შეხვედრა 10: რამდენიმე სენსორი ერთად – გავხადოთ ჩვენი მოწყობილობა კიდევ უფრო “ჭკვიანი”

გამარჯობა! წინა შეხვედრებზე ჩვენ არდუინოს ვასწავლეთ, როგორ ემუშავა სხვადასხვა სენსორთან ცალ-ცალკე. მან ისწავლა სინათლის, ტემპერატურისა და მანძილის გაზომვა. დღეს კი გადავდივართ ახალ, ბევრად საინტერესო ეტაპზე – ჩვენ მას ვასწავლით, როგორ ამუშაოს რამდენიმე სენსორი ერთდროულად. კონკრეტულად, შევქმნით მეტეოსადგურის პროტოტიპს, რომელიც ერთდროულად შეაგროვებს რამდენიმე მონაცემს.

1. კომპლექსური სქემის აწყობა

როგორ ვუთხრათ არდუინოს, რომ ერთდროულად რამდენიმე წყაროდან მიიღოს ინფორმაცია, ერთდროულად რამდენიმე მონაცემი მოაგროვოს?

1.1. რამდენიმე სენსორის ერთდროულად შეერთება არდუინოსთან

AnITa-ს პლატფორმაზე ეს პროცესი გამარტივებულია – ჩვენს ვირტუალურ დაფაზე სენსორები უკვე დაკავშირებულია შესაბამის პინებთან. თუმცა, მნიშვნელოვანია კარგად გავიაზროთ მონაცემთა შეგროვების პროცესი.

რეალურ პროექტში, როცა რამდენიმე სენსორს ვიყენებთ, თითოეულ მათგანს თავისი უნიკალური „მისამართი“, ანუ პინი სჭირდება. მაგალითად, ჩვენს სიმულატორში:

- **სინათლის სენსორი (ფოტორეზისტორი)** დაკავშირებულია **A0** ანალოგურ პინთან.
- **ტემპერატურის სენსორი (TMP36)** დაკავშირებულია **A1** ანალოგურ პინთან.

1.2. კვების და დამიწების ხაზების სწორად განაწილება

შენს სახლში, სავარაუდოდ, ერთიანი ელექტროგაყვანილობაა. მისგან ენერგიას იღებს ტელევიზორი, კომპიუტერი, ნათურა და ყველა სხვა მოწყობილობა.

არდუინოს შემთხვევაშიც ასეა. მას აქვს **5V** (კვება) და **GND** (დამიწება) პინები. როცა რამდენიმე სენსორს ვაერთებთ, ყველა მათგანს სჭირდება ენერგია. იმისათვის, რომ სენსორებმა გამართულად იმუშაოს, აუცილებელია წრედი იყოს სწორად შეკრული. ამისთვის სენსორის GND ფეხი უნდა იყოს დაკავშირებული მიწასთან (GND პინთან), რაც უზრუნველყოფს მთელი სისტემის სტაბილურ მუშაობას.

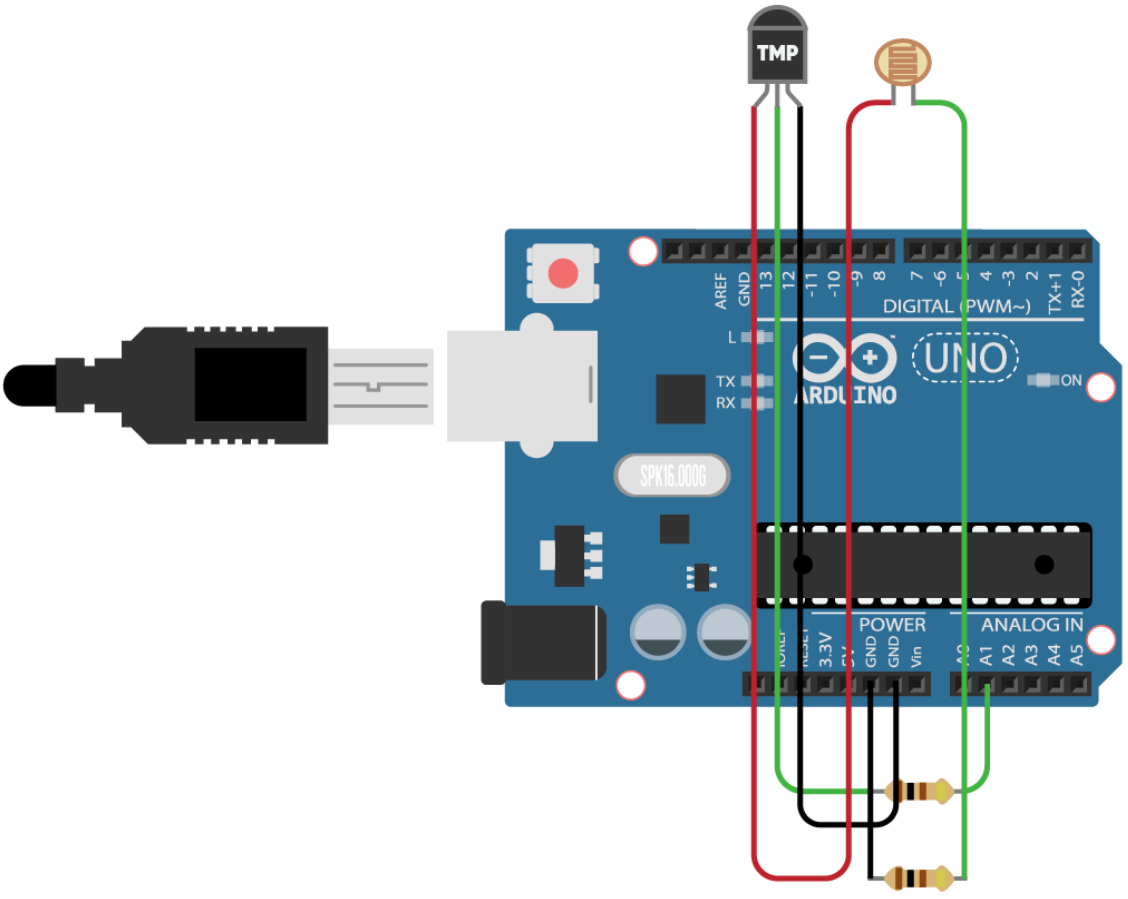
ჩვენს სიმულატორში ესეც ავტომატურად არის მოგვარებული.

1.3. სხვადასხვა პინის გამოყენება თითოეული სენსორისთვის

ეს ყველაზე მნიშვნელოვანი წესია: **ერთი პინი – ერთი დანიშნულება**. იმისთვის, რომ არდუინომ არ აურიოს, რომელი მონაცემი საიდან მოდის, ჩვენ მკაცრად უნდა განვსაზღვროთ, რომელი სენსორი რომელ პინზეა.

როცა კოდში ვწერთ `analogRead(A0);`, არდუინო ზუსტად იგებს, რომ ჩვენ სინათლის სენსორის მონაცემი გვაინტერესებს.

როცა ვწერთ `analogRead(A1);`, ის ხვდება, რომ ახლა ტემპერატურის სენსორის ჯერია.



როგორც ხედავთ, მწვანე კაბელები დაკავშირებულია არდუინოს ანალოგურ პინებთან. სწორედ ამ პინების დახმარებით ხდება არდუინოდან მონაცემების წაკითხვა. წითელი კაბელებით სენსორი არდუინოდან იღებს „ენერგიას“, რომელიც მისი მუშაობისათვის

არის საჭირო. ხოლო შავი კაბელები დაკავშირებულია დამიწების (GND) პინებთან, რათა წრედი შეიკრას.

2. მონაცემების წაკითხვა ერთდროულად სხვადასხვა სენსორიდან

როგორ ვასწავლოთ არდუინოს ორი სხვადასხვა წყაროდან ერთ ციკლში მონაცემების წაკითხვა?

2.1. loop() ფუნქციაში თითოეული სენსორის მონაცემის თანმიმდევრულად წაკითხვა

loop() ფუნქცია ბრძანებებს ასრულებს ზემოდან ქვემოთ, სათითაოდ, მაგრამ წარმოუდგენელი სისწრაფით ისე, რომ ჩვენ ვერც კი ვამჩნევთ

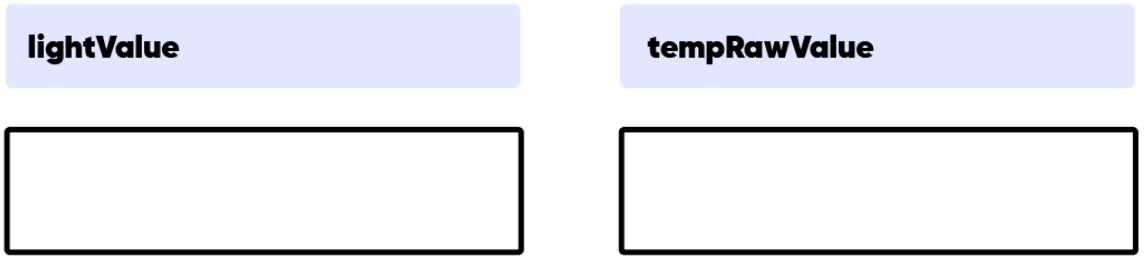
იმისთვის, რომ მონაცემები ორი სენსორიდან წავიკითხოთ, ჩვენ loop() ფუნქციაში ერთმანეთის მიყოლებით უნდა დავწეროთ ორივე წაკითხვის ბრძანება:

```
void loop() {
  // 1. წაიკითხე მონაცემი პირველი სენსორიდან
  int lightValue = analogRead(A0);

  // 2. წაიკითხე მონაცემი მეორე სენსორიდან
  int tempRawValue = analogRead(A1);

  // ... შემდეგი გამოთვლები ...
}
```

ამ კოდის გაშვებისას არდუინოს მეხსიერებაში იქმნება ორი „უჯრა“, სადაც ინახება A0 და A1 პინებზე მიერთებული სენსორების მიერ წაკითხული მონაცემები. ამ „უჯრებში“ ჩაწერილი რიცხვები ძალიან სწრაფად იცვლება და მათი მნიშვნელობა პირდაპირ დამოკიდებულია იმ გარემოზე, რომელშიც სენსორები იმყოფებიან.



2.2. წაკითხული მონაცემების შენახვა სხვადასხვა ცვლადში

როგორც ზემოთ მოცემულ მაგალითში ვნახეთ, კრიტიკულად მნიშვნელოვანია, თითოეული სენსორიდან წაკითხული მონაცემი შევინახოთ **სხვადასხვა ცვლადში**.

გამოიყენე აღწერითი სახელები, რათა კოდი გასაგები იყოს. `val1` და `val2`-ის ნაცვლად, დაარქვი `lightSensorValue` და `tempSensorValue`. ეს დაგეხმარება, თავიდან აიცილო დაბნეულობა და შეცდომები.

2.3. კოდის სტრუქტურირება და ორგანიზება, როცა პროგრამა რთულდება

როცა შენი პროგრამა იზრდება, მისი ორგანიზება აუცილებელი ხდება. ამაში დაგეხმარება **კომენტარები**, რათა კოდი ლოგიკურ ნაწილებად დაყო:

```
void loop() {
  // --- სენსორების მონაცემების წაკითხვა ---
  int lightValue = analogRead(A0);
  int tempRawValue = analogRead(A1);

  // --- გამოთვლების შესრულება ---
  float voltage = tempRawValue * (5.0 / 1023.0);
  float temperatureC = (voltage - 0.5) * 100;

  // --- შედეგების გამოტანა ---
  // ... დაბეჭდვის კოდი ...
}
```

3. მონაცემების ერთდროული გამოტანა

როგორ წარმოვადგინოთ ორი სხვადასხვა სენსორის მონაცემი ერთდროულად და გასაგებად?

3.1. მონაცემების ფორმატირება სერიულ მონიტორზე (მძიმით, ტაბულაციით გამოყოფა)

აქამდე ჩვენ ვიყენებდით `Serial.println()`, რომელიც ყოველი დაბეჭდვის შემდეგ კურსორს ახალ ხაზზე გადაიტანდა. ახლა ჩვენ **რამდენიმე მონაცემის ერთ ხაზზე დაბეჭდვა** გვჭირდება. ამისთვის ვიყენებთ `Serial.print()`. გავიხსენოთ განსხვავება Serial.print() სა და Serial.println() ს შორის

`print` **vs** `println`**:**

`Serial.print()` – ბეჭდავს ტექსტს და კურსორს ტოვებს იმავე ხაზზე.

`Serial.println()` – ბეჭდავს ტექსტს და კურსორი გადააქვს ახალ ხაზზე.

მაგალითი:

```
Serial.print("სინათლე: ");

Serial.print(lightValue);

Serial.print(", ტემპერატურა: ");

Serial.print(temperatureC);

Serial.println(" C");
```

შედეგი სერიულ მონიტორზე: სინათლე: 250, ტემპერატურა: 24.5 C

3.2. ბევრი მონაცემის ერთდროულად გამოტანა Serial Plotter-ზე

Serial Plotter-ს შეუძლია ერთდროულად რამდენიმე გრაფიკი ააგოს. ამისთვის, ჩვენ ერთ ხაზზე უნდა დავბეჭდოთ რამდენიმე რიცხვი, რომლებიც ერთმანეთისგან **მძიმით (",")** ან **ინტერვალით (" ")** იქნება გამოყოფილი.

მაგალითი:

```
Serial.print(lightValue);

Serial.print(","); // გამყოფი

Serial.println(temperatureC);
```

ამ შემთხვევაში, პლოტერი დახაზავს ორ გრაფიკს სხვადასხვა ფერში: ერთს სინათლის სენსორისთვის, მეორეს კი – ტემპერატურისთვის.

3.3. ლოგიკური ოპერატორების (&&, ||) გამოყენება მრავალ პირობაზე დამოკიდებული ლოგიკის შესაქმნელად

ახლა, როცა ორი სენსორის მონაცემი გვაქვს, შეგვიძლია კიდევ უფრო რთული მოწყობილობა ავაწყოთ.

&& **(AND - „და“):** ეს ოპერატორი true-ს აბრუნებს მხოლოდ მაშინ, თუ **ორივე** პირობა ჭეშმარიტია.

მაგალითი: შევქმნათ განგაში სათბურისთვის, რომელიც ჩაირთვება მხოლოდ მაშინ, როცა **ბნელა და თან ცივა**.

```
if (lightValue > 700 && temperatureC < 15) {
  Serial.println("განგაში! სათბურში ბნელა და ცივა!");
}
```

||

AND (&&) ოპერატორი

A	B	A && B
false	false	false
false	true	false
true	false	false
true	true	true

ეს ცხრილი (რომელიც ფოტოზეა მოცემული) დაგვეხმარება უკეთ გავიგოთ როგორ მუშაობს AND ოპერატორი. ცხრილში ნაჩვენებია სხვადასხვა შემთხვევები. მაგალოთად, როცა ერთი პირობა სრულდება და მეორე არ სრულდება, ან პირიქით. **მთავარი წესი ასეთია: AND ოპერატორი *true* (ჭეშმარიტს) აბრუნებს მხოლოდ მაშინ, როცა ორივე პირობა ერთდროულად სრულდება. თუ ორი პირობიდან რომელიმე არ სრულდება, ან ორივე პირობა არ სრულდება მაშინ AND ოპერატორი აბრუნებს მცდარს (false) ანუ პირობა მცდარია.**

ამის გასაგებად შეგვიძლია მოვიყვანოთ მაგალითი ჩვენი ყოველდღიური ცხოვრებიდან:

წარმოიდგინე, რომ ტელეფონზე თამაშის უფლება გაქვს მხოლოდ მაშინ, თუ:

- საშინაო დავალება შეასრულე **და**
- ოთახიც დაალაგე

თუ მხოლოდ დავალება შეასრულე, მაგრამ ოთახი არ დაგილაგებია - ტელეფონზე ვერ ითამაშებ. თუ მხოლოდ ოთახი დაგილაგებია, მაგრამ დავალება არ შეგისრულებია - ისევ ვერ ითამაშებ. თუ არც დავალება არ შეასრულე და ოთახიც არ დაალაგე, ასეთ შემთხვევაშიც, პირობა არ არის შესრულებული. **მხოლოდ მაშინ შეგიძლია ითამაშო, როცა ორივე საქმე გაკეთებული გაქვს!**

სწორედ, ასე მუშაობს AND ოპერატორი - ორივე პირობა უნდა იყოს მართალი.

(OR - „ან“): ეს ოპერატორი true-ს აბრუნებს, თუ **ერთ-ერთი** პირობა მაინც ჭეშმარიტია. ისევ იგივე მაგალითი რომ მოვიყვანოთ თუკი ან საშინაო დავალება შეასრულე ან ოთახი დაალაგე, მაშინ პირობა შესრულებული იქნება.

მაგალითი: შევქმნათ განგაშის სისტემა, რომელიც იმ შემთხვევაში ჩაირთვება, თუ **ტემპერატურა გარკვეული ნიშნულის ზევით აიწევს ან დაბნელდება**.

```
if (temperatureC > 30 || lightValue > 900) {

  Serial.println("განგაში! არანორმალური პირობები!");

}
```

ცხრილი OR ოპერატორისთვის ასეთია:

OR (||) ოპერატორი

A	B	A || B
false	false	false
false	true	true
true	false	true
true	true	true

- თუ ორივე პირობა მცდარია (false და false) → შედეგიც მცდარი (false) იქნება.
- თუ ერთი ჭეშმარიტია და მეორე მცდარია (true და false, ან false და true) → შედეგი უკვე ჭეშმარიტია (true).
- თუ ორივე ჭეშმარიტია (true და true) → შედეგიც ჭეშმარიტი (true) იქნება.

ყოველდღიური მაგალითი:
წარმოიდგინე, რომ მეგობართან შეხვედრა მხოლოდ იმ შემთხვევაში შეგიძლია, თუკი საშინაო დავალება შესრულებული გაქვს, ან ოთახი გაქვს დალაგებული.

- თუ არც საშინაო დავალებები გაქვს მომზადებული და არც ოთახი დალაგებული → ვერ გახვალ.

- თუ მხოლოდ საშინაო დავალებები გაქვს მომზადებული, მაგრამ ოთახ არ გაქვს დალადებული → მაინც გახვალ.

- თუ მხოლოდ ოთახი გაქვს დალაგებული, მაგრამ საშინაო დავალება არ მოგიმზადებია → ისევ გახვალ.

- თუ ორივე გაქვს შესრულებული → მით უმეტეს გახვალ.

განსხვავება AND-თან ისაა, რომ AND-ში ორივე პირობა ერთად უნდა შესრულდეს, რომ მთლიანი პირობა შესრულებულად ჩაითვალოს (დაგვიბრუნოს true), ხოლო OR-ში საკმარისია – ერთი მაინც.

სავარჯიშო:

პროგრამირებაში პირობა შეიძლება იყოს **სწორი** ან **არასწორი**.

თუ პირობა **ჭეშმარიტია**, მას ვანიჭებთ მნიშვნელობას **true**.

თუ პირობა **მცდარია**, მას ვანიჭებთ მნიშვნელობას **false**.

=ქვემოთ მოცემულიარამდენიმე მაგალითი ლოგიკური ოპერატორების გამოყენებით.შენი დავალებაა დააკვირდე და დაადგინო: საბოლოოდ ეს მაგალითი **ჭეშმარიტია (true)** თუ **მცდარია (false)**.

მინიშნება: როგორც მათემატიკურ ოპერატორებში, აქაც ჯერ სრულდება ფრჩხილებში ჩაწერილი მოქმედება

```
true && true = ?            //true

true && false = ?           //false

false && false = ?          //false

true || false = ?           //true

false || false = ?          //false

(true && false) || true = ?       //true
```

```
(true || false) && false = ?      false
```

დავალება 10: „მეტეოსადგურის მონაცემები“

შენი ამოცანაა, დაწერო პროგრამა, რომელიც ერთდროულად წაიკითხავს მონაცემებს სინათლის სენსორიდან (A0 პინი) და ტემპერატურის სენსორიდან (A1 პინი). შემდეგ, დაბეჭდე ორივე სენსორის ნედლი მონაცემი (0-1023) ერთ ხაზზე სერიულ მონიტორზე, ერთმანეთისგან მძიმით გამოყოფილი.

დავალების სიმულაცია tinkercad-ში:

https://drive.google.com/drive/folders/1pzRu7t3LTUaN4cxNjfQZ7rOiu0-UBz_S//10.1

როგორც ხედავთ, სენსორების მნიშვნელობების ცვლილება იწვევს serial monitor– ზე გამოტანილი რიცხვების შეცვლას.

დავალება 6.1: შეცდომის პოვნა და გასწორება

პროგრამისტმა კოდის წერისას შეცდომა დაუშვა. ის არასწორ ბრძანებას იყენებს მონაცემების ერთ ხაზზე დასაბეჭდად, რის გამოც რიცხვები სხვადასხვა ხაზზე იბეჭდება. შენი ამოცანაა, იპოვო და გაასწორო შეცდომა, რათა მონაცემები ერთ ხაზზე, მძიმით გამოყოფილი გამოვიდეს.

მინიშნება: გაიხსენე, რა განსხვავებაა `Serial.print()` და `Serial.println()` ბრძანებებს შორის. რომელი ბრძანება გადაიტანს კურსორს ახალ ხაზზე?

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

void loop() {
  int lightValue = analogRead(A0);
  int tempValue = analogRead(A1);
```

```
  // იპოვე შეცდომა ამ ხაზში
  Serial.println(lightValue);

  Serial.print(",");
  Serial.println(tempValue);

  delay(500);
}
```

დავალება 10.2: კოდის დასრულება

პროგრამის ძირითადი ნაწილი უკვე აგებულია, კოდი კითხულობს მონაცემებს ორივე სენსორიდან. თუმცა, მთავარი ნაწილი – ამ მონაცემების ერთ ხაზზე, მძიმით გამოყოფილი დაბეჭდვა – გამორჩენილია. შენი ჯერია! დაამატე გამოტოვებული ბრძანებები, რათა პროგრამა დასრულდეს.

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

void loop() {
  int lightValue = analogRead(A0);
  int tempValue = analogRead(A1);

  // --- ჩაამატე კოდი აქ ---
  // 1. დაბეჭდე lightValue ცვლადის მნიშვნელობა (კურსორი იმავე ხაზზე უნდა დარჩეს).
  // 2. დაბეჭდე მძიმე (",").
```

```
  // 3. დაბეჭდე tempValue ცვლადის მნიშვნელობა (კურსორი ახალ ხაზზე უნდა გადავიდეს).


  delay(500);

}
```

დავალება 10.3: შეიმუშავე პროგრამული კოდი
დაწერე პროგრამული კოდი, შექმენი პროგრამა, რომელიც შეასრულებს შემდეგ სამუშაოს:

1. setup ფუნქციაში მოამზადებს სერიულ მონიტორს სამუშაოდ.
2. loop ფუნქციაში წაიკითხავს მონაცემებს A0 და A1 პინებიდან.
3. დაბეჭდავს ამ ორ მონაცემს ერთ ხაზზე, მძიმით გამოყოფილს.
4. გააკეთებს ნახევარწამიან პაუზას.

```
void setup() {

  // დაწერე setup ფუნქციის კოდი აქ.

}


void loop() {

  // დაწერე loop ფუნქციის კოდი აქ.

}
```

სწორი პასუხი (პროგრამული კოდი სრულად):

```
void setup() {

  Serial.begin(9600);

}
```

```
void loop() {
  // 1. წაიკითხე მონაცემი A0 პინიდან და შეინახე ცვლადში.
  int lightValue = analogRead(A0);

  // 2. წაიკითხე მონაცემი A1 პინიდან და შეინახე მეორე ცვლადში.
  int tempValue = analogRead(A1);

  // 3. დაბეჭდე პირველი ცვლადი (გამოიყენე print).
  Serial.print(lightValue);

  // 4. დაბეჭდე მძიმე, როგორც გამყოფი.
  Serial.print(",");

  // 5. დაბეჭდე მეორე ცვლადი (გამოიყენე println, რათა კურსორი გადავიდეს).
  Serial.println(tempValue);

  delay(500);
}
```

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": "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": "analogread_a1",
          "name": "Read temperature sensor A1",
          "kind": "require_regex",
          "pattern": "analogRead\\s*\\(\\s*A1\\s*\\)",
```

```
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "ტემპერატურის სენსორი უნდა
წაიკითხოთ: analogRead(A1);",
          "msg_pass": "A1 პინიდან წაკითხვა სწორადაა."
        },
        {
          "id": "serial_print_used",
          "name": "Serial.print() used",
          "kind": "require_regex",
          "pattern": "Serial\\.print\\s*\\(",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "გამოიყენეთ Serial.print() ფუნქცია
მონაცემების დასაბეჭდად.",
          "msg_pass": "Serial.print() გამოყენებულია."
        },
        {
          "id": "serial_println_used",
          "name": "Serial.println() used",
          "kind": "require_regex",
          "pattern": "Serial\\.println\\s*\\(",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "გამოიყენეთ Serial.println()
ფუნქცია ხაზის დასასრულებლად.",
          "msg_pass": "Serial.println() გამოყენებულია."
        },
        {
          "id": "comma_separator",
          "name": "Comma separator used",
          "kind": "require_regex",
```

```
          "pattern":
"Serial\\.print\\s*\\(\\s*\",\"\\s*\\)",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "გამოიყენეთ მძიმე გამყოფად:
Serial.print(\",\");",
          "msg_pass": "მძიმე გამყოფი სწორად არის
გამოყენებული."
        },
        {
          "id": "read_sequence",
          "name": "Correct reading sequence",
          "kind": "require_ordered_regex",
          "patterns": [
            "analogRead\\s*\\(\\s*A0\\s*\\)",
            "analogRead\\s*\\(\\s*A1\\s*\\)"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "წაკითხვის თანმიმდევრობა: ჯერ
analogRead(A0), შემდეგ analogRead(A1).",
          "msg_pass": "სენსორების წაკითხვის თანმიმდევრობა
სწორია."
        },
        {
          "id": "print_format_sequence",
          "name": "Correct print format sequence",
          "kind": "require_ordered_regex",
          "patterns": [

"Serial\\.print\\s*\\(\\s*(?:lightValue|light|sensorValue
|sensor|a0|val|value)\\s*\\)",
            "Serial\\.print\\s*\\(\\s*\",\"\\s*\\)",
```

```
"Serial\\.println\\s*\\(\\s*(?:tempValue|temp|temperature
|a1|val|value)"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "ბეჭდვის თანმიმდევრობა:
Serial.print(lightValue) → Serial.print(\",\") →
Serial.println(tempValue)",
          "msg_pass": "ბეჭდვის ფორმატი სწორია."
        },
        {
          "id": "loop_logic_sequence",
          "name": "Correct loop logic sequence",
          "kind": "require_ordered_regex",
          "patterns": [
            "analogRead\\s*\\(\\s*A0\\s*\\)",
            "analogRead\\s*\\(\\s*A1\\s*\\)",
            "Serial\\.print\\s*\\("
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "loop() ლოგიკა: ჯერ ორივე
analogRead(), შემდეგ Serial.print().",
          "msg_pass": "loop() ლოგიკის თანმიმდევრობა
სწორია."
        }
      ]
    }
  ]
}
```

Sim Elements HTML

```
<div class="sensors" style="display: flex; gap: 30px; margin-bottom: 20px; align-items: center;">
    <div style="display: flex; flex-direction: column; align-items: center; gap: 10px;">
        <wokwi-photoresistor-sensor id="light-sensor" pin="A0"></wokwi-photoresistor-sensor>
        <label style="font-weight: bold;">Light Sensor (A0)</label>
    </div>

    <div style="display: flex; flex-direction: column; align-items: center; gap: 10px;">
        <wokwi-ntc-temperature-sensor id="temp-sensor" pin="A1"></wokwi-ntc-temperature-sensor>
        <label style="font-weight: bold;">Temperature Sensor (A1)</label>
    </div>
 </div>

 <div class="sensors">
    <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>

    <div class="distance-control">
        <label>A1 Temperature Sensor: <span data-sensor-display="A1"></span></label>
        <input
            type="range"
            data-sensor="A1"
            min="0"
```

```
            max="1023"
            class="distance-slider"
        />
    </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 = 512;
        }

        // Initialize A1 (temperature sensor) if not set
        if (sensorValues.value.A1 === undefined) {
            sensorValues.value.A1 = 300;
        }
    }
};
```