შეხვედრა 13: ჭკვიანი განათების სისტემა - ვირტუალური პროტოტიპი

გამარჯობა! ჩვენი მოგზაურობა ვირტუალურ ლაბორატორიაში დასასრულს უახლოვდება. დღეს შენ აღარ ხარ ის მოსწავლე, რომელიც ცალკეულ ბრძანებებს სწავლობდა. დღეს შენ ხარ ინჟინერი, დიზაინერი და პროგრამისტი ერთდროულად. ჩვენი ამოცანაა, გავაერთიანოთ მთელი ჩვენი ცოდნა და შევქმნათ პროექტი, რომელიც რეალურ პრობლემას გადაჭრის – ავტომატური სანათი, რომელიც თავისით მიხვდება, როდის უნდა აინთოს.

1. პროექტის დეკომპოზიცია

დიდი პროექტები ყოველთვის პატარა, მართვად ნაწილებად იყოფა. ამ პროცესს **დეკომპოზიცია** ჰქვია. **„გაყავი და იბატონე“ პრინციპი** მუშაობს პროგრამირებაშიც.

მისი არსი ისაა, რომ **დიდი და რთული ამოცანა დავყოთ პატარა ამოცანებად**, რომლებიც უფრო მარტივად შესასრულებელია.

შემდეგ, როცა თითოეულ პატარა ამოცანას გადავჭრით, საბოლოოდ მთელ პრობლემასაც მოვაგვარებთ. მოდი, დავშალოთ ჩვენი ამოცანა ნაწილებად.

1.1. პროექტის საბოლოო მიზნის განსაზღვრა: ავტომატური სანათი, რომელიც რეაგირებს ირგვლივ არსებულ განათების ცვლილებაზე

რა პრობლემას ვჭრით? წარმოიდგინე ქუჩის განათება. არავინ რთავს და თიშავს მას ხელით ყოველ დილით და საღამოს. ის ავტომატურია. ჩვენი მიზანია, შევქმნათ ასეთი სისტემის მინიატურული პროტოტიპი.

საბოლოო შედეგი: ჩვენი პროექტი მუდმივად უნდა აკონტროლებდეს გარემოს განათების დონეს. როცა საკმარისად დაბნელდება, მან ავტომატურად უნდა ჩართოს შუქდიოდი, ხოლო როცა გათენდება –გამოირთოს.

1.2. საჭირო კომპონენტების ჩამონათვალი: არდუინო, ფოტორეზისტორი, რეზისტორი, შუქდიოდი

იმისთვის, რომ ჩვენი მიზანი რეალობად ვაქციოთ, ჩვენ გვჭირდება გუნდი, რომლის წევრებიც არიან:

- **არდუინო:** ჩვენი პროექტის „ტვინი“, რომელიც მიიღებს გადაწყვეტილებას.
- **ფოტორეზისტორი (სინათლის სენსორი):** ჩვენი სისტემის „თვალი“, რომელიც გაზომავს სინათლის დონეს.
- **შუქდიოდი:** ჩვენი სისტემის „ხელი“, რომელიც შეასრულებს მოქმედებას – აინთება.
- **რეზისტორი:** შუქდიოდის „მცველი“, რომელიც მას გადაწვისგან დაიცავს.

AnITa-ს პლატფორმაზე ეს კომპონენტები უკვე დაკავშირებულია შესაბამის პინებთან, რაც ჩვენს სამუშაოს ამარტივებს.

1.3. პროექტის ალგორითმი და ფსევდოკოდის შედგენა

ალგორითმი: პროგრამამ უსასრულოდ უნდა გაიმეოროს შემდეგი მოქმედებები: წაიკითხოს, რას ხედავს სინათლის სენსორი. თუ სენსორის მონაცემი მიუთითებს, რომ ბნელა, ჩართოს ნათურა. თუ სენსორის მონაცემი მიუთითებს, რომ გარემო განათებულია, გამორთოს ნათურა.

ფსევდოკოდი (ჩვენი გეგმა):

```
პროგრამის დაწყება (SETUP):
  განსაზღვრე, რომ შუქდიოდის პინი არის გამომავალი (OUTPUT).
  ჩართე სერიული მონიტორი (ტესტირებისთვის).

მთავარი ციკლი (LOOP):
  წაიკითხე მონაცემი სინათლის სენსორიდან.
  შეინახე ეს მონაცემი ცვლადში.

  თუ სენსორის მონაცემი მეტია "სიბნელის ზღვარზე":
    ჩართე შუქდიოდი.
  სხვა შემთხვევაში:
    გამორთე შუქდიოდი.
```

2. კოდის სტრუქტურის დაგეგმვა

ახლა, როცა გეგმა გვაქვს, გადავთარგმნოთ ის არდუინოსთვის გასაგებ ენაზე.

2.1. რა უნდა მოხდეს `setup()` ფუნქციაში: პინების რეჟიმების დაყენება, სერიული პორტის ინიციალიზაცია

`setup()` ფუნქცია ჩვენი მოსამზადებელი ეტაპია. აქ ვასრულებთ ყველა იმ მოქმედებას, რომელიც მხოლოდ ერთხელ, პროგრამის დასაწყისშია საჭირო.

`pinMode(ledPin, OUTPUT);` – ჩვენ ვეუბნებით არდუინოს, რომ პინი, რომელზეც შუქდიოდია მიერთებული, იმუშავებს როგორც გამომავალი, ანუ ის გააგზავნის სიგნალს.

`Serial.begin(9600);` – ჩვენ ვრთავთ სერიულ მონიტორს. ეს არ არის პროექტის მუშაობისთვის აუცილებელი, მაგრამ ძალიან სასარგებლოა შეცდომების მოძიებასა და გასწორებისათვის (debugging) პროცესში. მონიტორი საშუალებას მოგვცემს დავინახოთ, რას ზომავს სენსორი.

2.2. რა უნდა მოხდეს `loop()` ფუნქციაში: სენსორიდან ინფორმაციის ამოკითხვა, მონაცემის შედარება ზღვრულ პარამეტრთან, შუქდიოდის მართვა

`loop()` ფუნქცია ჩვენი პროექტის გულია. აქ ხდება ყველა მთავარი უსასრულოდ გამეორებადი მოქმედება.

1. `int lightValue = analogRead(A0);` – ვკითხულობთ მონაცემს სინათლის სენსორიდან.

2. `if (lightValue > threshold)` – ვამოწმებთ პირობას.
3. `digitalWrite(ledPin, HIGH);` – ვანთებთ ნათურას, თუ პირობა სრულდება.
4. `digitalWrite(ledPin, LOW);` – ვთიშავთ ნათურას, თუ პირობა არ სრულდება.

2.3. სინათლის დონის ზღვრული მნიშვნელობის (threshold) შერჩევა და დაკალიბრება

რა არის ზღვრული პარამეტრი? ეს არის ჩვენი პროგრამის „გადაწყვეტილების მიღების წერტილი“. ეს არის რიცხვი, რომელსაც სენსორის მონაცემს ვადარებთ.

როგორ შევარჩიოთ სწორი ზღვარი? ამ პროცესს **კალიბრაცია** ჰქვია.

1. ჯერ დაწერე მარტივი კოდი, რომელიც მხოლოდ სენსორის მონაცემს წაიკითხავს და დაბეჭდავს სერიულ მონიტორზე (როგორც მე-5 დავალებაში).
2. გაუშვი სიმულაცია და გამოიყენე სლაიდერი. დააკვირდი, რა რიცხვებს გაძლევს სენსორი, როცა „ძალიან ნათელია“ და რა რიცხვებს – როცა „ძალიან ბნელა“.
3. შეარჩიე რიცხვი ამ ორ უკიდურესობას შორის. მაგალითად, თუ განათებულ გარემოში რიცხვი არის ~200, ხოლო სიბნელეში ~800, ზღვრის სასურველი მნიშვნელობა შეიძლება იყოს 500 ან 600.

ექსპერიმენტი: ნუ შეგეშინდება, სცადო ზღვრის სხვადასხვა მნიშვნელობა და მიაგნო იმ მაჩვენებელს, რომელიც შენი პროექტისათვის ოპტიმალური იქნება.

3. პროექტის აწყობა და ტესტირება

3.1. სქემის აწყობა ვირტუალურ სიმულატორში

AnITa-ს პლატფორმაზე სქემა უკვე აწყობილია. შენ მხოლოდ უნდა იცოდე, რომელი კომპონენტი რომელ პინზეა მიერთებული:

- **ფოტორეზისტორი:** A0 პინი.
- **შუქდიოდი:** მე-13 პინი.

3.2. კოდის დაწერა და კომპილაცია

ახლა შენი ჯერია! გახსენი კომპილატორი და ჩვენი გეგმის მიხედვით დაწერე სრული კოდი `setup` და `loop` ფუნქციებისთვის.

3.3. პროექტის ტესტირება და გამართვა (debugging) სიმულაციისას

ტესტირება: კოდის დაწერის შემდეგ, გაუშვი სიმულაცია.

1. გამოიყენე სინათლის სლაიდერი. როცა სლაიდერს „სიბნელისკენ“ წაიღებ, შუქდიოდი უნდა აინთოს.
2. როცა სლაიდერს „სინათლისკენ“ წაიღებ, შუქდიოდი უნდა ჩაქრეს.

პროგრამირებასა და ინჟინერიაში ხშირია, **როცა ერთი შეხედვით ყველაფერი სწორად არის გაკეთებული,** მაგრამ შედეგი მაინც არ ემთხვევა ჩვენს მოლოდინს. ასეთ შემთხვევაში საჭიროა თითოეული დეტალის ყურადღებით გადამოწმება.

- **ლოგიკა სწორია?**
  ხშირად `<` და `>` სიმბოლოები `if` პირობაში ერევათ. ერთმა შეცვლილმა სიმბოლომ შეიძლება სრულიად შეცვალოს პროგრამის მუშაობა.

- **ზღვარი სწორია?**
  გახსენი სერიული მონიტორი და ნახე, რა რიცხვებს აბრუნებს სენსორი. შეიძლება შენი არჩეული ზღვარი ძალიან მაღალი ან ძალიან დაბალი იყოს, რის გამოც პროგრამა არ რეაგირებს.

- **პინების ნომრები სწორია?**
  გადაამოწმე, კოდში მითითებული პინები ზუსტად ემთხვევა თუ **არა** შენს სქემას. მაგალითად, A0 უნდა იყოს სენსორისთვის, ხოლო 13 – **შუქდიოდისთვის.**

დამატებითი რჩევები:

- ნუ შეგეშინდება შეცდომების — ისინი სწავლების პროცესის ნაწილია.

- შეცდომის აღმოჩენისას, **სცადე სირთულის პატარა ნაწილებად დაყოფა და სათითაოდ** შემოწმება.

- ყოველთვის გამოიყენე სერიული მონიტორი „დასაფიქსირებლად“, რათა გაიგო სასურველი ცვლადის მნიშვნელობა და მიხვდე სად ჩერდება ან ცდება პროგრამა.

- თუ პრობლემა ვერ მოაგვარე, გააკეთე შესვენება და შემდეგ ახალი თვალით შეხედე კოდს — ეს ხშირად დაგეხმარება.

დავალება 13: „ააწყე ავტომატური სანათი“

შენი ამოცანაა, დამოუკიდებლად დაწერო სრული პროგრამა „ავტომატური სანათისთვის“. პროგრამამ უნდა წაიკითხოს მონაცემები სინათლის სენსორიდან (A0) და თუ გარემო საკმარისად ბნელია (სენსორის მნიშვნელობა შენ მიერ შერჩეულ ზღვარზე მეტია), უნდა აანთოს **შუქდიოდი** (პინი 13).

დავალების სიმულაცია tinkercad-ში:

https://drive.google.com/drive/folders/1pzRu7t3LTUaN4cxNjfQZ7rOiu0-UBz_S //13.1

დავალება 13.1: შეცდომის პოვნა და გასწორება

პროგრამისტმა კოდის წერისას შეცდომა დაუშვა. მას setup ფუნქციაში შუქდიოდის პინის რეჟიმის განსაზღვრა დაავიწყდა. შენი ამოცანაა, იპოვო და დაამატო ერთი გამოტოვებული ბრძანება, რათა პროგრამამ პირობის შესაბამისად იმუშაოს.

მინიშნება: გაიხსენე, რომელი ფუნქციით ვეუბნებით არდუინოს, რომ პინი იმუშავებს როგორც გამომავალი (OUTPUT)?

```
void setup() {
  Serial.begin(9600);
  // --- აქ აკლია ერთი ბრძანება ---
}

void loop() {
  int lightValue = analogRead(A0);
  int threshold = 500;

  if (lightValue > threshold) {
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }
}
```

დავალება 13.2: კოდის დასრულება

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

```
void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int lightValue = analogRead(A0);
  int threshold = 500;

  // --- ჩაამატე if/else ბლოკი აქ ---
  // 1. შეამოწმე, თუ lightValue მეტია threshold-ზე.
```

```
  // 2. თუ პირობა სრულდება, ჩართე მე-13 პინი.
  // 3. სხვა შემთხვევაში, გამორთე მე-13 პინი.
}
```

დავალება 13.3: შეიმუშავე პროგრამული კოდი

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

1. `setup` ფუნქციაში მოამზადებს მე-13 პინს `OUTPUT` რეჟიმში.
2. `loop` ფუნქციაში წაიკითხავს მონაცემს A0 პინიდან.
3. თუ მონაცემი 500-ზე მეტია, ჩართავს მე-13 პინს, სხვა შემთხვევაში – გამორთავს.

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

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

სწორი პასუხი (პროგრამული კოდი სრულად):

```
void setup() {
  // LED ნათურის პინის მომზადება გამომავალ რეჟიმში
  pinMode(13, OUTPUT);
  // სერიული მონიტორის ჩართვა (სურვილისამებრ, ტესტირებისთვის)
  Serial.begin(9600);
}

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

  // 2. განსაზღვრე სიბნელის ზღვარი.
  int threshold = 500;

  // 3. შეამოწმე პირობა: თუ სენსორის მნიშვნელობა მეტია ზღვარზე (ანუ ბნელა)...
  if (lightValue > threshold) {
    // ...ჩართე LED ნათურა.
    digitalWrite(13, HIGH);
  } else {
```

```
    // ...სხვა შემთხვევაში (ანუ ნათელია), გამორთე LED ნათურა.
    digitalWrite(13, LOW);
  }
}
```

JSON

```
{
  "version": 1,
  "board": "arduino:avr:uno",
  "steps": [
    {
      "type": "compile"
    },
    {
      "type": "static",
      "rules": [
        {
          "id": "pin_led_output",
          "name": "LED pin mode",
          "kind": "require_regex",
          "pattern":
"pinMode\\s*\\(\\s*13(?!\\d)\\s*,\\s*OUTPUT\\s*\\)\\s*;?",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
```

```
          "msg_fail": "LED პინი (13) უნდა იყოს OUTPUT რეჟიმში: pinMode(13, OUTPUT);",
          "msg_pass": "LED პინი სწორად არის OUTPUT რეჟიმში."
        },
        {
          "id": "analog_read_a0",
          "name": "Read light sensor",
          "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": "სინათლის სენსორის წაკითხვა სწორადაა."
        },
        {
          "id": "threshold_variable",
          "name": "Threshold variable",
          "kind": "require_regex",
          "pattern": "\\b(?:int|float|long)\\s+(?:threshold|zghvari|limit)\\s*=\\s*\\d+",
          "flags": "iu",
          "must_pass": true,
```

```
          "source": "stripped",
          "msg_fail": "უნდა განსაზღვროთ ზღვრის ცვლადი: int
threshold = 500;",
          "msg_pass": "ზღვრის ცვლადი განსაზღვრულია."
        },
        {
          "id": "light_condition",
          "name": "Check darkness condition",
          "kind": "require_regex",
          "pattern":
"if\\s*\\(\\s*(?:lightValue|light|sensorValue|sensor)\\s*>\\s*(?:
threshold|zghvari|\\d+)\\s*\\)",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "შეამოწმეთ სიბნელის პირობა: if (lightValue
> threshold)",
          "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 LED-ის გასამორთად.",
          "msg_pass": "else ბლოკი არსებობს."
        },
        {
          "id": "loop_sequence",
          "name": "Correct loop sequence",
          "kind": "require_ordered_regex",
          "patterns": [
            "analogRead\\s*\\(\\s*A0\\s*\\)",
"if\\s*\\(\\s*(?:lightValue|light|sensorValue|sensor)\\s*>\\s*(?:threshold|zghvari|\\d+)\\s*\\)"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "loop()-ში თანმიმდევრობა: ჯერ analogRead(A0), შემდეგ if შედარება.",
          "msg_pass": "loop() თანმიმდევრობა სწორია."
        },
        {
          "id": "if_then_high",
          "name": "LED HIGH in if block",
```

```
          "kind": "require_ordered_regex",
          "patterns": [

"if\\s*\\(\\s*(?:lightValue|light|sensorValue|sensor)\\s*>\\s*(?:threshold|zghvari|\\d+)\\s*\\)",

"digitalWrite\\s*\\(\\s*13(?!\\d)\\s*,\\s*HIGH\\s*\\)"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "როცა lightValue > threshold, if ბლოკში უნდა იყოს digitalWrite(13, HIGH);",
          "msg_pass": "if ბლოკში LED სწორად ირთვება (HIGH)."
        },
        {
          "id": "else_then_low",
          "name": "LED LOW in else block",
          "kind": "require_ordered_regex",
          "patterns": [
            "\\}\\s*else\\s*\\{",
            "digitalWrite\\s*\\(\\s*13(?!\\d)\\s*,\\s*LOW\\s*\\)"
          ],
          "flags": "iu",
          "must_pass": true,
```

```
            "source": "stripped",
            "msg_fail": "else ბლოკში უნდა იყოს digitalWrite(13,
  LOW);",
            "msg_pass": "else ბლოკში LED სწორად ირთვება (LOW)."
          }
        ]
      }
    ]
  }
```

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="leds" style="margin-bottom: 20px;">
    <div style="display: flex; align-items: center; gap: 10px;">
        <label style="font-weight: bold;">LED on Pin 13:</label>
        <wokwi-led color="red" pin="13" id="led-13"></wokwi-led>
```

```
    </div>
</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, ultrasonicDistance) {
        // Initialize A0 sensor if not set
        if (sensorValues.value.A0 === undefined) {
            sensorValues.value.A0 = 300; // Default: Light (below
threshold)
        }

        // Set up LED listener on Port B (pin 13 is PB5, bit 5)
        runner.portB.addListener(function(value) {
            // Pin 13 is bit 5 of Port B
            const pin13State = (value >> 5) & 1;
```

```
            // Update the LED element
            const ledElement = document.getElementById('led-13');
            if (ledElement) {
                ledElement.value = pin13State ? true : false;
            }
        });
    }
};
```