
  // Calculates the distance
  distance = (duration / 2) / 29;

  return distance; // centimeters
}
```