შეხვედრა 8: მანძილის განმსაზღვრელი სენსორი – შენი პირადი „პარკინგ-კონტროლი“

გამარჯობა! ჩვენ უკვე ვიცით, თუ როგორ რეაგირებს არდუინო სინათლესა და ტემპერატურაზე. დღეს ჩვენ მას „შევასწავლით“, როგორ „დაინახოს“ დაბრკოლებები და გაზომოს მანძილი დაბრკოლებამდე. ამისთვისგამოვიყენებთ ულტრაბგერით მანძილის განმსაზღვრელ სენსორს და ავაწყობთ ჩვენს პირად „პარკინგის ასისტენტს“.

1. ულტრაბგერითი სენსორის მუშაობის პრინციპი

ჩვენი დღევანდელი გმირია HC-SR04 ულტრაბგერითი სენსორი. ის ჰგავს მცირე ზომის რობოტს ორი დიდი „თვალით“.

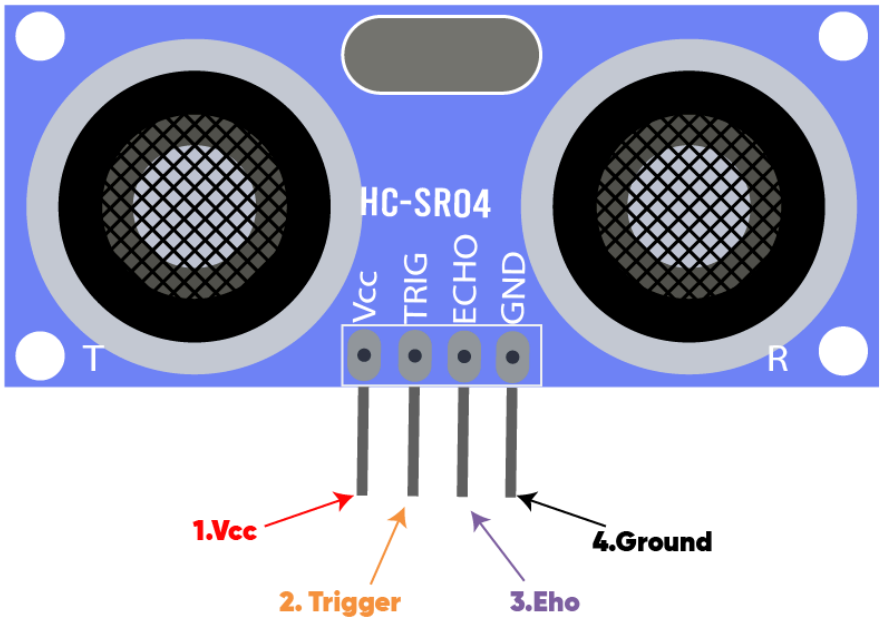
1.1. როგორ მუშაობს სენსორი ექოლოკაციის (ექოს) პრინციპით

ბიოლოგიური რადიოლოკატორი: გინახავს, როგორ დაფრინავს ღამურა სრულ სიბნელეში და არაფერს ეჯახება? ის იყენებს **ექოლოკაციას**. ღამურა გამოსცემს ძალიან მაღალი სიხშირის ბგერას (რომელიც ჩვენ არ გვესმის), შემდეგ კი უსმენს, რა დროში დაუბრუნდება მას ამ ბგერის ექო, რომელიც წინ არსებულ ობიექტებზე ირეკლება. დაბრუნების დროის მიხედვით ღამურა ზუსტად ხვდება, რა მანძილზეა დაბრკოლება.

ჩვენი სენსორიც ღამურას ჰგავს: ულტრაბგერითი სენსორი ზუსტად იგივე პრინციპით მუშაობს. მისი ერთი „თვალი“ (Trig) არის „პირი“, რომელიც უშვებს ულტრაბგერით იმპულსს, ხოლო მეორე „თვალი“ (Echo) არის „ყური“, რომელიც ელოდება ამ იმპულსის ექოს დაბრუნებას.

1.2. Trig (გამშვები) და Echo (მიმღები) პინების დანიშნულება

ულტრაბგერით სენსორს ოთხი მთავარი პინი აქვს, მაგრამ ჩვენ ორ მათგანზე ვფოკუსირდებით:



Trig (Trigger - გამშვები): ეს არის „დაწყების“ პინი. იმისთვის, რომ სენსორმა მანძილის გაზომვა დაიწყოს, ჩვენ არდუინოდან ამ პინზე უნდა გავაგზავნოთ ძალიან ხანმოკლე ელექტრული იმპულსი. ეს იგივეა, რომ სენსორს ვუთხრათ: „აბა, გაუშვი ბგერა!“.

Echo (ექო - მიმღები): ეს არის „პასუხის“ პინი. მას შემდეგ, რაც Trig პინზე იმპულსს გავაგზავნით, Echo პინი გადადის ლოდინის რეჟიმში. როგორც კი ექო დაბრუნდება, ეს პინი ჩაირთვება (გახდება HIGH) და გამორთული იქნება ზუსტად იმ დროის განმავლობაში, რა დროც ექოს დაბრუნებას დასჭირდა. ჩვენი ამოცანაა, გავზომოთ, რა ხანგრძლივობის განმავლობაში იყო ის ჩართული.

1.3. ბგერის სიჩქარის როლი მანძილის გამოთვლაში

ჩვენ ვიცით ფიზიკიდან, რომ ბგერას აქვს გავრცელების კონკრეტული სიჩქარე ჰაერში (დაახლოებით 343 მეტრი წამში).

თუ ჩვენ ვიცით **სიჩქარე** და გავზომავთ **დროს**, რომელიც ბგერას უკან დასაბრუნებლად დასჭირდა, ჩვენ შეგვიძლია მარტივად გამოვთვალოთ **მანძილი**. ეს არის ჩვენი პროექტის მთავარი მათემატიკური გასაღები.

უკეთესი ვიზუალიზაციისთვის შეგვიძლია ვნახოთ ვიდეო.

https://drive.google.com/drive/folders/1pzRu7t3LTUaN4cxNjfQZ7rOiu0-UBz_S //8.1

2. დროის გაზომვა და მანძილის გამოთვლა

როგორ ვთარგმნოთ ექოს დაბრუნების დრო სანტიმეტრებში? ამისთვის რამდენიმე ახალ ბრძანებას ვისწავლით.

2.1. `pulseIn(pin, VALUE)` ფუნქცია: პინზე იმპულსის ხანგრძლივობის გაზომვა

ეს არის სპეციალური ფუნქცია, რომელიც ზუსტად იმას აკეთებს, რაც ჩვენ გვჭირდება. ის ელოდება, როდის გახდება კონკრეტული პინი HIGH, რთავს წამმზომს, შემდეგ ელოდება, როდის გახდება იგივე პინი ისევ LOW და აჩერებს წამმზომს.

შედეგი: `pulseIn()` გვიბრუნებს დროს **მიკროწამებში (µs)**, ანუ წამის მემილიონედებში. ეს არის იმ დროის ხანგრძლივობა, რა დროის განმავლობაშიც Echo პინი იყო ჩართული.

სინტაქსი: `long duration = pulseIn(echoPin, HIGH);` (ეს ნიშნავს: „გაზომე, რა ხანგრძლივობით იყო `echoPin` ჩართულ მდგომარეობაში და შეინახე შედეგი `duration` ცვლადში“).

2.2. Trig პინზე მოკლე იმპულსის გაგზავნა მანძილის გაზომვის დასაწყებად

იმისთვის, რომ სენსორმა ექო გამოუშვას, ჩვენ Trig პინზე უნდა გავაგზავნოთ ძალიან ხანმოკლე, 10 მიკროწამის ხანგრძლივობის იმპულსი. კოდში ეს ასე გამოიყურება:

```
digitalWrite(trigPin, LOW);   // ვრწმუნდებით, რომ პინი გამორთულია
delayMicroseconds(2);         // მოკლე პაუზა
digitalWrite(trigPin, HIGH);  // ვრთავთ იმპულსს
delayMicroseconds(10);        // ველოდებით 10 მიკროწამს
digitalWrite(trigPin, LOW);   // ვთიშავთ იმპულსს
```

```delayMicroseconds()` `delay()`-ს ჰგავს, მაგრამ ის დროს წამის მემილიონედებში ზომავს.

2.3. მიღებული დროის (მიკროწამებში) გარდაქმნა მანძილად (სანტიმეტრებში)

ახლა, როცა `pulseIn()`-ით მივიღეთ დრო (`duration`), ვიყენებთ ფიზიკის ფორმულას: **მანძილი = სიჩქარე × დრო**
```

ბგერის სიჩქარე არის დაახლოებით `0.0343` სანტიმეტრი მიკროწამში.

მაგრამ არის ერთი მნიშვნელოვანი დეტალი: `duration` არის დრო, რომელიც ბგერამ გაიარა **ორმაგ გზაზე** (წინ და უკან). ჩვენ კი მანძილი მხოლოდ ერთ გზაზე გვჭირდება. ამიტომ, მიღებული შედეგი 2-ზე უნდა გავყოთ.

საბოლოო ფორმულა:
`float distance = (duration * 0.0343) / 2;`

3. მანძილის სენსორის გამოყენება პროექტებში

ახლა, როცა მანძილის გამოთვლა შეგვიძლია, მისი გამოყენების უამრავი შესაძლებლობა გვაქვს.

3.1. გაზომილი მანძილის შედარება ზღვრულ მნიშვნელობასთან

ისევე, როგორც წინა სენსორების შემთხვევაში, აქაც შეგვიძლია, დავაწესოთ **ზღვარი (threshold)**. მაგალითად, შეგვიძლია, განვსაზღვროთ „საშიში სიახლოვის“ ზღვარი.
`int dangerZone = 15; // 15 სანტიმეტრი`

3.2. `if/else` ლოგიკის გამოყენება ობიექტის სიახლოვის დეტექციისთვის

`if/else` კონსტრუქციით შეგვიძლია შევამოწმოთ მოხვდა თუ არა ობიექტი ჩვენს „საშიშ ზონაში“:

```cpp
if (distance < dangerZone) {
  Serial.println("ყურადღება! ობიექტი ძალიან ახლოსაა!");
} else {
  Serial.println("გზა თავისუფალია.");
}
```

სავარჯიშო:

პროგრამისტების ჯგუფს მისცეს დავალება: შექმნან კოდი არდუინოსთვის, რომელიც სენსორის (HC-SR04) საშუალებით მუდმივად გაზომავს მანძილს ობიექტამდე და სერიულ მონიტორზე აჩვენებს, თუ რამდენად შეიცვალა მანძილი წინა გაზომვასთან შედარებით. თუმცა მათი კოდი გაურკვეველი მიზეზის გამო არ მუშაობს. შენი დავალებაა გადახედო მათ კოდს და იპოვო შეცდომა.

მითითება: გაიხსენე განსხვავება trig და echo პინებს შორის და დაუკვირდი pinmode ბრძანებას
```
const int TRIG = 9;
const int ECHO = 10;
float lastDistance = -1; // წინა გაზომვა
```

```
void setup() {
      Serial.begin(9600);
      pinMode(TRIG,INPUT);
      pinMode(ECHO, OUTPUT);
      }
      void loop() {
      // ულტრაბგერითი იმპულსის გაგზავნა
      digitalWrite(TRIG, LOW);
      delayMicroseconds(2);
      digitalWrite(TRIG, HIGH);
      delayMicroseconds(10);
      digitalWrite(TRIG, LOW);
// ექოს დროის გაზომვა
long duration = pulseIn(ECHO, HIGH);
// მანძილის გამოთვლა სმ-ში
float distance = (duration * 0.0343) / 2;
// მანძილის ცვლილება ბოლო გაზომვასთან შედარებით
if (lastDistance >= 0) {
 float delta = distance - lastDistance;
 Serial.print("მანძილის ცვლილება: ");
 Serial.print(delta);
Serial.println(" სმ");
}
else
{ Serial.println("პირველი გაზომვა: " + String(distance) + " სმ");
}
lastDistance = distance; // ახალი გაზომვა შეინახე შემდეგისთვის delay(500); }
```

სწორი პასუხი:

რადგანაც `Trig` პინზე სიგნალი იგზავნება არდუინოდან, ის უნდა იყოს `OUTPUT`, ხოლო `Echo` პინი იღებს სიგნალს სენსორიდან არდუინოში, ამიტომ ის უნდა იყოს `INPUT`.

3.3. პოტენციური პროექტები: რობოტი, რომელიც დაბრკოლებას არიდებს თავს, ავტომატური კარი და სხვა

მანძილის განმსაზღვრელი სენსორი ერთ-ერთი ყველაზე სახალისო კომპონენტია. მისი გამოყენებით შეგიძლია შექმნა:

- **რობოტი, რომელიც კედლებს არ ეჯახება:** თუ სენსორი დაინახავს, რომ წინ კედელია, რობოტი ავტომატურად მოუხვევს.

- **ინტერაქტიული მუსიკალური ინსტრუმენტი (თერმენი):** რაც უფრო ახლოს მიიტან ხელს სენსორთან, მით უფრო მაღალ ხმას გამოსცემს ინსტრუმენტი.
- **ავტომატური ნაგვის ურნა:** როცა ხელს მიიტან ნაგვის ურნასთან, ანუ ურნაზე დამაგრებულ სენსორთან, თავსახური თავისით გაიღება.

პრაქტიკული დავალება 8: „უხილავი კედელი“

შენი ამოცანაა დაწერო პროგრამა, რომელიც მუდმივად ზომავს მანძილს ობიექტამდე. თუ ობიექტი 25 სანტიმეტრზე ახლოსაა, სერიულ მონიტორზე დაბეჭდე შეტყობინება: "object is close", ხოლო თუ 25 სანტიმეტრზე შორსაა, დაბეჭდე: " road is free".

🎮 თქვენი პროექტის სიმულაცია Tinkercad-ში

https://drive.google.com/drive/folders/1pzRu7t3LTUaN4cxNjfQZ7rOiu0-UBz_S //8.2

დავალება 8.1: შეცდომის პოვნა და გასწორება

პროგრამისტმა კოდის წერისას შეცდომა დაუშვა. მანძილის გამოსათვლელ ფორმულაში ერთი მნიშვნელოვანი მოქმედება გამორჩა, რის გამოც პროგრამა მანძილს ორჯერ დიდი მნიშვნელობით აჩვენებს. შენი ამოცანაა, იპოვო და გაასწორო შეცდომა ფორმულაში.

მინიშნება: გაიხსენე, `duration` ცვლადი ზომავს დროს ორმაგ გზაზე (წინ და უკან). რა მოქმედება უნდა შევასრულოთ, რომ მხოლოდ ერთი გზის მანძილი მივიღოთ?

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

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);

  // იპოვე შეცდომა ამ ფორმულაში
  float distance = (duration * 0.0343);
```

```
  if (distance < 25) {
    Serial.println("object is close");
  } else {
    Serial.println("road is free");
  }

  delay(500);
}
```

დავალება 8.2: კოდის დასრულება

პროგრამის ძირითადი ნაწილი უკვე აგებულია, კოდი აგზავნის ულტრაბგერით იმპულსს. თუმცა, მთავარი ნაწილი – ექოს დაბრუნების დროის გაზომვა და მისი მანძილად გადაქცევა – გამორჩენილია. შენი ჯერია! დაამატე ორი გამოტოვებული ბრძანება, რათა პროგრამა დასრულდეს.

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

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // --- ჩაამატე პირველი ბრძანება აქ ---
  // გაზომე ექოს დაბრუნების დრო pulseIn() ფუნქციით და შეინახე long ტიპის ცვლადში.

  // --- ჩაამატე მეორე ბრძანება აქ ---
  // გამოთვალე მანძილი სანტიმეტრებში და შეინახე float ტიპის ცვლადში.

  if (distance < 25) {
    Serial.println("object is close");
  } else {
    Serial.println(“road is free");
```

```
  }

  delay(500);
}
```

დავალება 6.3: შეიმუშავე პროგრამული კოდი
დაწერე პროგრამული კოდი, შექმენი პროგრამა, რომელიც შეასრულებს შემდეგ სამუშაოს:

1. განსაზღვრავს Trig და Echo პინებს (9 და 10).
2. setup ფუნქციაში მოამზადებს სერიულ მონიტორს და პინების რეჟიმებს.
3. loop ფუნქციაში გააგზავნის ულტრაბგერით იმპულსს.
4. გაზომავს ექოს დაბრუნების დროს.
5. გამოითვლის მანძილს სანტიმეტრებში.
6. if/else პირობით შეამოწმებს, არის თუ არა მანძილი 25-ზე ნაკლები და დაბეჭდავს შესაბამის შეტყობინებას.

```
void setup() {
  // დაწერე setup ფუნქციის კოდი აქ.
}

void loop() {
  // დაწერე loop ფუნქციის კოდი აქ.
}
```

სწორი პასუხი (პროგრამული კოდი სრულად):

```
// პინების განსაზღვრა
int trigPin = 9;
int echoPin = 10;

void setup() {
  Serial.begin(9600);
  // პინების რეჟიმების განსაზღვრა
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // 1. გააგზავნე 10 მიკროწამიანი იმპულსი trigPin-ზე.
  digitalWrite(trigPin, LOW);
```

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

  // 2. გაზომე ექოს დაბრუნების დრო pulseIn() ფუნქციით და შეინახე ცვლადში.
  long duration = pulseIn(echoPin, HIGH);

  // 3. გამოთვალე მანძილი სანტიმეტრებში.
  float distance = (duration * 0.0343) / 2;

  // 4. დაბეჭდე მანძილი სერიულ მონიტორზე.
  Serial.print("distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // 5. გამოიყენე if/else, რათა შეამოწმო, არის თუ არა მანძილი 15-ზე ნაკლები
  //    და დაბეჭდე შესაბამისი შეტყობინება.
  if (distance < 25) {
    Serial.println("object is close");
  } else {
    Serial.println("road is free");
  }

  delay(500); // პაუზა გაზომვებს შორის
}
```

None

```
{
  "version": 1,
  "board": "arduino:avr:uno",
  "steps": [
    {
      "type": "compile"
    },
    {
      "type": "static",
      "rules": [
```

```
        {
          "id": "serial_begin",
          "name": "Serial communication initialized",
          "kind": "require_regex",
          "pattern": "Serial\\.begin\\s*\\(\\s*9600\\s*\\)",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "Serial კომუნიკაცია უნდა ინიციალიზდეს:
Serial.begin(9600);",
          "msg_pass": "Serial კომუნიკაცია სწორად არის
ინიციალიზებული."
        },
        {
          "id": "pin_trig_output",
          "name": "TRIG pin mode",
          "kind": "require_regex",
          "pattern":
"pinMode\\s*\\(\\s*(?:\\btrigPin\\b|\\bTRIG(?:_PIN)?\\b|9(?!\\d))
\\s*,\\s*OUTPUT\\s*\\)\\s*;?",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "trigPin (პინი 9) უნდა იყოს OUTPUT რეჟიმში:
pinMode(trigPin, OUTPUT);",
          "msg_pass": "TRIG პინი სწორად არის OUTPUT რეჟიმში."
        },
        {
          "id": "pin_echo_input",
          "name": "ECHO pin mode",
          "kind": "require_regex",
          "pattern":
"pinMode\\s*\\(\\s*(?:\\bechoPin\\b|\\bECHO(?:_PIN)?\\b|10(?!\\d)
)\\s*,\\s*INPUT\\s*\\)\\s*;?",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "echoPin (პინი 10) უნდა იყოს INPUT რეჟიმში:
pinMode(echoPin, INPUT);",
          "msg_pass": "ECHO პინი სწორად არის INPUT რეჟიმში."
        },
```

```
        {
          "id": "pulsein",
          "name": "Measure echo with pulseIn",
          "kind": "require_regex",
          "pattern":
"pulseIn\\s*\\(\\s*(?:\\bechoPin\\b|10(?!\\d))\\s*,\\s*HIGH\\s*\\
)",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "echoPin-ის პულსი უნდა გაიზომოს:
pulseIn(echoPin, HIGH);",
          "msg_pass": "echoPin-ის გაზომვა გამოყენებულია
pulseIn()-ით."
        },
        {
          "id": "distance_formula",
          "name": "Distance formula",
          "kind": "require_regex",
          "pattern":
"\\b(distance|dist)\\b\\s*=\\s*[^;]*duration[^;]*0\\.034\\d*[^;]*
/\\s*2\\b",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "მანძილი გამოთვალეთ სწორად: distance =
(duration * 0.0343) / 2;",
          "msg_pass": "მანძილის ფორმულა სწორადაა."
        },
        {
          "id": "threshold_25",
          "name": "Threshold check for 25cm",
          "kind": "require_regex",
          "pattern":
"if\\s*\\(\\s*distance\\s*<\\s*25(?:\\.0+)?\\s*\\)",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "შედარება უნდა იყოს: if (distance < 25)",
          "msg_pass": "25 სმ-ის ზღვარი სწორადაა გამოყენებული."
        },
```

```
        {
          "id": "else_structure",
          "name": "else statement exists",
          "kind": "require_regex",
          "pattern": "\\}\\s*else\\s*\\{",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "გამოიყენეთ else ბლოკი შორი ობიექტის
შეტყობინებისთვის.",
          "msg_pass": "else ბლოკი არსებობს."
        },
        {
          "id": "msg_close",
          "name": "Close object message",
          "kind": "require_regex",
          "pattern":
"Serial\\.println\\s*\\(\\s*\"[^\"]*object\\s+is\\s+close[^\"]*\"
\\s*\\)",
          "flags": "iu",
          "must_pass": true,
          "source": "raw",
          "msg_fail": "როცა distance < 25, დაბეჭდეთ: \"object is
close\"",
          "msg_pass": "ახლო ობიექტის შეტყობინება სწორადაა."
        },
        {
          "id": "msg_free",
          "name": "Free road message",
          "kind": "require_regex",
          "pattern":
"Serial\\.println\\s*\\(\\s*\"[^\"]*road\\s+is\\s+free[^\"]*\"\\s
*\\)",
          "flags": "iu",
          "must_pass": true,
          "source": "raw",
          "msg_fail": "როცა distance ≥ 25, დაბეჭდეთ: \"road is
free\"",
          "msg_pass": "თავისუფალი გზის შეტყობინება სწორადაა."
        },
        {
```

```
          "id": "trig_pulse_seq",
          "name": "TRIG pulse sequence",
          "kind": "require_ordered_regex",
          "patterns": [

"digitalWrite\\s*\\(\\s*(?:\\btrigPin\\b|9(?!\\d))\\s*,\\s*LOW\\s
*\\)",
            "delayMicroseconds\\s*\\(\\s*\\d+\\s*\\)",

"digitalWrite\\s*\\(\\s*(?:\\btrigPin\\b|9(?!\\d))\\s*,\\s*HIGH\\
s*\\)",
            "delayMicroseconds\\s*\\(\\s*\\d+\\s*\\)",

"digitalWrite\\s*\\(\\s*(?:\\btrigPin\\b|9(?!\\d))\\s*,\\s*LOW\\s
*\\)"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "TRIG იმპულსის თანმიმდევრობა: LOW → delay →
HIGH → delay → LOW",
          "msg_pass": "TRIG იმპულსის თანმიმდევრობა სწორია."
        },
        {
          "id": "loop_main_sequence",
          "name": "Correct main loop sequence",
          "kind": "require_ordered_regex",
          "patterns": [

"digitalWrite\\s*\\(\\s*(?:\\btrigPin\\b|9(?!\\d))\\s*,\\s*LOW\\s
*\\)",

"pulseIn\\s*\\(\\s*(?:\\bechoPin\\b|10(?!\\d))\\s*,\\s*HIGH\\s*\\
)",
            "\\b(distance|dist)\\b\\s*=\\s*[^;]*duration",
            "if\\s*\\(\\s*distance\\s*<\\s*25(?:\\.0+)?\\s*\\)"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
```

```
          "msg_fail": "loop() თანმიმდევრობა: TRIG პულსი → pulseIn
→ distance გამოთვლა → if შედარება",
          "msg_pass": "loop() მთავარი ლოგიკის თანმიმდევრობა
სწორია."
        },
        {
          "id": "distance_print_format",
          "name": "Distance print format",
          "kind": "require_ordered_regex",
          "patterns": [

"Serial\\.print\\s*\\(\\s*\"[^\"]*distance[^\"]*\"\\s*\\)",
            "Serial\\.print\\s*\\(\\s*(?:distance|dist)\\s*\\)",

"Serial\\.println\\s*\\(\\s*\"[^\"]*cm[^\"]*\"\\s*\\)"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "მანძილის ბეჭდვის ფორმატი:
Serial.print(\"distance: \") → Serial.print(distance) →
Serial.println(\" cm\")",
          "msg_pass": "მანძილის ბეჭდვის ფორმატი სწორია."
        },
        {
          "id": "if_then_close",
          "name": "Close message in if block",
          "kind": "require_ordered_regex",
          "patterns": [
            "if\\s*\\(\\s*distance\\s*<\\s*25(?:\\.0+)?\\s*\\)",

"Serial\\.println\\s*\\(\\s*\"[^\"]*object\\s+is\\s+close[^\"]*\"
\\s*\\)"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "raw",
          "msg_fail": "if (distance < 25) ბლოკში უნდა იყოს
Serial.println(\"object is close\");",
          "msg_pass": "if ბლოკში ახლო ობიექტის შეტყობინება
სწორადაა."
```

```
        },
        {
          "id": "else_then_free",
          "name": "Free message in else block",
          "kind": "require_ordered_regex",
          "patterns": [
            "\\}\\s*else\\s*\\{",

"Serial\\.println\\s*\\(\\s*\"[^\"]*road\\s+is\\s+free[^\"]*\"\\s
*\\)"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "raw",
          "msg_fail": "else ბლოკში უნდა იყოს
Serial.println(\"road is free\");",
          "msg_pass": "else ბლოკში თავისუფალი გზის შეტყობინება
სწორადაა."
        }
      ]
    }
  ]
}
```

Sim Elements HTML

```
<div style="display: flex; flex-direction: column; align-items: center;
gap: 20px; margin-bottom:
 20px;">
    <wokwi-hc-sr04 id="ultrasonic" distance="100"></wokwi-hc-sr04>
    <label style="font-weight: bold;">Ultrasonic Distance Sensor
(HC-SR04)</label>
 </div>

 <div class="distance-control">
    <label>Distance (cm): <span
data-ultrasonic-display></span></label>
```

```
    <input
        type="range"
        data-ultrasonic-distance
        min="2"
        max="400"
        class="distance-slider"
    />
    <div class="distance-labels">
        <span>2 cm (Very Close)</span>
        <span>400 cm (Far)</span>
    </div>
</div>
```

Sim Hooks js

```
return {
    onInit: function(runner, sensorValues, ultrasonicDistance) {
        if (ultrasonicDistance.value === undefined) {
            ultrasonicDistance.value = 100;
        }
    }
};
```