შეხვედრა 6: სინათლის დეტექტორი და პროგრამული გადაწყვეტილებები

გამარჯობა! წინა შეხვედრაზე ჩვენ ვისწავლეთ, თუ როგორ წავიკითხოთ მისგან მონაცემები სერიულ მონიტორზე. ასევე, ჩვენ ვნახეთ, როგორ იცვლება რიცხვები 0-დან 1023-მდე, თუმცა არ განგვიმარტავს რას ნიშნავს ეს რიცხვები? და რაც მთავარია, როგორ ვაქციოთ ეს რიცხვები ჭკვიან მოქმედებად? დღეს ჩვენ სწორედ ამ კითხვებს ვუპასუხებთ.

1. ფოტორეზისტორის მუშაობის პრინციპი

მოდი, უფრო ღრმად ჩავიხედოთ, როგორ მუშაობს ჩვენი სინათლის სენსორი.

1.1. როგორ იცვლება ფოტორეზისტორის წინაღობა სინათლის ინტენსივობის მიხედვით

რა არის წინაღობა? წარმოიდგინე წყლის მილი. თუ მილი ფართოა, წყალი ადვილად გაივლის მასში, მაგრამ თუ მილი ვიწროა, წყალს გაუჭირდება მასში გავლა. ელექტრონულ სქემებში დენის გავლას ხელს უშლის **წინაღობა**. რაც მეტია წინაღობა, მით უფრო უჭირს დენს გავლა.

ფოტორეზისტორი ანუ სინათლეზე მგრძნობიარე რეზისტორი: ფოტორეზისტორი არის განსაკუთრებული ტიპის რეზისტორი, რომელსაც შეუძლია, თავისი წინაღობა შეცვალოს. მისი წინაღობა დამოკიდებულია სინათლის რაოდენობაზე:

- **როცა გარემო სენსორის ირგვლივ ძალიან ნათელია:** ფოტორეზისტორის წინაღობა ძალიან დაბალია (წყლის მილი ფართოვდება).
- **როცა გარემო სენსორის ირგვლივ ძალიან ბნელია:** ფოტორეზისტორის წინაღობა ძალიან მაღალია (წყლის მილი ვიწროვდება).

1.2. ძაბვის გამყოფი (voltage divider)

არდუინო პირდაპირ წინაღობას ვერ ზომავს, მაგრამ მას შეუძლია, გაზომოს **ძაბვა**. იმისთვის, რომ წინაღობის ცვლილება არდუინოსთვის გასაგები გავხადოთ, ჩვენ ვიყენებთ ხრიკს, რომელსაც „ძაბვის გამყოფი“ ჰქვია.

როგორ მუშაობს? ჩვენს ელექტრულ სქემაში (რომელიც AnITa-ს პლატფორმაზე უკვე აწყობილია) ფოტორეზისტორი შეერთებულია მეორე, ჩვეულებრივ რეზისტორთან ერთად. როცა სინათლე იცვლება, იცვლება ფოტორეზისტორის წინაღობაც, რაც, თავის მხრივ, ცვლის ძაბვას იმ წერტილში, სადაც ისინი ერთმანეთს უკავშირდებიან და საიდანაც არდუინო კითხულობს მონაცემს (A0 პინი). სწორედ ამ ძაბვის ცვლილებას კითხულობს `analogRead()` ფუნქცია.

1.3. სენსორის მონაცემების ინტერპრეტაცია

რას ნიშნავს რიცხვები 0-დან 1023-მდე? ამ რიცხვების მნიშვნელობა ასეთია:

- **როცა გარემო განათებულია:** ფოტორეზისტორის წინაღობა დაბალია. შესაბამისად, A0 პინზე ძაბვაც დაბალია. analogRead(A0) დაგვიბრუნებს **დაბალ რიცხვს** (მაგალითად, 50-დან 300-მდე).
- **როცა ძალიან ბნელა:** ფოტორეზისტორის წინაღობა მაღალია. შესაბამისად, A0 პინზე ძაბვაც მაღალია. analogRead(A0) დაგვიბრუნებს **მაღალ რიცხვს** (მაგალითად, 700-დან 950-მდე).
- **შენიშვნა:** ეს რიცხვები შეიძლება ოდნავ იცვლებოდეს, მაგრამ მთავარი პრინციპი იგივე რჩება: **განათებულ გარემოში – დაბალი რიცხვითი მნიშვნელობა, ბნელ გარემოში კი – მაღალი.**

ფოტორეზისტორის მუშაობის პრინციპის უკეთ გასააზრებლად შეგვიძლია შევხედოთ ვიდეოს

https://drive.google.com/drive/folders/1pzRu7t3LTUaN4cxNjfQZ7rOiu0-UBz_S //6.1

როგორც ვიდეოში ჩანს სინათლის დონის გაზრდა იწვევს წინაღობის შემცირებას, შესაბამისად მასში დამუხტული ნაწილაკები**, დენი** უკეთ მიედინება.

დენი - ეს არის ელექტრონების (დამუხტული ნაწილაკების) მოძრაობა გამტარში, როგორიცაა, მაგალითად, მავთული. როდესაც ვამბობთ "დენი მიედინება", ეს ნიშნავს, რომ ელექტრონები მოძრაობენ გამტარში ერთი მიმართულებით. ისევე როგორც წყალი მიედინება მილში, ელექტრონები "მიედინება" მავთულში და ქმნიან ელექტრულ დენს.

2. პირობითი ოპერატორები (if/else)

ახლა, როცა ვიცით, რას ნიშნავს სენსორის მონაცემები, დროა, არდუინოს ვასწავლოთ გადაწყვეტილების მიღება. ამისთვის ვიყენებთ if/else კონსტრუქციას.

2.1. if/else ოპერატორების სტრუქტურა და გამოყენება

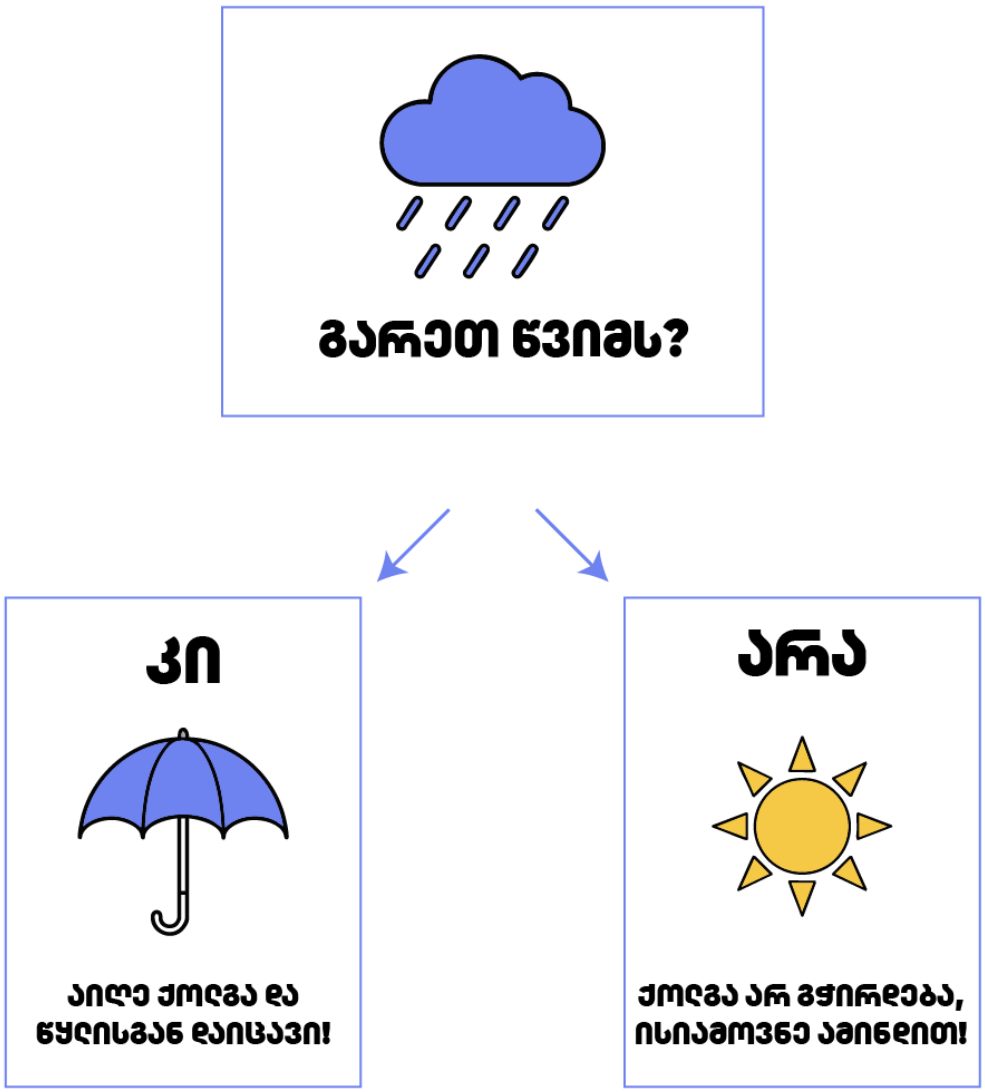
if ინგლისურად ნიშნავს „თუ“. ეს არის ჩვენი მთავარი ინსტრუმენტი პირობის შესამოწმებლად. მისი სტრუქტურა ასეთია:

```
if (პირობა) {
  // კოდი, რომელიც შესრულდება, თუ პირობა ჭეშმარიტია (true)
}
```

ჩვენ შეგვიძლია, დავამატოთ else ბლოკიც, რომელიც ნიშნავს „სხვა შემთხვევაში“.

```
if (პირობა) {
  // კოდი, რომელიც შესრულდება, თუ პირობა ჭეშმარიტია (true)
} else {
  // კოდი, რომელიც შესრულდება, თუ პირობა მცდარია (false)
}
```

ცხოვრებისეული მაგალითი:



კოდის სახით მაგალითი:

```
if (წვიმს == true) {
    Serial.println("აიღე ქოლგა!");
} else {
    Serial.println("ქოლგა არ გჭირდება!");
}
```

2.2. შედარების ოპერატორების (<, >, ==) გამოყენება if პირობაში

if-ის ფრჩხილებში მოსათავსებელი პირობის შესაქმნელად ვიყენებთ შედარების ოპერატორებს:

- > (მეტია): if (sensorValue > 500) – თუ სენსორის მნიშვნელობა 500-ზე მეტია.
- < (ნაკლებია): if (sensorValue < 300) – თუ სენსორის მნიშვნელობა 300-ზე ნაკლებია.
- == (უდრის): if (sensorValue == 1023) – თუ სენსორის მნიშვნელობა ზუსტად 1023-ს უდრის. **(მნიშვნელოვანია: არ აგერიოს = ერთ ცალ სიმბოლოში, რომელიც მნიშვნელობის მინიჭებას ნიშნავს!)**

2.3. ლოგიკური ზღვრის (threshold) კონცეფცია

როდის უნდა ჩავთვალოთ, რომ „ბნელა“? როცა სენსორის მნიშვნელობა 700-ია? 750? თუ 800? იმისთვის, რომ ჩვენი პროგრამა მოქნილი იყოს, ვიყენებთ ცვლადს, რომელსაც **ზღვარს (threshold)** ვუწოდებთ.

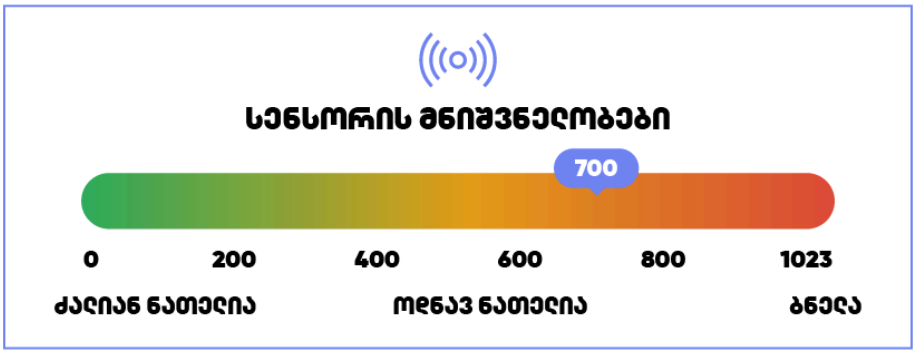
რა არის ზღვარი? ეს არის ჩვენ მიერ დაწესებული „ჯადოსნური რიცხვი“, რომელსაც სენსორის მონაცემს ვადარებთ. მაგალითად, შეგვიძლია, კოდის დასაწყისში შევქმნათ ცვლადი:

int darkThreshold = 700;

შემდეგ კი if პირობაში გამოვიყენოთ ეს ცვლადი:

if (sensorValue > darkThreshold)

ასეთი მიდგომის უპირატესობა ისაა, რომ თუ მოგვიანებით გადავწყვეტთ, რომ „სიბნელის“ ზღვარი შევცვალოთ (მაგალითად, 750-ით), ჩვენ კოდში მხოლოდ ერთ რიცხვს შევცვლით და არა ყველა if პირობას.



სავარჯიშო:

#შექმენი ორი int ტიპის ცვლადი, მიანიჭე შენვის სასურველი მნიშვნელობები და დაბეჭდე მხოლოდ ისინი რომელთა მნიშვნელობაც 10-ზე მეტია.

```
void setup(){

Serial.begin(9600);

int firstNum=5;

int second=25;
```

```
//დაწერე if ოპერატორი შესაბამისი პირობით
}

void loop(){
//loop ფუნქცია ცარიელი რჩება.
}
```

სწორი პასუხი:

```
void setup(){
      Serial.begin(9600);
      //შეიქმნა ცვლადი, რომლის მნიშვნელოვაც არის 5
      int firstNum=5;
      //შეიქმნა მეორე ცვლადი, რომლის მნიშვნელობაც არის 25
      int second=25;
      // შემოწმდა პირველი რიცხვი იყო თუ არა 10-ზე მეტი
      if(firstNum>10){
//თუ if -ში ჩაწერილი პირობა შესრულდება დაიბეჭდება პირველი რიცხვი
            Serial.println(firstNum);
}
//ანალოგიურად მოწმდება მეორე რიცხვიც
      if(secondNum>10){
            Serial.println(secondNum);
```

```
}
}

void loop(){
//loop ფუნქცია ცარიელი რჩება.
}
```

3. ავტომატური სანათის ალგორითმი

მოდი, გავაერთიანოთ ჩვენი ახალი ცოდნა და დავგეგმოთ ჩვენი პირველი ჭკვიანი პროექტი.

3.1. პროექტის ალგორითმის შემუშავება: თუ სინათლე ნაკლებია ზღვარზე, შეასრულე მოქმედება

მიზანი: შევქმნათ პროგრამა, რომელიც მიხვდება, გარემო განათებულია თუ ირგვლის ბნელა.

ალგორითმი:

1. წავიკითხოთ სინათლის სენსორის მონაცემი.
2. შევადაროთ ეს მონაცემი ჩვენ მიერ დაწესებულ სიბნელის ზღვარს.
3. **თუ** მონაცემი ზღვარზე **მეტია** (ანუ ბნელა), დავბეჭდოთ სერიულ მონიტორზე „ბნელა“.
4. **სხვა შემთხვევაში** (ანუ როცა გარემო განათებულია), დავბეჭდოთ, რომ „ნათელია“.

3.2. სენსორის მონაცემზე დაყრდნობით გადაწყვეტილების მიღება კოდში

ეს ალგორითმი კოდში ასე ითარგმნება:

```
int sensorValue = analogRead(A0);
int darkThreshold = 700;
```

```
if (sensorValue > darkThreshold) {
  Serial.println("ბნელა!");
} else {
  Serial.println("ნათელა!");
}
```

3.3. ფსევდოკოდის დაწერა პროექტისთვის

სანამ საბოლოო კოდს დავწერთ, ყოველთვის სასარგებლოა, შევადგინოთ **ფსევდოკოდი**. ეს არის ჩვენი ალგორითმის აღწერა ადამიანური ენით, რომელიც ძალიან ჰგავს ნამდვილ კოდს.

ჩვენი პროექტის ფსევდოკოდი:

```
პროგრამის დაწყება:
  ჩართე სერიული მონიტორი.

მთავარი ციკლი:
  წაიკითხე მნიშვნელობა A0 პინიდან.
  თუ წაკითხული მნიშვნელობა მეტია 700-ზე:
    დაბეჭდე "ბნელა!"
  სხვა შემთხვევაში:
    დაბეჭდე "ნათელია!"
  დაიცადე ცოტა ხანი.
  დაბრუნდი მთავარი ციკლის დასაწყისში.
```

ახლა შენ მზად ხარ შექმნა შენი პირველი პროგრამა, რომელიც დამოუკიდებლად იღებს გადაწყვეტილებას!

დავალება 6: „სინათლის დეტექტორი“

შენი ამოცანაა, დაწერო პროგრამა, რომელიც წაიკითხავს სინათლის სენსორის მონაცემს და `if/else` პირობის გამოყენებით სერიულ მონიტორზე დაბეჭდავს შეტყობინებას „dark!“ ან “bright!“.

პროექტის სიმულაცია Tinkercad-ში

ახლა, როცა თქვენ უკვე გაიგეთ, როგორ მუშაობს ჩვენი სინათლის დეტექტორი და რა პრინციპით უნდა დაიწეროს კოდი, მოდი ვნახოთ, როგორ გამოიყურება ეს პროექტი **Tinkercad**-ის ვირტუალურ გარემოში.

https://drive.google.com/drive/folders/1pzRu7t3LTUaN4cxNjfQZ7rOiu0-UBz_S //6.2

რადგანაც darkThreshold ცვლადის მნიშვნელობა 400-ს უდრის, როდესაც ფოტორეზისტორის მნიშვნელობა 400-ზე მაღალი ხდება, სერიულ მონიტორზე "bright"-ის ნაცვლად "dark" ჩნდება.

დავალება 6.1: შეცდომის პოვნა და გასწორება

პროგრამისტმა კოდის წერისას ლოგიკური შეცდომა დაუშვა.პროგრამა მუშაობს, მაგრამ არასწორად რეაგირებს განათებულობის დონის ცვლილებაზე – როცა ბნელა, წერს „ნათელია!“ და პირიქით. შენი ამოცანაა, იპოვო და გაასწორო ლოგიკური შეცდომა `if` პირობაში.

მინიშნება: გაიხსენე, სენსორის მაღალი რიცხვი სიბნელეს ნიშნავს თუ სინათლეს? შედარების ოპერატორი (`<` ან `>`) სწორად არის გამოყენებული?

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

void loop() {
  int sensorValue = analogRead(A0);
  int darkThreshold = 400; // სიბნელის ზღვარი

  // იპოვე ლოგიკური შეცდომა ამ პირობაში
```

```
  if (sensorValue < darkThreshold) {
    Serial.println("dark!");
  } else {
    Serial.println("bright!");
  }

  delay(200);
}
```

დავალება 6.2: კოდის დასრულება

პროგრამის ძირითადი ნაწილი უკვე აგებულია, კოდი კითხულობს სენსორის მონაცემს. თუმცა, მთავარი ნაწილი – if/else კონსტრუქცია, რომელიც მონაცემს ამოწმებს და შესაბამის ტექსტს ბეჭდავს – გამორჩენილია. შენი ჯერია! დაამატე if/else ბლოკი, რათა პროგრამა დასრულდეს.

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

void loop() {
  int sensorValue = analogRead(A0);
  int darkThreshold = 400;

  // --- ჩაამატე if/else ბლოკი აქ ---
  // 1. შეამოწმე, თუ sensorValue მეტია darkThreshold-ზე.
  // 2. თუ პირობა სრულდება, დაბეჭდე "ბნელა!".
```

```
  // 3. სხვა შემთხვევაში, დაბეჭდე "ნათელია!".


  delay(200);
}
```

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

1. `setup` ფუნქციაში მოამზადებს სერიულ მონიტორს სამუშაოდ.
2. `loop` ფუნქციაში მუდმივად წაიკითხავს მნიშვნელობას ანალოგური პინიდან A0.
3. შეადარებს წაკითხულ მნიშვნელობას შენ მიერ შერჩეულ ზღვარს (მაგ. 400).
4. თუ მნიშვნელობა ზღვარზე მეტია, დაბეჭდავს „ბნელა!“, თუ ნაკლებია – „ნათელია!“.

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


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

სწორი პასუხი (პროგრამული კოდი სრულად):

```
void setup() {
  // სერიული მონიტორის მომზადება
```

```
  Serial.begin(9600);
}

void loop() {
  // სენსორის მონაცემის წაკითხვა A0 პინიდან
  int sensorValue = analogRead(A0);

  // ლოგიკური ზღვრის განსაზღვრა
  int darkThreshold = 400;

  // პირობის შემოწმება: თუ სენსორის მნიშვნელობა მეტია ზღვარზე (ანუ ბნელა)
  if (sensorValue > darkThreshold) {
    Serial.println("dark!");
  } else {
    // სხვა შემთხვევაში (ანუ ნათელია)
    Serial.println("bright!");
  }

  // მცირე პაუზა წაკითხვებს შორის
  delay(200);
}
```

JSON

```
{
  "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": "analogread_a0",
          "name": "Read light sensor A0",
          "kind": "require_regex",
          "pattern": "analogRead\\s*\\(\\s*A0\\s*\\)",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "სინათლის სენსორი უნდა წაიკითხოთ:
analogRead(A0);",
          "msg_pass": "A0 პინიდან წაკითხვა სწორადაა."
        },
```

```
        {
          "id": "threshold_variable",
          "name": "Threshold variable defined",
          "kind": "require_regex",
          "pattern":
"\\b(?:int|float|long)\\s+(?:darkThreshold|threshold|zghv
ari|limit)\\s*=\\s*\\d+",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "უნდა განსაზღვროთ ზღვრის ცვლადი:
int darkThreshold = 400;",
          "msg_pass": "ზღვრის ცვლადი განსაზღვრულია."
        },
        {
          "id": "light_condition",
          "name": "Check darkness condition",
          "kind": "require_regex",
          "pattern":
"if\\s*\\(\\s*(?:sensorValue|sensor|lightValue|light|valu
e|val)\\s*>\\s*(?:darkThreshold|threshold|zghvari|\\d+)\\
s*\\)",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "შეამოწმეთ სიბნელის პირობა: if
(sensorValue > darkThreshold)",
          "msg_pass": "სიბნელის პირობა სწორადაა
შემოწმებული."
        },
        {
          "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 ნათელი
გარემოს შეტყობინებისთვის.",
          "msg_pass": "else ბლოკი არსებობს."
        },
        {
          "id": "msg_dark",
          "name": "Dark message",
          "kind": "require_regex",
          "pattern":
"Serial\\.println\\s*\\(\\s*\"[^\"]*dark!?[^\"]*\"\\s*\\)
",
          "flags": "iu",
          "must_pass": true,
          "source": "raw",
          "msg_fail": "როცა ბნელა (sensorValue >
threshold), დაბეჭდეთ: \"dark!\"",
          "msg_pass": "ბნელის შეტყობინება არსებობს."
        },
        {
          "id": "msg_bright",
          "name": "Bright message",
          "kind": "require_regex",
          "pattern":
"Serial\\.println\\s*\\(\\s*\"[^\"]*bright!?[^\"]*\"\\s*\
\)",
          "flags": "iu",
          "must_pass": true,
          "source": "raw",
          "msg_fail": "როცა ნათელა (sensorValue ≤
threshold), დაბეჭდეთ: \"bright!\"",
          "msg_pass": "ნათელის შეტყობინება არსებობს."
        },
        {
          "id": "delay_exists",
```

```
          "name": "delay() function used",
          "kind": "require_regex",
          "pattern": "delay\\s*\\(\\s*\\d+\\s*\\)",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "გამოიყენეთ delay() ფუნქცია
პაუზისთვის.",
          "msg_pass": "delay() ფუნქცია გამოყენებულია."
        },
        {
          "id": "loop_sequence",
          "name": "Correct loop sequence",
          "kind": "require_ordered_regex",
          "patterns": [
            "analogRead\\s*\\(\\s*A0\\s*\\)",

"if\\s*\\(\\s*(?:sensorValue|sensor|lightValue|light|valu
e|val)\\s*>\\s*(?:darkThreshold|threshold|zghvari|\\d+)\\
s*\\)"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "loop() თანმიმდევრობა: ჯერ
analogRead(A0), შემდეგ if შედარება.",
          "msg_pass": "loop() თანმიმდევრობა სწორია."
        },
        {
          "id": "if_then_dark",
          "name": "Dark message in if block",
          "kind": "require_ordered_regex",
          "patterns": [

"if\\s*\\(\\s*(?:sensorValue|sensor|lightValue|light|valu
```

```
e|val)\\s*>\\s*(?:darkThreshold|threshold|zghvari|\\d+)\\
s*\\)",

"Serial\\.println\\s*\\(\\s*\"[^\"]*dark!?[^\"]*\"\\s*\\)
"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "raw",
          "msg_fail": "if (sensorValue > threshold)
ბლოკში უნდა იყოს Serial.println(\"dark!\");",
          "msg_pass": "if ბლოკში ბნელის შეტყობინება
სწორადაა."
        },
        {
          "id": "else_then_bright",
          "name": "Bright message in else block",
          "kind": "require_ordered_regex",
          "patterns": [
            "\\}\\s*else\\s*\\{",

"Serial\\.println\\s*\\(\\s*\"[^\"]*bright!?[^\"]*\"\\s*\
\)"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "raw",
          "msg_fail": "else ბლოკში უნდა იყოს
Serial.println(\"bright!\");",
          "msg_pass": "else ბლოკში ნათელის შეტყობინება
სწორადაა."
        }
      ]
    }
  ]
}
```

Sim Elements html

```
<div style="display: flex; flex-direction: column; align-items: center;
gap: 20px; margin-bottom:
 20px;">
     <wokwi-photoresistor-sensor id="light-sensor"
pin="A0"></wokwi-photoresistor-sensor>
     <label style="font-weight: bold;">Light Sensor (A0)</label>
 </div>

 <div class="distance-control">
     <label>A0 Light Sensor: <span
data-sensor-display="A0"></span></label>
     <input
         type="range"
         data-sensor="A0"
         min="0"
         max="1023"
         class="distance-slider"
     />
     <div class="distance-labels">
         <span>0 (Bright)</span>
         <span>1023 (Dark)</span>
     </div>
 </div>
```

Sim Hooks js

```
return {
     onInit: function(runner, sensorValues) {
         // Initialize A0 (light sensor) if not set
         if (sensorValues.value.A0 === undefined) {
             sensorValues.value.A0 = 300;
         }
     }
 };
```