შეხვედრა 2: არდუინო – შენი პირადი ელექტრონული კონსტრუქტორი

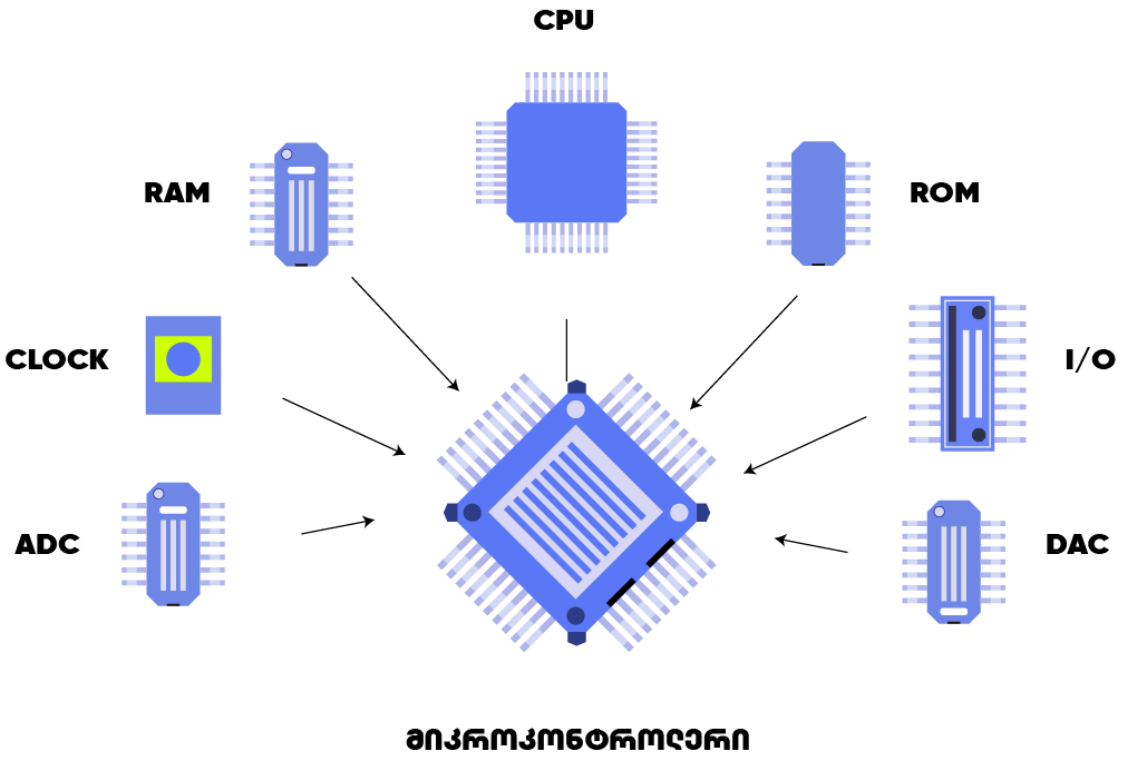
გამარჯობა! პირველ შეხვედრაზე ჩვენ აღმოვაჩინეთ, რომ STEAM-ის სამყარო ჩვენს გარშემოა. დღეს კი ხელში ავიღებთ ამ სამყაროს გასაღებს – პატარა, ლურჯ დაფას, რომელსაც არდუინო ჰქვია. წარმოიდგინე ის, როგორც ელექტრონული კონსტრუქტორი, რომლითაც შეგიძლია ააწყო უამრავი ელექტრონული მოწყობილობა. ისეთი, რასაც შენი ფანტაზია გიკარნახებს.

1. მიკროკონტროლერების სამყარო

ვიდრე უშუალოდ არდუინოს გავიცნობთ, მოდი, გავიგოთ, რა დგას მის უკან.

1.1. რა არის მიკროკონტროლერი და მისი როლი ელექტრონულ მოწყობილობებში

მიკროკონტროლერი – მოწყობილობის „ტვინი“: წარმოიდგინე სარეცხი მანქანა, მიკროტალღური ღუმელი ან თუნდაც ტელევიზორის პულტი. თითოეულ მათგანს აქვს თავისი პატარა, სპეციალური „ტვინი“, რომელიც მხოლოდ ერთ კონკრეტულ საქმეს აკეთებს. სწორედ ეს არის მიკროკონტროლერი. ის არის ჩიპი, რომელიც შეიცავს პროცესორს, მეხსიერებას და სხვადასხვა პორტს (შემავალ/გამომავალ კონტაქტებს (პინებს)). ჩვეულებრივი კომპიუტერისგან განსხვავებით, რომელიც უამრავ სხვადასხვა საქმეს აკეთებს, მიკროკონტროლერი, როგორც წესი, ერთი ამოცანის შესასრულებლად არის შექმნილი.

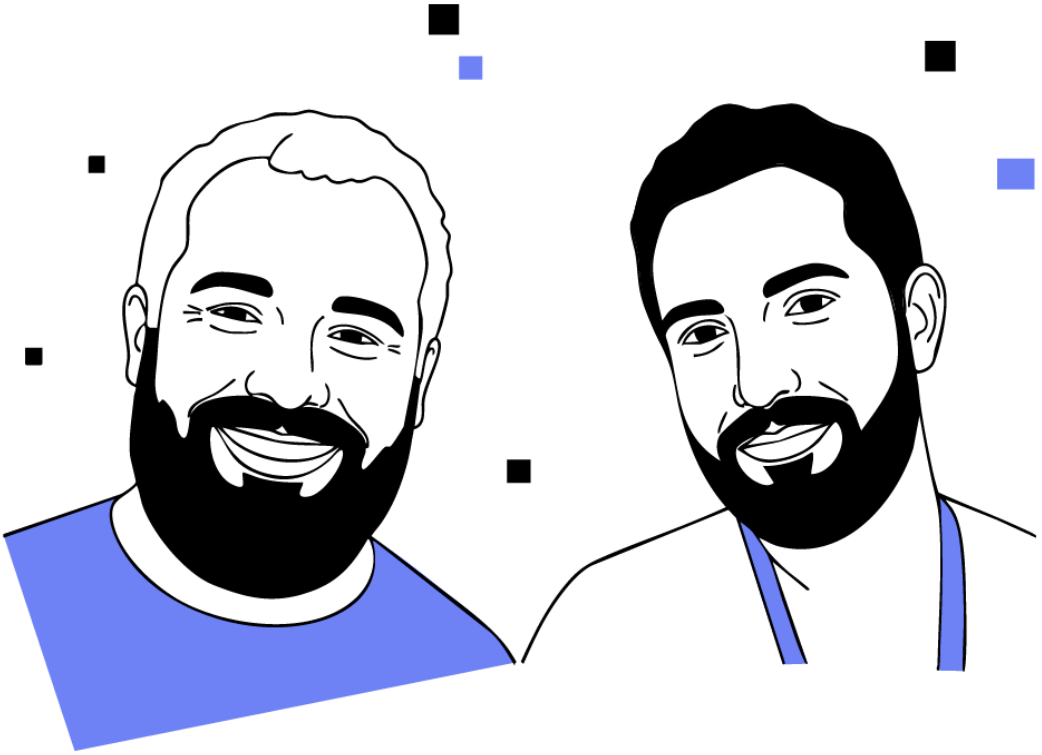


როგორ მუშაობს ის? მიკროკონტროლერი კითხულობს ინფორმაციას სენსორებიდან (მაგალითად, ღილაკზე დაჭერას) და ამ ინფორმაციის საფუძველზე მართავს სხვა მოწყობილობებს, ანუ აქტუატორებს (მაგალითად, ანთებს ნათურას ან რთავს ძრავას).

1.2. არდუინოს ისტორია, შექმნის მიზანი და ფილოსოფია

შექმნილია სტუდენტებისთვის: არდუინო 2005 წელს იტალიაში, ქალაქ ივრეაში, ინტერაქციის დიზაინის ინსტიტუტში შეიქმნა. მისი მთავარი ავტორები იყვნენ მასიმო ბანცი და დევიდ კუარტიელესი. მათი მიზანი იყო შეექმნათ იაფი და მარტივი ინსტრუმენტი სტუდენტებისა და სხვადასხვა სფეროს წარმომადგენლებისთვის, მათ შორის ხელოვანებისთვის, რომლებსაც არ ჰქონდათ ელექტრონიკასა და პროგრამირებაში დიდი გამოცდილება, მაგრამ სურდათ ინტერაქტიული პროექტების შექმნა. თანამედროვე ხელოვნების გამოფენებზე ხშირად შეხვედებით ისეთ

ნამუშევრებს, რომელიც, მაგალითად, დამთვალიერებლის მოძრაობაზე ან ხმაზე რეაგირებს. ასეთი ნამუშევრების შესაქმნელად ხელოვანები ხშირად არდუინოსა და მის სენსორებს იყენებენ.



სახელის ისტორია: სახელი „არდუინო“ მომდინარეობს ქალაქ ივრეაში არსებული ბარის სახელიდან, სადაც პროექტის ავტორები ხშირად იკრიბებოდნენ. ეს ბარი, თავის მხრივ, იტალიის მეფის, არდუინ ივრეელის სახელს ატარებდა.

არდუინოს ფილოსოფია: არდუინოს მთავარი იდეა არის ხელმისაწვდომობა და სიმარტივე. ის შექმნილია იმისთვის, რომ ტექნოლოგიები ყველასთვის გასაგები და ხელმისაწვდომი გახადოს, მიუხედავად მათი პროფესიისა თუ გამოცდილებისა.

1.3. ღია კოდის (Open-source) ტექნიკისა და პროგრამული უზრუნველყოფის კონცეფცია

რა არის Open-source? წარმოიდგინე, რომ შენმა მეგობარმა უგემრიელესი ნამცხვრის რეცეპტი მოიგონა. მას შეუძლია ეს რეცეპტი საიდუმლოდ შეინახოს (ეს არის „დახურული კოდი“), ან შეუძლია, ყველას გაუზიაროს. როცა რეცეპტი საჯაროა, ნებისმიერ მსურველს შეუძლია, ისწავლოს მისი მომზადება, გააუმჯობესოს (მაგალითად, დაამატოს შოკოლადი) და გაუზიაროს სხვებს თავისი გაუმჯობესებული ვერსია. სწორედ ეს არის Open-source, ანუ „ღია კოდის“ პრინციპი.

არდუინო, როგორც Open-source პროექტი: არდუინოს შემთხვევაში, მისი პროგრამული უზრუნველყოფა (კოდი) და ტექნიკური ნაწილი (დაფის ნახაზები) ღია წვდომაშია. ეს ნიშნავს იმას, რომ:

1.3.1. ნებისმიერს შეუძლია ნახოს, თუ როგორ არის აწყობილი არდუინოს დაფა და შექმნას თავისი საკუთარი ვერსია.

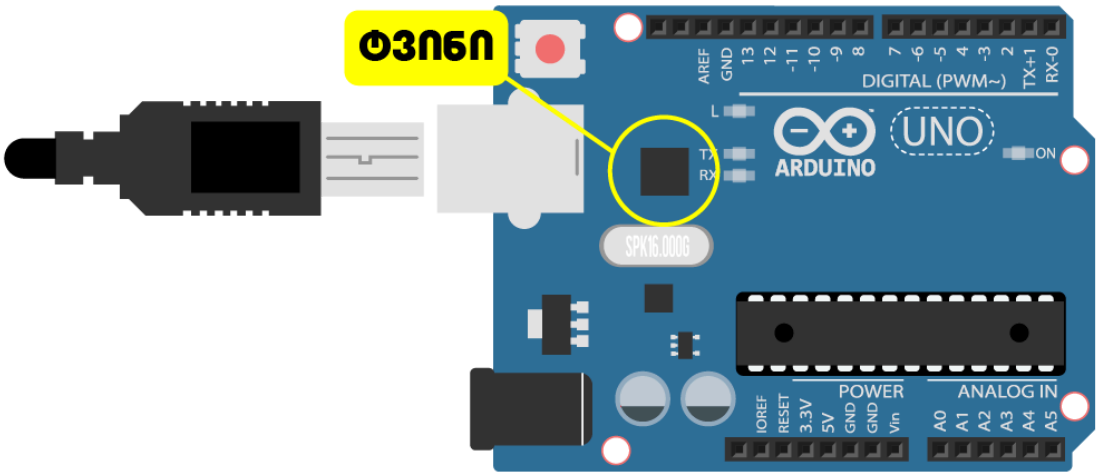
1.3.2. ნებისმიერს შეუძლია, ნახოს არდუინოს პროგრამული უზრუნველყოფის კოდი, შეცვალოს ის და მოარგოს თავის საჭიროებებს. ამ ღიაობამ განაპირობა არდუინოს გარშემო უზარმაზარი საზოგადოების (community) შექმნა, სადაც ადამიანები მუდმივად უზიარებენ ერთმანეთს პროექტებს, იდეებსა და ცოდნას.

2. არდუინოს ანატომია

მოდი, ახლა დეტალურად დავათვალიეროთ ჩვენი არდუინოს დაფა (ჩვენს შემთხვევაში, Arduino Uno-ს მოდელი) და გავიგოთ, თითოეული კომპონენტი რას ემსახურება.

2.1. მიკროკონტროლერი (მაგ. ATmega328P) - დაფის "ტვინი"

დაფაზე ყველაზე თვალსაჩინო დეტალი არის დიდი, შავი, მართკუთხა ჩიპი. ეს არის ATmega328P მიკროკონტროლერი – ჩვენი პროექტის გული და ტვინი. სწორედ ამ ჩიპში იწერება ჩვენი პროგრამა (სკეტჩი) და სწორედ ის ასრულებს ყველა ლოგიკურ ოპერაციას. როცა ჩვენ ვწერთ კოდს, ჩვენ, ფაქტობრივად, ვესაუბრებით ამ პატარა ტვინს და ვასწავლით რა გააკეთოს.



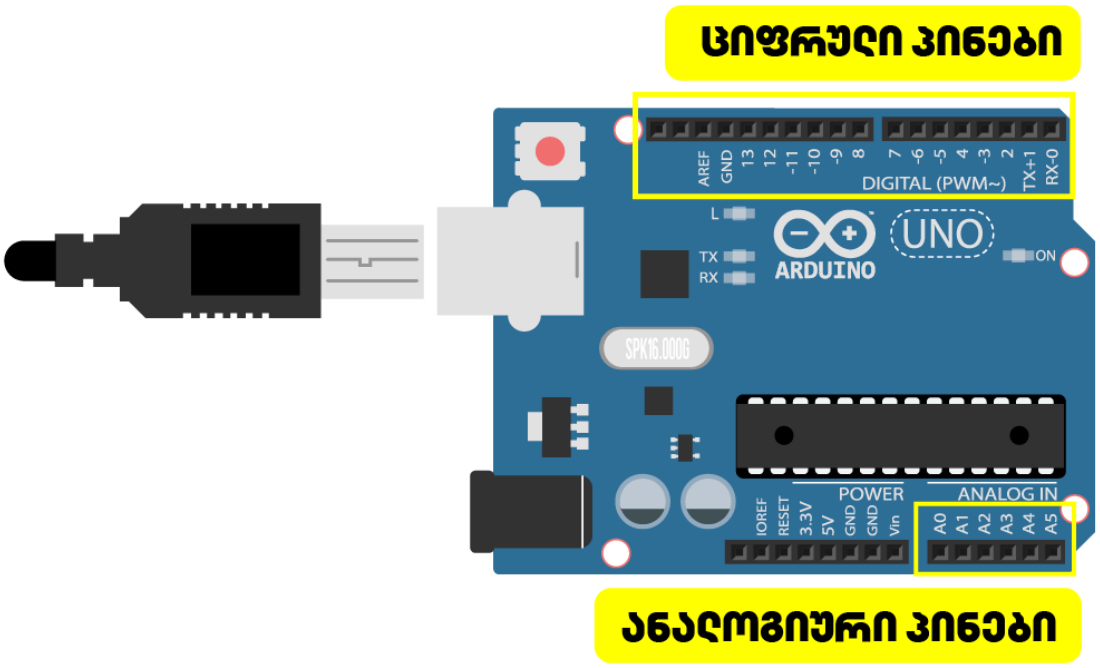
2.2. ციფრული და ანალოგური პინები (შემავალი/გამომავალი კონტაქტები (პინები))

პინები – არდუინოს „ხელები“ და „გრძნობის ორგანოები“: დაფის კიდეებზე განლაგებულია პატარა ნახვრეტები, რომლებსაც პინები (pins) ეწოდება. მათი საშუალებით არდუინო უკავშირდება გარე სამყაროს. პინების საშუალებით ის უკავშირდება სენსორებს, ნათურებს, ძრავებს და სხვა კომპონენტებს. პინები ორ ძირითად ტიპად იყოფა:

- **ციფრული პინები (Digital Pins):** დაფაზე ისინი დანომრილია 0-დან 13-მდე. ციფრული პინები მუშაობს „ან-ან“ პრინციპით, როგორც შუქის ჩამრთველი. ისინი მხოლოდ ორ მდგომარეობაში არიან: ჩართული (HIGH, ანუ 5 ვოლტი) ან გამორთული (LOW, ანუ 0 ვოლტი). მათ ვიყენებთ ისეთი კომპონენტებისთვის,

როგორიცაა შუქდიოდი (ჩართე/გამორთე), რომელსაც ასევე LED ნათურებს უწოდებენ ან ღილაკი (დაჭერილია/არ არის დაჭერილი).

- **ანალოგური პინები (Analog Pins):** დაფის მეორე მხარეს ნახავ პინებს წარწერით A0-დან A5-მდე. ციფრულისგან განსხვავებით, მათ შეუძლიათ, გაზომონ ძაბვის დონეების მთელი დიაპაზონი (0-დან 5 ვოლტამდე), რასაც არდუინო გარდაქმნის რიცხვებში 0-დან 1023-მდე. ისინი გვჭირდება ისეთი სენსორებისთვის, რომლებიც გვაწვდიან ცვალებად ინფორმაციას, მაგალითად, ფოტორეზისტორს შეუძლია განსაზღვროს ოთახში სინათლის დონე ან ტემპერატურის სენსორი.

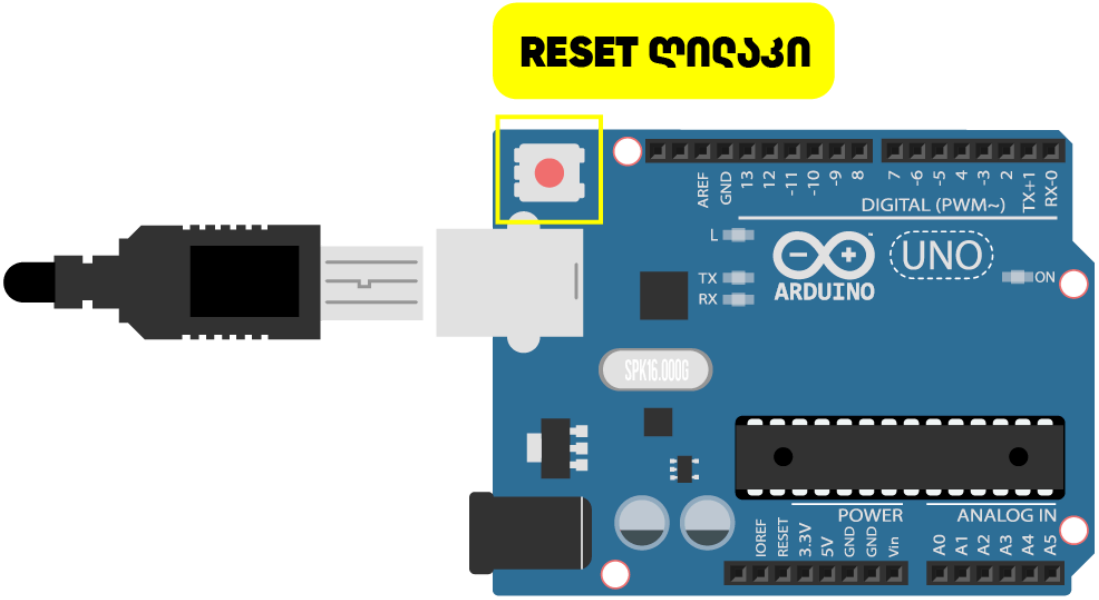


2.3. USB პორტი, კვების ბლოკი და Reset ღილაკი

USB პორტი: ეს არის კვადრატული ფორმის მეტალის პორტი, რომლითაც არდუინო კომპიუტერს უკავშირდება. მას ორი მთავარი ფუნქცია აქვს:

2.3.1. **პროგრამის ატვირთვა:** ამ პორტის საშუალებით ჩვენ ვტვირთავთ ჩვენ მიერ დაწერილ კოდს მიკროკონტროლერის მეხსიერებაში.
2.3.2. **კვება:** როცა არდუინო კომპიუტერთანაა შეერთებული, ის კვებასაც USB პორტიდან იღებს.

კვების ბლოკი (Power Jack): ეს არის მრგვალი ფორმის პორტი. როცა გინდა, შენი პროექტი კომპიუტერისგან დამოუკიდებლად ამუშაო, შეგიძლია, მას ადაპტერის ან ელემენტის საშუალებით კვება მიაწოდო ამ პორტიდან.



Reset ღილაკი: პატარა, წითელი ან შავი ღილაკი დაფაზე. მისი დაჭერა იგივეა, რაც კომპიუტერის გადატვირთვა. ის არ შლის კოდს, უბრალოდ აიძულებს მიკროკონტროლერს, პროგრამის შესრულება თავიდან, setup() ფუნქციიდან დაიწყოს.

3. არდუინოს გამოყენების სფეროები

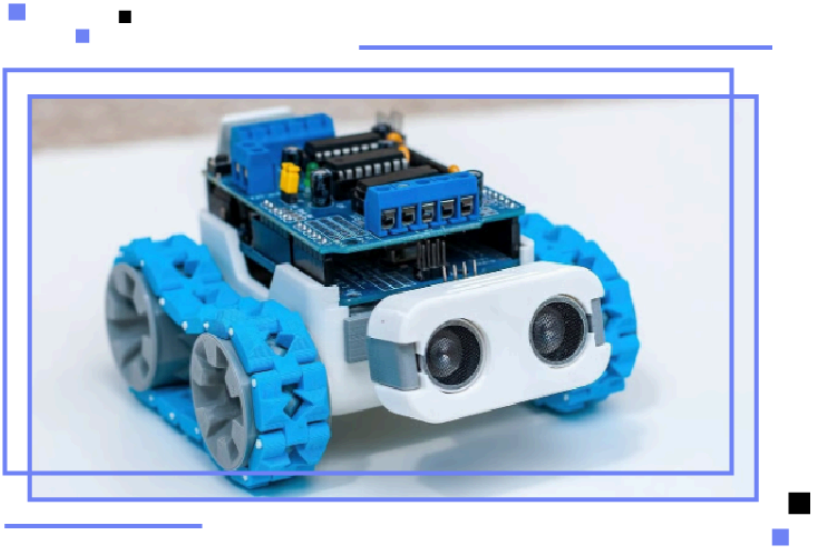
არდუინოს მიმზიდველობა მის მრავალფეროვნებაშია – მისი გამოყენება თითქმის ყველგან შეიძლება.**3.1. სწრაფი პროტოტიპირება და საკუთარი გამოგონებების შექმნა**

წარმოიდგინე, რომ დაგებადა იდეა შექმნა „ჭკვიანი“ ყვავილის ქოთანი, რომელიც თავად მიხვდება, როდის სჭირდება მცენარეს წყალი და შეგატყობინებს. არდუინოს საშუალებით შეგიძლია ძალიან სწრაფად ააწყო ამ იდეის პირველი, მომუშავე ვერსია – პროტოტიპი. ამისთვის შეაერთებ ნიადაგის ტენიანობის სენსორს, შუქდიოდს და დაწერ კოდს. თუ პროტოტიპი იმუშავებს, შემდეგ უკვე შეგიძლია, იფიქრო მის დიზაინსა და გაუმჯობესებაზე.

3.2. რობოტიკა და ავტომატიზირებული სისტემები

არდუინო ბევრი საგანმანათლებლო და სამოყვარულო რობოტის „ტვინია“. მისი საშუალებით შეგიძლია ააწყო:

- ბორბლებიანი რობოტი, რომელიც დამოუკიდებლად მოძრაობს და თავს არიდებს დაბრკოლებებს.
- რობოტი-ხელი, რომელიც სხვადასხვა ზომის საგნებს იღებს.
- სახლის ავტომატიზაციის სისტემები, მაგალითად, ფარდები, რომლებიც მზის ამოსვლისას ავტომატურად იხსნება.



3.3. ინტერაქტიული ხელოვნების ნიმუშები და "ჭკვიანი" მოწყობილობები

ხელოვანები და დიზაინერები აქტიურად იყენებენ არდუინოს, რათა შექმნან ინტერაქტიული ნამუშევრები, რომლებიც მაყურებლის მოძრაობაზე, ხმაზე ან შეხებაზე რეაგირებენ. მაგალითად:

- მუსიკალური კიბე, სადაც თითოეული საფეხური სხვადასხვა ნოტს გამოსცემს.
- „ჭკვიანი“ ტანსაცმელი, რომელზეც შუქდიოდები პულსის ცემის მიხედვით იცვლიან ფერს.
- ნახატები, რომლებიც ხმას გამოსცემენ, როცა მათ ეხები.

დავალება 2: „პროექტების დაფა“

ახლა, როცა უკვე იცი, რა არის არდუინო და რისი გაკეთება შეუძლია, დროა, შთაგონება მიიღო! შენი დავალებაა, მოიძიო ინტერნეტში (მაგალითად, საიტებზე: Arduino Project Hub, Instructables, YouTube) ორი შენთვის ყველაზე საინტერესო პროექტი არდუინოს გამოყენებით.შემდეგ კი დაწერე პროგრამა, რომელიც სერიულ მონიტორზე დაბეჭდავს ამ პროექტების სახელებს და თითოეულისთვის ერთწინადადებიან აღწერას ინგლისურად.

მაგალითად:

```
პროექტი 1: ამინდის სადგური
აღწერა: ეს პროექტი ზომავს ტემპერატურასა და ტენიანობას და აჩვენებს მონაცემებს ეკრანზე.

პროექტი 2: ლაზერული სიგნალიზაცია
აღწერა: ეს არის სიგნალიზაცია, რომელიც რთავს განგაშს, როცა ვინმე ლაზერის სხივს გადაკვეთს.
```

კოდის ნიმუში დასახმარებლად:

```
void setup() {
  // სერიული პორტის ჩართვა
  Serial.begin(9600);

  // პირველი პროექტის ინფორმაციის დაბეჭდვა
  Serial.println("project 1:weather station");
  Serial.println("Description: this project measures temperature and humidity and displays the data on the screen.");
  Serial.println(); // ცარიელი ხაზი გამოყოფისთვის

  // მეორე პროექტის ინფორმაციის დაბეჭდვა
  Serial.println("project 2: Laser signaling");
  Serial.println("Description: This is a security alarm that sounds an alarm when someone crosses the laser beam");
}

void loop() {
}
```

ეს დავალება დაგეხმარება უკეთ დაინახო არდუინოს შესაძლებლობების მრავალფეროვნება და იქნებ შენი მომავალი პროექტის იდეაც იპოვო!

დავალება 2.1: შეცდომის პოვნა და გასწორება

პროგრამისტს კოდის წერისას რაღაც შეცდომა მოუვიდა. მას დაავალეს რომ მოცემული ინფორმაცია მხოლოდ ერთხელ დაბეჭდილიყო, მაგრამ მის მიერ მოძებული ინფორმაცია

უწყვეტად იბეჭდება. შენი მიზანია, იპოვო შეცდომა და დაეხმარო პროგრამისტს კოდის გამართვაში.

მინიშნება : გაიხსენე განსხვავება setup და loop ფუნქციებს შორის.

```
void setup() {

}

void loop(){
  Serial.begin(9600);

 Serial.println("project 1:weather station");
  Serial.println("Description: this project measures temperature and humidity and displays the data on the screen.");

  Serial.println();

  // მეორე პროექტის ინფორმაციის დაბეჭდვა
  Serial.println("project 2: Laser signaling");
  Serial.println("Description: This is a security alarm that sounds an alarm when someone crosses the laser beam.");
}
```

დავალება 2.2: კოდის დასრულება

პროგრამის ძირითადი ნაწილი უკვე აგებულია, ახლა საჭიროა setup ფუნქციაში რამდენიმე ბრძანების დამატება რათა კოდმა სრულყოფილად იმუშაოს

განმარტება :

- წინა დავალების მსგავსად, ახლაც მხოლოდ ორი ბრძანების გამოყენება გვიწევს. `Serial.begin(9600);` ეს ბრძანება საჭიროა იმისთვის, რომ არდუინომ და კომპიუტერმა “ერთ ენაზე საუბარი” შეძლონ.
- მეორე ბრძანება პასუხისმგებელია **იმაზე,** რომ ჩვენთვის სასურველი ტექსტი ეკრანზე დაიბეჭდოს. ამისათვის `Serial.println();` ბრძანება გამოგვადგება და დაბეჭდვის შემდეგ კურსორი ავტომატურად გადავა ახალ ხაზზე. თუ დავალების გარკვეულ ეტაპზე საჭირო იქნება ორი სხვადასხვა ინფორმაციის ერთმანეთის გვერდით დაბეჭდვა, შეგიძლიათ გამოიყენოთ `Serial.print();`
- `Serial.println();`-ში ჩაწერილი ტექსი უნდა იყოს ბრჭყალებში, მაგალითად, `Serial.println(“ჩემი პირველი პროექტი:”);`

- იმისათვის, რომ თქვენ მიერ მოძიებული პროექტები ერთმანეთისგან მარტივი გასარჩევი იყოს, პროექტების დაბეჭდვებს შორის გამოყავით ერთი ცარიელი ხაზი `Serial.println();` ბრძანების დახმარებით

```
void setup() {
  //მიუთითეთ შესაბამისი რიცხვი
  Serial.begin();

  //  შეიყვანეთ პირველი მოძიებული პროექტის სათაური
  Serial.println();


  //დაბეჭდეთ თქვენი პროექტის აღწერა



 // გამოყავით ცარიელი ხაზი მეორე პროექტისთვის

//ჩაწერეთ მეორე პროექის სახელი
Serial.println();

//დაბეჭდეთ მეორე პროექტის აღწერა
Serial.println();

}

void loop() {
  // loop ფუნქცია ცარიელი რჩება, რადგან ინფორმაციის დაბეჭდვა მხოლოდ ერთხელ გვჭირდება.
}
```

დავალება 2.3: შეიმუშავე პროგრამული კოდი

შენს წინაშეა ცარიელი ფურცელი და შენი დავალებაა შენ მიერ მოძიებული არდუინოს პროექტები გაგვიზიარო. ამისათვის საჭიროა რამდენიმე მოქმედების შესრულება:

1. მოიძიე პროექტები **არდუინოს გამოყენებით;**
2. მოამზადე სერიული მონიტორი სამუშაოდ (კომუნიკაციის სიჩქარე 9600);
3. დაბეჭდე პროექტის სახელი და მომდევნო ხაზზე მისი აღწერა;
4. პირველი პროექტის აღწერის შემდეგ სერიულ მონიტორზე გამოტოვე ერთი ხაზი, რათა პროექტები თვალსაჩინოდ იყოს გამოყოფილი.
5. დაბეჭდე მეორე პროექტის სახელი და მომდევნო ხაზზე მისი აღწერა.

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

}

void loop() {
  // ეს ფუნქცია ცარიელი რჩება.
}
```

პროგრამულად სწორი კოდი:

```
void setup() {
  Serial.begin(9600);
  Serial.println("project 1:weather station");
  Serial.println("Description: this project measures temperature and humidity and displays the data on the screen.");
  Serial.println(); // ცარიელი ხაზი გამოყოფისთვის

  Serial.println("project 2: Laser signaling");
  Serial.println("Description: This is a security alarm that sounds an alarm when someone crosses the laser beam");
}

void loop() {
}
```

JSON

```
{
  "version": 1,
  "board": "arduino:avr:uno",
  "steps": [
    {
      "type": "compile"
    },
    {
      "type": "static",
      "rules": [
        {
          "id": "serial_begin",
          "name": "Serial initialization",
```

```
          "kind": "require_regex",
          "pattern":
"Serial\\.begin\\s*\\(\\s*9600\\s*\\)\\s*;?",
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "სერიული პორტი უნდა ჩაირთოს:
Serial.begin(9600);",
          "msg_pass": "სერიული პორტი სწორად არის ჩართული."
        },
        {
          "id": "project_1",
          "name": "Project 1 name",
          "kind": "require_regex",
          "pattern":
"Serial\\.println?\\s*\\(\\s*\"[^\"]*[Pp]roject\\s*1[^\"]*\"\\s*\
\)",
          "flags": "iu",
          "must_pass": true,
          "source": "raw",
          "msg_fail": "დაბეჭდე პირველი პროექტის სახელი:
Serial.println(\"Project 1: ...\");",
          "msg_pass": "პროექტი 1-ის სახელი დაბეჭდილია."
        },
        {
          "id": "project_2",
          "name": "Project 2 name",
          "kind": "require_regex",
          "pattern":
"Serial\\.println?\\s*\\(\\s*\"[^\"]*[Pp]roject\\s*2[^\"]*\"\\s*\
\)",
          "flags": "iu",
          "must_pass": true,
          "source": "raw",
          "msg_fail": "დაბეჭდე მეორე პროექტის სახელი:
Serial.println(\"Project 2: ...\");",
          "msg_pass": "პროექტი 2-ის სახელი დაბეჭდილია."
        },
        {
          "id": "description_1",
          "name": "Project 1 description",
```

```
          "kind": "require_regex",
          "pattern":
"Serial\\.println?\\s*\\(\\s*\"[^\"]*[Dd]escription[^\"]*\"\\s*\\
)",
          "flags": "iu",
          "must_pass": true,
          "source": "raw",
          "msg_fail": "დაბეჭდე პირველი პროექტის აღწერა:
Serial.println(\"Description: ...\");",
          "msg_pass": "პროექტის აღწერა დაბეჭდილია."
        },
        {
          "id": "multiple_prints",
          "name": "Multiple Serial prints",
          "kind": "require_min_matches",
          "pattern": "Serial\\.println?\\s*\\(",
          "min_matches": 4,
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "უნდა იყოს მინიმუმ 4 Serial.print/println
(ორი პროექტი + ორი აღწერა).",
          "msg_pass": "საკმარისი Serial.print ოპერაციებია."
        }
      ]
    }
  ]
}
```