Ministerul Educației și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Ingineria Software și Automatică

Lucrare de laborator Nr. 3

Disciplina: Internetul lucrurilor (IOT)
Tema: Achiziție și condiționare semnal

A efectuat: Popa Cătălin, gr. TI-211

A verificat: Cristian Lupan, asist. univ.

Chișinău 2024

2.1 Problema propusă:

Sa se realizeze o aplicație in baza de MCU care va prelua un semnal de la o sursa de semnal, si va afișa parametrul fizic la un terminal (LCD si/sau Serial). Fiecare student va selecta un senzor fie analogic fie digital (nu binar) din PDF atașat.

- Sa se achiziționeze semnalul de la senzor;
- Să se afișeze datele pe afișor LCD și / sau Serial.

2.2 Problema propusă:

- Sa se achiziționeze semnalul de la senzor (vezi Lab 3.1);
- Să se condiționeze semnalul implicând filtre digitale și alte metode;
- Să se afișeze datele pe afișor LCD și / sau Serial.

Obiective

1. Utilizarea senzorilor pentru a măsura parametrii de mediu (lumina și distanța).
2. Implementarea unui sistem de monitorizare care să capteze și să proceseze datele de la senzori în timp real.
3. Aplicarea unor tehnici de filtrare digitală (de exemplu, filtrul mediu) pentru a îmbunătăți calitatea semnalului și a reduce zgomotul în datele citite de la senzorii de lumină.
4. Calcularea distanței cu senzorul ultrasonic și interpretarea corectă a valorilor obținute.
5. Implementarea unui sistem de afișare a datelor pe monitorul serial pentru a permite utilizatorului să vizualizeze în timp real informațiile captate de senzori.

Introducere

Microcontrolerele permit controlul și monitorizarea dispozitivelor fizice, aducând tehnologia mai aproape de utilizator prin intermediul unei interfețe intuitive și simplificate. Un exemplu clasic de interacțiune între utilizator și dispozitive fizice este controlul unui LED prin apăsarea unui buton sau trimiterea de comenzi printr-o interfață serială. Această tehnologie este utilizată într-o gamă largă de aplicații, de la sisteme de iluminat inteligente până la dispozitive electronice complexe utilizate în automatizări industriale. Unul dintre avantajele microcontrolerelor este costul redus și flexibilitatea lor, permițând implementarea rapidă de prototipuri și teste pentru diverse aplicații IoT. Microcontrolerele precum cele din familia Arduino sau ESP sunt folosite frecvent pentru prototiparea sistemelor IoT datorită ecosistemului extins de biblioteci și comunități de dezvoltatori care susțin aceste platforme .

Materiale și metode

Arduino UNO - platformă de microcontroler open-source bazată pe microcontrolerul ATmega328P. Este ușor de utilizat și permite conectarea ușoară a componentelor externe prin pini digitali și analogici. Acesta este ideal pentru prototiparea de sisteme IoT și proiecte electronice interactive. Afișarea este în figura 1.1.

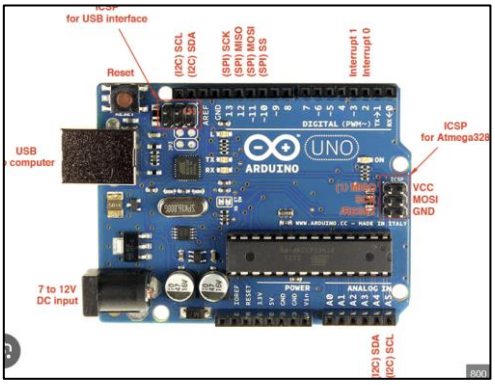


Fig. 1.1. Arduino UNO

- 14 pini digitali I/O (din care 6 sunt PWM);
- 6 intrări analogice;
- memorie flash de 32 KB;
- tensiune de operare: 5V;
- interfață de programare: Arduino IDE.

LED-ul - (diodă emițătoare de lumină) este utilizată pentru a semnaliza starea sistemului în funcție de comenzile primite sau de validitatea codului introdus pe tastatură. Afișarea este în figura 1.2.

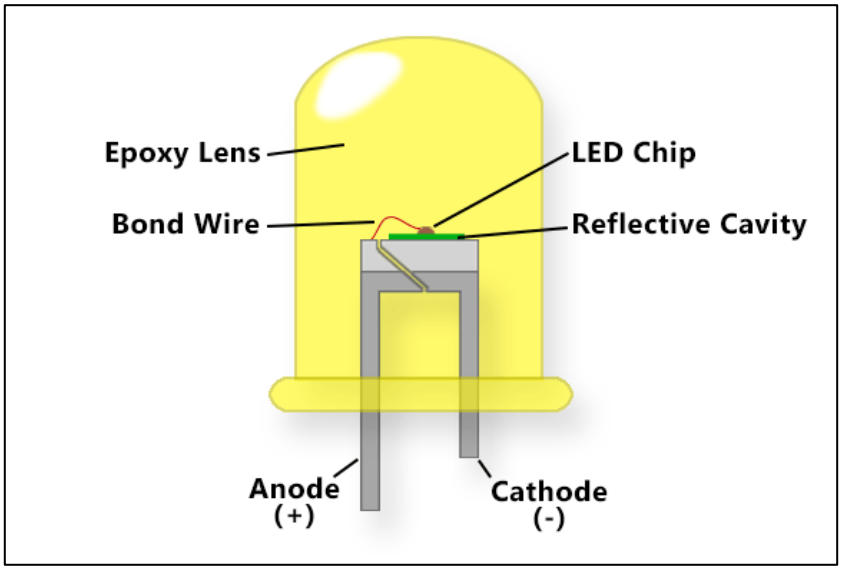


Fig. 1.2. LED

- tensiune de operare: 2V-3V;
- curent: ~20mA;
- LED verde pentru semnalizarea codurilor valide;
- LED roșu pentru semnalizarea codurilor invalide.

Rezistor - Rezistorul este utilizat pentru a limita curentul ce trece prin LED, prevenind arderea acestuia. Pentru a proteja LED-urile, se vor folosi rezistoare de aproximativ 220Ω. Afișarea este în figura 1.3.

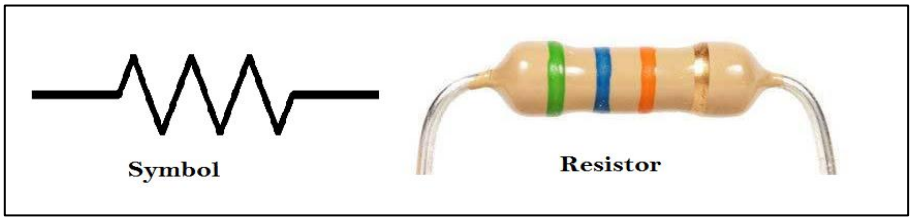


Fig. 1.3. Rezistor

Fire Jumper - Firele jumper sunt utilizate pentru a face legături electrice între diverse componente și plăcuța Arduino. Acestea permit conectarea componentelor fără a fi necesară lipirea acestora. Afișarea este în figura 1.6.

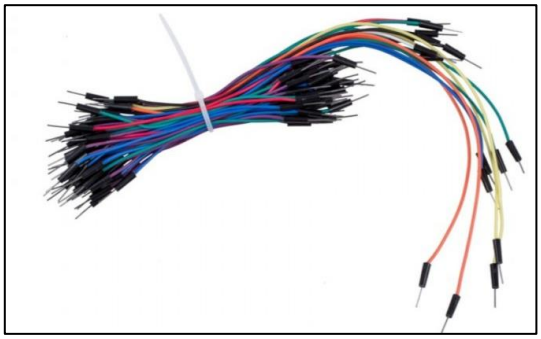


Fig. 1.6. Fire Jumper

Buton - În aplicații de acest gen, un buton (push button) este utilizat pentru a oferi utilizatorului o modalitate de a interacționa cu sistemul printr-un eveniment de apăsare. În această situație, ar putea fi utilizat pentru a reseta sistemul sau pentru a iniția o acțiune nouă. Afișarea este în figura 1.7.

Fig. 1.7. Buton

Metode

Configurarea Arduino UNO: Arduino va fi configurat folosind Arduino IDE. Se va implementa codul pentru a citi inputurile de la tastatura 4x4 și pentru a controla LED-urile și afișarea pe LCD.

Interacțiunea cu utilizatorul: Sistemul va primi comenzi fie prin terminalul serial, fie prin tastatura matricială, și va răspunde prin intermediul LCD-ului și al LED-urilor. Comenzile "led on" și "led off" vor aprinde și stinge LED-ul respectiv, iar codurile introduse vor fi validate, aprinzând LED-ul verde pentru coduri corecte și LED-ul roșu pentru coduri greșite.

Validare în simulator: Codul și conexiunile vor fi testate inițial într-un simulator de circuite, cum ar fi Tinkercad sau Proteus, pentru a verifica funcționalitatea corectă. După simulare, se va testa și pe un circuit real.

2.1

Pentru prima sarcină am avut nevoie de 3 rezistoare, 2 leduri un fotorezistor și pentru sarcina individuală am luat inca un senzor ultrasonic. Mai întâi am conectat toate componentele necesare, apoi am trecut la partea de cod. Componentele conectate sunt reprezentate în figura 1.8.

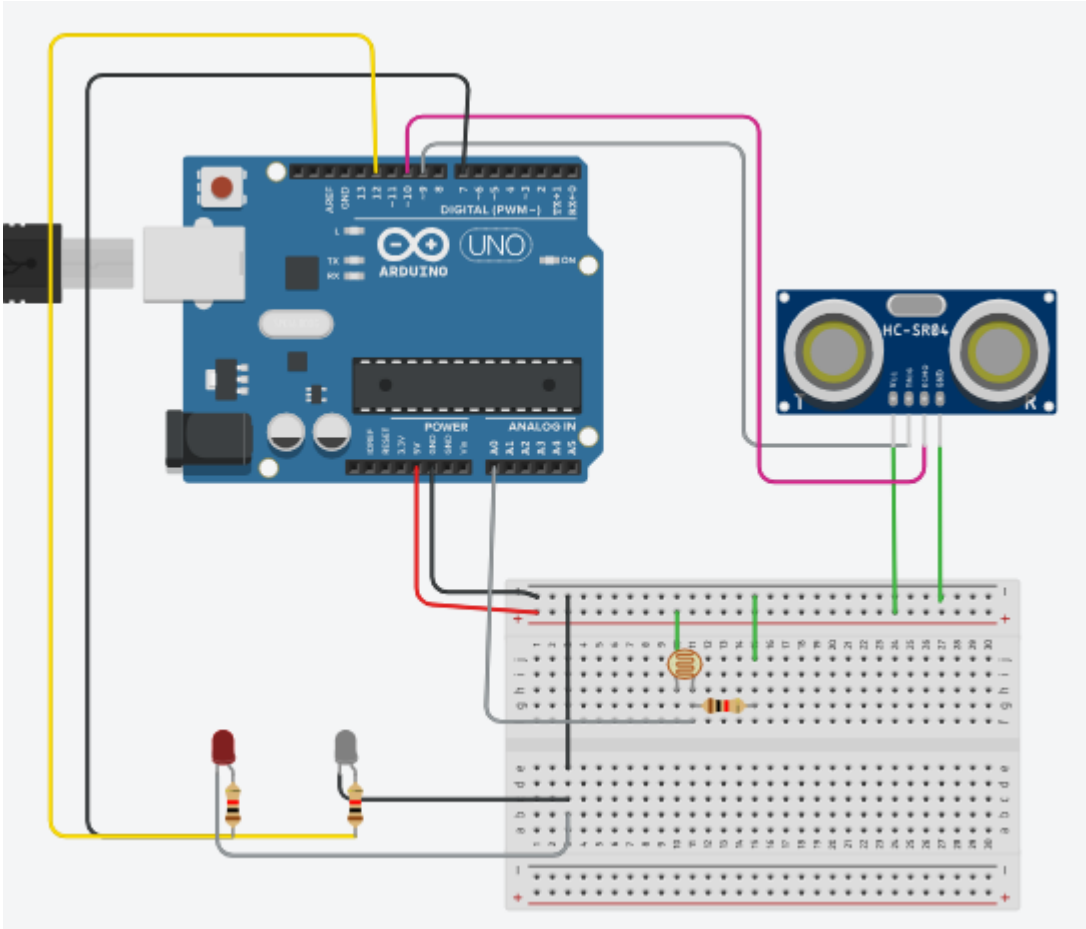


Figura 1.8 – Simulare microcontroler

Mai întâi am pornit o comunicare serială pentru a monitoriza datele. Apoi am setat senzorul ultrasonic și pinii pentru LED-uri.

```
const int ldrPin = A0;
void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledDistancePin, OUTPUT);
  pinMode(ledPin12, OUTPUT);
}
```

În continuare, în loop am măsurat distanța cu HC-SR04, apoi am convertit in cm.

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

durata = pulseIn(echoPin, HIGH);
distanta = durata * 0.034 / 2;
```

Am afișat distanța în serial monitor.

```
Serial.print("Distanta: ");
Serial.print(distanta);
Serial.println(" cm");
```

La fel am făcut și pentru fotorezistor, mai întâi citim datele.

```
valoareLumina = analogRead(ldrPin);
 Serial.print("Valoarea luminii: ");
 Serial.println(valoareLumina);
```

Pentru a controla LED-ul de distanță pe bază distanței, am folosit if și else. Atunci când distanța este sub 20 cm LED-ul va începe a clipi rapid. Dacă distanța este mai mare de 20 cm, viteza de clipire va încetina pe baza distanței.

```
int delayTime = map(distanta, 20, 200, 100, 1000);
    delayTime = constrain(delayTime, 100, 1000);
    digitalWrite(ledDistancePin, HIGH);
    delay(delayTime);
    digitalWrite(ledDistancePin, LOW);
    delay(delayTime);
```

Pe lângă aceasta, al doilea LED se va aprinde atunci când distanța este mai mică de 50 și valorea luminii este mai mică de 300.

cm

```
if (distanta < 50 && valoareLumina < 300) {
  digitalWrite(ledPin12, HIGH);
} else {
  digitalWrite(ledPin12, LOW);
}
```

Pentru realizarea punctului 2, am fost nevoit să condiționăm semnalul. Am aplicat filtre cu media, atât pe fotorezistor, cât și pentru ultrasonic. Aceste filtre ajută la obținerea unor măsurări mai stabile, eliminând eventuale valori aberante datorate zgomotului.

Mai întâi se ia valoarea veche din array. După se citește valoare nouă și o adăugăm în listă și avansăm indexul. După calculăm media și afișăm datele in serial monitor. La fel la final calculăm tensiunea și o afișăm.

```
total = total - readings[readIndex];
  readings[readIndex] = analogRead(ldrPin);
  total = total + readings[readIndex];
  readIndex = (readIndex + 1) % numReadings;
  average = total / numReadingsș

  Serial.print("Valoarea medie a luminii (ADC): ");
  Serial.println(average);

  float voltage = (average / 1023.0) * 5.0;
  Serial.print("Tensiune (V): ");
  Serial.println(voltage, 4);
```

Pentru ulstrasonic mai întâi am aplicat calcularea distanței în cm, apoi deja am aplica filtru mediu. Principiul de calculare a medie ieste la fel ca și la fotoresistor.

```
durata = pulseIn(echoPin, HIGH);
  distanta = durata * 0.034 / 2;

  ultrasonicTotal = ultrasonicTotal - ultrasonicReadings[ultrasonicIndex];
  ultrasonicReadings[ultrasonicIndex] = distanta;  // Citim noua distanță
  ultrasonicTotal = ultrasonicTotal + ultrasonicReadings[ultrasonicIndex];
  ultrasonicIndex = (ultrasonicIndex + 1) % numReadingsș
  ultrasonicAverage = ultrasonicTotal / numReadings;

  Serial.print("Distanta medie: ");
  Serial.print(ultrasonicAverage);
  Serial.println(" cm");
```

Concluzii

Acest proiect demonstrează cum se pot utiliza senzori simpli, precum fotorezistorul și senzorul ultrasonic, pentru a prelua și condiționa date de mediu, aplicând filtre digitale pentru a obține măsurători precise și stabile. Valorile mediate sunt afișate pe monitorul serial, iar datele procesate sunt folosite pentru a controla LED-uri, indicând atât intensitatea luminii cât și distanța față de obstacole, în funcție de valorile citite de senzori. Acest tip de implementare poate fi extins și adaptat pentru diverse aplicații de monitorizare și automatizare.

Anexă

3.1

```
const int trigPin = 9;
const int echoPin = 10;

const int ldrPin = A0;

const int ledDistancePin = 7;

const int ledPin12 = 12;

const int numReadings = 14;
int readings[numReadings];
int readIndex = 0;
int total = 0;
int average = 0;

int ultrasonicReadings[numReadings];
int ultrasonicIndex = 0;
long durata;
int distanta;

void setup() {
  Serial.begin(9600);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
```

```
  pinMode(ledDistancePin, OUTPUT);
  pinMode(ledPin12, OUTPUT);

  for (int i = 0; i < numReadings; i++) {
    readings[i] = 0;
    ultrasonicReadings[i] = 0;
  }
}

void loop() {

  total = total - readings[readIndex];
  readings[readIndex] = analogRead(ldrPin);
  total = total + readings[readIndex];
  readIndex = (readIndex + 1) % numReadings;

  int recentReadings[9];
  for (int i = 0; i < 9; i++) {
    recentReadings[i] = readings[(readIndex - 1 - i + numReadings) % numReadings];
  }

  for (int i = 0; i < 8; i++) {
    for (int j = i + 1; j < 9; j++) {
      if (recentReadings[i] > recentReadings[j]) {
        int temp = recentReadings[i];
        recentReadings[i] = recentReadings[j];
```

```
      recentReadings[j] = temp;
    }
  }
}

int median = recentReadings[4];
float ponderi[] = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9};
float ponderata = 0.0;
float sumaPonderi = 0.0;

for (int i = 0; i < 9; i++) {
  ponderata += recentReadings[i] * ponderi[i];
  sumaPonderi += ponderi[i];
}
ponderata = ponderata / sumaPonderi;

Serial.print("Valoarea mediană a luminii (ADC): ");
Serial.println(median);

Serial.print("Valoarea ponderată a luminii (ADC): ");
Serial.println(ponderata);

float voltage = (median / 1023.0) * 5.0;
Serial.print("Tensiune (V): ");
Serial.println(voltage, 4);
```

```
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  durata = pulseIn(echoPin, HIGH);
  distanta = durata * 0.034 / 2;



  //filtru median

  ultrasonicReadings[ultrasonicIndex] = distanta;
  ultrasonicIndex = (ultrasonicIndex + 1) % numReadings;

  int recentUltrasonic[9];
  for (int i = 0; i < 9; i++) {
    recentUltrasonic[i] = ultrasonicReadings[(ultrasonicIndex - 1 - i + numReadings) %
numReadings];
  }

  for (int i = 0; i < 8; i++) {
    for (int j = i + 1; j < 9; j++) {
      if (recentUltrasonic[i] > recentUltrasonic[j]) {
        int temp = recentUltrasonic[i];
        recentUltrasonic[i] = recentUltrasonic[j];
        recentUltrasonic[j] = temp;
      }
```

```
  }
}

int ultrasonicMedian = recentUltrasonic[4];

    //mediana ponderata
float ponderataUltrasonic = 0.0;
sumaPonderi = 0.0;

for (int i = 0; i < 9; i++) {
  ponderataUltrasonic += recentUltrasonic[i] * ponderi[i];
  sumaPonderi += ponderi[i];
}
ponderataUltrasonic = ponderataUltrasonic / sumaPonderi;

Serial.print("Distanta mediană: ");
Serial.print(ultrasonicMedian);
Serial.println(" cm");

Serial.print("Distanta ponderată: ");
Serial.print(ponderataUltrasonic);
Serial.println(" cm");

if (ultrasonicMedian < 20) {
  digitalWrite(ledDistancePin, HIGH);
  delay(100);
  digitalWrite(ledDistancePin, LOW);
```

```
    delay(100);
  } else {
    int delayTime = map(ultrasonicMedian, 20, 200, 100, 1000);
    delayTime = constrain(delayTime, 100, 1000);
    digitalWrite(ledDistancePin, HIGH);
    delay(delayTime);
    digitalWrite(ledDistancePin, LOW);
    delay(delayTime);
  }

  if (ultrasonicMedian < 50 && median < 300) {
    digitalWrite(ledPin12, HIGH);
  } else {
    digitalWrite(ledPin12, LOW);
  }

  delay(1000);
}
```

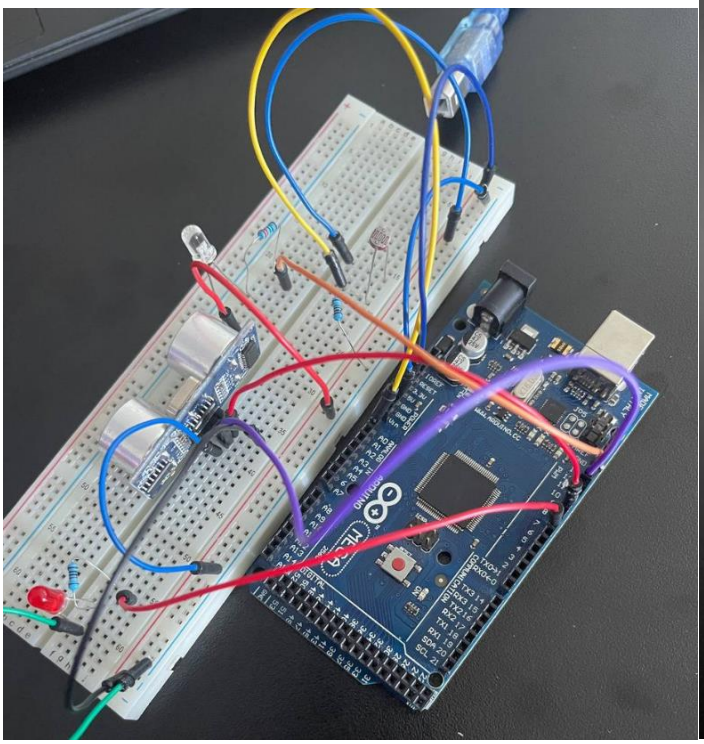
3.2

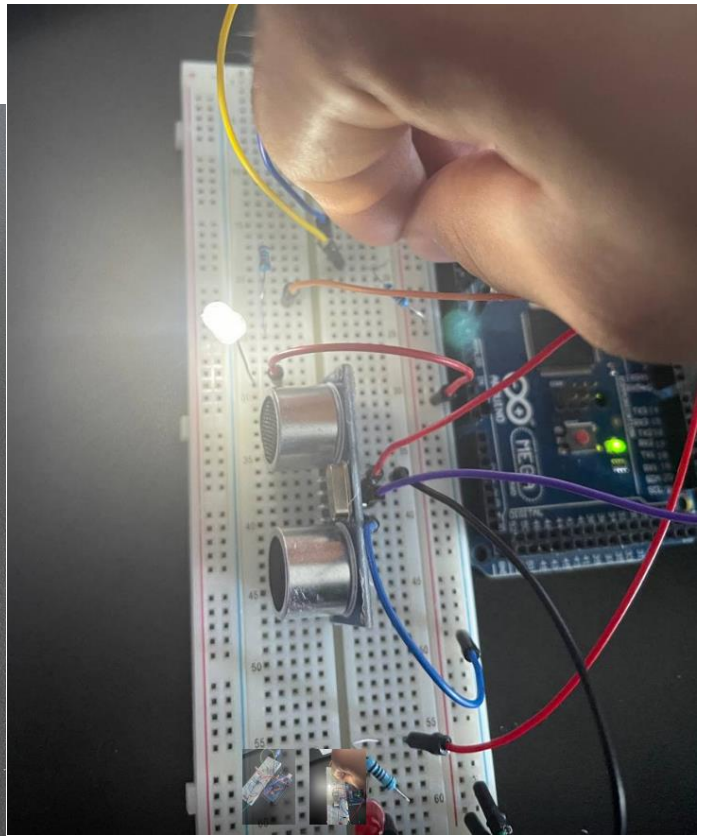
```
const int trigPin = 9;
const int echoPin = 10;

const int ldrPin = A0;

const int ledDistancePin = 7;

const int ledPin12 = 12;

const int numReadings = 10;
```

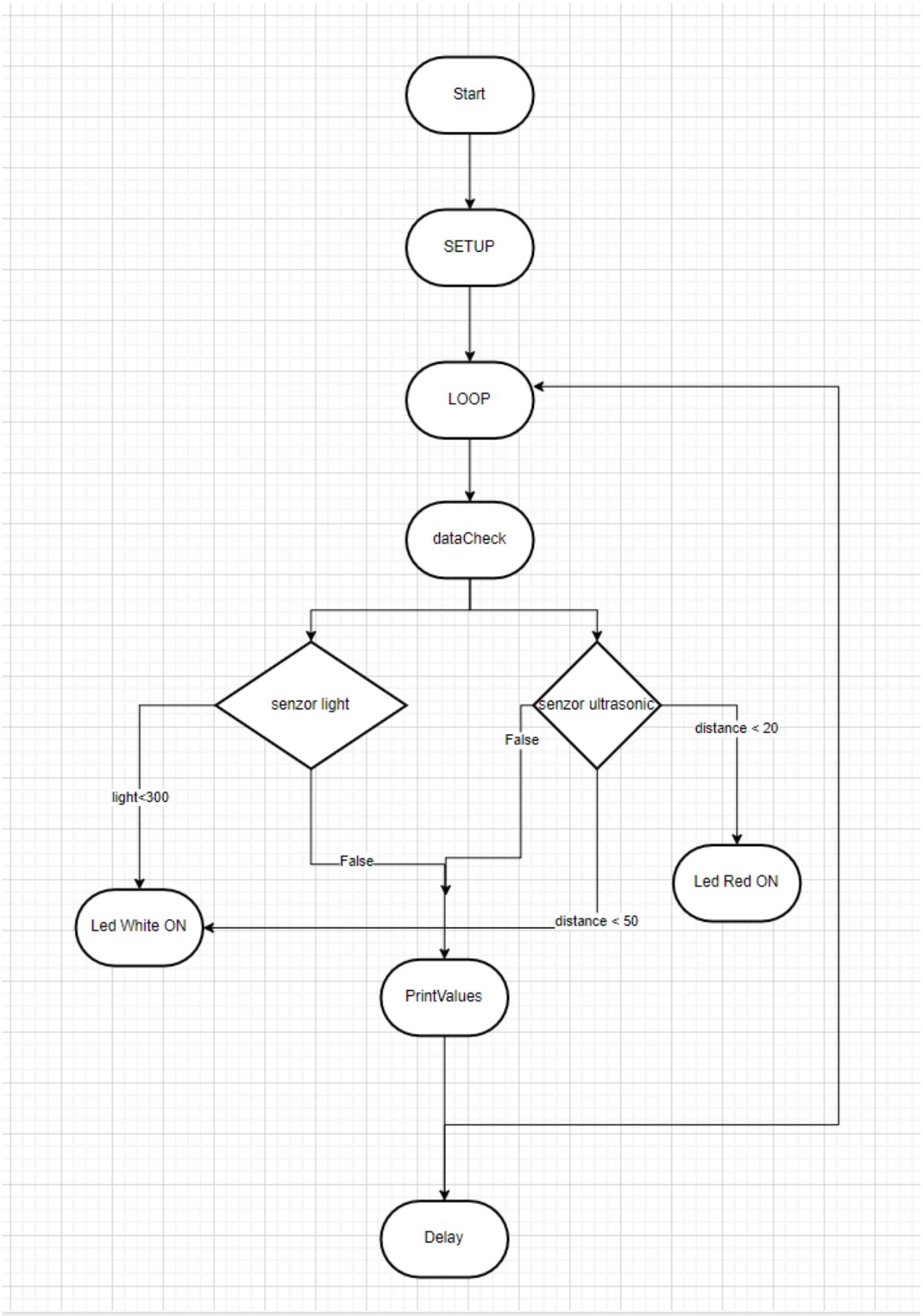
```
int readings[numReadings];
int readIndex = 0;
int total = 0;
int average = 0;

int ultrasonicReadings[numReadings];
int ultrasonicTotal = 0;
int ultrasonicIndex = 0;
int ultrasonicAverage = 0;

long durata;
int distanta;

void setup() {
  Serial.begin(9600);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(ledDistancePin, OUTPUT);
  pinMode(ledPin12, OUTPUT);

  for (int i = 0; i < numReadings; i++) {
    readings[i] = 0;
    ultrasonicReadings[i] = 0;
  }
}

void loop() {
  total = total - readings[readIndex];
```

```
readings[readIndex] = analogRead(ldrPin);
total = total + readings[readIndex];
readIndex = (readIndex + 1) % numReadings;
average = total / numReadings;

Serial.print("Valoarea medie a luminii (ADC): ");
Serial.println(average);

float voltage = (average / 1023.0) * 5.0;
Serial.print("Tensiune (V): ");
Serial.println(voltage, 4);

digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

durata = pulseIn(echoPin, HIGH);
distanta = durata * 0.034 / 2;

ultrasonicTotal = ultrasonicTotal - ultrasonicReadings[ultrasonicIndex];
ultrasonicReadings[ultrasonicIndex] = distanta;
ultrasonicTotal = ultrasonicTotal + ultrasonicReadings[ultrasonicIndex];
ultrasonicIndex = (ultrasonicIndex + 1) % numReadings;
ultrasonicAverage = ultrasonicTotal / numReadings;

Serial.print("Distanta medie: ");
Serial.print(ultrasonicAverage);
Serial.println(" cm");
```

```
  if (ultrasonicAverage < 20) {
    digitalWrite(ledDistancePin, HIGH);
    delay(100);
    digitalWrite(ledDistancePin, LOW);
    delay(100);
  } else {
    int delayTime = map(ultrasonicAverage, 20, 200, 100, 1000);
    delayTime = constrain(delayTime, 100, 1000);
    digitalWrite(ledDistancePin, HIGH);
    delay(delayTime);
    digitalWrite(ledDistancePin, LOW);
    delay(delayTime);
  }

  if (ultrasonicAverage < 50 && average < 300) {
    digitalWrite(ledPin12, HIGH);
  } else {
    digitalWrite(ledPin12, LOW);
  }

  delay(1000);
}
```






Start
SETUP
LOOP
dataCheck
senzor light
senzor ultrasonic
light<300
False
False
distance < 20
Led Red ON
Led White ON
distance < 50
PrintValues
Delay