შეხვედრა 9: მონაცემთა ვიზუალიზაცია – დავინახოთ ინფორმაცია

გამარჯობა! წინა შეხვედრებზე ჩვენ ვმუშაობდით სენსორებთან და მათგან მიღებულ მონაცემებს სერიულ მონიტორზე ვბეჭდავდით. ჩვენ ვხედავდით რიცხვების უწყვეტ ნაკადს, მაგრამ რთული იყო, ამ რიცხვებში რაიმე კანონზომიერება ან ტენდენცია დაგვენახა. დღეს ჩვენ ვისწავლით, როგორ ვაქციოთ ეს მოსაწყენი რიცხვები ცოცხალ, მოძრავ გრაფიკად და როგორ „დავინახოთ“ ინფორმაცია სრულიად ახლებურად.

1. მონაცემთა ვიზუალიზაციის მნიშვნელობა

1.1. რატომ არის გრაფიკული წარმოდგენა უფრო ინფორმაციული, ვიდრე რიცხვების ნაკადი

მარტივი აღსაქმელია: ჩვენი ტვინი ბევრად უკეთ აღიქვამს ვიზუალურ ინფორმაციას, ვიდრე ტექსტს ან რიცხვებს. წარმოიდგინე, ექიმი უყურებს პაციენტის გულისცემის მონაცემებს. რა უფრო ინფორმაციული იქნება მისთვის: ასობით რიცხვის გრძელი სია თუ ელექტროკარდიოგრამის (ეკგ) გრაფიკი, სადაც გულისცემის რიტმი ნათლად ჩანს? რა თქმა უნდა, გრაფიკი.

მყისიერი შთაბეჭდილება: გრაფიკზე ერთი შეხედვითაც კი შეგვიძლია, მივიღოთ ზოგადი წარმოდგენა პროცესზე. ჩვენ მაშინვე ვხედავთ, მონაცემები იზრდება, მცირდება, სტაბილურია თუ მკვეთრად იცვლება. რიცხვების ნაკადში ამის დასადგენად დიდი დრო და კონცენტრაცია დაგვჭირდებოდა.

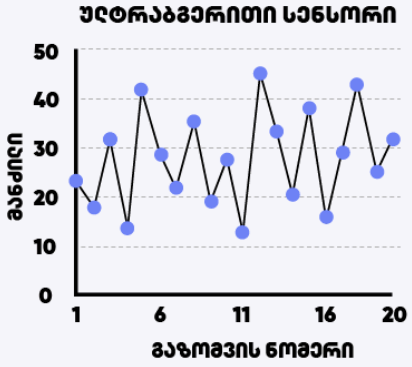
ნაცნობი ხელსაწყოები:მონაცემთა ვიზუალიზაციის მნიშვნელობის უკეთ გასაგებად, გამოვიყენოთ ჩვენთვის უკვე ნაცნობი ულტრაბგერითი სენსორის მაგალითი.

ციფრული მონაცემები

#1: 23.5სმ (1368მკ) #2: 18.7სმ (1088მკ) #3: 32.1სმ (1868მკ) #4: 15.4სმ (885მკ) #5: 41.8სმ (2433მკ) #6: 20.9სმ (1628მკ) #7: 22.3სმ (1299მკ) #8: 35.6სმ (2072მკ) #9: 19.8სმ (1152მკ) #10: 27.4სმ (1595მკ) #11: 12.9სმ (751მკ) #12: 45.3სმ (2636მკ) #13: 33.7სმ (1961მკ) #14: 21.1სმ (1228მკ) #15: 38.4სმ (2235მკ) #16: 16.5სმ (960მკ) #17: 29.8სმ (1734მკ) #18: 43.2სმ (2514მკ) #19: 25.7სმ (1496მკ) #20: 31.6სმ (1839მკ)

სტატისტიკა: საშუალო: 22.2სმ მინიმალური: 12.9სმ მაქსიმუმი: 45.3სმ დისპერსია: 9.3სმ

ვიზუალური გრაფიკი

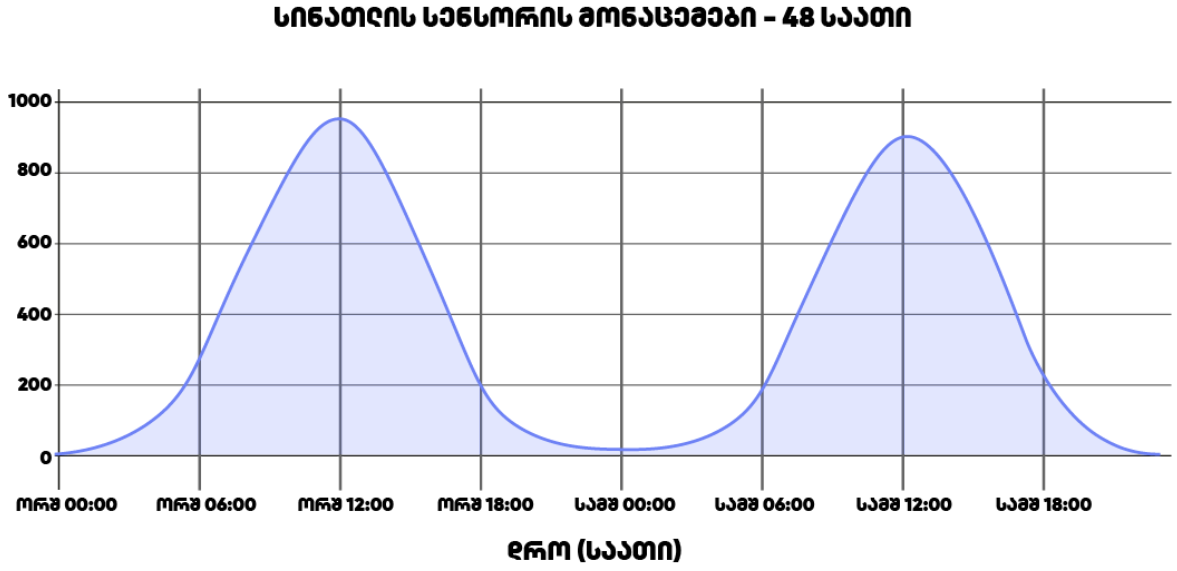


პირველ სურათზე ნაჩვენებია სენსორის მიერ დათვლილი მანძილი მხოლოდ რიცხვების სახით, ხოლო მეორე სურათზე იგივე მონაცემები გრაფიკულად არის წარმოდგენილი. როგორც ხედავთ, მხოლოდ რიცხვებიდან კონკრეტული დასკვნის გამოტანა საკმაოდ რთულია მაშინ, როდესაც გრაფიკულ ჩანახატზე ნათლად ჩანს, რომ სენსორსა და ობიექტს შორის მანძილი თანმიმდევრულად იზრდება და შემდეგ მცირდება.

1.2. ვიზუალიზაციის როლი ტენდენციებისა და კანონზომიერებების აღმოჩენაში

ტენდენციის დანახვა: როცა სენსორის მონაცემებს გრაფიკზე ვხედავთ, ჩვენ შეგვიძლია მარტივად დავინახოთ **ტენდენცია**. მაგალითად, თუ ტემპერატურის სენსორის გრაფიკი ნელ-ნელა ზემოთ მიიწევს, ეს ნიშნავს, რომ ოთახში ტემპერატურა თანდათან იმატებს.

კანონზომიერების (Pattern) აღმოჩენა: ზოგჯერ მონაცემები პერიოდულად იცვლება. მაგალითად, ავტომატური განათების სისტემის სენსორმა შეიძლება აჩვენოს, რომ სინათლის დონე ყოველ 24 საათში ერთხელ მკვეთრად ეცემა (ღამით) და შემდეგ ისევ იმატებს (დღისით). გრაფიკზე ეს განმეორებადი „ტალღების“ სახით გამოჩნდება.



მოცემულ გრაფიკზე გამოსახულია სინათლის სენსორის ჩვენება 48 საათის განმავლობაში. სწორედ, ამგვარი გრაფიკი გვეხმარება დავინახოთ კანონზომიერება მონაცემთა შორის.

ანომალიების შემჩნევა: გრაფიკზე ძალიან მარტივია უჩვეულო მოვლენების, ანუ **ანომალიების** შემჩნევა. მაგალითად, თუ ტემპერატურის გრაფიკი უეცრად მკვეთრად გაიზარდა,ეს შეიძლება რაიმე პრობლემაზე მიუთითებდეს. რიცხვების ნაკადში ასეთი ერთი უჩვეულო რიცხვი შეიძლება უბრალოდ გამოგვრჩეს.

1.3. ვიზუალიზაციის მაგალითები მეცნიერებასა და ყოველდღიურ ცხოვრებაში

მონაცემთა ვიზუალიზაცია ჩვენ გარშემო ყველგანაა:

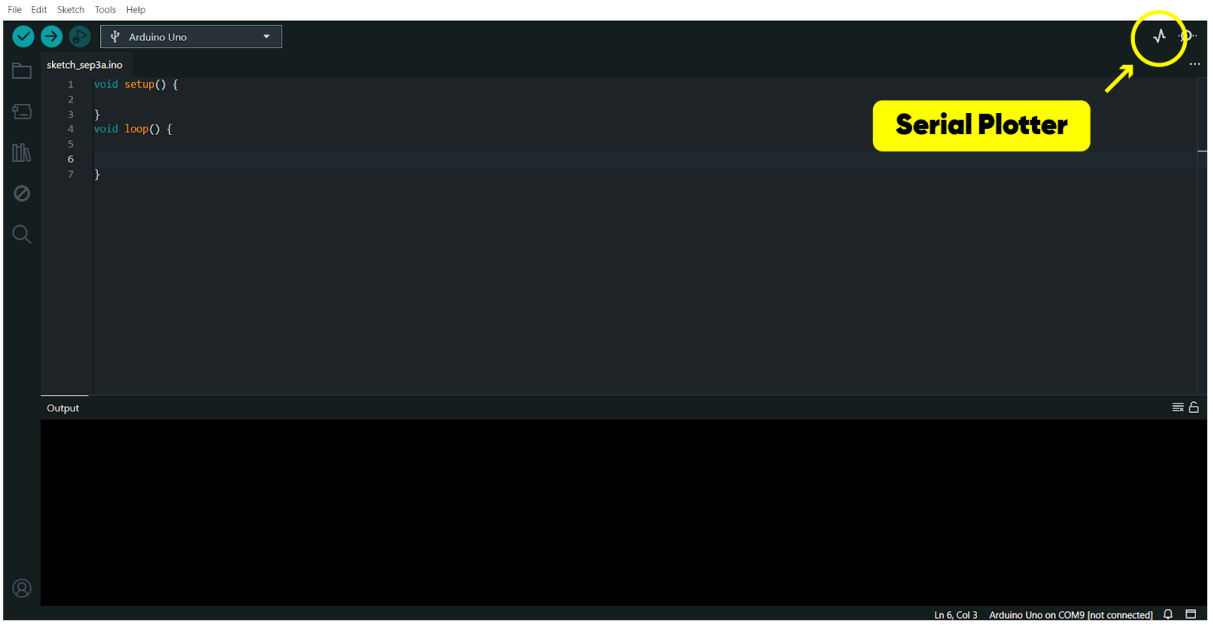
- **ამინდის პროგნოზი:** ტემპერატურის, ნალექისა და ქარის სიჩქარის გრაფიკები გვეხმარება გავიგოთ, თუ როგორი ამინდი გველის.
- **ფიტნეს აპლიკაციები:** გრაფიკები გვიჩვენებს, რამდენი ნაბიჯი გავიარეთ, როგორი იყო ჩვენი პულსი ვარჯიშის დროს ან როგორ გვეძინა ღამით.
- **ეკონომიკური სიახლეები:** აქციების ფასების ან ვალუტის კურსის ცვლილებას ყოველთვის გრაფიკების სახით წარმოადგენენ.

2. Serial Plotter-ის გამოყენება

Arduino IDE-ს (და, შესაბამისად, AnITa-ს პლატფორმას) აქვს ჩაშენებული ძალიან მოსახერხებელი ინსტრუმენტი მონაცემების ვიზუალიზაციისთვის, რომელსაც **Serial Plotter** (სერიული გრაფიკის ამგები) ჰქვია.

2.1. Serial Plotter-ის გახსნა და ინტერფეისის გაცნობა

ისევე, როგორც სერიულ მონიტორს, Serial Plotter-საც თავისი ღილაკი აქვს სიმულატორის ინტერფეისში. როცა შენი პროგრამა მუშაობს და მონაცემებს ბეჭდავს, ამ ღილაკზე დაჭერით გაიხსნება ახალი ფანჯარა, სადაც რიცხვების ნაცვლად გრაფიკს დაინახავ.



ინტერფეისი: Serial Plotter-ის ფანჯარა ძალიან მარტივია. ის შედგება გრაფიკის არისგან, სადაც რეალურ დროში იხაზება სენსორის მონაცემები.

2.2. მონაცემების სწორი ფორმატირება Serial Plotter-ისთვის (`Serial.println()`)

ერთი რიცხვი ერთ ხაზზე: იმისთვის, რომ Serial Plotter-მა გამართულად იმუშაოს, მას სჭირდება, რომ სერიულ პორტში იბეჭდებოდეს **მხოლოდ რიცხვითი მონაცემები**, თანაც თითოეული რიცხვი ახალ ხაზზე. ამიტომ, ჩვენ უნდა გამოვიყენოთ `Serial.println()` ბრძანება.

რა არ უნდა გავაკეთოთ: თუ ჩვენ დავწერთ კოდს, რომელიც ტექსტსაც ბეჭდავს, მაგალითად: `Serial.print("ტემპერატურა: "); Serial.println(temp);`, პლოტერი დაიბნევა და გრაფიკს ვერ ააგებს. ვიზუალიზაციისთვის კოდი ასეთი უნდა იყოს: `Serial.println(temp);`

2.3. გრაფიკის ღერძების (X და Y) მნიშვნელობების გააზრება

გრაფიკს, როგორც წესი, ორი ღერძი აქვს:

- **Y ღერძი (ვერტიკალური):** ეს ღერძი გვიჩვენებს სენსორის **მნიშვნელობას**. მაგალითად, `analogRead()`-ის შემთხვევაში, Y ღერძი იქნება 0-დან 1023-მდე.
- **X ღერძი (ჰორიზონტალური):** ეს ღერძი წარმოადგენს **დროს** ან, უფრო ზუსტად, მონაცემთა წერტილების თანმიმდევრობას. ყოველი ახალი `println()` ბრძანება გრაფიკს მარჯვნივ, ერთი ნაბიჯით გადასწევს.

3. გრაფიკების ანალიზი

როგორ „წავიკითხოთ“ გრაფიკი და რა ინფორმაცია მივიღოთ მისგან?

3.1. სენსორის მონაცემების ცვლილების ინტერპრეტაცია გრაფიკიდან

ხაზის მოძრაობა:

- თუ გრაფიკის ხაზი **ზემოთ მიდის**, ეს ნიშნავს, რომ სენსორის მნიშვნელობა იზრდება. (მაგალითად, ოთახში ბნელდება).
- თუ გრაფიკის ხაზი **ქვემოთ ჩამოდის**, ეს ნიშნავს, რომ სენსორის მნიშვნელობა მცირდება. (მაგალითად, ოთახი ნათდება).
- თუ ხაზი **სწორია**, ეს ნიშნავს, რომ სენსორის მონაცემი სტაბილურია.

3.2. გრაფიკზე "ხმაურის" (noise) იდენტიფიცირება და მისი შესაძლო მიზეზები

ზოგჯერ შეიძლება შეამჩნიო, რომ გრაფიკის ხაზი იდეალურად სწორი არ არის. მცირედ და სწრაფად „ხტუნაობს" ზემოთ-ქვემოთ, მაშინაც კი, როცა გარემო პირობები არ იცვლება. ამ მოვლენას **„ხმაური" (noise)** ეწოდება.

რატომ ხდება ასე? ეს სრულიად ნორმალურია. ანალოგური სენსორები ძალიან მგრძნობიარეა და შეიძლება რეაგირებდეს:

- მცირე ელექტრულ ტალღებზე;
- ახლომდებარე ელექტრო მოწყობილობებზე;
- და ა. შ.

რა უნდა გააკეთო? მთავარია, ამ მცირე "ნახტომების" მიღმა დავინახოთ მთავარი მიმართულება. წარმოიდგინე, რომ ხარ მთაში და შორიდან უყურებ გზას - ზოგან გზას ბორცვები ან ხეები ეფარება, მიუხედავად იმისა, რომ ასეთ ადგილებში გზა არ ჩანს მთავარი მიმართულება მაინც ნათელია. ასევეა გრაფიკშიც - შენ შეგიძლია ამოიცნო მთავარი ტენდენცია: იზრდება თუ მცირდება მაჩვენებელი.

3.3. რამდენიმე მონაცემის ერთდროულად გამოტანა და შედარება გრაფიკზე

Serial Plotter-ს ერთდროულად რამდენიმე გრაფიკის აგებაც შეუძლია. ამისთვის, ჩვენ ერთ ხაზზე უნდა დავბეჭდოთ რამდენიმე რიცხვი, რომლებიც ერთმანეთისგან მძიმით ან ინტერვალით იქნება გამოყოფილი.

მაგალითი:

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

ამ შემთხვევაში, პლოტერი დახაზავს ორ განსხვავებული ფერის გრაფიკს ერთს სინათლის სენსორისთვის, მეორეს კი – ტემპერატურისთვის. ამ თემას უფრო დეტალურად შემდეგ შეხვედრაზე განვიხილავთ.

მაგალითის ვიზუალური მხარე tinkercad-ში

https://drive.google.com/drive/folders/1pzRu7t3LTUaN4cxNjfQZ7rOiu0-UBz_S //9.1

როგორც უკვე ვთქვით, მოცემული მაგალითით ორი განსხვავებული ფერის გრაფიკი მივიღეთ: ლურჯი დიაგრამა ასახავს სინათლის სენსორის მნიშვნელობებს, ხოლო ყვითელი დიაგრამა – ტემპერატურის ცვლილებებს.

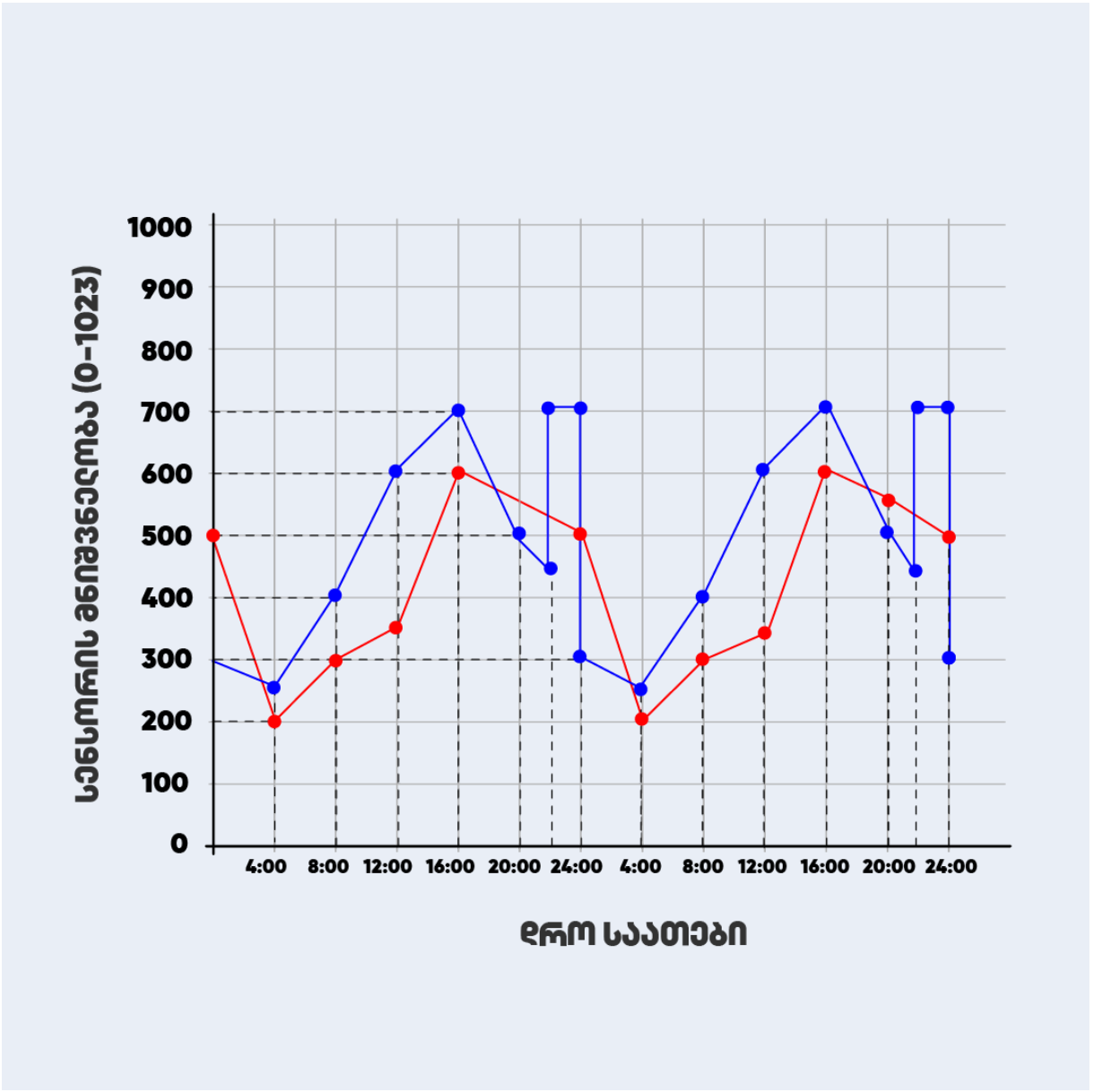
სავარჯიშო:

სიტუაცია:

ანას ოთახში სინათლის და ტემპერატურის სენსორებია დაყენებული და ის 48 საათიან ექსპერიმენტს ახორციელებს. მან Arduino-ზე ჩაწერა პროგრამა, რომელიც ორივე სენსორის მონაცემებს ერთდროულად აგროვებს და Serial Plotter-ზე აჩვენებს. თუმცა, დაავიწყდა რომელი გრაფიკი რომელი სენსორის მონაცემებს ასახავდა. შენი დავალებაა ქვემოთ მოცემული სურათის დახმარებით თითოეულ სენსორს თავისი გრაფიკი მოუძებნო.

მინიშნება1: ბოლო ორი დღის განმავლობაში ანა საღამოს 10 საათზე შედიოდა თავის ოთახში, ორი საათის განმავლობაში ანთებულ შუქზე კითხულობდა წიგნს, ხოლო 12 საათისთვის სინათლეს აქრობდა და დასაძინებლად ემზადებოდა.

მინიშნება2: ტემპერატურა 12 საათის შემდეგ მკვეთრად მცირდება



დავალება 9: „სენსორის კარდიოგრამა“

შენი ამოცანაა, დაწერო პროგრამა, რომელიც წაიკითხავს სინათლის სენსორის მონაცემს და გამართავს მას Serial Plotter-ზე ვიზუალიზაციისთვის. პროგრამის გაშვების შემდეგ გამოიყენე სლაიდერი, შეცვალე ვირტუალური განათების დონე და დააკვირდი, როგორ იცვლება გრაფიკი.

დავალება 9.1: შეცდომის პოვნა და გასწორება

პროგრამისტმა კოდის წერისას შეცდომა დაუშვა. ის სენსორის მონაცემთან ერთად ტექსტსაც ბეჭდავდა, რის გამოც Serial Plotter–ი გრაფიკს ვერ აგებდა. შენი ამოცანაა, იპოვო და წაშალო კოდის ის ნაწილი, რომელიც პლოტერს მუშაობაში ხელს უშლის.

მინიშნება: Serial Plotter-ს მხოლოდ რიცხვების „ესმის“. დარწმუნდი, რომ `loop` ფუნქციაში `Serial.println()` ბრძანებას გადაეცემა მხოლოდ რიცხვითი ცვლადი.

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

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

  // იპოვე და წაშალე ზედმეტი ბრძანება ამ ნაწილში
  Serial.print("Sensor Value: ");
  Serial.println(sensorValue);

  delay(100);
}
```

დავალება 9.2: კოდის დასრულება

პროგრამის ძირითადი ნაწილი უკვე აგებულია, სენსორის მონაცემი იკითხება და ინახება ცვლადში. თუმცა, მთავარი ნაწილი – ამ მონაცემის Serial Plotter-ისთვის სწორ ფორმატში დაბეჭდვა – გამორჩენილია. შენი ჯერია! დაამატე ერთი ბრძანება, რათა პროგრამა დასრულდეს.

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

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

  // --- ჩაამატე ბრძანება აქ ---
  // დაბეჭდე sensorValue ცვლადის მნიშვნელობა ისე, რომ კურსორი ახალ ხაზზე გადავიდეს.

  delay(100);
}
```

დავალება 9.3: 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. დაბეჭდავს ამ მნიშვნელობას ახალ ხაზზე (Serial Plotter-ისთვის).
4. გააკეთებს 100 მილიწამიან პაუზას.

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

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

სწორი პასუხი (პროგრამული კოდი სრულად):

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

void loop() {
  // წაიკითხე მნიშვნელობა ანალოგური პინიდან A0
  int sensorValue = analogRead(A0);

  // დაბეჭდე მხოლოდ ცვლადის მნიშვნელობა, რათა პლოტერმა იმუშაოს.
  Serial.println(sensorValue);

  // დაამატე მცირე პაუზა.
  delay(100);
}
```

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": "serial_println_used",
          "name": "Serial.println() used",
          "kind": "require_regex",
```

```
          "pattern":
"Serial\\.println\\s*\\(\\s*(?:sensorValue|sensor|value|v
al|light)\\s*\\)",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "გამოიყენეთ Serial.println()
სენსორის მნიშვნელობის დასაბეჭდად:
Serial.println(sensorValue);",
          "msg_pass": "Serial.println() სწორად არის
გამოყენებული."
        },
        {
          "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 for plotter",
          "kind": "require_ordered_regex",
          "patterns": [
            "analogRead\\s*\\(\\s*A0\\s*\\)",

"Serial\\.println\\s*\\(\\s*(?:sensorValue|sensor|value|v
al|light)\\s*\\)",
            "delay\\s*\\(\\s*\\d+\\s*\\)"
          ],
          "flags": "iu",
```

```
            "must_pass": true,
            "source": "stripped",
            "msg_fail": "loop() თანმიმდევრობა უნდა იყოს:
  analogRead(A0) → Serial.println(sensorValue) → delay()",
            "msg_pass": "loop() თანმიმდევრობა პლოტერისთვის
  სწორია."
          }
        ]
      }
    ]
  }
```

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>Light Level: <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 (Dark)</span>
        <span>1023 (Bright)</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 = 512;
         }
     }
 };
```