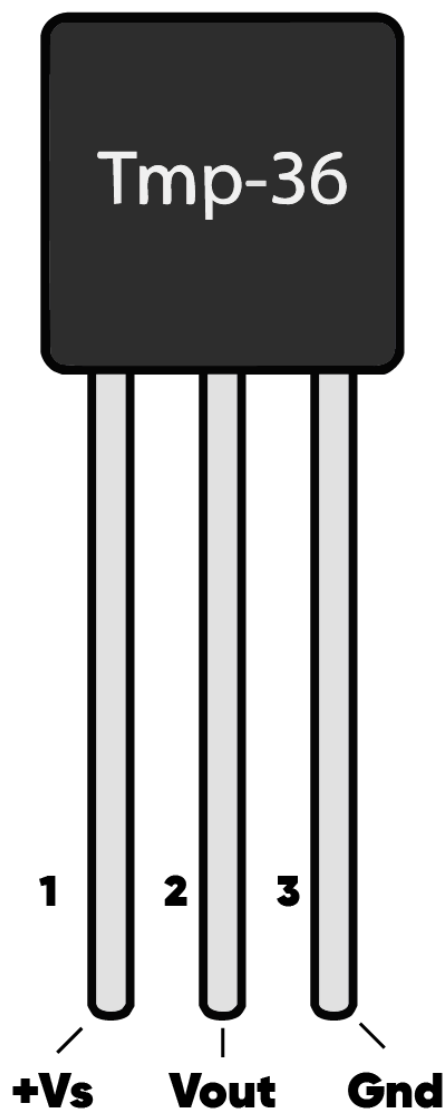


## შეხვედრა 7: ტემპერატურის სენსორი – ჭკვიანი თერმომეტრი

გამარჯობა! წინა შეხვედრაზე ჩვენ არდუინოს ვასწავლეთ თუროგორ „დაინახოს“ სინათლე და მიიღოს გადანყვეტილება იმის მიხედვით თუ როგორი განათებულობის ღონეა ჩვენს გარშემო. დღეს ჩვენ მას კიდევ ერთ, ძალიან მნიშვნელოვან უნარს შევძენთ – ტემპერატურის „შეგრძნებას“. ჩვენ გამოვიყენებთ ტემპერატურის სენსორს, რათა ავანყოთ ჩვენი საკუთარი ციფრული თერმომეტრი.

### 1. ტემპერატურის სენსორის მუშაობის პრინციპი

ჩვენს პროექტში გამოვიყენებთ ერთ-ერთ ყველაზე პოპულარულ ტემპერატურის სენსორს – **TMP36**.



#### 1.1. TMP36 სენსორის მიმოხილვა და მისი პინების დანიშნულება

- **როგორ გამოიყურება ეს სენსორი:** TMP36 არის პატარა, შავი, პლასტმასის ელექტრონული კომპონენტი სამი „ფეხით“ (პინით). მისი ერთი მხარე ბრტყელია, მეორე კი – მომრგვალებული.
- **პინების დანიშნულება:**
  1. **კვება (VCC ან Vs):** მარცხენა პინი. მას ვაერთებთ არდუინოს 5V პინთან, რათა სენსორმა ენერგია მიიღოს.
  2. **გამომავალი ძაბვა (Vout):** შუა პინი. ეს არის მთავარი პინი, რომელიც გვანვდის ინფორმაციას. მას ვაერთებთ არდუინოს ანალოგურ პინთან (მაგ. A0).
  3. **დამინება (GND):** მარჯვენა პინი. მას ვაერთებთ არდუინოს GND პინთან, რათა ელექტრული წრედი შეიკრას.

## 1.2. როგორ გარდაქმნის სენსორი ტემპერატურას ძაბვად

TMP36-ის მუშაობის პრინციპი ძალიან მარტივი და ზუსტია. ის ზომავს გარემოს ტემპერატურას და ამის შესაბამისად, თავის შუა (Vout) პინზე გამოიმუშავებს კონკრეტული დონის ძაბვას.

**პირდაპირი კავშირი:** რაც უფრო მაღალია ტემპერატურა, მით უფრო მაღალ ძაბვას გამოიმუშავებს სენსორი. ეს კავშირი არის **წრფივი**, რაც ნიშნავს, რომ ტემპერატურის ყოველი ერთი გრადუსით მომატებაზე ძაბვაც ყოველთვის ერთი და იმავე რაოდენობით იზრდება. ეს მას ძალიან სანდოს ხდის.

## 1.3. სენსორის მონაცემთა ცხრილის (datasheet) მნიშვნელობა

**რა არის Datasheet?** წარმოიდგინე, რომ იყიდე ახალი ტელეფონი. მას მოჰყვება ინსტრუქცია, სადაც დეტალურად წერია მისი ყველა ფუნქცია, მახასიათებლები და გამოყენების წესები. Datasheet-ი არის ელექტრონული კომპონენტის ასეთი „ინსტრუქცია“.

**რატომ გვჭირდება?** Datasheet-ში მწარმოებელი გვანვდის უმნიშვნელოვანეს ინფორმაციას, მათ შორის, ფორმულას, თუ როგორ გადავიყვანოთ სენსორის მიერ გამოიმუშავებული ძაბვა ტემპერატურის ზუსტ მაჩვენებელში (მაგალითად, ცელსიუსის გრადუსებში). სწორედ ამ დოკუმენტის დახმარებით შეგვიძლია ზუსტად გავიგოთ, თუ რას „გვეუბნება“ სენსორი.

## 2. მონაცემების გარდაქმნა (კონვერტაცია)

`analogRead()` ფუნქცია გვაძლევს რიცხვს 0-დან 1023-მდე. ეს არ არის ტემპერატურა. იმისთვის, რომ მივიღოთ ცელსიუსის გრადუსები, ჩვენ ორსაფეხურიანი გარდაქმნა უნდა შევასრულოთ.

### 2.1. `analogRead()`-ით მიღებული მნიშვნელობის გარდაქმნა ძაბვაში (ვოლტებში)

# ქაბზილან ტემპერატურამდე

## როგორ იმუშავებს ცნობილი ფორმულა?

TMP36 ყოველ გრადუს სადსიუსზე 10mV ქაბზას აძლევს

1

**analogRead() შედეგი**

Arduino კითხულობს  
0-1023 რიცხვს

```
sensorValue =  
analogRead(A0);
```

2

**ქრავად გარდაქმნა**

რიცხვი ქრავად  
გარდაიქმნება

```
voltage =  
sensorValue * (5.0  
/ 1023.0);
```

3

**ტემპერატურის გამოთვლა**

ქაბზა ტემპერატურად  
იქცევა

```
temperature =  
(voltage - 0.5) * 100 ;
```

**ნაბიჯი 1:** ჯერ `analogRead()`-ის შედეგი (0-1023) უნდა გადავიყვანოთ ძაბვაში (0-5V). ჩვენ ვიცით, რომ 1023 შეესაბამება 5 ვოლტს. ამიტომ, ფორმულა ასეთია:

```
float voltage = sensorValue * (5.0 / 1023.0);
```

ყურადრება მიაქციეთ `voltage` ცვლადის "ტიპს". რადგანაც მას უნდა მიენიჭოს უკვე ძაბვად გარდაქმნილი რიცხვი და ეს რიცხვი შეიძლება წილადიც იყოს, გვჭირდება შესაბამისი "უჯრა". `Float` ტიპი კი სწორედ ასეთი ცვლადების შენახვის საშუალებას გვაძლევს.

**რა განსხვავებაა 5 და 5.0 ს შორის და რატომ გვჭირდება 5.0-ის დანერა მოცემულ შემთხვევაში?** Arduino-ში 5 და 5.0 სხვადასხვა ტიპის მონაცემებია. როდესაც ვწერთ 5-ს, ის ინახება როგორც `integer` (მთელი რიცხვი), ხოლო 5.0 ინახება როგორც `float` (ათწილადი რიცხვი). Arduino-ში მოქმედების შესასრულებლად საჭიროა, რომ ყველა

ცვლადი ერთნაირი ტიპის იყოს. რადგანაც `sensorValue` არის `float` ტიპის, სხვა ცვლადებიც `float` ტიპის უნდა გავხადოთ.

## 2.2. მიღებული ძაბვის გარდაქმნა ტემპერატურად ცელსიუსში მათემატიკური ფორმულის გამოყენებით

**ნაბიჯი 2:** როცა სენსორი ტემპერატურას ძაბვაში გადააკონვერტირებს და ძაბვის მაჩვენებლის სახით გვანვდის, ვიყენებთ ფორმულას TMP36-ის `datasheet`-იდან:  
$$\text{float temperatureC} = (\text{voltage} - 0.5) * 100;$$

### რას ნიშნავს ეს ფორმულა?

- **- 0.5:** სენსორს აქვს მცირე „ნანაცვლება“.  $0^{\circ}\text{C}$ -ზე ის 0 ვოლტს კი არ გამოიმუშავებს, არამედ  $- 0.5$  ვოლტს. ამიტომ, ჯერ ამ 0.5 ვოლტს ვაკლებთ.
- **\* 100:** სენსორი ძალიან მგრძნობიარეა. ტემპერატურის ყოველ  $1^{\circ}\text{C}$ -ით ცვლილებაზე ძაბვა 10 მილივოლტით (ანუ 0.01 ვოლტით) იცვლება. ამიტომ, იმისთვის, რომ მიღებული ძაბვა გრადუსებში გადავიყვანოთ, ის 100-ზე უნდა გავამრავლოთ.

## 2.3. გამოთვლილი ტემპერატურის გამოტანა სერიულ მონიტორზე

ახლა, როცა საბოლოო შედეგი გვაქვს, შეგვიძლია, ის ლამაზად გამოვიტანოთ სერიულ მონიტორზე:

```
Serial.print("ტემპერატურა: ");  
Serial.print(temperatureC);  
Serial.println(" C"); // ვამატებთ ცელსიუსის აღნიშვნას
```

სავარჯიშო:

შენს წინაშეა Arduino პროგრამა, სადაც `testValue` ცვლადში ჩაწერილია სენსორის ტესტური მნიშვნელობა - 256. შენი ამოცანაა, ეს რიცხვი ძაბვაში გარდაქმნა და ლამაზად გამოიტანო სერიულ მონიტორზე.

```
void setup() {  
  
    Serial.begin(9600);  
  
}  
  
void loop() {  
  
    int testValue = 256;  
  
    //გამოთვალე testValue-ს შესაბამისი ძაბვა  
    //დაბეჭდე შესაბამის ფორმატში 1023.->5v  
  
    delay(2000);  
  
}
```

სწორი პასუხი:

შენს წინაშეა Arduino პროგრამა, სადაც `testValue` ცვლადში ჩანერილია სენსორის ტესტური მნიშვნელობა - 256. შენი ამოცანაა, ეს რიცხვი ძაბვაში გარდაქმნა და სერიულ მონიტორზე გამოიტანო.

```
void setup() {  
  
    Serial.begin(9600);  
  
}  
  
void loop() {  
  
    int testValue = 256;  
  
    // ძაბვის გამოთვლა  
  
    float voltage = testValue * (5.0 / 1023.0);  
  
    // შედეგის ლამაზად გამოტანა  
  
    Serial.print(testValue);  
  
    Serial.print(" - ");  
  
    Serial.print(voltage);  
  
    Serial.println("V");  
  
    delay(2000);  
  
}
```

### 3. პროექტის ალგორითმის გაფართოება

ტემპერატურის გაზომვა საინტერესო პროცესია, მაგრამ ჩვენ შეგვიძლია, ჩვენი პროექტი კიდევ უფრო ჭკვიანი გავხადოთ.

#### 3.1. ტემპერატურის მონაცემების შედარება ზღვრულ მნიშვნელობასთან

როგორც ფოტორეზისტორის შემთხვევაში, აქაც შეგვიძლია, დავანესოთ ზღვარი (threshold). მაგალითად, შეგვიძლია, განვსაზღვროთ, რა არის „ცხელი“ ტემპერატურა ჩვენი პროექტისთვის.

```
float hotThreshold = 28.0;
```

### 3.2. if/else პირობის გამოყენება ტემპერატურული განგაშის შესაქმნელად

if/else კონსტრუქციის გამოყენებით, ჩვენ შეგვიძლია, შევქმნათ მარტივი განგაშის სისტემა:

```
if (temperatureC > hotThreshold) {  
  Serial.println("ყურადღება: ოთახში ძალიან ცხელა!");  
} else {  
  Serial.println("ტემპერატურა ნორმალურია.");  
}
```

### 3.3. კოდის ოპტიმიზაცია და ცვლადების ეფექტურად გამოყენება

როცა კოდი რთულდება, მნიშვნელოვანია, ის გასაგები და ადვილად წასაკითხი იყოს. კარგი პრაქტიკაა:

- **გამოიყენო აღწერითი სახელები ცვლადებისთვის:** t-ს ნაცვლად დაარქვა temperatureC.
- **გამოთვლები ნაწილ-ნაწილ აწარმოო:** ნუ ეცდები, ყველაფერი ერთ, გრძელ ხაზში დაწერო. გამოიყენე შუალედური ცვლადები (როგორც voltage), რათა კოდის ლოგიკა ნათელი იყოს.
- **დატოვო კომენტარები:** აუხსენი, რას აკეთებს კოდის ესა თუ ის ნაწილი.

#### დავალბა 7: „გადახურების სიგნალიზაცია“

შენი ამოცანაა, დაწერო პროგრამა, რომელიც წაიკითხავს ტემპერატურის სენსორის მონაცემს, გადაიყვანს მას ცელსიუსის გრადუსებში და დაბეჭდავს სერიულ მონიტორზე. დამატებით, თუ ტემპერატურა გადააჭარბებს 28°C-ს, პროგრამამ უნდა დაბეჭდოს გამაფრთხილებელი შეტყობინება: "Attention: high temperature!".

## თქვენი პროექტის სიმულაცია Tinkercad-ში

ახლა, როცა თქვენ უკვე გაიგეთ, როგორ მუშაობს ტემპერატურის სენსორი 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//7.1](https://drive.google.com/drive/folders/1pzRu7t3LTUaN4cxNjfQZ7rOiu0-UBz_S//7.1)

როგორც ვიდეოში ხედავთ, როდესაც ტემპერატურა გადაცდება hotThreshold ცვლადისთვის მინიჭებულ მნიშვნელობას არდუინო გვაფრთხილებს: "warning High temperature!".

### დავალება 7.1: შეცდომის პოვნა და გასწორება

პროგრამისტს კოდის წერისას შეცდომა მოუვიდა. ტემპერატურის გამოსათვლელ ფორმულაში ერთი რიცხვი არასწორად არის მითითებული, რის გამოც პროგრამა ტემპერატურას არასწორად აჩვენებს. შენი ამოცანაა, იპოვო და გაასწორო შეცდომა ფორმულაში.

**მინიშნება:** გაიხსენე, რა რიცხვს ვაკლებთ მიღებულ ძაბვას, სანამ 100-ზე გავამრავლებთ?

```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  int sensorValue = analogRead(A0);  
  float voltage = sensorValue * (5.0 / 1023.0);  
  
  // იპოვე შეცდომა ამ ფორმულაში  
  float temperatureC = (voltage - 0.8) * 100;  
  
  Serial.print("temperature: ");  
  Serial.print(temperatureC);  
  Serial.println(" C");  
  
  if (temperatureC > 28) {  
    Serial.println("Attention:hihg temperature!");  
  }  
  
  delay(1000);  
}
```

### დავალება 7.2: კოდის დასრულება

პროგრამის ძირითადი ნაწილი უკვე აგებულია, კოდი კითხულობს სენსორის მონაცემს და ძაბვაში გადაჰყავს. თუმცა, მთავარი ნაწილი – ძაბვის ტემპერატურაში გადაყვანა და შედეგის დაბეჭდვა – გამოჩენილია. შენი ჯერია! დაამატე გამოტოვებული კოდის ნაწილები, რათა პროგრამა დასრულდეს.

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



```

void loop() {
  int sensorValue = analogRead(A0);
  float voltage = sensorValue * (5.0 / 1023.0);

  // --- ჩაამატე კოდი აქ ---
  // 1. შექმენი float ტიპის ცვლადი temperatureC.
  // 2. გამოთვალე ტემპერატურა ცელსიუსში (გაიხსენე ფორმულა).
  // 3. დაბეჭდე შედეგი სერიულ მონიტორზე შესაბამისი ტექსტით.
  // 4. დამატე if პირობა, რომელიც შეამოწმებს ტემპერატურას და საჭიროების
შემთხვევაში დაბეჭდავს განგაშის შეტყობინებას.

  delay(1000);
}

```

### დავალება 7.3: შეიმუშავე პროგრამული კოდი

დაწერე პროგრამული კოდი, შექმენი პროგრამა, რომელიც შეასრულებს შემდეგ სამუშაოს:

1. მოამზადებს სერიულ მონიტორს სამუშაოდ.
2. წაიკითხავს მონაცემს A0 პინიდან.
3. გადაიყვანს მონაცემს ჯერ ძაბვაში, შემდეგ კი ტემპერატურაში (ცელსიუსი).
4. დაბეჭდავს ტემპერატურას სერიულ მონიტორზე.
5. თუ ტემპერატურა 28 გრადუსზე მეტია, დამატებით დაბეჭდავს გამაფრთხილებელ შეტყობინებას.

```

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

```

```

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

```

სწორი პასუხი (პროგრამული კოდი სრულად):

```

void setup() {
  // 1. მოამზადე სერიული მონიტორი სამუშაოდ.
  Serial.begin(9600);
}

```

```

void loop() {
  // 2. წაიკითხე მნიშვნელობა ანალოგური პინიდან A0 და შეინახე ცვლადში.
  int sensorValue = analogRead(A0);

```

```

// 3. გადაიყვანე ნაკითხული მნიშვნელობა ძაბვაში.
float voltage = sensorValue * (5.0 / 1023.0);

// 4. გადაიყვანე ძაბვა ტემპერატურაში (ცელსიუსი).
float temperatureC = (voltage - 0.5) * 100;

// 5. დაბეჭდე ტემპერატურის მიმდინარე მნიშვნელობა.
Serial.print("temperature: ");
Serial.print(temperatureC);
Serial.println(" C");

// 6. გამოიყენე if პირობა, რათა შეამოწმო, ხომ არ არის ტემპერატურა 28-ზე
მაღალი.
//     თუ ასეა, დაბეჭდე გამაფრთხილებელი შეტყობინება.
if (temperatureC > 28) {
    Serial.println("Attention:high temperature!");
}

// 7. დააყოვნე პროგრამა 1 წამით, რათა მონაცემები ადვილად წასაკითხი იყოს.
delay(1000);
}

```

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 temperature sensor A0",
        "kind": "require_regex",
        "pattern": "analogRead\\s*\\(\\s*A0\\s*\\)",
        "flags": "iu",
        "must_pass": true,
        "source": "stripped",
        "msg_fail": "ტემპერატურის სენსორი უნდა წაკითხოთ:
analogRead(A0);",
        "msg_pass": "A0 პინიდან წაკითხვა სწორადაა."
    },
    {
        "id": "voltage_conversion",
        "name": "Voltage conversion formula",
        "kind": "require_regex",
        "pattern":
"\\b(voltage|volt)\\b\\s*=\\s*[^;]*sensorValue[^;]*5\\.0[^;]*/[^;
]*1023",
        "flags": "iu",
        "must_pass": true,
        "source": "stripped",
        "msg_fail": "ძაბვის გამოთვლა: voltage = sensorValue *
(5.0 / 1023.0);",
        "msg_pass": "ძაბვის გამოთვლის ფორმულა სწორია."
    },
    {
        "id": "temperature_conversion",
        "name": "Temperature conversion formula",
        "kind": "require_regex",

```

```

        "pattern":
"\b(temperature|temp)\w*\b\s*=\s*\b(?:\b|[\^;]*voltage[\^;]*-[\^;]*0
\\.5[\^;]*\b)[\^;]*\b*[\^;]*100",
        "flags": "iu",
        "must_pass": true,
        "source": "stripped",
        "msg_fail": "ტემპერატურის გამოთვლა: temperatureC =
(voltage - 0.5) * 100;",
        "msg_pass": "ტემპერატურის გამოთვლის ფორმულა სწორია.",
    },
    {
        "id": "threshold_28",
        "name": "Threshold check for 28°C",
        "kind": "require_regex",
        "pattern":
"if\s*\b(?:\b|[\^;]*voltage[\^;]*-[\^;]*0\\.5[\^;]*\b)\s*>\s*28(?:\\.0+)?\s*\b",
        "flags": "iu",
        "must_pass": true,
        "source": "stripped",
        "msg_fail": "შედადება უნდა იყოს: if (temperatureC >
28)",
        "msg_pass": "28°C-ის ზღვარი სწორადაა გამოყენებული.",
    },
    {
        "id": "warning_message",
        "name": "High temperature warning",
        "kind": "require_regex",
        "pattern":
"Serial\\.println\s*\b(?:\b|[\^;]*voltage[\^;]*-[\^;]*0\\.5[\^;]*\b)\s*[\^;]*[Aa]ttention[\^;]*[Hh]i[gh]h[
^\^;]*[Tt]emperature[\^;]*\b(?:\b|[\^;]*voltage[\^;]*-[\^;]*0\\.5[\^;]*\b)",
        "flags": "iu",
        "must_pass": true,
        "source": "raw",
        "msg_fail": "როცა temperatureC > 28, დაბეჭდეთ:
\\\"Attention: high temperature!\\\"",
        "msg_pass": "მაღალი ტემპერატურის გამაფრთხილებელი
შეტყობინება არსებობს.",
    },
    {
        "id": "delay_exists",

```

```

        "name": "delay() function used",
        "kind": "require_regex",
        "pattern": "delay\\s*\\(\\s*1000\\s*\\)",
        "flags": "iu",
        "must_pass": true,
        "source": "stripped",
        "msg_fail": "გამოიყენეთ delay(1000) 1 წამიანი
        პაუზისთვის.",
        "msg_pass": "delay(1000) გამოყენებულია."
    },
    {
        "id": "conversion_sequence",
        "name": "Correct conversion sequence",
        "kind": "require_ordered_regex",
        "patterns": [
            "analogRead\\s*\\(\\s*A0\\s*\\)",
            "\\b(voltage|volt)\\b\\s*=\\s*\\([^;]*sensorValue",

            "\\b(temperature|temp)\\w*\\b\\s*=\\s*\\([^;]*voltage"
        ],
        "flags": "iu",
        "must_pass": true,
        "source": "stripped",
        "msg_fail": "გადაყვანის თანმიმდევრობა: analogRead(A0) →
        voltage გამოთვლა → temperatureC გამოთვლა",
        "msg_pass": "გადაყვანის თანმიმდევრობა სწორია."
    },
    {
        "id": "temperature_print_format",
        "name": "Temperature print format",
        "kind": "require_ordered_regex",
        "patterns": [
            "Serial\\.print\\s*\\(\\s*\\\"[^\\"]*temperature[^\\"]*\\\"\\s*\\)",
            "Serial\\.print\\s*\\(\\s*(?:temperature|temp)\\w*\\s*\\)",
            "Serial\\.println\\s*\\(\\s*\\\"[^\\"]*[CcCc][^\\"]*\\\"\\s*\\)",
        ],
        "flags": "iu",
        "must_pass": true,
    }

```

```

        "source": "stripped",
        "msg_fail": "ტემპერატურის ბეჭდვის ფორმატი:
Serial.print(\"temperature: \") → Serial.print(temperatureC) →
Serial.println(\" C\")",
        "msg_pass": "ტემპერატურის ბეჭდვის ფორმატი სწორია."
    },
    {
        "id": "loop_main_sequence",
        "name": "Correct main loop sequence",
        "kind": "require_ordered_regex",
        "patterns": [
            "analogRead\\s*\\(\\s*A0\\s*\\)",
            "\\b(voltage|volt)\\b\\s*=",
            "\\b(temperature|temp)\\w*\\b\\s*=",
            "Serial\\.print\\s*\\(",
            "if\\s*\\(\\s*(?:temperature|temp)\\w*\\s*>\\s*28"
        ],
        "flags": "iu",
        "must_pass": true,
        "source": "stripped",
        "msg_fail": "loop() თანმიმდევრობა: analogRead → voltage
→ temperatureC → Serial.print → if შედარება",
        "msg_pass": "loop() მთავარი ლოგიკის თანმიმდევრობა
სწორია."
    },
    {
        "id": "if_then_warning",
        "name": "Warning in if block",
        "kind": "require_ordered_regex",
        "patterns": [

            "if\\s*\\(\\s*(?:temperature|temp)\\w*\\s*>\\s*28(?:\\.0+)?\\s*\\)
)",
            "Serial\\.println\\s*\\(\\s*\\[\\^\\]*[Aa]ttention[\\^\\]*[Hh]i[gh]h[\\^\\]*[Tt]emperature"
        ],
        "flags": "iu",
        "must_pass": true,
        "source": "raw",

```

```

    "msg_fail": "if (temperatureC > 28) ბლოკში უნდა იყოს
Serial.println(\"Attention: high temperature!\");",
    "msg_pass": "if ბლოკში გამაფრთხილებელი შეტყობინება
სწორადაა."
  }
]
}
]
}

```

## Sim Elements HTML

```

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

```

```

</div>

<div class="distance-control">
  <label>A0 Sensor Value: <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 (Cold)</span>
    <span>1023 (Hot)</span>
  </div>
</div>

```

### Sim Hooks js

```

return {
  onInit: function(runner, sensorValues) {
    // Initialize A0 (temperature sensor) if not set
    // Default: ADC 307 ≈ 30°C (will show warning)
    if (sensorValues.value.A0 === undefined) {
      sensorValues.value.A0 = 307;
    }
  }
}

```



```
    }  
  }  
};
```