

1-variant

1. IoT qishloq xo'jaligida qanday ishlatiladi? IoT qishloq xo'jaligida qanday yordam beradi? Uchta misol keltiring, ularni ishlash prinsipini batafsil tushuntiring. Har bir misolda qanday sensorlar va qurilmalar ishlatilishini tushuntiring.

Javob:

IoT (Internet of Things — narsalar interneti) qishloq xo'jaligida samaradorlikni oshirish, resurslardan oqilona foydalanish va hosildorlikni nazorat qilishda katta yordam beradi. Quyida IoT texnologiyasi qishloq xo'jaligida qanday qo'llanilishi va uchta aniq misol keltirilgan, ularning ishlash prinsipi va ishlatiladigan sensorlar bilan:

1. Aqlli sug'orish tizimi

Ishlash prinsipi:

Tuproqdagi namlik darajasi sensorlar yordamida o'lchanadi. Agar tuproq quruqligi ma'lum chegaradan past bo'lsa, IoT tizimi nasosni avtomatik ishga tushirib, ekinlarni sug'oradi. Tizim quyosh energiyasi bilan ishlashi mumkin va simsiz aloqada bo'ladi.

Qo'llaniladigan sensor va qurilmalar:

- Tuproq namligi sensori (Soil Moisture Sensor):** Tuproqdagi suv miqdorini o'lchaydi.
- Temperatura sensori (DHT11 yoki DHT22):** Havoning harorati va namligini aniqlaydi.
- Arduino/ESP32:** Sensor ma'lumotlarini o'qib, qaror qabul qiladi.
- Nasos va rele moduli:** Suvni haydash uchun ishlatiladi.
- Wi-Fi moduli:** Masofadan boshqarish yoki monitoring qilish uchun.

Foyda: Suv sarfi kamayadi, o'simliklar doimo kerakli miqdorda suv oladi, ishchi kuchiga ehtiyoj kamayadi.

2. Ekinlar monitoringi uchun dron va kamera tizimi

Ishlash prinsipi:

Dronlar IoT bilan integratsiya qilingan kameralar orqali dalalarni aylanib chiqadi. Ular NDVI (Normalized Difference Vegetation Index) asosida o'simlik salomatligini tahlil qiladi. Dronlar yig'gan ma'lumotlar tahlil qilinib, muammo mavjud joylar aniq belgilanadi.

Qo'llaniladigan sensor va qurilmalar:

- Multispektral kamera:** O'simlik salomatligini turli yorug'lik to'lqinlarida ko'rib baholaydi.
- GPS moduli:** Dronning aniq joylashuvini aniqlash uchun.
- Wi-Fi/LoRa moduli:** Uzoq masofaga ma'lumot uzatish uchun.
- IoT platformasi (masalan, ThingsBoard yoki Blynk):** Ma'lumotlarni ko'rsatish va saqlash.

Foyda: O'simlik kasalliklari erta aniqlanadi, o'g'itlash va purkash manzilli bajariladi, resurslar tejiladi.

3. Chorvachilik monitoring tizimi

Ishlash prinsipi:

Har bir hayvonga o'rnatilgan IoT qurilmalar orqali ularning harakatlanishi, tana harorati va joylashuvi kuzatib boriladi. Bu orqali sog'liq muammolari, homiladorlik holati yoki g'ayritabiiy harakatlar aniqlanadi.

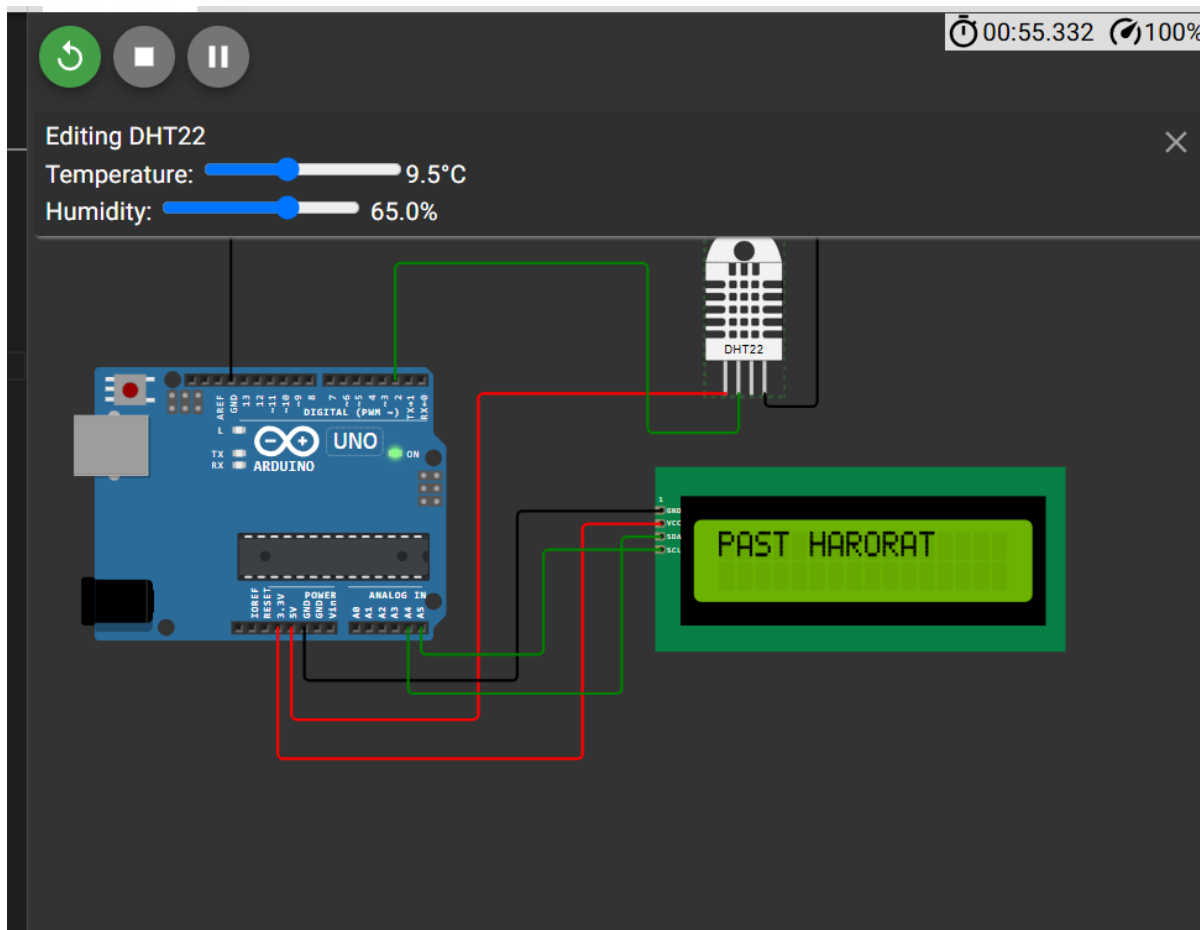
Qo'llaniladigan sensor va qurilmalar:

- **GPS treker:** Hayvonning joylashuvini aniqlaydi.
- **Tana harorati sensori:** Ichki tana haroratini o'lchaydi (masalan, RFID bilan birga ishlovchi implant).
- **Accelerometr:** Harakat faolligini aniqlaydi.
- **LoRa/ NB-IoT moduli:** Qishloq hududida uzoq masofali aloqani ta'minlaydi.

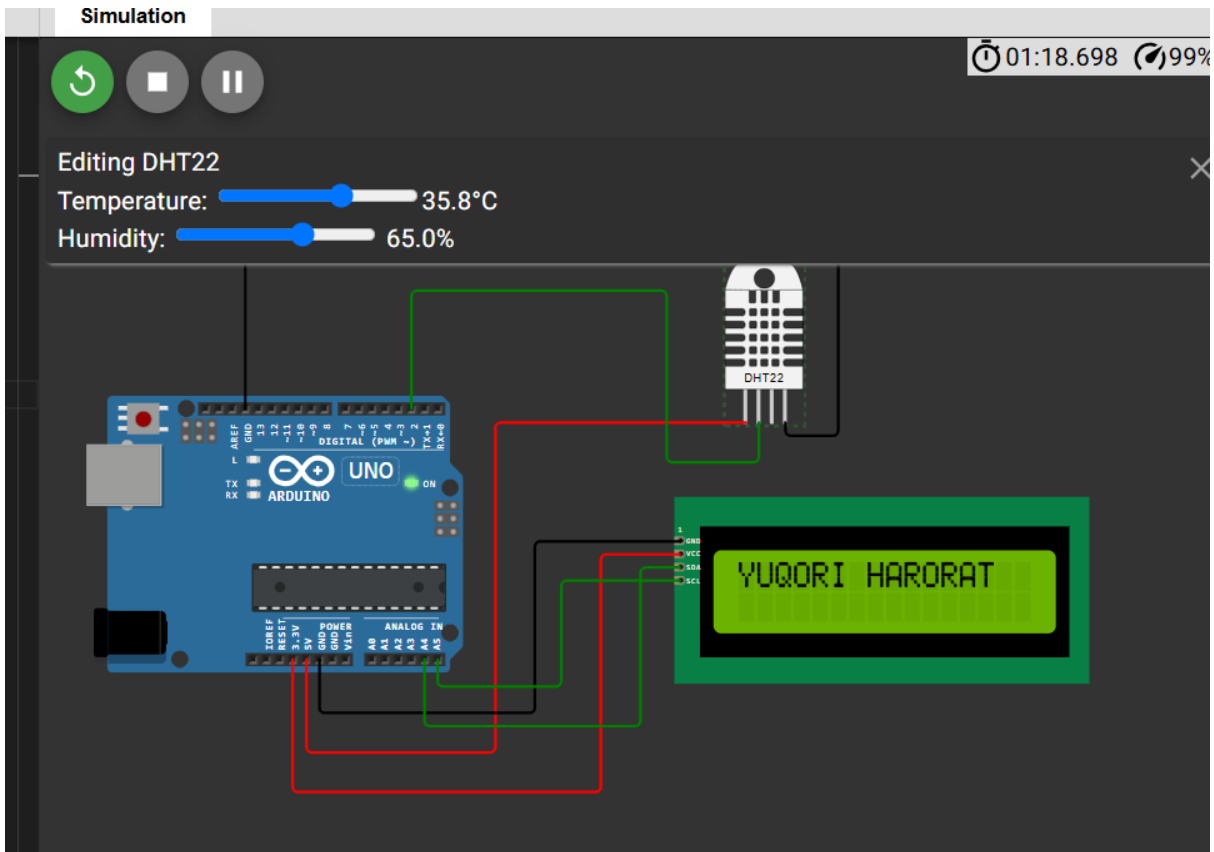
Foyda: Hayvonlar sog'lig'i nazoratda bo'ladi, yo'qolgan hayvonlar tez topiladi, veterinar aralashuvi samaraliroq bo'ladi.

2. LCD1602 va DHT22 bilan issiqxona tizimi yaratish. LCD1602 hamda DHT22 harorat va namlik sensori yordamida arduinoda issiqxona tizimini yarating, unda quyidagi shartlarni bajaring: - LCD 1602'da harorat va namlik ko'rsatilsin. - Harorat 30 darajadan oshsa, "YUQORI HARORAT" yozuvi LCDga chiqsin - Harorat 10 darajadan tushsa, "PAST HARORAT" yozuvi LCDga chiqsin

10 dan past harorat natijasi



10 dan yuqori harorat natijasi



Dastur kodi

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

// LCD1602 I2C manzili odatda 0x27 bo'ladi
LiquidCrystal_I2C lcd(0x27, 16, 2);

// DHT22 sensor
#define DHTPIN 2
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  lcd.begin(16,2);
  lcd.backlight();
  dht.begin();
}

void loop() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();

  lcd.clear();

  if (isnan(temp) || isnan(hum)) {
    lcd.setCursor(0, 0);
```

```

    lcd.print("Sensor xatosi!");
    delay(2000);
    return;
}

lcd.setCursor(0, 0);
lcd.print("T: ");
lcd.print(temp);
lcd.print((char)223); // Gradus belgisi
lcd.print("C");

lcd.setCursor(0, 1);
lcd.print("H: ");
lcd.print(hum);
lcd.print("%");

delay(2000); // ma'lumotni 2 soniyada yangilash

// Harorat shartlarini tekshirish
if (temp > 30) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("YUQORI HARORAT");
    delay(2000);
} else if (temp < 10) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("PAST HARORAT");
    delay(2000);
}
}

```

Diagram.json

```

{
  "version": 1,
  "author": "ww",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-arduino-uno", "id": "uno", "top": 0, "left": 0, "attrs": {} },
    {
      "type": "wokwi-lcd1602",
      "id": "lcd1",
      "top": 73.6,
      "left": 428,
      "attrs": { "pins": "i2c" }
    },
    { "type": "wokwi-dht22", "id": "dht1", "top": -95.7, "left": 465, "attrs": {} }
  ],

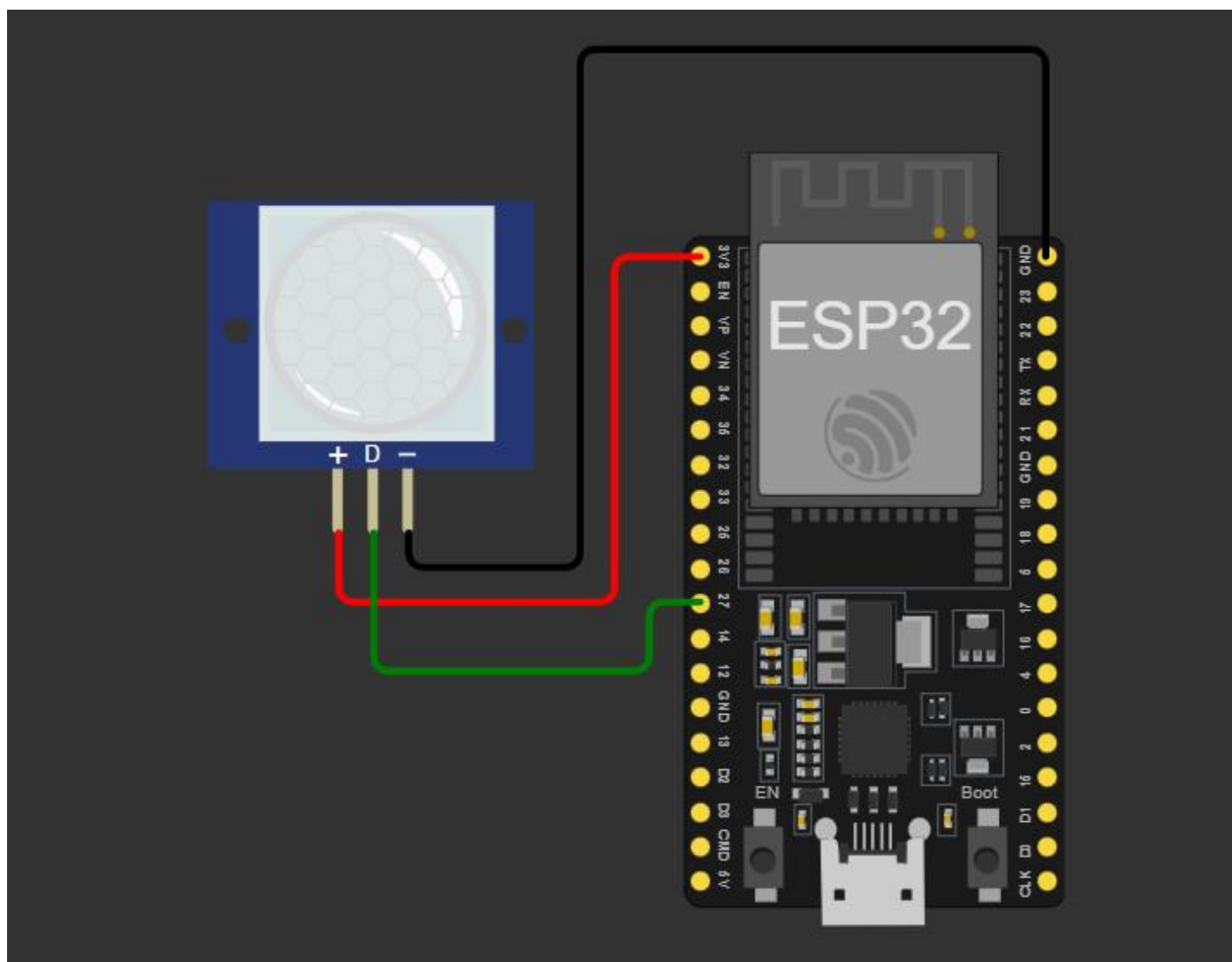
```

```





"connections": [
  [ "dht1:VCC", "uno:5V", "red", [ "v0", "h-182.4", "v240", "h-124.8" ] ],
  [ "dht1:SDA", "uno:2", "green", [ "v28.8", "h-67.1", "v-124.8", "h-163.2" ] ],
  [ "dht1:GND", "uno:GND.1", "black", [ "v9.6", "h38.4", "v-144", "h-422.4" ] ],
  [ "lcd1:GND", "uno:GND.2", "black", [ "h-105.6", "v124.8", "h-144" ] ],
  [ "lcd1:VCC", "uno:3.3V", "red", [ "h-57.6", "v172.9", "h-230.4" ] ],
  [ "lcd1:SDA", "uno:A4", "green", [ "h-28.8", "v115.4", "h-144" ] ],
  [ "lcd1:SCL", "uno:A5", "green", [ "h-86.4", "v77.1", "h-86.4" ] ]
],
"dependencies": {}
}

```

3. ESP32 dan foydalangan holda MQTT brokerga xabar yuborish (MQTT broker: broker.hivemq.com) ESP32, PIR sensori va MQTT broker orqali quyidagi shartlarni bajaruvchi loyiha tuzing: - ESP32 WiFi tarmog'iga ulansin hamda, Serial monitorga "WiFi ga ulandi" yozuvi chiqarilsin. - ESP32 MQTT tizimiga ulansin hamda, Serial monitorga "MQTT brokerga ulandi" yozuvi chiqarilsin. - Ulanish amalga oshirilgach PIR sensori holatini MQTT brokerga yuboring: "Yuborildi ->Harakat mavjud" yoki "Yuborildi ->Harakat mavjud emas".



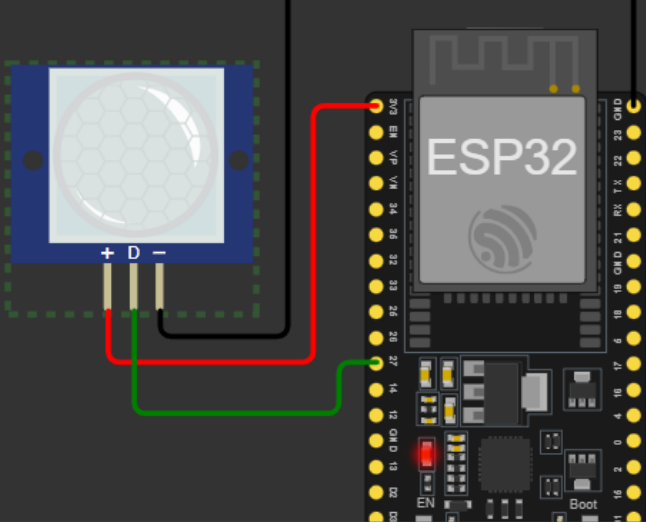
Simulation






00:23.632 49%

PIR Motion Sensor

Simulate motion



Wi-Fi ga ulanmoqda: Wokwi-GUEST
.....
WiFiga ulandi
IP manzil: 10.10.0.2
MQTT brokerga ulanmoqda...MQTT brokerga ulandi
Yuborildi -> Harakat mavjud



Simulation

00:27.398 47%

PIR Motion Sensor

Simulate motion

```

Wi-Fi ga ulanmoqda: Wokwi-GUEST
.....
WiFiga ulandi
IP manzil: 10.10.0.2
MQTT brokerga ulanmoqda...MQTT brokerga ulandi
Yuborildi -> Harakat mavjud
Yuborildi -> Harakat mavjud emas

```

Kodi

```

#include <WiFi.h>
#include <PubSubClient.h>

// WiFi sozlamalari
const char* ssid = "Wokwi-GUEST";
const char* password = "";

// MQTT broker manzili
const char* mqtt_server = "broker.hivemq.com";

// PIR sensor GPIO pini
const int pirPin = 27;

// MQTT ob'ektlari
WiFiClient espClient;
PubSubClient client(espClient);

```

```
// O'zgaruvchilar
int lastPirState = LOW;

void setup_wifi() {
    delay(10);
    Serial.begin(9600);
    Serial.println();
    Serial.print("Wi-Fi ga ulanmoqda: ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("\nWiFiga ulandi");
    Serial.print("IP manzil: ");
    Serial.println(WiFi.localIP());
}

// MQTT brokerga qayta ulanish
void reconnect() {
    while (!client.connected()) {
        Serial.print("MQTT brokerga ulanmoqda...");
        String clientId = "ESP32Client-" + String(random(0xffff), HEX);
        if (client.connect(clientId.c_str())) {
            Serial.println("MQTT brokerga ulandi");
        } else {
            Serial.print(" Xatolik, kod: ");
            Serial.print(client.state());
            delay(5000);
        }
    }
}

void setup() {
    pinMode(pirPin, INPUT);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    int currentPirState = digitalRead(pirPin);
    if (currentPirState != lastPirState) {
        if (currentPirState == HIGH) {
```

```

    Serial.println("Yuborildi -> Harakat mavjud");
    client.publish("iot/pir", "Harakat mavjud");
  } else {
    Serial.println("Yuborildi -> Harakat mavjud emas");
    client.publish("iot/pir", "Harakat mavjud emas");
  }
  lastPirState = currentPirState;
}
}

```

Diagram.json

```

{
  "version": 1,
  "author": "ww",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": {} },
    { "type": "wokwi-pir-motion-sensor", "id": "pir1", "top": 13.6, "left": -132.18,
"attrs": {} }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "pir1:VCC", "esp:3V3", "red", [ "v19.2", "h76.8", "v-96" ] ],
    [ "pir1:GND", "esp:GND.2", "black", [ "v9.6", "h47.74", "v-144", "h124.8" ] ],
    [ "pir1:OUT", "esp:27", "green", [ "v38.4", "h76.66", "v-9.6" ] ]
  ],
  "dependencies": {}
}

```

2-variant

1. IoT sogʻliqni saqlashda qanday ishlatiladi? IoT sogʻliqni saqlashda qanday yordam beradi? Masofadan bemor monitoringi (RPM) tizimi qanday ishlaydi? RPM uchun qanday sensorlar ishlatiladi va bemorlar uchun qanday foyda keltiradi? Uchta misol keltiring.

IoT sogʻliqni saqlashda qanday ishlatiladi?

IoT (Narsalarning Interneti) sogʻliqni saqlash sohasida **real vaqtda maʼlumot toʻplash**, masofadan tibbiy kuzatuv, va **favqulodda holatlarda tezkor xabar berish** orqali tibbiy xizmat sifatini oshiradi. IoT qurilmalari yordamida bemorning sogʻligʻi haqidagi maʼlumotlar avtomatik tarzda yigʻilib, shifokorga uzatiladi.

📶 Masofadan bemor monitoringi (RPM) tizimi nima va qanday ishlaydi?

RPM (Remote Patient Monitoring) — bu texnologiya orqali bemorlar o'z uylarida bo'lgan holda tibbiy holati doimiy ravishda kuzatiladi. Sensorlar bemorning haroratini, yurak urish tezligini, qon bosimini va boshqa ko'rsatkichlarini o'lchaydi va **Wi-Fi yoki mobil tarmoqlar orqali shifokorga uzatadi**.

RPM orqali shifokorlar tezda og'ishlarni aniqlaydi va erta choralar ko'radi, bu esa bemorning hayotini saqlab qolish imkonini oshiradi.

🔧 RPM tizimida ishlatiladigan asosiy sensorlar:

Sensor turi	O'lchaydigan ma'lumot
ECG sensori	Yurak ritmi (elektrokardiogramma)
Pulse oximeter	Qonda kislorod miqdori, yurak urishi
Body temperature sensor	Tana harorati
Blood pressure sensor	Qon bosimi
Gliko-sensor (CGM)	Qand miqdori (diabetiklar uchun)

🌟 3 ta real misol:

1. 📶 Yurak ritmini kuzatish (ECG monitoring):

- **Sensorlar:** ECG patch (elektrodlar) va ESP32 kabi IoT moduli.
- **Prinsipi:** Sensor yurak impulsini aniqlaydi va Wi-Fi orqali serverga uzatadi.
- **Foydasi:** Arterial fibrillatsiya yoki yurak urishidagi nosozliklar erta aniqlanadi, shifokor ogohlantiriladi.

2. 🌡️ Tana haroratini doimiy kuzatish:

- **Sensor:** DS18B20 yoki MAX30205 tana harorat sensori.
- **Prinsipi:** Sensor tana yuzasidagi haroratni o'lchaydi va ma'lumotlarni MQTT yoki HTTP orqali yuboradi.
- **Foydasi:** Infektsiyalar, isitma yoki hipotermiya holatlari tezda aniqlanadi.

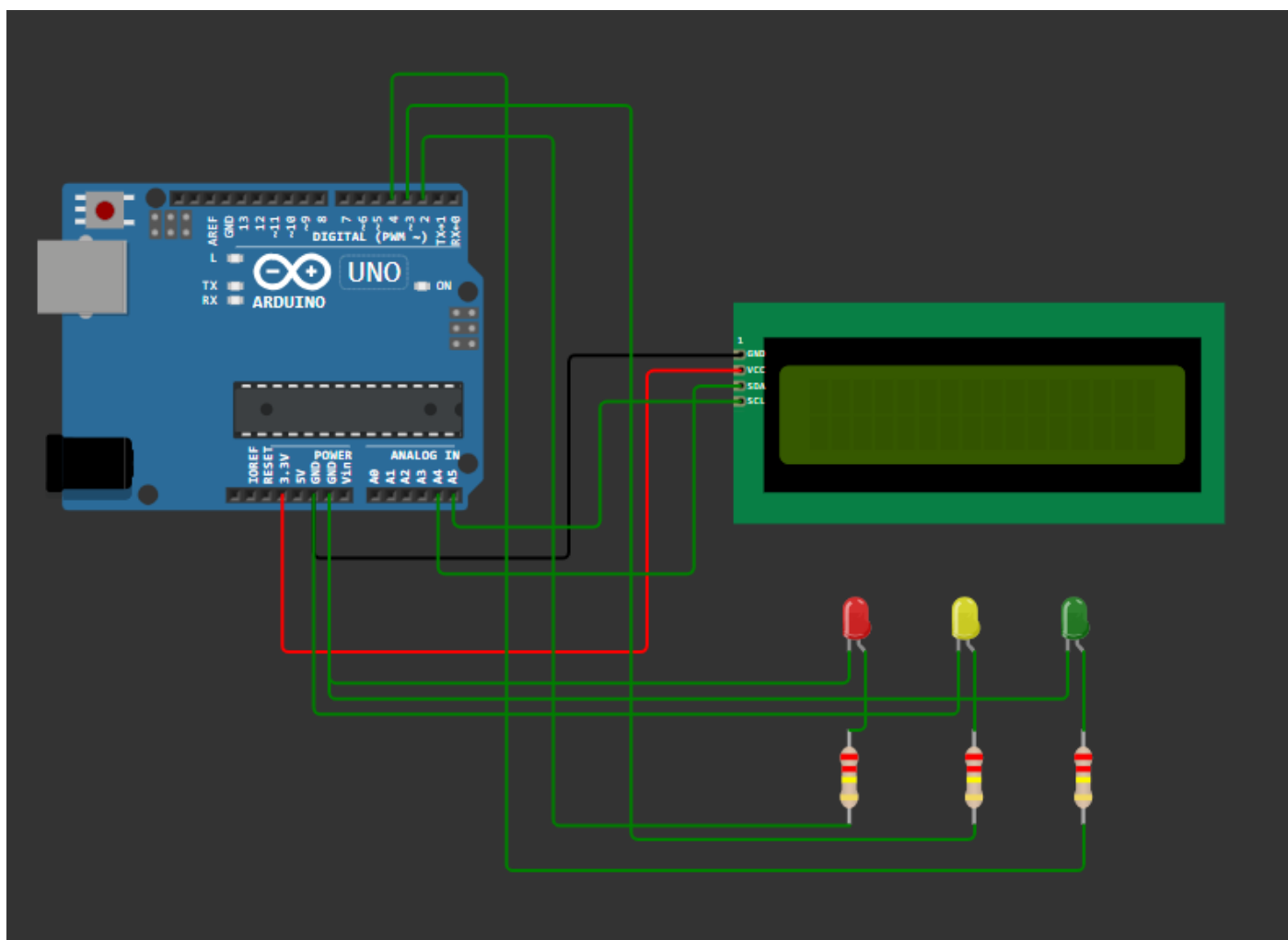
3. 🩸 Diabetik bemorlar uchun qon shakar miqdorini kuzatish:

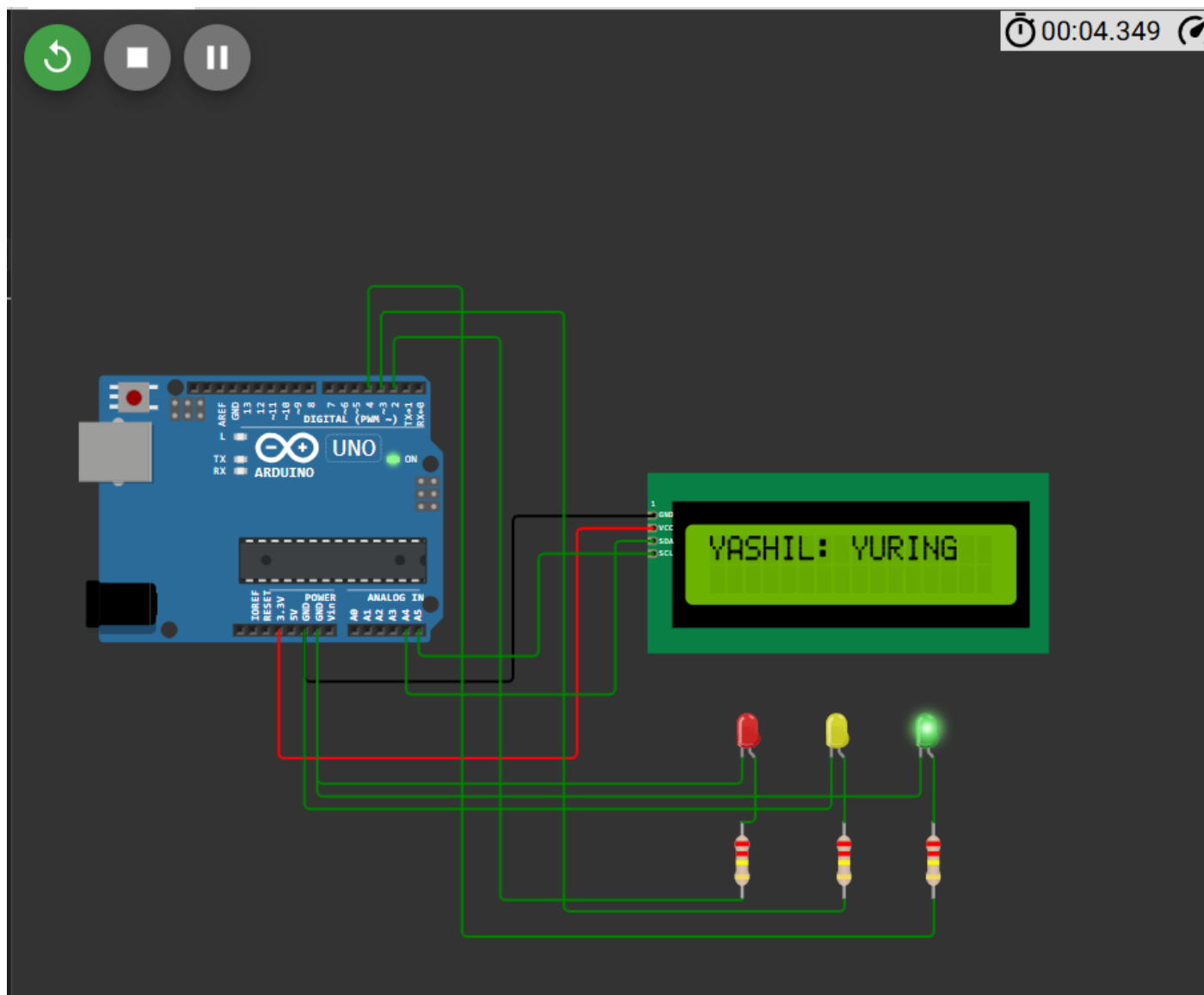
- **Sensor:** Gluco-sensor (Continuous Glucose Monitor – CGM)
- **Prinsipi:** Sensor teri ostidagi qand darajasini o'lchaydi va mobil ilovaga yoki tibbiy markazga yuboradi.
- **Foydasi:** Qand miqdorining ko'tarilishi yoki tushishini real vaqt ichida kuzatish orqali diabetni nazorat qilish osonlashadi.

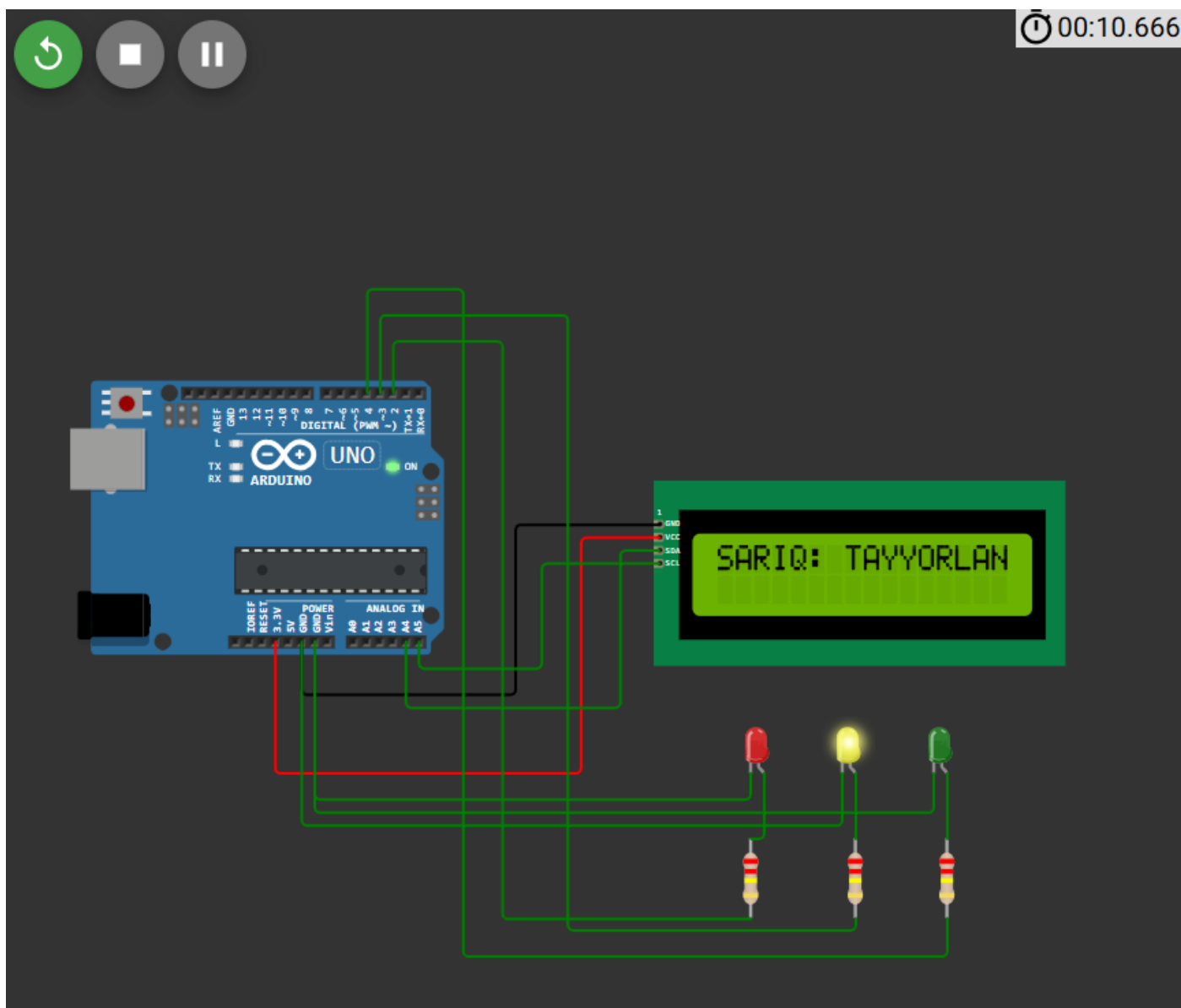
□ Xulosa:

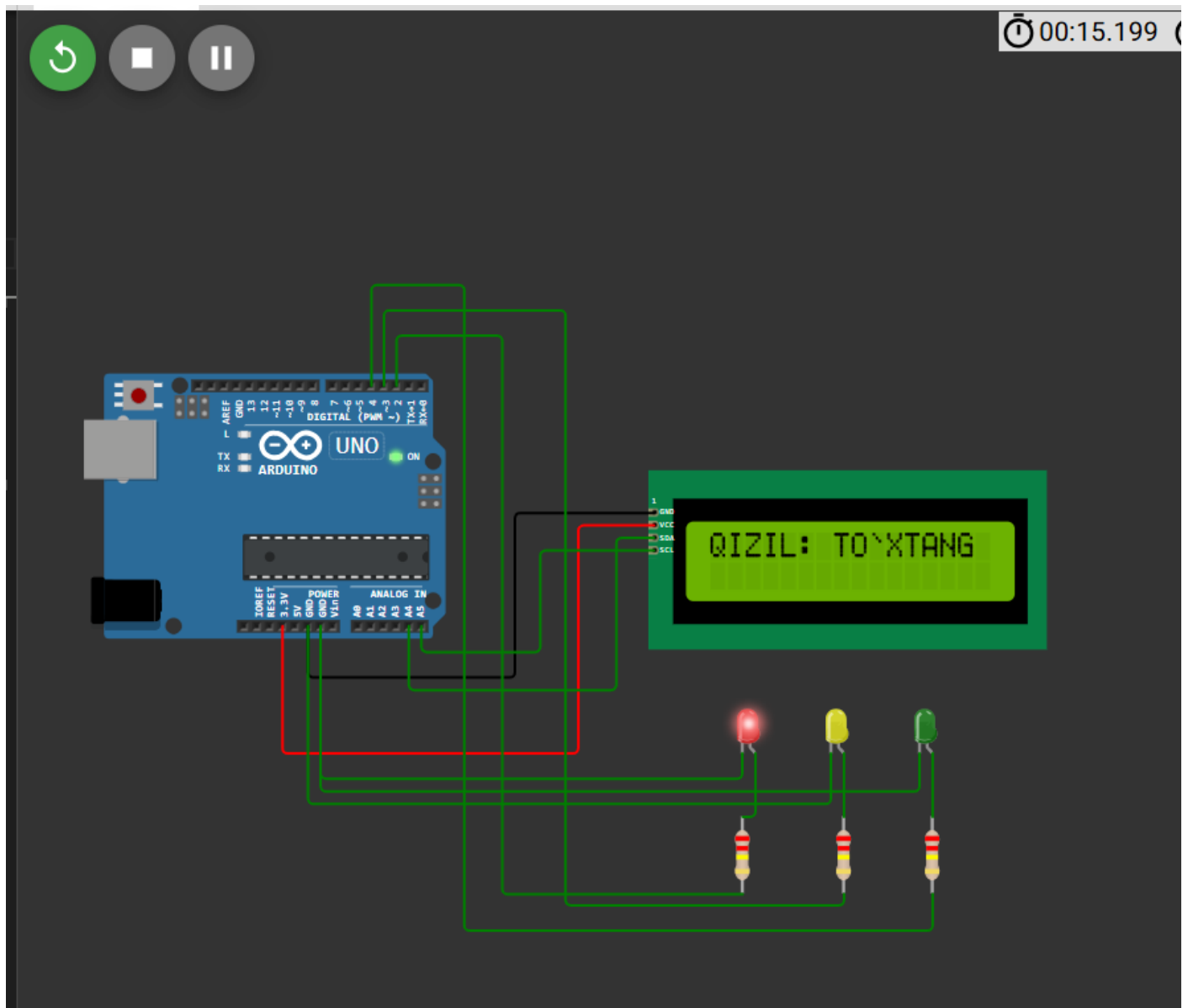
IoT texnologiyalari sog'liqni saqlashda inqilobiy o'zgarishlar kiritmoqda. Ayniqsa RPM tizimlari orqali bemorlar masofadan xavfsiz, qulay va doimiy nazorat ostida bo'ladi. Bu esa kasalliklarning oldini olish va erta davolash imkonini yaratadi.

2. LCD1604 va LED chiroqlar orqali svetafor tizimini yaratish. LCD1604 va LED chiroqlari (qizil, sariq, yashil) orqali arduinoda svetafor tizimini yarating, unda quyidagi shartlarni bajaring: - Rezistorlardan foydalaning - Rang almashinishlarida sariq chiroq 3 marta miltillasin - Ranglar ketma-ketligi svetafor ishlash tartibida bo'lsin









Dastur kodi

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// LCD1602 uchun I2C manzil 0x27 bo'lishi mumkin
LiquidCrystal_I2C lcd(0x27, 16, 2);

// LED pinlar
const int redLED = 2;
const int yellowLED = 3;
const int greenLED = 4;

void setup() {
  // LED pinlar chiqish sifatida
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
}
```



```

// LCD boshlanishi
lcd.init();
lcd.backlight();
lcd.setCursor(0, 0);
lcd.print("Svetafor Tayyor!");
delay(2000);
lcd.clear();
}

void loop() {
// Yashil chiroq YONADI
digitalWrite(greenLED, HIGH);
lcd.setCursor(0, 0);
lcd.print("YASHIL: YURING ");
delay(5000);

// Sariq chiroq 3 marta MILTILLAYDI
digitalWrite(greenLED, LOW);
for (int i = 0; i < 3; i++) {
    digitalWrite(yellowLED, HIGH);
    lcd.setCursor(0, 0);
    lcd.print("SARIQ: TAYYORLAN");
    delay(500);
    digitalWrite(yellowLED, LOW);
    delay(500);
}

// Qizil chiroq YONADI
digitalWrite(redLED, HIGH);
lcd.setCursor(0, 0);
lcd.print("QIZIL: TO`XTANG ");
delay(5000);

// Sariq chiroq 3 marta MILTILLAYDI yana
digitalWrite(redLED, LOW);
for (int i = 0; i < 3; i++) {
    digitalWrite(yellowLED, HIGH);
    lcd.setCursor(0, 0);
    lcd.print("SARIQ: TAYYORLAN");
    delay(500);
    digitalWrite(yellowLED, LOW);
    delay(500);
}

// Yana boshidan davom etadi
}

```

```

{
  "version": 1,
  "author": "ww",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-arduino-uno", "id": "uno", "top": 0, "left": 0, "attrs": {} },
    {
      "type": "wokwi-lcd1602",
      "id": "lcd1",
      "top": 73.6,
      "left": 428,
      "attrs": { "pins": "i2c" }
    },
    { "type": "wokwi-led", "id": "led1", "top": 246, "left": 483.8, "attrs": { "color": "red" } },
    { "type": "wokwi-led", "id": "led2", "top": 246, "left": 551, "attrs": { "color": "yellow" } },
    {
      "type": "wokwi-led",
      "id": "led3",
      "top": 246,
      "left": 618.2,
      "attrs": { "color": "green" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": 360,
      "left": 469.85,
      "rotate": 90,
      "attrs": { "value": "220000" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r2",
      "top": 360,
      "left": 546.65,
      "rotate": 90,
      "attrs": { "value": "220000" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r3",
      "top": 360,
      "left": 613.85,
      "rotate": 90,
      "attrs": { "value": "220000" }
    }
  ],
  "connections": [
    [ "lcd1:GND", "uno:GND.2", "black", [ "h-105.6", "v124.8", "h-144" ] ],
    [ "lcd1:VCC", "uno:3.3V", "red", [ "h-57.6", "v172.9", "h-230.4" ] ],

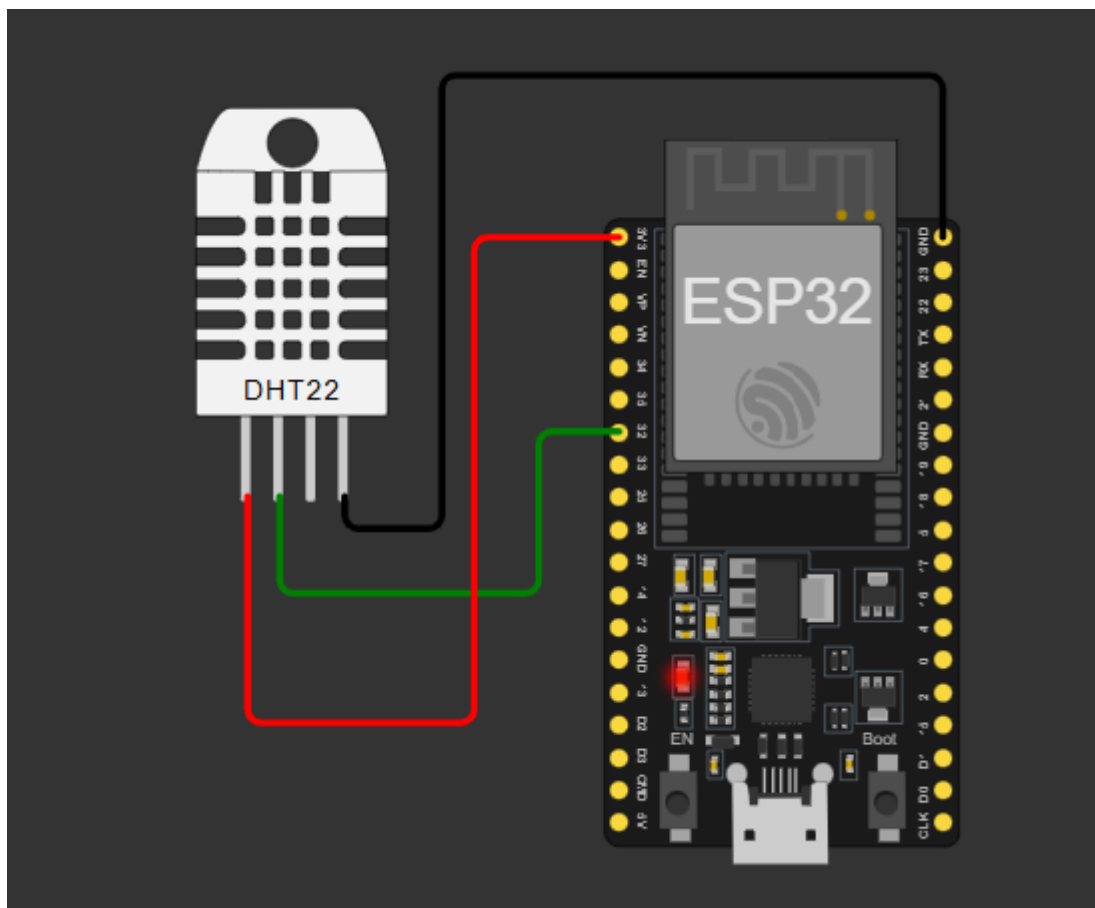
```

```

[ "lcd1:SDA", "uno:A4", "green", [ "h-28.8", "v115.4", "h-144" ] ],
[ "lcd1:SCL", "uno:A5", "green", [ "h-86.4", "v77.1", "h-86.4" ] ],
[ "led1:A", "r1:1", "green", [ "v0" ] ],
[ "r1:2", "uno:2", "green", [ "h-182.4", "v-423.6", "h-48" ] ],
[ "led1:C", "uno:GND.3", "green", [ "v19.2", "h-239.6" ] ],
[ "led2:C", "uno:GND.2", "green", [ "v38.4", "h-402.8" ] ],
[ "led3:C", "uno:GND.3", "green", [ "v28.8", "h-450.8" ] ],
[ "led2:A", "r2:1", "green", [ "v0" ] ],
[ "led3:A", "r3:1", "green", [ "v57.6" ] ],
[ "r3:2", "uno:4", "green", [ "h0", "v27.6", "h-355.2", "v-489.6", "h-67.2" ] ],
[ "r2:2", "uno:3", "green", [ "h0", "v8.4", "h-211.2", "v-451.2", "h-134.4" ] ]
],
"dependencies": {}
}

```

3. ESP32 dan foydalangan holda MQTT brokerga xabar yuborish (MQTT broker: broker.hivemq.com) ESP32, DHT22 sensori va MQTT broker orqali quyidagi shartlarni bajaruvchi loyiha tuzing: - ESP32 WiFi tarmog'iga ulansin hamda, Serial monitorga "WiFiga ulandi" yozuvi chiqarilsin. - ESP32 MQTT tizimiga ulansin hamda, Serial monitorga "MQTT brokerga ulandi" yozuvi chiqarilsin. - Ulanish amalga oshirilgach DHT22 sensori ma'lumotlarini MQTT brokerga yuboring: "Yuborildi -> Namlik: 20%, Harorat: 18C".



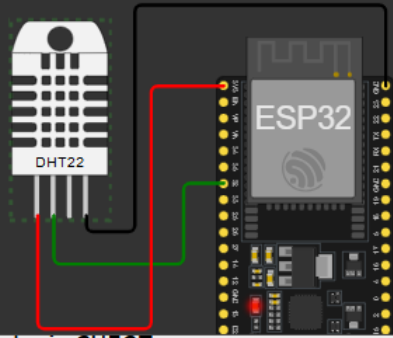
Simulation

00:31.298 44%

Editing DHT22

Temperature: 12.0°C

Humidity: 32.0%



```

WiFi ga ulanmoqda: Wokwi-GUEST
.....
WiFiga ulandi
IP manzil: 10.10.0.2
MQTT brokerga ulanmoqda... Ulandi!
Yuborildi -> Namlik: 40.0%, Harorat: 24.0C
Yuborildi -> Namlik: 62.5%, Harorat: 62.2C
Yuborildi -> Namlik: 62.5%, Harorat: 62.2C
Yuborildi -> Namlik: 62.5%, Harorat: 62.2C
Yuborildi -> Namlik: 32.0%, Harorat: 12.0C

```

24°C ENG 11:36:05 PM

Dastur kodi

```

#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

// Wi-Fi sozlamalari
const char* ssid = "Wokwi-GUEST"; // WiFi nomi
const char* password = ""; // WiFi paroli

// MQTT broker manzili
const char* mqtt_server = "broker.hivemq.com";

// DHT sensor konfiguratsiyasi
#define DHTPIN 32
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

```

```
// MQTT va Wi-Fi obyektlari
WiFiClient espClient;
PubSubClient client(espClient);

// Xabar yuborish intervalli (ms)
unsigned long lastMsg = 0;
const long interval = 5000; // 5 soniya

void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("WiFi ga ulanmoqda: ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("\nWiFiga ulandi");
    Serial.print("IP manzil: ");
    Serial.println(WiFi.localIP());
}

// MQTT brokerga qayta ulanadigan funksiya
void reconnect() {
    while (!client.connected()) {
        Serial.print("MQTT brokerga ulanmoqda...");
        String clientId = "ESP32Client-" + String(random(0xffff), HEX);
        if (client.connect(clientId.c_str())) {
            Serial.println(" Ulandi!");
        } else {
            Serial.print(" Xatolik, kod: ");
            Serial.print(client.state());
            Serial.println(" | 5 soniyadan keyin qayta urinish");
            delay(5000);
        }
    }
}

void setup() {
    Serial.begin(115200);
    dht.begin();
    setup_wifi();
    client.setServer(mqtt_server, 1883);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
}
```

```

}
client.loop();

unsigned long now = millis();
if (now - lastMsg > interval) {
    lastMsg = now;

    float h = dht.readHumidity();
    float t = dht.readTemperature();

    if (isnan(h) || isnan(t)) {
        Serial.println("Sensor o'qish xatosi!");
        return;
    }

    // Xabar tayyorlash
    char message[64];
    snprintf(message, sizeof(message), "Yuborildi -> Namlik: %.1f%%, Harorat: %.1fC", h,
t);

    // MQTT ga yuborish
    client.publish("iot/sensor", message);
    Serial.println(message);
}
}

```

Diagram.json

```

{
  "version": 1,
  "author": "ww",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": {} },
    { "type": "wokwi-dht22", "id": "dht1", "top": -9.3, "left": -120.6, "attrs": {} }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "dht1:SDA", "esp:32", "green", [ "v28.8", "h76.9", "v-48" ] ],
    [ "dht1:VCC", "esp:3V3", "red", [ "v67.2", "h67.2", "v-115.2" ] ],
    [ "dht1:GND", "esp:GND.2", "black", [ "v9.6", "h28.8", "v-134.4", "h134.4" ] ]
  ],
  "dependencies": {}
}

```

3 variant

1. IoT aqlli shaharda qanday ishlatiladi? IoT aqlli shahar loyihalarida qanday yordam beradi? Yo'l harakati, chiqindi boshqaruvi yoki energiya tejashda IoT qanday ishlaydi? Uchta misol keltiring va har birida qanday sensorlar ishlatilishini tushuntiring.

IoT (Internet of Things) texnologiyasi **aqlli shahar** (Smart City) loyihalarida samaradorlikni oshirish, resurslarni tejash va aholiga qulaylik yaratish uchun keng qo'llaniladi. Quyida IoT'ning aqlli shahardagi ishlatilishi, foydasi va uchta aniq misoli keltirilgan:

◆ IoT nima uchun aqlli shaharda muhim?

- Real vaqt ma'lumotlar asosida qarorlar qabul qilish imkonini beradi.
- Avtomatlashtirilgan tizimlar bilan inson mehnatini kamaytiradi.
- Tashkilotlarga resurslar (elektr, suv, chiqindi)ni samarali boshqarishga yordam beradi.
- Aholi xavfsizligini oshiradi va yashash sifatini yaxshilaydi.

✓ 1. Yo'l harakati boshqaruvi (Traffic Management)

★ Tushuntirish:

IoT yordamida tirbandliklar aniqlanadi, svetaforlar avtomatik boshqariladi va haydovchilarga bo'sh yo'llar haqida ma'lumot yuboriladi.

🔍 Ishlatiladigan sensorlar va qurilmalar:

- **IR yoki ultrasonik datchiklar** – avtomobillar sonini aniqlash.
- **Kamera + AI** – yo'l holatini aniqlash va xavfsizlikni nazorat qilish.
- **RFID** – jamoat transportlarini kuzatish.

⚙️ Ishlash prinsipi:

Sensorlar yo'l bo'yida avtomobillar oqimini o'lchaydi. Ma'lumotlar markaziy tizimga yuboriladi. Svetaforlar tirbandlik bo'yicha avtomatik ravishda yashil-qizil vaqtni o'zgartiradi.

✓ 2. Chiqindi boshqaruvi (Smart Waste Management)

★ Tushuntirish:

IoT chiqindi qutilari to'lish darajasini nazorat qiladi va chiqindilarni yig'ish marshrutlarini optimallashtiradi.

🦋 Ishlatiladigan sensorlar va qurilmalar:

- **Ultrasonik sensor** – chiqindi konteyneri to'lish darajasini o'lchaydi.
- **GPS moduli** – chiqindi mashinalari joylashuvini aniqlaydi.
- **GPRS/LoRa moduli** – markaziy tizimga ma'lumot yuborish.

🦋 Ishlash prinsipi:

Chiqindi qutilariga o'rnatilgan sensorlar to'lish darajasini o'lchab, serverga yuboradi. Tizim to'la qutilar asosida eng qisqa chiqindi yig'ish yo'nalishini tuzadi.

✓ 3. Aqlli yoritish (Smart Street Lighting / Energiya tejash)

🦋 Tushuntirish:

Ko'cha chiroqlari atrof-muhit yorug'ligi yoki harakatga qarab avtomatik yoqiladi yoki o'chiriladi.

🦋 Ishlatiladigan sensorlar:

- **Harakat (PIR) sensori** – odam yoki avtomobilni aniqlash.
- **Yorug'lik (LDR) sensori** – atrofda yorug'lik darajasini aniqlash.
- **ZigBee/WiFi moduli** – chiroqni masofadan boshqarish.

🦋 Ishlash prinsipi:

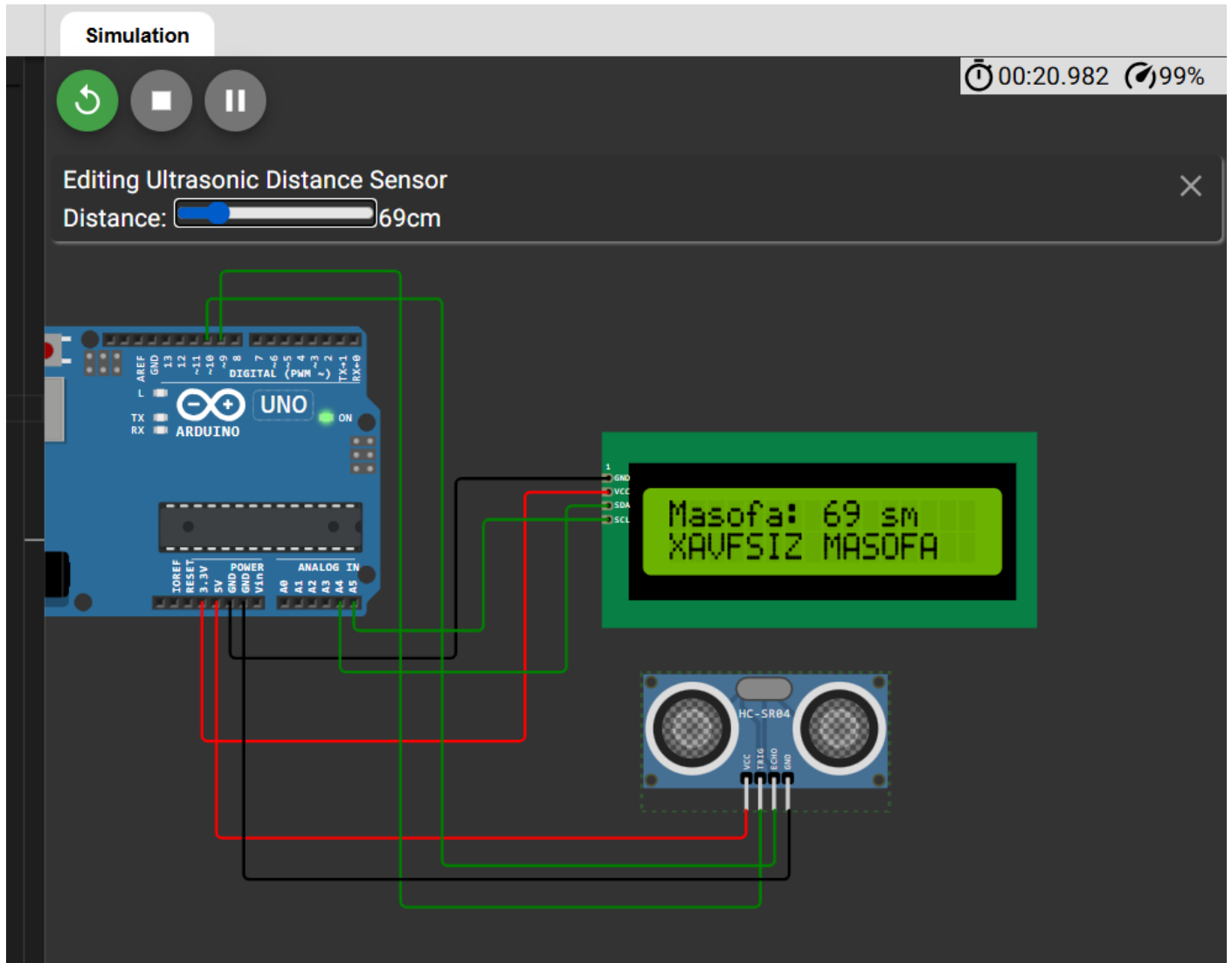
Agar atrof qorong'i bo'lsa va harakat aniqlansa, chiroq avtomatik yoqiladi. Harakat tugagach yoki yorug'lik kuchaygach, chiroq o'chiriladi. Bu energiya sarfini sezilarli kamaytiradi.

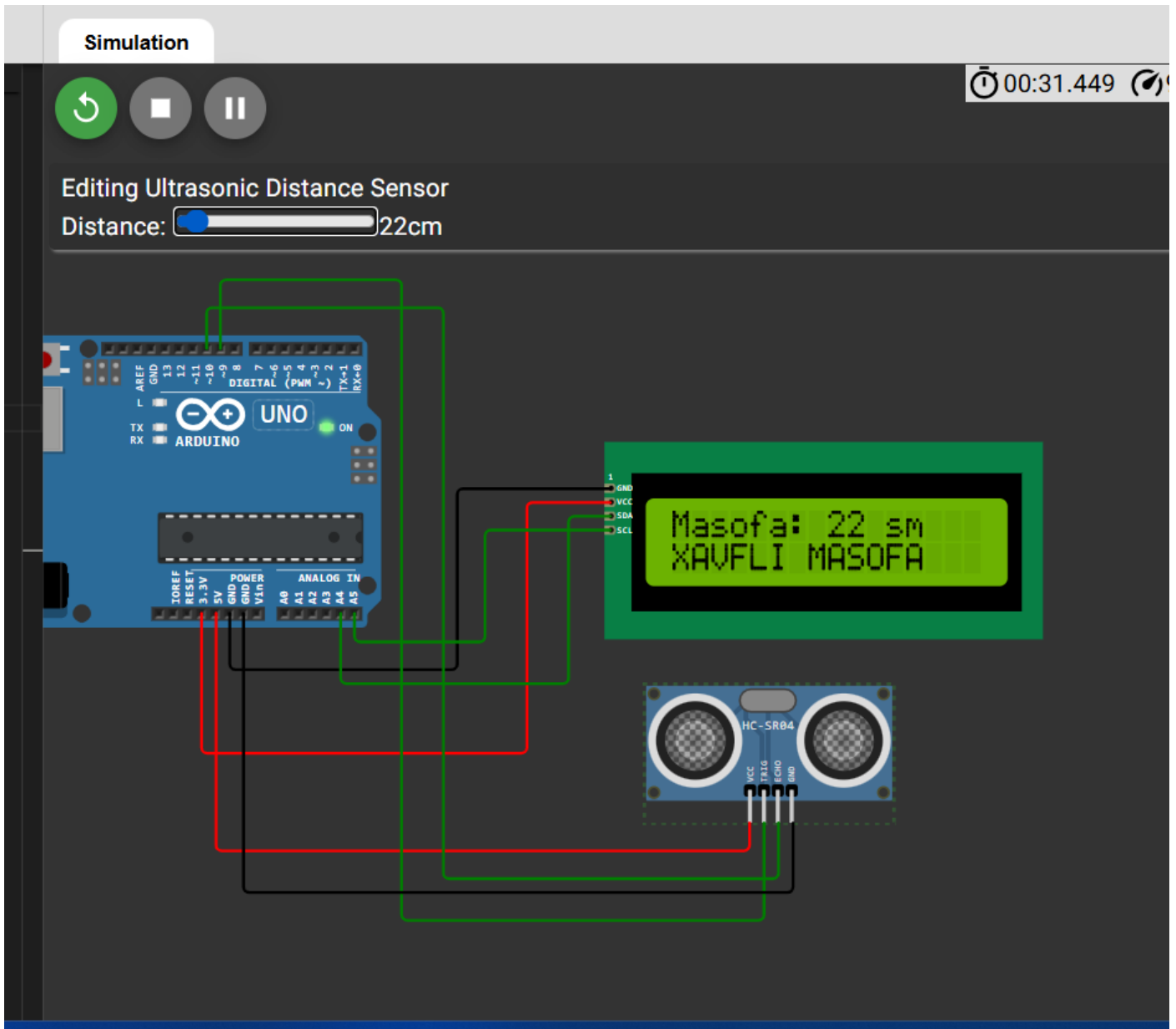
☐ Xulosa:

IoT texnologiyasi aqlli shaharlarni:

- xavfsiz,
- samarali,
- ekologik toza va
- qulay qilishga yordam beradi.

2. LCD1602 va HC-SR04 bilan masofani hisoblash tizimini yaratish. LCD1602 va HC-SR04 ultratovush sensori yordamida arduinoda masofani aniqlash tizimi yarating, unda quyidagi shartlarni bajaring: - LCD 1602'da masofa ko'rsatilsin. - Masofa 65 dan kam bo'lsa, "XAVFLI MASOFA" yozuvi LCDga chiqsin - Masofa 65 yoki undan ko'p bo'lsa, "XAVFSIZ MASOFA" yozuvi LCDga chiqsin





Dastur kodi

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// LCD1602 (I2C) adresi odatda 0x27 bo'ladi
LiquidCrystal_I2C lcd(0x27, 16, 2);

// HC-SR04 pinlari
const int trigPin = 9;
const int echoPin = 10;

void setup() {
  lcd.begin(16,2);
  lcd.backlight();
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}
```

```

void loop() {
  // Ultratovush signali yuborish
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Signaldan qaytish vaqtini o'lchash
  long duration = pulseIn(echoPin, HIGH);

  // Masofani sm da hisoblash
  float distance = duration * 0.034 / 2;

  // LCD tozalash
  lcd.clear();

  // Masofani LCD ga chiqarish
  lcd.setCursor(0, 0);
  lcd.print("Masofa: ");
  lcd.print(distance, 0);
  lcd.print(" sm");

  lcd.setCursor(0, 1);
  if (distance >= 65) {
    lcd.print("XAVFSIZ MASOFA");
  } else {
    lcd.print("XAVFLI MASOFA");
  }

  delay(1000);
}

```

Diagram.json

```

{
  "version": 1,
  "author": "ww",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-arduino-uno", "id": "uno", "top": 0, "left": 0, "attrs": {} },
    {
      "type": "wokwi-lcd1602",
      "id": "lcd1",
      "top": 73.6,
      "left": 428,
      "attrs": { "pins": "i2c" }
    }
  ]
}

```

```

    },
    { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": 241.5, "left": 456.7, "attrs":
  {} }
  ],
  "connections": [
    [ "lcd1:GND", "uno:GND.2", "black", [ "h-105.6", "v124.8", "h-144" ] ],
    [ "lcd1:VCC", "uno:3.3V", "red", [ "h-57.6", "v172.9", "h-230.4" ] ],
    [ "lcd1:SDA", "uno:A4", "green", [ "h-28.8", "v115.4", "h-144" ] ],
    [ "lcd1:SCL", "uno:A5", "green", [ "h-86.4", "v77.1", "h-86.4" ] ],
    [ "ultrasonic1:VCC", "uno:5V", "red", [ "v19.2", "h-336" ] ],
    [ "ultrasonic1:TRIG", "uno:9", "green", [ "v67.2", "h-250", "v-441.6", "h-124.8" ] ],
    [ "ultrasonic1:ECHO", "uno:10", "green", [ "v38.4", "h-231.2", "v-393.6", "h-172.8" ]
  ],
  [ "ultrasonic1:GND", "uno:GND.3", "black", [ "v48", "h-154.8" ] ]
  ],
  "dependencies": {}
}

```

3. ESP32 dan foydalangan holda MQTT brokerga xabar yuborish (MQTT broker: broker.hivemq.com) ESP32, potensiometer va MQTT broker orqali quyidagi shartlarni bajaruvchi loyiha tuzing: - ESP32 WiFi tarmog'iga ulansin hamda, Serial monitorga "WiFiga ulandi" yozuvi chiqarilsin. - ESP32 MQTT tizimiga ulansin hamda, Serial monitorga "MQTT brokerga ulandi" yozuvi chiqarilsin. - Ulanish amalga oshirilgach potensiometer qiymatini MQTT brokerga yuboring: "Yuborildi -> Qiymat – 60".

Simulation

00:22.814 47%

```

.....
WiFi ulandi
IP manzil: 10.10.0.2
MQTT brokerga ulanmoqda...MQTT brokerga ulandi
Yuborildi -> Qiymat - 0
Yuborildi -> Qiymat - 0
Yuborildi -> Qiymat - 85

```

Dastur kodi

```

#include <WiFi.h>
#include <PubSubClient.h>

// Wi-Fi sozlamalari
const char* ssid = "Wokwi-GUEST"; // O'zingizning SSID ni yozing
const char* password = ""; // Agar kerak bo'lsa, parolni yozing

// MQTT broker
const char* mqtt_server = "broker.hivemq.com";

// Potensiometr pin
const int potPin = 34; // Analog kirish pin (ESP32 da A0 yo'q)

// Ob'ektlar
WiFiClient espClient;
PubSubClient client(espClient);

```

```

// Ulash vaqtini tekshirish
unsigned long lastMsg = 0;
const long interval = 5000; // 5 soniyada bir marta yuboriladi

// Wi-Fi ulanish funksiyasi
void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("WiFiga ulanmoqda: ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFiga ulandi");
    Serial.print("IP manzil: ");
    Serial.println(WiFi.localIP());
}

// MQTT brokerga qayta ulanish
void reconnect() {
    while (!client.connected()) {
        Serial.print("MQTT brokerga ulanmoqda...");
        String clientId = "ESP32Client-" + String(random(0xffff), HEX);
        if (client.connect(clientId.c_str())) {
            Serial.println("MQTT brokerga ulandi");
        } else {
            Serial.print("Xatolik, qayta uriniladi. Kod: ");
            Serial.println(client.state());
            delay(5000);
        }
    }
}

void setup() {
    Serial.begin(9600);
    setup_wifi();
    client.setServer(mqtt_server, 1883); // 1883 - standart port
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}

```

```

unsigned long now = millis();
if (now - lastMsg > interval) {
    lastMsg = now;

    int potValue = analogRead(potPin); // 0 - 4095 oralig'ida o'qiladi
    int percentValue = map(potValue, 0, 4095, 0, 100); // 0-100% ga aylantirish

    char msg[50];
    sprintf(msg, "Yuborildi -> Qiymat - %d", percentValue);

    client.publish("iot/potensiometr", msg);
    Serial.println(msg);
}
}

```

Diagram.json

```

{
  "version": 1,
  "author": "ww",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": {} },
    { "type": "wokwi-potentiometer", "id": "pot1", "top": 17.9, "left": -134.6, "attrs": {} }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "pot1:GND", "esp:GND.2", "black", [ "v38.4", "h76.8", "v-163.2", "h115.2" ] ],
    [ "pot1:VCC", "esp:3V3", "red", [ "v19.2", "h76", "v-76.8" ] ],
    [ "pot1:SIG", "esp:34", "green", [ "v67.2", "h76.4", "v-19.2" ] ]
  ],
  "dependencies": {}
}

```

4 – variant

1. IoT transport va logistika sohasida qanday ishlatiladi? IoT transport va logistika sohasida qanday yordam beradi? Tovar kuzatuv, ombor boshqaruvi yoki transport optimallashtirishda IoT qanday ishlaydi? Uchta misol keltiring va qanday qurilmalar ishlatilishini tushuntiring.

JAVOBLAR

IoT (Internet of Things) transport va logistika sohasida real vaqtli ma'lumot almashinuvi, kuzatuv va avtomatlashtirish orqali samaradorlikni oshiradi. Quyida IoTning bu sohalarda qanday yordam berishini, ishlash mexanizmlarini va ishlatiladigan qurilmalarni uchta **aniq misol** bilan tushuntiraman:

1. Tovarlar harakatini real vaqtli kuzatish (Shipment Tracking)

 Qanday ishlaydi:

IoT qurilmalari yordamida yuk mashinalari, konteynerlar yoki qadoqlangan tovarlar harakati GPS orqali kuzatilib, ularning joylashuvi va harakat holati real vaqt rejimida serverga uzatiladi.

 Ishlatiladigan qurilmalar:

- **GPS modul** (tovar yoki transportga o'rnatiladi)
- **GSM yoki NB-IoT moduli** (mobil tarmoqlar orqali ma'lumot yuborish uchun)
- **Sensorli trekerlar** (vibratsiya, harorat, zarba o'lchash uchun)

 Foyda:

- Yukning yo'qolishi yoki kechikishi aniqlanadi
- Mijozlar yukning qayerdaligini aniq biladi
- Sifatli va ishonchli yetkazib berish ta'minlanadi

2. Omborxona boshqaruvi (Smart Warehouse Management)

 Qanday ishlaydi:

IoT texnologiyalari ombor ichidagi mahsulotlar joylashuvi, harorati va harakati haqidagi ma'lumotlarni avtomatik yig'ib, tizimga uzatadi.

 Ishlatiladigan qurilmalar:

- **RFID teglar va skanerlari** (har bir mahsulotni aniqlash)
- **Zigbee yoki Wi-Fi tarmog'idagi sensorlar**
- **Avtonom mobil robotlar (AGV)** (ombor ichida yuk tashish uchun)

 Foyda:

- Inventarizatsiya avtomatik tarzda yuritiladi
- Xatoliklar kamayadi, ishchi kuchi ehtiyoji qisqaradi
- Tovarlar harorati va holati kuzatilib, sifati saqlanadi

3. Transport yo'llarini optimallashtirish (Fleet Optimization)

JAVOBLAR

Qanday ishlaydi:

IoT qurilmalari transport vositalarining yo'l holati, tezlik, yonilg'i sarfi va texnik holatini tahlil qilib, eng optimal marshrutlarni tanlashga yordam beradi.

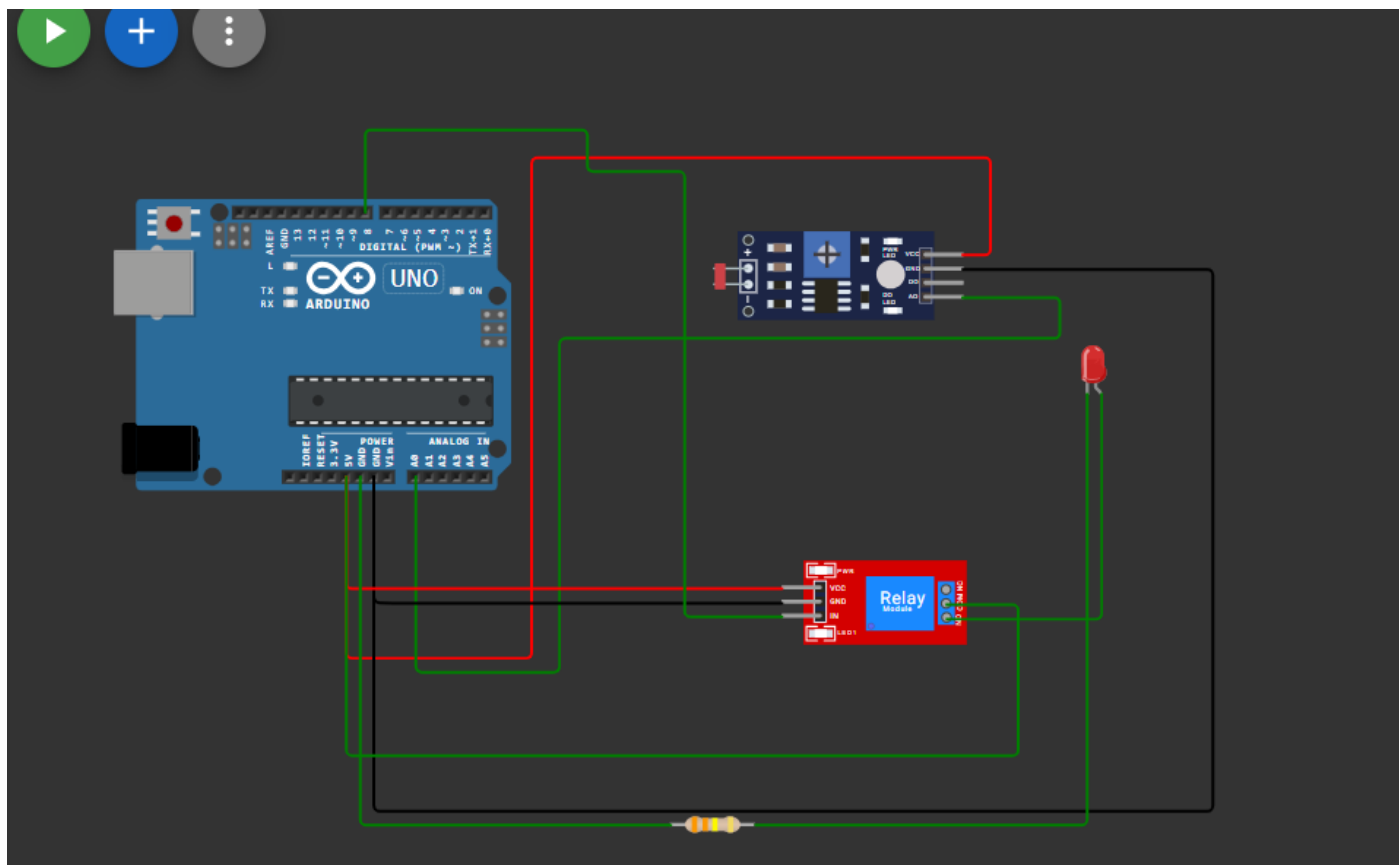
Ishlatiladigan qurilmalar:

- **OBD-II trekerlar** (avtomobil tizimlariga ulanadi)
- **GPS moduli + akselerometrlar**
- **Bulutga ulangan boshqaruv panellari**

Foyda:

- Yonilg'i tejaladi, transport xarajatlari kamayadi
- Mashina nosozliklari oldindan aniqlanadi
- Operatorlar marshrutlarni avtomatik belgilaydi

2. LDR fotorezistor va Rele orqali aqlli chiroq tizimi yaratish. LDR, Rele va LED yordamida arduinoda aqlli chiroq tizimini yarating, unda quyidagi shartlarni bajaring: - Chegara qiymatini 500 ga o'rnatish. - Qiymat 500 dan kamaysa LED chiroq o'chirilsin - Qiymat 500 va undan oshsa LED chiroq yoqilsin

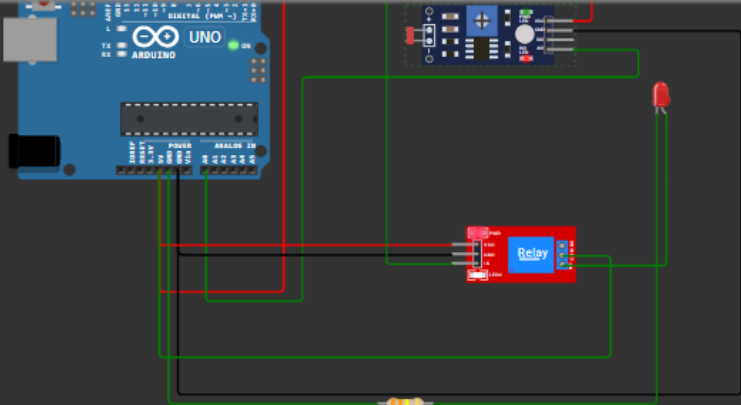


Simulation

00:22.681 102%

Photoresistor (LDR)

ILLUMINATION (LUX) 331 331 lux



LED O'CHDI

LDR qiymati: 308

LED O'CHDI

LDR qiymati: 308

LED O'CHDI

LDR qiymati: 308

LED O'CHDI

LED O'CHDI

Simulation

00:33.314 98%

Photoresistor (LDR)

ILLUMINATION 8 (UX)

8 lux

LED O'CHDI

LDR qiymati: 870

LED YONDI

LDR qiymati: 870

LED YONDI

LDR qiymati: 870

LED YONDI

Dastur kodi

```
#define LDR_PIN A0          // LDR ulangani analog pin
#define RELAY_PIN 8         // Rele IN pin
#define THRESHOLD 500      // Yorug'lik chegarasi

void setup() {
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW); // Dastlab chiroq o'chirilgan
  Serial.begin(9600);
}

void loop() {
  int ldrValue = analogRead(LDR_PIN);
  Serial.print("LDR qiymati: ");
  Serial.println(ldrValue);

  if (ldrValue >= THRESHOLD) {
```

```

    digitalWrite(RELAY_PIN, HIGH); // LED yoqiladi
    Serial.println("LED YONDI");
  } else {
    digitalWrite(RELAY_PIN, LOW); // LED o'chiriladi
    Serial.println("LED O'CHDI");
  }

  delay(1000); // 1 soniya kutish
}

```

Diagram.json

```

{
  "version": 1,
  "author": "ww",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-arduino-uno", "id": "uno", "top": 0, "left": 0, "attrs": {} },
    {
      "type": "wokwi-photoresistor-sensor",
      "id": "ldr1",
      "top": 22.4,
      "left": 413.6,
      "attrs": {}
    },
    { "type": "wokwi-relay-module", "id": "relay1", "top": 249.8, "left": 460.8, "attrs": {} },
    { "type": "wokwi-led", "id": "led1", "top": 92.4, "left": 656.6, "attrs": { "color": "red" } },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": 426.35,
      "left": 384,
      "attrs": { "value": "330000" }
    }
  ],
  "connections": [
    [ "ldr1:VCC", "uno:5V", "red", [ "h19.2", "v-67.2", "h-316.8", "v345.6", "h-124.8" ] ],
    [ "ldr1:A0", "uno:A0", "green", [ "h67.2", "v28.1", "h-345.6", "v230.4", "h-86.4" ] ],
    [ "ldr1:GND", "uno:GND.3", "black", [ "h172.8", "v374", "h-576" ] ],
    [ "relay1:GND", "uno:GND.3", "black", [ "h-268.8", "v-0.4", "h-19.2" ] ],
    [ "relay1:VCC", "uno:5V", "red", [ "h-288" ] ],
    [ "relay1:IN", "uno:8", "green", [ "h-67.2", "v-307.4", "h-67.2", "v-28.8", "h-153.6" ] ],
    [ "relay1:COM", "uno:5V", "green", [ "h49.2", "v104.2", "h-259.2" ] ],
    [ "relay1:NO", "led1:A", "green", [ "h0" ] ],
  ]
}

```

```
[ "r1:2", "led1:C", "green", [ "v0", "h238.8" ] ],
[ "r1:1", "uno:GND.2", "green", [ "v0", "h-211.2" ] ]
],
"dependencies": {}
}
```

3. ESP32 dan foydalangan holda MQTT brokerga xabar yuborish (MQTT broker: broker.hivemq.com) ESP32 va HC-SR04 sensori va MQTT broker orqali quyidagi shartlarni bajaruvchi loyiha tuzing: - ESP32 WiFi tarmog'iga ulansin hamda, Serial monitorga "WiFiga ulandi" yozuvi chiqarilsin. - ESP32 MQTT tizimiga ulansin hamda, Serial monitorga "MQTT brokerga ulandi" yozuvi chiqarilsin. - Ulanish amalga oshirilgach HC-SR04 sensori orqali olingan masofa qiymatini MQTT brokerga yuboring: "Yuborildi -> Qiymat - 150sm"

Simulation

00:22.713 47%

Editing Ultrasonic Distance Sensor
Distance: 233cm

```
.....
WiFiga ulandi
IP manzil: 10.10.0.2
MQTT brokerga ulanmoqda... MQTT brokerga ulandi
Yuborildi -> Qiymat - 399sm
Yuborildi -> Qiymat - 226sm
Yuborildi -> Qiymat - 232sm
```

```
#include <WiFi.h>
#include <PubSubClient.h>

// WiFi sozlamalari
const char* ssid = "Wokwi-GUEST";
const char* password = "";

// MQTT broker manzili
const char* mqtt_server = "broker.hivemq.com";

// HC-SR04 pinlari
const int trigPin = 5;
const int echoPin = 18;

// Ob'ektlar
WiFiClient espClient;
PubSubClient client(espClient);

// So'nggi yuborilgan vaqt
unsigned long lastMsg = 0;
const long interval = 5000; // 5 soniya

// WiFi ga ulanish funksiyasi
void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("WiFiga ulanmoqda: ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFiga ulandi");
    Serial.print("IP manzil: ");
    Serial.println(WiFi.localIP());
}

// MQTT brokerga qayta ulanish funksiyasi
void reconnect() {
    while (!client.connected()) {
        Serial.print("MQTT brokerga ulanmoqda...");
        String clientId = "ESP32Client-" + String(random(0xffff), HEX);
        if (client.connect(clientId.c_str())) {
            Serial.println(" MQTT brokerga ulandi");
        }
    }
}
```

```

    } else {
        Serial.print(" Xatolik. Kod: ");
        Serial.print(client.state());
        Serial.println(" 5 soniyadan so'ng qayta urinish...");
        delay(5000);
    }
}
}

// Masofa o'lchash funksiyasi
long getDistanceCM() {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    long duration = pulseIn(echoPin, HIGH);
    long distance = duration * 0.034 / 2;
    return distance;
}

void setup() {
    Serial.begin(115200);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    unsigned long now = millis();
    if (now - lastMsg > interval) {
        lastMsg = now;

        long distance = getDistanceCM();
        char msg[50];
        sprintf(msg, "Yuborildi -> Qiymat - %ldsm", distance);
        client.publish("iot/distance", msg);
        Serial.println(msg);
    }
}

```

Diagram.json

```
{
  "version": 1,
  "author": "ww",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": {} },
    { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -36.9, "left": -224.9, "attrs": {} }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "ultrasonic1:VCC", "esp:3V3", "red", [ "v48", "h124.8", "v-48" ] ],
    [ "ultrasonic1:GND", "esp:GND.2", "black", [ "v9.6", "h85.2", "v-86.4", "h134.4" ] ],
    [ "ultrasonic1:TRIG", "esp:5", "green", [ "v163.2", "h306.8", "v-76.8" ] ],
    [ "esp:18", "ultrasonic1:ECHO", "green", [ "h24.04", "v38.4", "h-201.6", "v-67.2", "h-48" ] ]
  ],
  "dependencies": {}
}
```

5-variant

1. IoT atrof-muhit monitoringida qanday ishlatiladi? IoT havo sifati, suv resurslari va o'rmon yong'inlarini kuzatishda qanday yordam beradi? Uchta misol keltiring va har birida qanday sensorlar ishlatilishini tushuntiring.

IoT (Internet of Things) atrof-muhit monitoringida turli sensorlar va qurilmalar yordamida real vaqtli ma'lumotlarni yig'ish, tahlil qilish va masofadan kuzatishni ta'minlaydi. Bu texnologiya ekologik muammolarni erta aniqlash va ularga tezkor javob berishda katta yordam beradi.

Quyida IoT texnologiyasi yordamida **atrof-muhit monitoringi** qanday amalga oshirilishi bo'yicha 3 ta **misol** keltirilgan:

1. Havo sifati monitoringi

 *Tavsif:*

Shahar yoki sanoat hududlarida havodagi zararli gazlar va zarrachalar (PM2.5, CO₂, NO₂) ni o'lchash uchun IoT tizimi ishlatiladi.

 *Ishlaydigan sensorlar:*

- **MQ135** – CO₂, NH₃, benzen kabi gazlarni aniqlaydi

JAVOBLAR

- **PM2.5 sensor (Plantower PMS5003)** – mayda chang zarrachalarini aniqlaydi
- **DHT22** – harorat va namlikni o'lchash uchun

Foyda:

- Havoning sifati yomonlashsa avtomatik ogohlantirish yuboriladi
 - Fuqarolarga real vaqtli ma'lumot taqdim qilinadi (masalan, mobil ilova orqali)
 - Hukumat ekologik choralarni o'z vaqtida ko'rishi mumkin
-

2. Suv resurslarini monitoring qilish

Tavsif:

Daryolar, suv omborlari yoki quduqlarda suv sathi, ifloslanish darajasi va haroratni kuzatish uchun IoT tizimlar ishlatiladi.

Ishlaydigan sensorlar:

- **Ultrasonik sensor (HC-SR04 yoki JSN-SR04T)** – suv sathini aniqlash
- **TDS sensor** – suvning tozaligi va minerallasuv darajasini aniqlash
- **DS18B20** – suv haroratini aniqlash

Foyda:

- Suv sathi pasayishi yoki toshqin xavfi oldindan aniqlanadi
 - Suv sifati nazorat ostida bo'ladi (ichimlik suvi xavfsizligi uchun)
 - Qishloq xo'jaligi va sanoat uchun suvdan samarali foydalanish
-

3. O'rmon yong'inlarini erta aniqlash

Tavsif:

O'rmonlar yoki tog'li hududlarda yong'in xavfini oldindan aniqlash va erta ogohlantirishlar berish uchun IoT asosidagi yong'in kuzatuv tizimlari ishlatiladi.

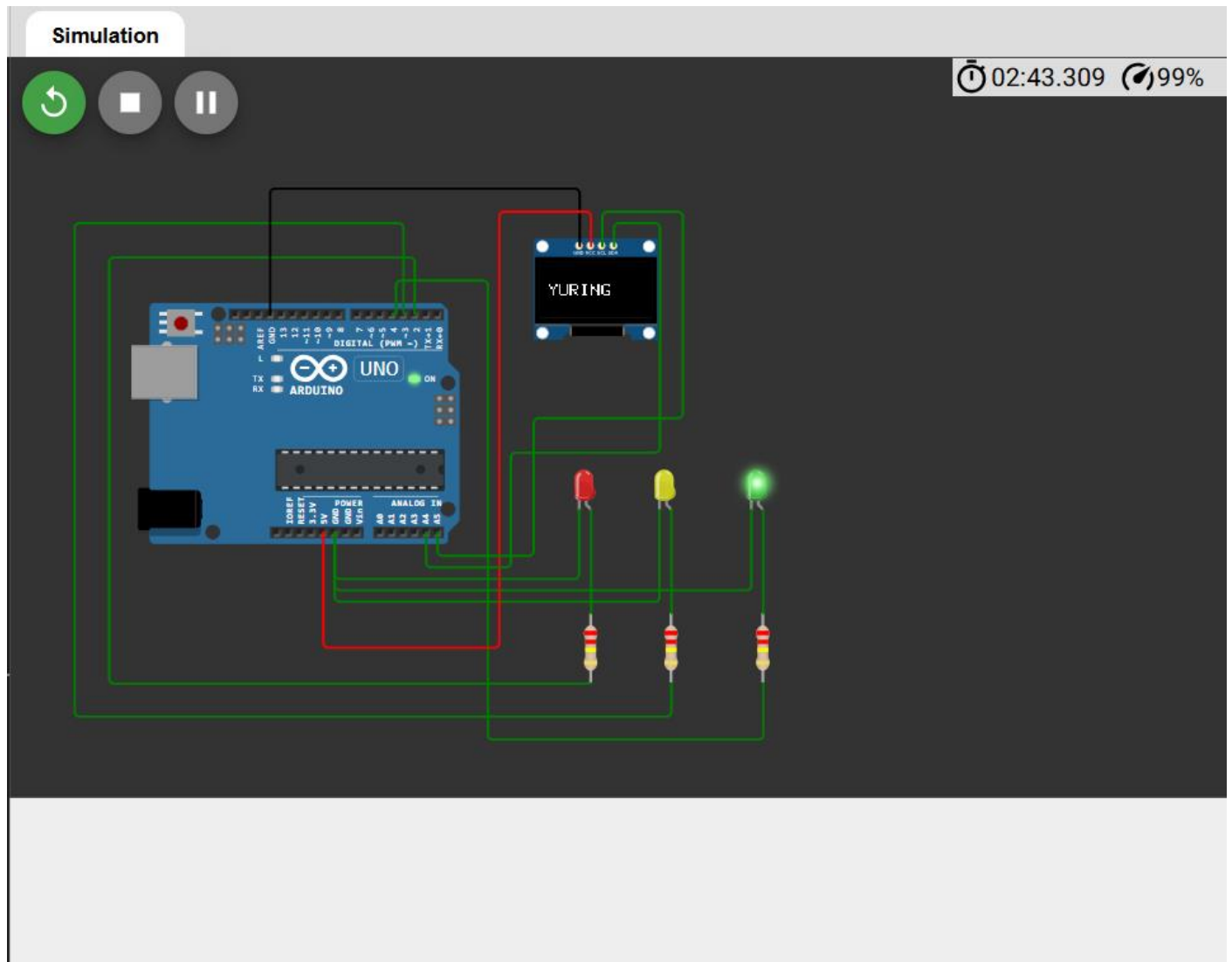
Ishlaydigan sensorlar:

- **Flame sensor / UV sensor** – olov yoki olovga xos nur to'lqinini aniqlash
- **MQ-2 yoki MQ-6** – tutun va yonuvchan gazlarni aniqlash
- **DHT22 / DS18B20** – haroratni o'lchash

Foyda:

- Yong'in chiqishidan oldin aniqlab, o'rmonni asrash
- Favqulodda vaziyatlar xizmatlariga avtomatik signal yuborish
- Ekotizimni saqlash va moliyaviy zararlarni kamaytirish

2. OLED display va LED orqali aqlli transport harakati tizimini yaratish. OLED display va LED chiroqlar orqali aqlli transport tizimini loyihalashtiring - Yashil chiroq yonganda OLEDda “YURING” yozuvi aks etsin. - Sariq chiroq yonganda OLEDda “KUTING” yozuvi aks etsin. - Yashil chiroq yonganda OLEDda “TO’XTANG” yozuvi aks etsin.



Dastur kodi

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// LED pinlari
const int redLED = 2;
const int yellowLED = 3;
const int greenLED = 4;
```

```

void setup() {
  // OLED boshlash
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);

  // LED pinlar chiqish sifatida
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
}

void loop() {
  // Yashil LED – YURING
  digitalWrite(greenLED, HIGH);
  digitalWrite(yellowLED, LOW);
  digitalWrite(redLED, LOW);
  showMessage("YURING");
  delay(4000);

  // Sariq LED – KUTING
  digitalWrite(greenLED, LOW);
  digitalWrite(yellowLED, HIGH);
  digitalWrite(redLED, LOW);
  showMessage("KUTING");
  delay(2000);

  // Qizil LED – TO‘XTANG
  digitalWrite(greenLED, LOW);
  digitalWrite(yellowLED, LOW);
  digitalWrite(redLED, HIGH);
  showMessage("TO‘XTANG");
  delay(4000);
}

void showMessage(String message) {
  display.clearDisplay();
  display.setCursor(10, 25);
  display.print(message);
  display.display();
}

```

Diagram.json

```

{
  "version": 1,

```

```

"author": "ww",
"editor": "wokwi",
"parts": [
  { "type": "wokwi-arduino-uno", "id": "uno", "top": 0, "left": 0, "attrs": {} },
  {
    "type": "board-ssd1306",
    "id": "oled1",
    "top": -54.46,
    "left": 336.23,
    "attrs": { "i2cAddress": "0x3c" }
  },
  {
    "type": "wokwi-led",
    "id": "led1",
    "top": 130.8,
    "left": 426.2,
    "attrs": { "color": "yellow" }
  },
  { "type": "wokwi-led", "id": "led2", "top": 130.8, "left": 359, "attrs": { "color":
"red" } },
  {
    "type": "wokwi-led",
    "id": "led3",
    "top": 130.8,
    "left": 503,
    "attrs": { "color": "green" }
  },
  {
    "type": "wokwi-resistor",
    "id": "r1",
    "top": 283.2,
    "left": 354.65,
    "rotate": 90,
    "attrs": { "value": "220000" }
  },
  {
    "type": "wokwi-resistor",
    "id": "r2",
    "top": 283.2,
    "left": 421.85,
    "rotate": 90,
    "attrs": { "value": "220000" }
  },
  {
    "type": "wokwi-resistor",
    "id": "r3",
    "top": 283.2,
    "left": 498.65,
    "rotate": 90,
    "attrs": { "value": "220000" }
  }
],

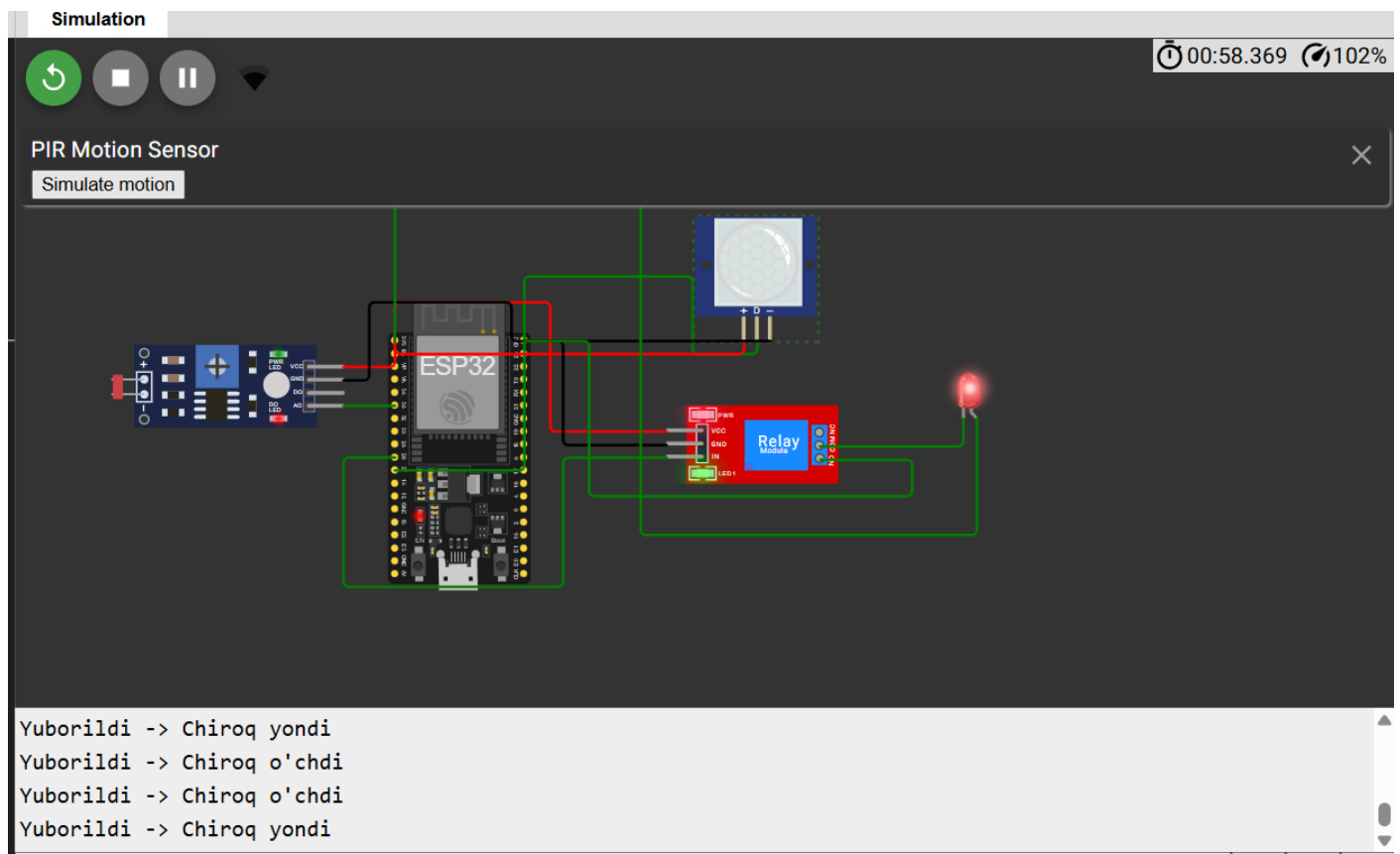
```

```

"connections": [
  [ "led2:C", "uno:GND.2", "green", [ "v57.6", "h-230" ] ],
  [ "led2:A", "r1:1", "green", [ "v0" ] ],
  [ "led1:C", "uno:GND.2", "green", [ "v76.8", "h-249.2" ] ],
  [ "led1:A", "r2:1", "green", [ "v0" ] ],
  [ "led3:C", "uno:GND.2", "green", [ "v67.2", "h-335.6" ] ],
  [ "led3:A", "r3:1", "green", [ "v0" ] ],
  [ "r1:2", "uno:2", "green", [ "h-403.2", "v-356.4", "h259.2" ] ],
  [ "r2:2", "uno:3", "green", [ "h0", "v27.6", "h-499.2", "v-412.8", "h259.2" ] ],
  [ "r3:2", "uno:4", "green", [ "h0", "v46.8", "h-230.4", "v-384", "h-76.8", "v19.2" ]
],
[ "oled1:GND", "uno:GND.1", "black", [ "v-48", "h-297.6" ] ],
[ "oled1:VCC", "uno:5V", "red", [ "v-28.8", "h-76.65", "v364.8", "h-144" ] ],
[
  "oled1:SCL",
  "uno:A5",
  "green",
  [ "v-28.8", "h67.5", "v172.8", "h-124.8", "v115.2", "h-96" ]
],
[ "oled1:SDA", "uno:A4", "green", [ "v-19.2", "h38.47", "v192", "h-124.8", "v96", "h-
67.2" ] ]
],
"dependencies": {}
}

```

3. ESP32 dan foydalangan holda MQTT brokerga xabar yuborish (MQTT broker: broker.hivemq.com) ESP32, LDR, RELE va MQTT broker orqali quyidagi shartlarni bajaruvchi loyiha tuzing: - ESP32 WiFi tarmog'iga ulansin hamda, Serial monitorga "WiFiga ulandi" yozuvi chiqarilsin. - ESP32 MQTT tizimiga ulansin hamda, Serial monitorga "MQTT brokerga ulandi" yozuvi chiqarilsin. - Ulanish amalga oshirilgach PIR sensori holatini MQTT brokerga yuboring: "Yuborildi -> Chiroq yondi" yoki "Yuborildi -> Chiroq o'chdi".



Dastur kodi

```
#include <WiFi.h>
#include <PubSubClient.h>

// Wi-Fi sozlamalari
const char* ssid = "Wokwi-GUEST";
const char* password = "";

// MQTT broker
const char* mqtt_server = "broker.hivemq.com";

// Pinlar
const int pirPin = 27;
const int ldrPin = 35; // A0
const int relayPin = 26;

// Chegara (LDR)
const int ldrThreshold = 500;

WiFiClient espClient;
PubSubClient client(espClient);

void setup_wifi() {
  delay(10);
```

```

Serial.begin(115200);
Serial.println();
Serial.print("WiFiga ulanmoqda: ");
Serial.println(ssid);

WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}

Serial.println();
Serial.println("WiFiga ulandi");
Serial.print("IP: ");
Serial.println(WiFi.localIP());
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("MQTT brokerga ulanmoqda...");
        String clientId = "ESP32Client-" + String(random(0xffff), HEX);
        if (client.connect(clientId.c_str())) {
            Serial.println("MQTT brokerga ulandi");
        } else {
            Serial.print("Xatolik. Kod: ");
            Serial.print(client.state());
            delay(5000);
        }
    }
}

void setup() {
    pinMode(pirPin, INPUT);
    pinMode(relayPin, OUTPUT);
    digitalWrite(relayPin, LOW);

    setup_wifi();
    client.setServer(mqtt_server, 1883);
}

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    int motion = digitalRead(pirPin);
    int ldrValue = analogRead(ldrPin);

    if (motion == HIGH && ldrValue < ldrThreshold) {
        digitalWrite(relayPin, HIGH); // Chiroq yoqilsin
        client.publish("esp32/chiroq", "Yuborildi -> Chiroq yondi");
    }
}

```

```

    Serial.println("Yuborildi -> Chiroq yondi");
  } else {
    digitalWrite(relayPin, LOW); // Chiroq o'chirilsin
    client.publish("esp32/chiroq", "Yuborildi -> Chiroq o'chdi");
    Serial.println("Yuborildi -> Chiroq o'chdi");
  }

  delay(2000); // 2 soniya kutish
}

```

Diagram.json

```

{
  "version": 1,
  "author": "wokwi",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": -19.2, "left": -4.76, "attrs": {} },
    {
      "type": "wokwi-photoresistor-sensor",
      "id": "ldr1",
      "top": 12.8,
      "left": -210.4,
      "attrs": {}
    },
    { "type": "wokwi-pir-motion-sensor", "id": "pir1", "top": -82.4, "left": 223.02, "attrs": {} },
    { "type": "wokwi-relay-module", "id": "relay1", "top": 57.8, "left": 201.6, "attrs": {} },
    { "type": "wokwi-led", "id": "led1", "top": 25.2, "left": 407, "attrs": { "color": "red" } }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "pir1:OUT", "esp:27", "green", [ "v9.6", "h-48.14", "v-57.6", "h-124.8", "v48" ] ],
    [ "pir1:VCC", "esp:3V3", "red", [ "v9.6", "h-124.8" ] ],
    [ "pir1:GND", "esp:GND.2", "black", [ "v0" ] ],
    [ "relay1:VCC", "esp:3V3", "red", [ "h-86.4", "v-96", "h-105.6" ] ],
    [ "relay1:GND", "esp:GND.2", "black", [ "h-76.8", "v-58" ] ],
    [ "relay1:IN", "esp:26", "green", [ "h-76.8", "v95.8", "h-163.2", "v-19.2" ] ],
    [ "ldr1:VCC", "esp:3V3", "red", [ "h0" ] ],
    [ "ldr1:GND", "esp:GND.2", "black", [ "h19.2", "v-58", "h105.6", "v9.6" ] ],
    [ "esp:35", "ldr1:AO", "green", [ "h0" ] ],
    [ "relay1:NO", "esp:GND.2", "green", [ "h68.4", "v27", "h-240", "v-96" ] ],
    [ "led1:C", "relay1:COM", "green", [ "v0" ] ],
    [ "led1:A", "esp:3V3", "green", [ "v86.4", "h-249.6", "v-259.2", "h-201.6" ] ]
  ],
  "dependencies": {}
}

```


JAVOBLAR

