შეხვედრა 4: კოდიდან მოქმედებამდე - როგორ ავანთოთ შუქდიოდი

გამარჯობა! წინა შეხვედრებზე ჩვენ გავეცანით არდუინოს და ვისწავლეთ პროგრამირების ენის ძირითადი ელემენტები. დღეს კი ამ ცოდნას გავაერთიანებთ და ჩვენს პირველ ნამდვილ პროექტს შევქმნით. ჩვენ შევაბიჯებთ AnITa-ს ვირტუალურ ლაბორატორიაში, სადაც თეორია რეალობად იქცევა.

1. ვირტუალური ლაბორატორიის გაცნობა

AnITa-ს სიმულატორი არის შენი პირადი ციფრული სახელოსნო. აქ შეგიძლია, უსაფრთხოდ და თავისუფლად ჩაატარო ექსპერიმენტები. მოდი, გავეცნოთ მის ძირითად ნაწილებს.

1.1. AnITa-ს პლატფორმის ინტერფეისის მიმოხილვა

როცა სიმულატორს გახსნი, დაინახავ სამ ძირითად სამუშაო ზონას, რომლებიც ერთად მუშაობს:

- **დავალების პირობა:** ეკრანის ერთ ნაწილში მოცემულია შენი მიმდინარე დავალების დეტალური აღწერა. სწორედ აქ გაიგებ, რა უნდა შეასრულოს შენმა პროგრამამ.
- **კომპილატორი (კოდის რედაქტორი):** ეს არის ცენტრალური სივრცე, სადაც შენ წერ, ასწორებ და ამუშავებ შენს სკეტჩს (პროგრამას).
- **არდუინოს პანელი:** ეს არის ვიზუალური ნაწილი, სადაც ხედავ არდუინოს დაფას და მასთან დაკავშირებულ კომპონენტებს.

1.2. არდუინოს პანელი და გარემოს მართვა

მზა სქემა: სხვა სიმულატორებისგან განსხვავებით, აქ სქემის აწყობა თავიდან არ გიწევს. თითოეული დავალებისთვის საჭირო ყველა კომპონენტი (ნათურები, სენსორები) უკვე წინასწარ არის მიერთებული არდუინოს დაფასთან. ეს გეხმარება ფოკუსირება მხოლოდ პროგრამის ლოგიკაზე მოახდინო.

ინტერაქტიული სლაიდერი: არდუინოს გამოსახულების თავზე ხშირად დაგხვდება სლაიდერი. მისი საშუალებით შეგიძლია შეცვალო ვირტუალური გარემოს პირობები. მაგალითად, თუ ფოტორეზისტორთან მუშაობ, სლაიდერით შეძლებ, „მოუმატო“ ან „დაუწიო“ ვირტუალურ განათებას და დააკვირდე, როგორ რეაგირებს შენი პროგრამა.

1.3. კოდთან მუშაობა და დახმარების ფუნქცია

კომპილატორი: ეს შენი მთავარი სამუშაო ინსტრუმენტია. აქ შენ წერ კოდს და ტვირთავ ვირტუალურ არდუინოში (ღილაკი დასრულება), რათა ნახო, როგორ მუშაობს ის.

დახმარების ღილაკი: თუ დავალების შესრულებისას სირთულეს წააწყდი ან გინდა შენი დაწერილი კოდი სწორ პასუხს შეადარო, „დავალების პირობის“ სექციაში ყოველთვის შეგიძლია გამოიყენო დახმარების ღილაკი. ამ ღილაკზე დაჭერისას

შეგიძლია დამატებითი მინიშნებები ნახო. ასევე, ღილაკი **სწორი პასუხი (პროგრამული კოდი სრულად)** დაგეხმარება ნახო გამართული პროგრამული კოდი სრულად.

2. ციფრული გამომავალი სიგნალი (Digital Output)

ჩვენი პირველი ამოცანაა, არდუინომ ფიზიკური მოქმედება შეასრულოს – აანთოს შუქდიოდი. ამისთვის მან უნდა გააგზავნოს სიგნალი, ანუ მიაწოდოს ძაბვა (5 ვოლტი) თავისი ერთ-ერთი პინიდან.

2.1. `pinMode(pin, OUTPUT)` ფუნქცია: პინის რეჟიმის გამომავალზე დაყენება

ვიდრე არდუინო პინს გამოიყენებს, მან უნდა იცოდეს, რა არის ამ პინის დანიშნულება. პინს შეიძლება ჰქონდეს ორი ძირითადი როლი:

- **OUTPUT (გამომავალი):** პინი აგზავნის სიგნალს (მაგალითად, ანთებს ნათურას).
- **INPUT (შემავალი):** პინი იღებს სიგნალს (მაგალითად, კითხულობს, დააჭირეს თუ არა ღილაკს თითი).

`pinMode()` ფუნქცია სწორედ ამ როლს განსაზღვრავს. რადგან ჩვენ შუქდიოდის ანთება გვინდა, პინის რეჟიმი უნდა იყოს `OUTPUT`. ეს ბრძანება იწერება `setup()` ფუნქციაში, რადგან პინის როლის განსაზღვრა მხოლოდ ერთხელ, პროგრამის დასაწყისში გვჭირდება.

სინტაქსი: `pinMode(13, OUTPUT);` (ეს ნიშნავს: „მე-13 პინი იმუშავებს როგორც გამომავალი“)

2.2. `digitalWrite(pin, VALUE)` ფუნქცია: პინზე ძაბვის მიწოდება (HIGH) და შეწყვეტა (LOW)

მას შემდეგ, რაც პინს თავის დავალებას მივცემთ, `digitalWrite()` ფუნქციით ვაძლევთ კონკრეტულ ბრძანებას. ციფრულ პინს მხოლოდ ორი ბრძანება ესმის:

- **HIGH (მაღალი):** ჩაირთოს, ანუ მიაწოდოს ძაბვა (5 ვოლტი).
- **LOW (დაბალი):** გამოირთოს, ანუ შეწყვიტოს ძაბვის მიწოდება (0 ვოლტი).

ეს ბრძანება, როგორც წესი, `loop()` ფუნქციაში იწერება, რადგან მოქმედების შესრულება პროგრამის მუშაობის განმავლობაში გვინდა.

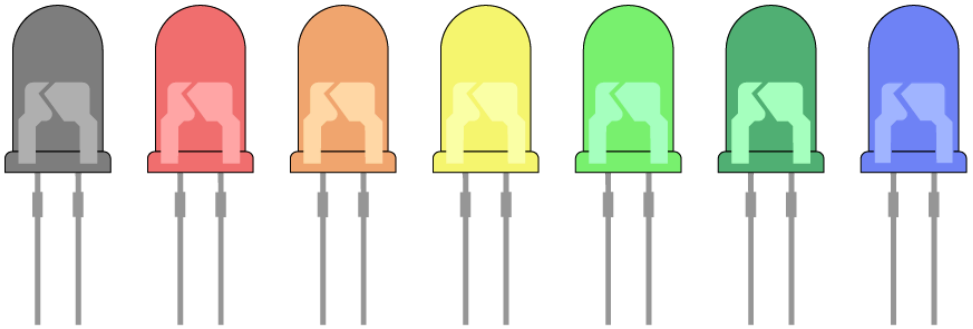
სინტაქსი: `digitalWrite(13, HIGH);` (ეს ნიშნავს: „მე-13 პინს მიაწოდე ძაბვა“).

2.3. შუქდიოდის, როგორც ციფრული სიგნალის ინდიკატორის, გამოყენება

შუქდიოდი (LED – Light Emitting Diode) არის მცირე ზომის ნათურა, რომელიც მაშინ ანათებს, როცა მასში დენი გადის. AnITa-ს პლატფორმაზე ის უკვე დაკავშირებულია არდუინოსთან. ჩვენ მხოლოდ მისი მართვა გვჭირდება კოდით.

"ნათურა = მცირე ტელევიზორი"

წარმოიდგინე, რომ შუქდიოდი არის მცირე ზომის ტელევიზორი, ხოლო არდუინო - მისი დისტანციური მართვა. digitalWrite() ფუნქცია ღილაკია პულტზე, რომელიც ნათურას უბრძანებს "ჩაირთე" ან "გამოირთე".



რეალურ პროექტებზე მუშაობისას შუქდიოდის სწორად ჩასართავად აუცილებელია მისი ფეხების განსხვავების გათვალისწინება. გრძელი ფეხი, ანუ ანოდი, უნდა მიერთდეს ძაბვის წყაროსთან (Arduino-ს შემთხვევაში – 5V), ხოლო შედარებით მოკლე ფეხი, ანუ კათოდი, მიწასთან (GND). წინააღმდეგ შემთხვევაში შუქდიოდი გადაიწვება.

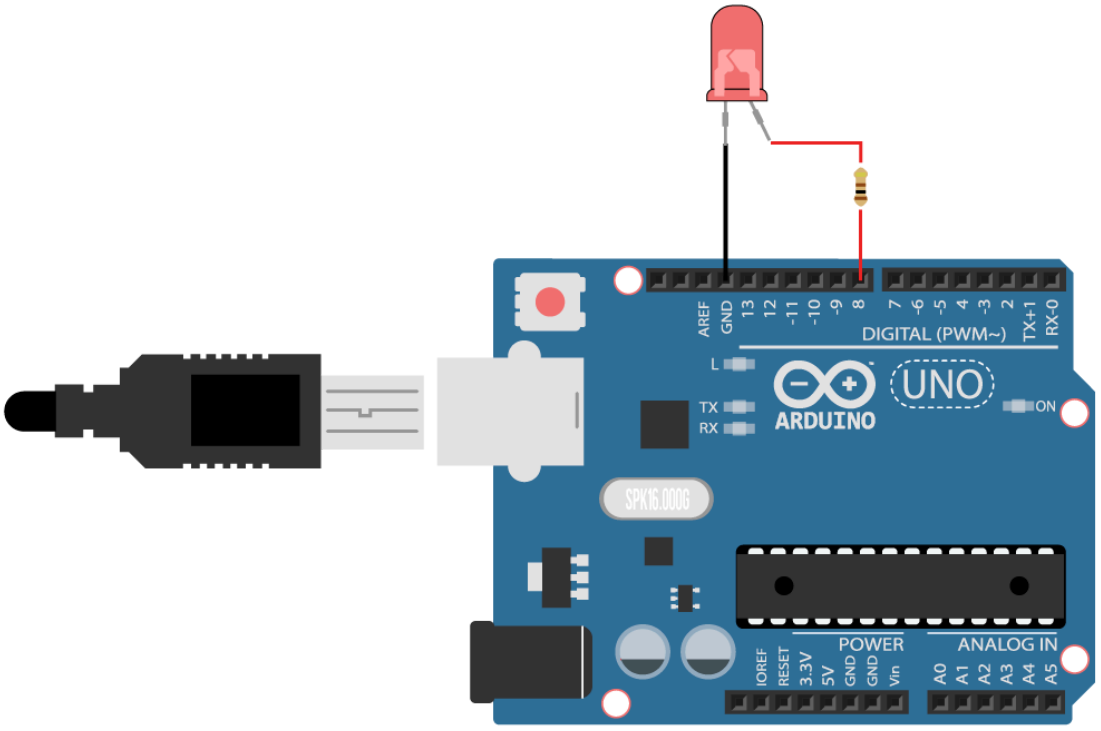
რეზისტორი – შუქდიოდის მცველი: რეალურ ცხოვრებაში შუქდიოდს გადაწვისგან რეზისტორი იცავს. ყველა რეზისტორს აქვს გარკვეული წინაღობა, რაც

უზრუნველყოფს რომ ნათურა არ გაფუჭდეს. რაც უფრო პატარაა წინაღობა, მით უფრო მკაფიოდ ანათებს ნათურაც. ჩვენს სიმულატორში ესეც უკვე გათვალისწინებულია და სქემაში ჩართულია, ამიტომ დამატებით მასზე ზრუნვა არ გვჭირდება.

სავარჯიშო 1:

#დააკვირდი ქვემოთ მოცემულ სურათს სურათზე აწყობილი წრედი პასუხისმგებელია LED ნათურის ანთებაზე. ქვემოთ მოცემული ბრძანებებიდან აირჩიე სწორი პასუხი

pinMode(8,INPUT); თუ pinmode(8,OUTPUT);



3. დროის კონტროლი პროგრამაში

არდუინო ბრძანებებს წარმოუდგენელი სისწრაფით ასრულებს (მილიონობით ოპერაცია წამში). თუ ჩვენ უბრალოდ დავწერთ კოდს, რომელიც ნათურას ჩართავს და მაშინვე

გამორთავს, ამას ადამიანის თვალი ვერც კი შეამჩნევს. ამიტომ, ჩვენ გვჭირდება დროის მართვის ინსტრუმენტი.

3.1. delay(ms) ფუნქცია: პროგრამის შესრულების შეჩერება მილიწამებში

- delay() ფუნქცია არის ჩვენი „პაუზის“ ღილაკი. ის აჩერებს პროგრამის შესრულებას ზუსტად იმ დროით, რასაც ფრჩხილებში მივუთითებთ.
- **მნიშვნელოვანია:** დრო იზომება **მილიწამებში (ms)**.
    - 1000 მილიწამი = 1 წამი.
- **სინტაქსი:** delay(1000); (ეს ნიშნავს: „დააპაუზე პროგრამა 1 წამით“).

3.2. დროის პარამეტრების ცვლილების გავლენა პროგრამის მუშაობაზე

- თუ გვინდა, რომ ნათურა 2 წამით ენთოს, დავწერთ delay(2000);.
- თუ გვინდა, რომ ნახევარი წამით ენთოს, დავწერთ delay(500);.
- ამ პარამეტრის ცვლილებით ჩვენ შეგვიძლია ვაკონტროლოთ ციმციმის სიხშირე და ჩვენი პროექტის რიტმი.

3.3. ციმციმის ეფექტის შექმნა digitalWrite-ისა და delay-ს კომბინაციით

მოდი, შევადგინოთ ნათურის ციმციმის ალგორითმი:

1. ჩართე ნათურა (digitalWrite(13, HIGH););
2. დაელოდე 1 წამი (delay(1000););
3. გამორთე ნათურა (digitalWrite(13, LOW););
4. დაელოდე 1 წამი (delay(1000);).

რადგან ამ კოდს loop() ფუნქციაში ვწერთ, არდუინო ამ ოთხ ნაბიჯს უსასრულოდ გაიმეორებს, რის შედეგადაც მივიღებთ მუდმივად მოციმციმე ნათურას.

AnITa სიმულატორის ალტერნატივა - Tinkercad პლატფორმა

AnITa სიმულატორთან ერთად, ანალოგიური პროექტების განხორციელება Tinkercad-ის გარემოშიც არის შესაძლებელი. Tinkercad-ი არის ინტუიტიური და მოსახერხებელი ონლაინ პლატფორმა, რომელსაც მომავალში უფრო დეტალურად გავეცნობით.

ამ ეტაპზე კი, მოდით, გავეცნოთ მოციმციმე შუქდიოდის ვიზუალიზაციას Tinkercad-ის ინტერფეისში და ვნახოთ, თუ როგორ გამოიყურება ეს მარტივი, მაგრამ ფუნდამენტური ელექტრონული ექსპერიმენტი ამ პლატფორმაზე.

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

დავალება 4: „საკუთარი შუქნიშანი“

შენი ამოცანაა, დაწერო პროგრამა, რომელიც არდუინოს პანელზე არსებულ შუქდიოდს აანთებს 2 წამით და ჩააქრობს ნახევარი წამით (500 მილიწამით).

დავალება 4.1: შეცდომის პოვნა და გასწორება
პროგრამისტმა კოდის წერისას შეცდომა დაუშვა. ერთ-ერთი ბრძანება არასწორად არის დაწერილი, რის გამოც პროგრამა არ მუშაობს. შენი ამოცანაა, იყო კოდის დეტექტივი, იპოვო და გაასწორო შეცდომა, რათა ნათურამ დავალების პირობის შესაბამისად იციმციმოს.

მინიშნება: ყურადღება მიაქციე, როგორ იწერება ბრძანებები და მათი პარამეტრები. ხომ არ არის რომელიმე სიტყვაში ასო გამორჩენილი? ასევე, გაიხსენე რა განსხვავებას INPUT-სა და OUTPUT-ს შორისდა დაფიქრდი მოცემულ შემთხვევაში რომლის გამოყენებაა საჭირო.

```
void setup() {
  pinMode(13,INPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(2000);

  // იპოვე შეცდომა ამ ხაზში
  dgtalWrite(13, LOW);

  delay(500);
}
```

დავალება 4.2: კოდის დასრულება

პროგრამის ძირითადი ნაწილი უკვე აგებულია, კოდი ნათურას რთავს და 2 წამით ანთებულ მდგომარეობაში ტოვებს. თუმცა, მთავარი ნაწილი – ნათურის გამორთვა და შესაბამისი პაუზა – გამორჩენილია. შენი ჯერია! დაამატე ორი გამოტოვებული ბრძანება, რათა პროგრამა დასრულდეს და ნათურამ ციმციმი დაიწყოს.

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(2000);
```

```
  // --- ჩაამატე პირველი ბრძანება აქ ---
  // გამორთე LED ნათურა.

  // --- ჩაამატე მეორე ბრძანება აქ ---
  // დააყოვნე პროგრამა ნახევარი წამით.
}
```

დავალება 4.3: შეიმუშავე პროგრამული კოდი
დაწერე პროგრამული კოდი, შექმენი პროგრამა, რომელიც შეასრულებს შემდეგ სამუშაოს:

1. setup ფუნქციაში განსაზღვრავს მე-13 პინის რეჟიმს როგორც გამომავალს (OUTPUT).
2. loop ფუნქციაში ჩართავს ნათურას, დააყოვნებს 2 წამით, გამორთავს ნათურას და დააყოვნებს ნახევარი წამით.

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

}

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

}
```

სწორი პასუხი (პროგრამული კოდი სრულად):

```
void setup() {
  // 1. უთხარი არდუინოს, რომ მე-13 პინი იმუშავებს როგორც გამომავალი (OUTPUT).
  pinMode(13, OUTPUT);
}

void loop() {
  // 2. ჩართე LED ნათურა (მიაწოდე HIGH სიგნალი).
  digitalWrite(13, HIGH);

  // 3. დააყოვნე პროგრამა 2 წამით.
  delay(2000);

  // 4. გამორთე LED ნათურა (მიაწოდე LOW სიგნალი).
  digitalWrite(13, LOW);
```

```
  // 5. დააყოვნე პროგრამა ნახევარი წამით.
  delay(500);
}
```

JSON

```
{
  "version": 1,
  "board": "arduino:avr:uno",
  "steps": [
    {
      "type": "compile"
    },
    {
      "type": "static",
      "rules": [
        {
          "id": "pin_13_output",
          "name": "Pin 13 as OUTPUT",
          "kind": "require_regex",
          "pattern":
"pinMode\\s*\\(\\s*13\\s*,\\s*OUTPUT\\s*\\)\\s*;?",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "პინი 13 უნდა იყოს OUTPUT რეჟიმში:
pinMode(13, OUTPUT);",
          "msg_pass": "პინი 13 სწორად არის OUTPUT რეჟიმში."
        },
        {
          "id": "led_on",
          "name": "Turn LED ON",
          "kind": "require_regex",
          "pattern":
"digitalWrite\\s*\\(\\s*13\\s*,\\s*HIGH\\s*\\)\\s*;?",
          "flags": "iu",
          "must_pass": true,
```

```
          "source": "stripped",
          "msg_fail": "LED უნდა ჩაირთოს: digitalWrite(13,
HIGH);",
          "msg_pass": "LED ჩართვა სწორადაა."
        },
        {
          "id": "delay_2_seconds",
          "name": "Delay 2 seconds",
          "kind": "require_regex",
          "pattern": "delay\\s*\\(\\s*2000\\s*\\)\\s*;?",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "LED უნდა ანთებდეს 2 წამით: delay(2000);",
          "msg_pass": "2 წამიანი დაყოვნება სწორადაა."
        },
        {
          "id": "led_off",
          "name": "Turn LED OFF",
          "kind": "require_regex",
          "pattern":
"digitalWrite\\s*\\(\\s*13\\s*,\\s*LOW\\s*\\)\\s*;?",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "LED უნდა გამოირთოს: digitalWrite(13,
LOW);",
          "msg_pass": "LED გამორთვა სწორადაა."
        },
        {
          "id": "delay_half_second",
          "name": "Delay 0.5 seconds",
          "kind": "require_regex",
          "pattern": "delay\\s*\\(\\s*500\\s*\\)\\s*;?",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "LED უნდა ჩაქრობდეს ნახევარი წამით:
delay(500);",
          "msg_pass": "0.5 წამიანი დაყოვნება სწორადაა."
        },
```

```
        {
          "id": "correct_blink_sequence",
          "name": "Correct blinking sequence",
          "kind": "require_ordered_regex",
          "patterns": [
            "digitalWrite\\s*\\(\\s*13\\s*,\\s*HIGH\\s*\\)",
            "delay\\s*\\(\\s*2000\\s*\\)",
            "digitalWrite\\s*\\(\\s*13\\s*,\\s*LOW\\s*\\)",
            "delay\\s*\\(\\s*500\\s*\\)"
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "loop()-ში თანმიმდევრობა უნდა იყოს: HIGH → delay(2000) → LOW → delay(500).",
          "msg_pass": "LED-ის მოციმციმე თანმიმდევრობა სწორია."
        }
      ]
    }
  ]
}
```

Sim Elements HTML

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

Sim Elements JS

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

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