შეხვედრა 1: მოგზაურობა STEAM-ის სამყაროში

გამარჯობა! მე ანITა ვარ და მოხარული ვარ, რომ ერთად ვიწყებთ ამ საოცარ მოგზაურობას. დღეს ჩვენი პირველი, ყველაზე მნიშვნელოვანი გაჩერებაა STEAM-ის სამყარო,სადაც ფანტაზია და ცოდნა ერთიანდება და ნამდვილ სასწაულებს ახდენს. მზად ხარ?

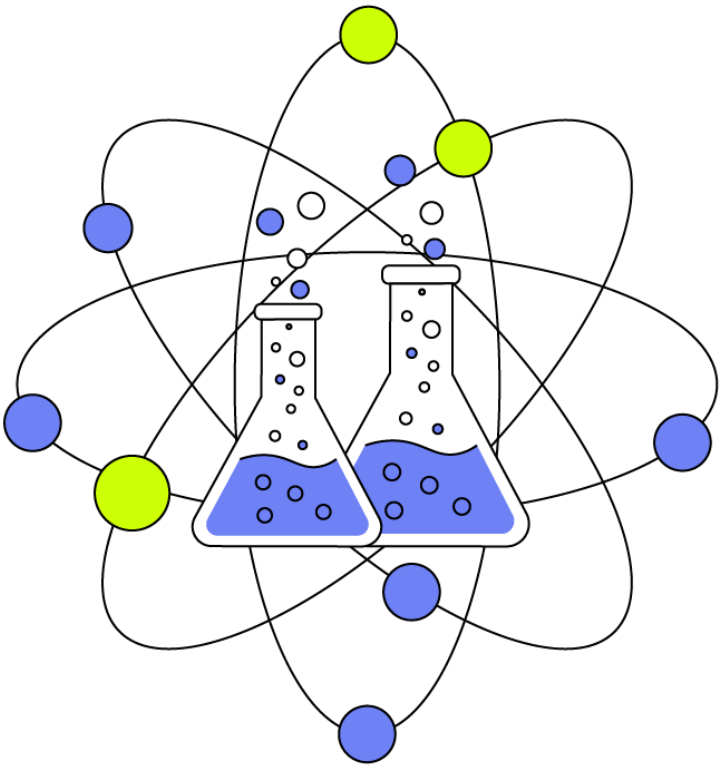
1. STEAM-ის კონცეფცია

წარმოიდგინე, რომ STEAM არის სუპერგმირების გუნდი, სადაც თითოეულ წევრს თავისი უნიკალური ძალა აქვს, მაგრამ ერთად ისინი უძლეველები არიან. მოდი, თითოეული მათგანი ახლოს გავიცნოთ:

1.1. მეცნიერება, ტექნოლოგია, ინჟინერია, ხელოვნება და მათემატიკა

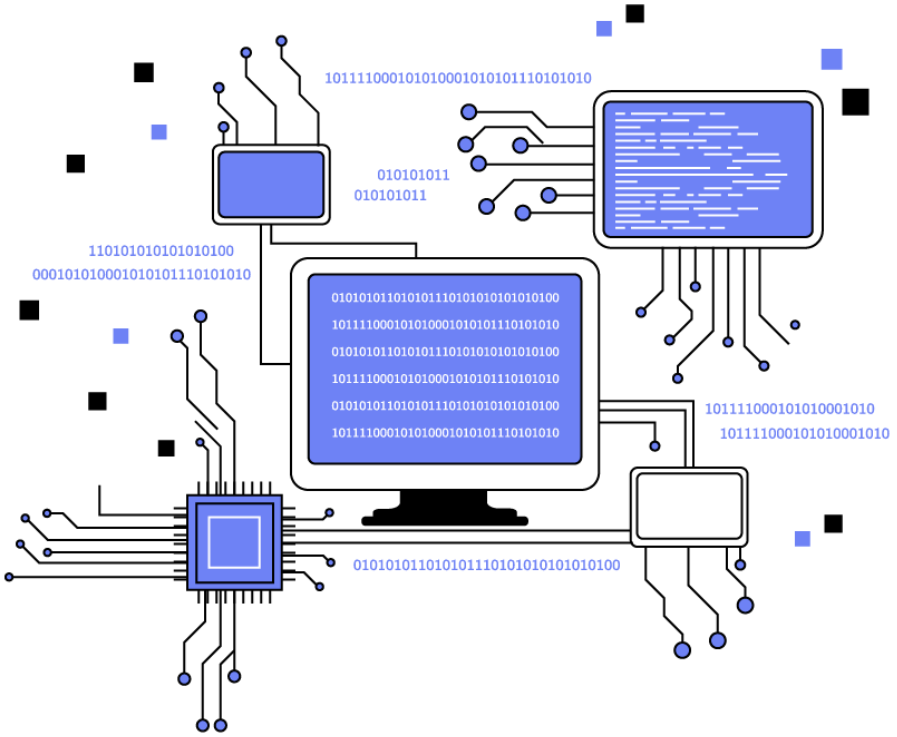
S (Science) - საბუნებისმეტყველო მეცნიერებები სამყაროს შეცნობის ხელოვნებაა. ეს არის ჩვენი დაუოკებელი ცნობისმოყვარეობა, რომელიც მუდმივად გვაიძულებს, დავსვათ კითხვები: „რატომ არის ცა ლურჯი?“, „როგორ იზრდება მცენარე?“, „რატომ გვესმის ხმა?“. მეცნიერება არ არის მხოლოდ ფაქტების კრებული; ის არის პროცესი, რომელიც მოიცავს დაკვირვებას, ჰიპოთეზის (ვარაუდის) ჩამოყალიბებას, ექსპერიმენტის ჩატარებას და დასკვნების გამოტანას. ის გვასწავლის, როგორ გავიგოთ სამყაროს კანონები.

ყოველდღიური მაგალითი: როცა საჭმელს ამზადებ, შენ ქიმიურ ექსპერიმენტს ატარებ. წყლის ადუღება, შაქრის გახსნა, ცომის აფუება – ეს ყველაფერი მეცნიერული პროცესია.



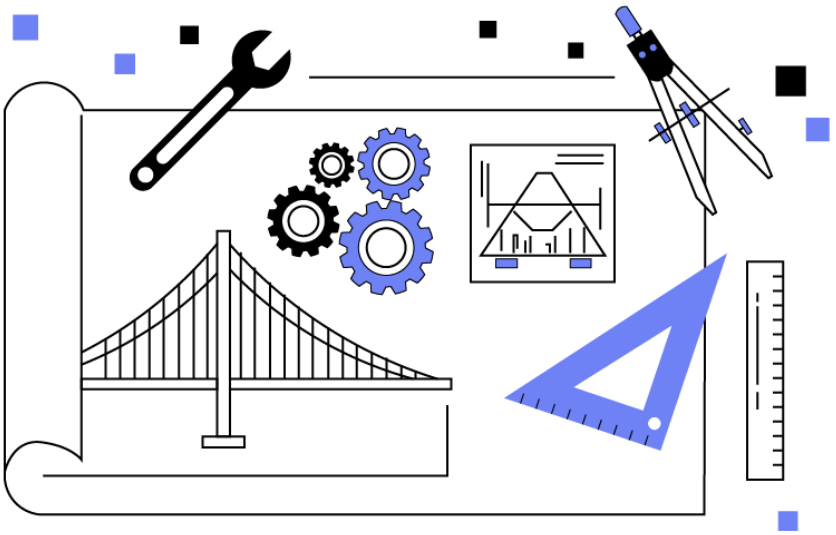
T (Technology) - ტექნოლოგიაიდეების რეალობად ქცევაში გვეხმარება. ტექნოლოგია ეს არის გარკვეული წესებისა და თანმიმდევრობის დაცვით განხორციელებული ოპერაციების ერთობლიობა. ტექნოლოგიის, როგორც მეცნიერების, ამოცანაა ფიზიკური, ქიმიური, მექანიკური და სხვა კანონზომიერებების გამოვლენა ყველაზე ეფექტური გზით.

ყოველდღიური მაგალითი: სათვალე არის შესანიშნავი ტექნოლოგიური მაგალითი. მეცნიერებამ შეისწავლა სინათლის გარდატეხის ფენომენი და მისი კანონები ფიზიკის დარგში – ოპტიკაში, ასევე ადამიანის თვალის მუშაობის პრინციპები. ტექნოლოგიის განვითარებამ კი შესაძლებელი გახადა ისეთი ლინზებისა და ჩარჩოს შექმნა, რომლებიც ადამიანის ბიოლოგიურ აგებულებაზეა მორგებული და გვეხმარება უკეთესად დავინახოთ.



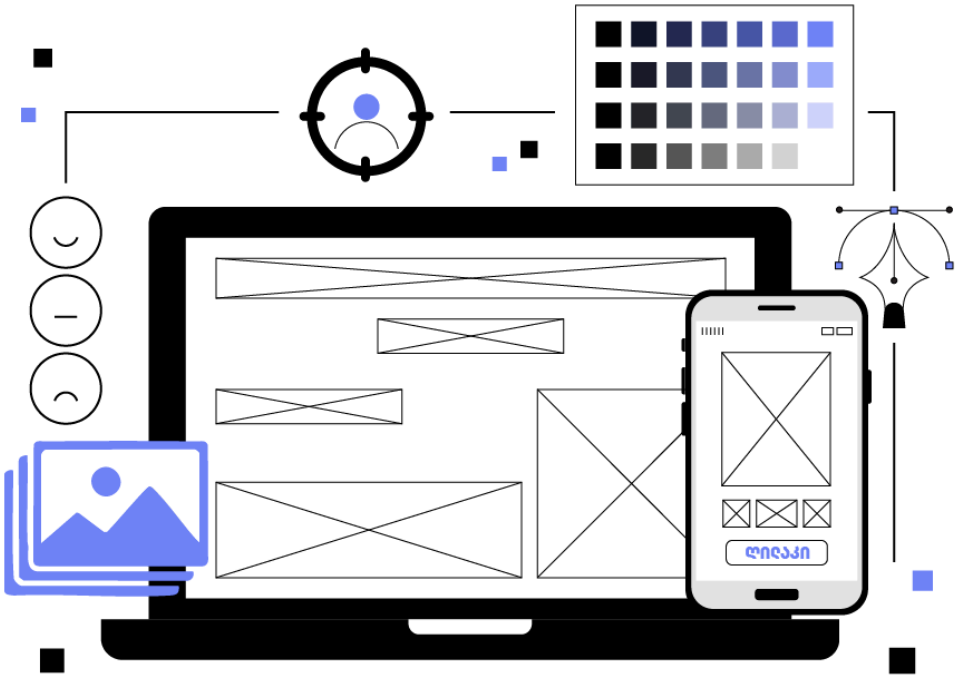
E (Engineering) - ინჟინერია: პრობლემების გადაჭრის ხელოვნებაა.
ინჟინერია იყენებს მეცნიერებისა და მათემატიკის პრინციპებს, რათა დააპროექტოს და შექმნას მუშა სისტემები. ინჟინრები ფიქრობენ არა მხოლოდ იმაზე, თუ რა უნდა შეიქმნას, არამედ იმაზეც, თუ როგორ უნდა შეიქმნას. შეიქმნას ისე, რომ იყოს მყარი, უსაფრთხო, ეფექტური და ხშირად, ეკონომიურიც.

ყოველდღიური მაგალითი: ველოსიპედი საინჟინრო ხელოვნების ერთ-ერთი ნიმუშია. ინჟინრებმა დააპროექტეს მისი ჩარჩო, მუხრუჭების სისტემა, ჯაჭვის მექანიზმი, რათა ადამიანებმა შეძლონ სწრაფად და უსაფრთხოდ გადაადგილება.



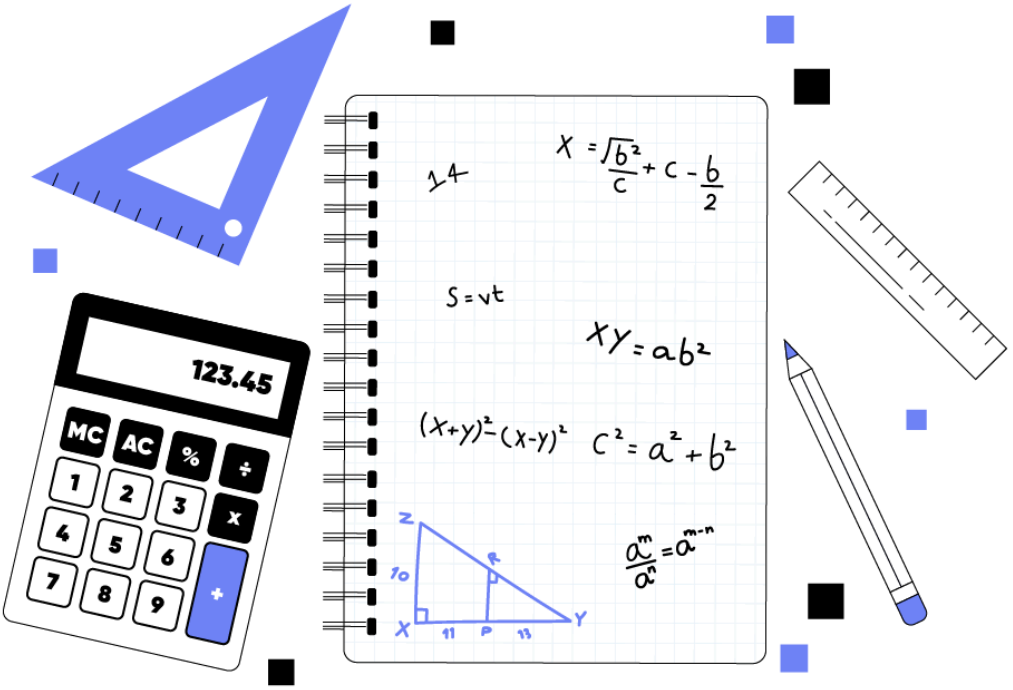
A (Art) - ხელოვნება: შთაგონება.
ეს არის ჩვენი ფანტაზია და ესთეტიკის შეგრძნება. ხელოვნება სძენს ჩვენს ქმნილებებს სულს. ის პასუხისმგებელია არა მხოლოდ სილამაზეზე, არამედ მომხმარებლის გამოცდილებაზე (User Experience) – რამდენად მოსახერხებელი, სასიამოვნო და ინტუიციურია ამა თუ იმ ნივთის გამოყენება. **ხელოვნების ერთ-ერთი ყველაზე მნიშვნელოვანი დანიშნულება ადამიანის შემოქმედებითი უნარის განვითარებაა.**

ყოველდღიური მაგალითი: დააკვირდი შენი საყვარელი აპლიკაციის დიზაინს მობილურში. ფერების შეხამება, ღილაკების განლაგება, შრიფტის სტილი – ეს ყველაფერი ხელოვნების ნაწილია, რომელიც აპლიკაციის გამოყენებას სასიამოვნოს ხდის.



M (Mathematics) - მათემატიკა სამყაროს უნივერსალური ენაა.
ეს არის წესრიგისა და ლოგიკის ენა, რომელიც ყველა დანარჩენ სფეროს აერთიანებს. მათემატიკა გვეხმარება, აღვწეროთ კანონზომიერებები, გავაკეთოთ ზუსტი გამოთვლები, შევქმნათ მოდელები და ვიწინასწარმეტყველოთ შედეგები. მის გარეშე ინჟინერია და მეცნიერება უბრალოდ ვერ იარსებებდა.

ყოველდღიური მაგალითი: როცა თამაშობ კომპიუტერულ თამაშს, ყველა პერსონაჟის მოძრაობა, ნახტომის სიმაღლე და ტრაექტორია ზუსტი მათემატიკური ფორმულებით აღიწერება.



1.2. STEAM-ის კომპონენტების კავშირი ყოველდღიურ საგნებში

მოდი, ახლა უფრო ღრმად ჩავიხედოთ და ავიღოთ ერთი კონკრეტული მაგალითი – **ვიდეოთამაში**:

საბუნებისმეტყველო მეცნიერებები: მაგალითად, ფიზიკის კანონები განსაზღვრავს, როგორ მოძრაობენ ობიექტები თამაშის სამყაროში (გრავიტაცია, ხახუნი). ბიოლოგია შეისწავლის ცოცხალი ორგანიზმების აგებულებას, რის საფუძველზეც შეიძლება შევქმნათ რეალისტური პერსონაჟები.

ტექნოლოგია: ეს არის სათამაშო ძრავა (Game Engine), პროგრამული უზრუნველყოფა, რომელიც საშუალებას აძლევს დეველოპერებს, შექმნან თამაში. ასევე, კომპიუტერის გრაფიკული ბარათი და პროცესორი, რომელიც ამ ყველაფერს ამუშავებს.

ინჟინერია: პროგრამული ინჟინრები წერენ მილიონობით ხაზისგან შემდგარ კოდს, რომელიც თამაშის ლოგიკას, წესებსა და ფუნქციონირებას უზრუნველყოფს. ისინი ქმნიან სტაბილურ და ოპტიმიზებულ სისტემას.

ხელოვნება: 2D და 3D არტისტები ხატავენ პერსონაჟებს, გარემოს, ქმნიან ანიმაციებს. სცენარისტები წერენ ისტორიას, კომპოზიტორები კი – მუსიკას. დიზაინერები აწყობენ თამაშის დონეებს.

- **მათემატიკა:** 3D გრაფიკა მთლიანად გეომეტრიასა და ალგებრაზეა დაფუძნებული. ხელოვნური ინტელექტი (AI), რომელიც თამაშის გარემოს მართავს, რთულ მათემატიკურ ალგორითმებს იყენებს.

როგორც ხედავ, ერთი საინტერესო ვიდეოთამაში ამ ხუთივე სფეროს ჰარმონიული თანამშრომლობის შედეგია.

1.3. ინტერდისციპლინური მიდგომის მნიშვნელობა

ინტერდისციპლინური მიდგომა ხაზს უსვამს იმას, რომ რეალური, კომპლექსური პრობლემების გადასაჭრელად საკითხების ერთი სფეროდან ცოდნა საკმარისი არ არის. ინოვაციური ნივთები სწორედ მაშინ იქმნება, როცა სხვადასხვა პროფესიის ადამიანები – მეცნიერები, ინჟინრები, დიზაინერები – ერთად მუშაობენ, უზიარებენ ერთმანეთს იდეებს და პოულობენ საუკეთესო გამოსავალს. ეს კურსი გასწავლის, იფიქრო სწორედ ასე – კომპლექსურად.

2. STEAM-ის გავლენა და მნიშვნელობა

კარგი, გავიგეთ, რა არის STEAM, მაგრამ რატომ არის ის ასეთი მნიშვნელოვანი პირადად შენთვის ?

2.1. ტექნოლოგიური ინოვაციების როლი

დაფიქრდი, როგორ შეცვალა ინტერნეტმა ჩვენი ცხოვრება. თუმცა ეს მხოლოდ ერთი მაგალითია. დღეს STEAM-ის წყალობით გვაქვს:

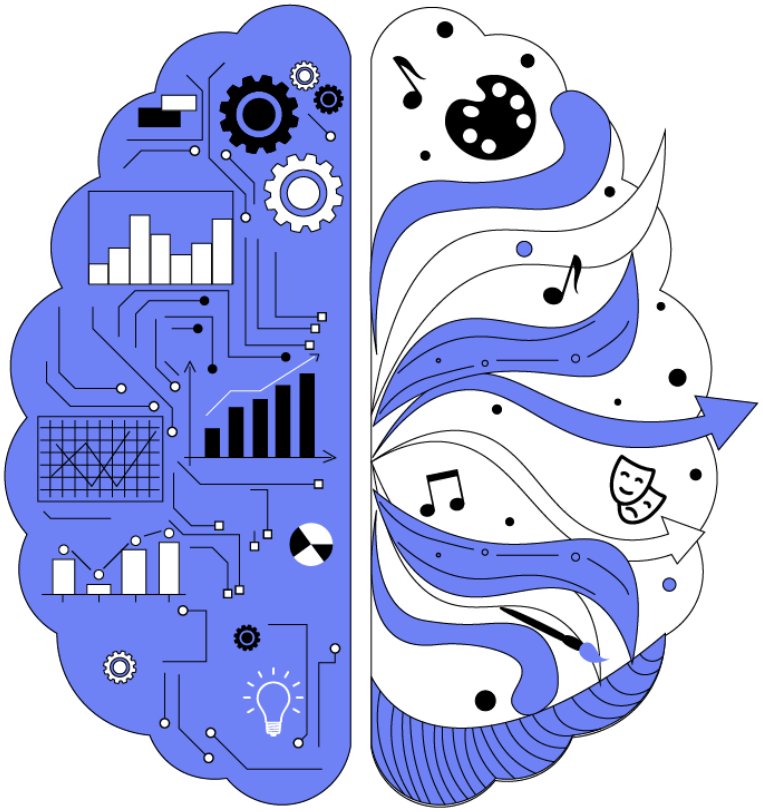
- **მედიცინაში:** მაგნიტურ-რეზონანსული ტომოგრაფიის (MRI) აპარატები ექიმებს საშუალებას აძლევს, ადამიანის სხეულის შიგნით დეტალურად შეისწავლონ ორგანოები და ქსოვილები.
- **კოსმოსის კვლევაში:** მარსმავლები, რომლებიც პლანეტა მარსს შეისწავლიან და გვიგზავნიან მონაცემებს.
- **ყოველდღიურობაში:** ელექტრომობილები, რომლებიც ამცირებენ გარემოს დაბინძურებას.

ეს ყველაფერი STEAM სფეროებში მომუშავე ადამიანების შექმნილია. ისინი არიან ჩვენი დროის ნამდვილი ნოვატორები.

2.2. შემოქმედებითი და კრიტიკული აზროვნების განვითარება

STEAM არ არის მხოლოდ ფორმულების და ფაქტების დაზეპირება. ის ავითარებს ორ უმნიშვნელოვანეს უნარს:

- **კრიტიკული აზროვნება:** ეს არის პრობლემის სიღრმისეულად გაანალიზების, მისი შემადგენელ ნაწილებად დაშლის და ლოგიკური კავშირების პოვნის უნარი. როცა კოდში შეცდომას (bug) ეძებ, შენ სწორედ კრიტიკულ აზროვნებას იყენებ.
- **შემოქმედებითი აზროვნება:** ეს არის პრობლემის გადაჭრის ახალი, არასტანდარტული გზების პოვნის, სხვადასხვა იდეის ერთმანეთთან დაკავშირების და რაღაც სრულიად ახლის შექმნის უნარი. როცა ფიქრობ, „კიდევ რისი გაკეთება შეუძლია ჩემს პროექტს?“, შენ შემოქმედებითად აზროვნებ.



2.3. STEAM უნარების საჭიროება მომავლის პროფესიებში

მომავალში ბევრი პროფესია, რომელზეც დღეს არც კი გვიფიქრია, სწორედ STEAM-თან იქნება დაკავშირებული. წარმოიდგინე:

- **ვირტუალური რეალობის არქიტექტორი:** ადამიანი, რომელიც დააპროექტებს შენობებსა და ქალაქებს ვირტუალურ სამყაროში.
- **მონაცემთა ანალიტიკოსი:** ადამიანი, რომელიც უზარმაზარ ინფორმაციაში პოულობს მნიშვნელოვან კანონზომიერებებს და ეხმარება კომპანიებს მიიღონ სწორი გადაწყვეტილებები.
- **რობოტების ეთიკის სპეციალისტი:** ადამიანი, რომელიც შეიმუშავებს წესებს, თუ როგორ უნდა მოიქცნენ რობოტები სხვადასხვა სიტუაციაში.

ეს პროფესიები ფანტასტიკის სფერო აღარ არის. ამ კურსის გავლა კი შენი პირველი ნაბიჯია ამ საინტერესო მომავლისკენ.

3. კურსის შესავალი

3.1. კურსის მიზნები და ამოცანები

ამ კურსის მთავარი მიზანია, არა მხოლოდ გასწავლოთ პროგრამირებისა და ელექტრონიკის საფუძვლები, არამედ გაჩვენოთ, რომ თქვენ შეგიძლიათ იყოთ არა მხოლოდ ტექნოლოგიების მომხმარებელი, არამედ მისი შემქმნელიც. ჩვენ ერთად გავივლით გზას იდეიდან მის განხორციელებამდე, ვისწავლით, როგორ შევქმნათ ელექტრონული კომპონენტების „ტვინი“ და როგორ მივანიჭოთ მათ „გრძნობები“. როგორ ავაწყოთ ჩვენი პირველი „ჭკვიანი“ მოწყობილობები.

3.2. AnITa-ს პლატფორმა და ვირტუალური სიმულატორი

ჩვენი სამუშაო სივრცე AnITa-ს პლატფორმაა. აქ ნახავ გაკვეთილებს, დავალებებს და რაც მთავარია – ვირტუალურ სიმულატორს. წარმოიდგინე ის, როგორც ჯადოსნური, ციფრული სახელოსნო. ამ სახელოსნოში გაქვს ელექტრონული ნაწილების უსასრულო მარაგი და შეგიძლია ააწყო ნებისმიერი სირთულის სქემა. ყველაზე კარგი ისაა, რომ აქ ვერაფერს გააფუჭებ, **გადაწვავ, ჩაატარებ უსაფრთხო ექსპერიმენტებს უსასრულო რაოდენობით.**

3.3. სერიული მონიტორი – ჩვენი „ჩატის ფანჯარა“ არდუინოსთან

როგორ გავიგოთ, რას „ფიქრობს“ ჩვენი არდუინო, როცა პროგრამას ვუშვებთ? როგორ დავრწმუნდეთ, რომ ის ჩვენს ბრძანებებს სწორად ასრულებს? ამისთვის არსებობს **სერიული მონიტორი**. ეს არის ფანჯარა სიმულატორში, რომელიც ჩვენსა და არდუინოს შორის ცალმხრივი „ჩატის“ როლს ასრულებს. არდუინოს შეუძლია, ამ ფანჯარაში მოგვწეროს ტექსტური შეტყობინებები, მაგალითად, გვითხრას, რას ზომავს სენსორი, შეგვატყობინოს, რომ პროგრამის კონკრეტული ნაწილი შესრულდა, ან უბრალოდ მოგვესალმოს. ეს არის ერთ-ერთი ყველაზე მნიშვნელოვანი ინსტრუმენტი პროექტების შექმნისა და გამართვის პროცესში.

და როგორ შეგვიძლია მივახვედროთ არდუინო რა მომენტში რა ინფორმაცია გვჭირდება მისგან? ამისათვის გამოიყენება არდუინოს “კოდის სახელოსნო”, რომელიცთავიდან ასე გამოიყურება:

```
void setup()
{

}

void loop()
{

}
```

როგორც ხედავ, ის შედგება ორი ცარიელი ფუნქციისგან რომელშიც ბრძანებების ჩაწერა თქვენი მოვალეობა. იმისათვის რომ ფუნქციას ბრძანებები ჩავუწეროთ საჭიროა გავიგოთ კომპიუტერის ენაზე რას ნიშნავს ფუნქცია და რაში გამოიყენება ის.

არდუინოს ენაზე ყველა ფუნქციას თავისი სახელი აქვს.მაგალითად, void setup() ფუნქციაში setup მისი სახელია ხოლო დასაწყისი და დასასრული განისაზღვრება { } მოცემული სიმბოლოებით.({-ეს ნიშნავს დასაწყისს } - ეს კი დასასრულს).

იგივე სტრუქტურას ვხვდებით loop ფუნქციასთანაც, მხოლოდ ამ ორ ფუნქციას შორის ერთი მნიშვნელოვანი განსხვავებაა – setup-ში ჩაწერილი ბრძანებები სრულდება მხოლოდ ერთხელ, როდესაც არდუინო ჩაირთვება ან გადაიტვირთება, ხოლო loop-ში ჩაწერილი ბრძანებები გამუდმებით მეორდება, ვიდრე არდუინო ჩართულია.

დავალება 1: „გამარჯობა, STEAM!“

პროგრამის „საძირკველი“ უკვე აშენებულია – სერიული მონიტორი მზად არის სამუშაოდ. ახლა შენი ჯერია! დაამატე ორი ბრძანება მოცემულ კოდში, რათა დაასრულო პროგრამა და მიიღო სასურველი შედეგი.

სასურველი შედეგი:

```
გამარჯობა! მე ვარ ნინო.

ჩემი პულტიანი მანქანა STEAM-ია!
```

დავალება 1.1: შეცდომის პოვნა და გასწორება

კოდი თითქმის მზად არის, მაგრამ პროგრამისტს წერისას რამდენიმე შეცდომა მოუვიდა და პროგრამა არ მუშაობს. შენი ამოცანაა, იყო ნამდვილი კოდის დეტექტივი, იპოვო და გაასწორო ორივე შეცდომა, რათა პროგრამამ სწორად იმუშაოს.

განმარტება:

- კოდის რედაქტორში, `setup()` ფუნქციის შიგნით, მოცემულია ორი ტიპის ბრძანება. პირველი, `Serial.begin(9600);`, რომელიც რთავს ჩვენს „ჩატის ფანჯარას“ და ამზადებს მას კომუნიკაციისთვის.
- მეორე ბრძანება `Serial.println()` ტექსტის დასაბეჭდად არის განკუთვნილი. `println` ნიშნავს, რომ დაბეჭდვის შემდეგ კურსორი ავტომატურად გადავა ახალ ხაზზე.
- თავის მხრივ `void loop()` ფუნქცია პასუხისმგებელია კოდის მუდმივ გამეორებაზე. რადგან დაბეჭდვა მხოლოდ ერთხელ გვინდა ეს ფუნქცია ცარიელი რჩება.

მინიშნება: ყურადღება მიაქციე ბრძანებების დაბოლოებებს და იმას, თუ რომელი ბრძანება გადაიტანს ტექსტს ახალ ხაზზე.

```
void setup() {
  // ეს ბრძანება ამზადებს სერიულ მონიტორს სამუშაოდ.
  Serial.begin(9600) // შეცდომა #1

  // ეს ბრძანებები ბეჭდავს ტექსტს.
  Serial.print("hello! i am [your name]");  // შეცდომა #2
  Serial.println("my  [object] is STEAM!");
}

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

დავალება 1.2: კოდის დასრულება

პროგრამის ძირითადი ნაწილი უკვე აგებულია, სერიული მონიტორი მზად არის სამუშაოდ. ახლა შენი ჯერია! დაამატე ორი ბრძანება მოცემულ კოდში, რათა დაასრულო პროგრამა და მიიღო სასურველი შედეგი.

განმარტება:

- კოდის რედაქტორში, `setup()` ფუნქციის შიგნით, მოცემულია ორი ტიპის ბრძანება. პირველი, `Serial.begin(9600);`, რომელიც რთავს ჩვენს „ჩატის ფანჯარას“ და ამზადებს მას კომუნიკაციისთვის.
- მეორე ბრძანება `Serial.println()` ტექსტის დასაბეჭდად არის განკუთვნილი. `println` ნიშნავს, რომ დაბეჭდვის შემდეგ კურსორი ავტომატურად გადავა ახალ ხაზზე.
- თავის მხრივ `void loop()` ფუნქცია პასუხისმგებელია კოდის მუდმივ გამეორებაზე. რადგან დაბეჭდვა მხოლოდ ერთხელ გვინდა ეს ფუნქცია ცარიელი რჩება.

```
void setup() {
  // ეს ბრძანება ამზადებს სერიულ მონიტორს სამუშაოდ.
  Serial.begin(9600);

  // --- ჩაამატე პირველი ბრძანება აქ ---
  // დაბეჭდე მისალმება შენი სახელით.

  // --- ჩაამატე მეორე ბრძანება აქ ---
  // დაბეჭდე შენი STEAM მაგალითი.

}

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

დავალება 1.3: შეიმუშავე პროგრამული კოდი

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

1. მოამზადებს სერიულ მონიტორს სამუშაოდ (კომუნიკაციის სიჩქარე: 9600).
2. დაბეჭდავს პირველ ხაზს: მისალმებას შენი სახელით.
3. დაბეჭდავს მეორე ხაზს: შენს პირად STEAM მაგალითს.

განმარტება:

- კოდის რედაქტორში, setup() ფუნქციის შიგნით, მოცემული უნდა იყოს ორი ტიპის ბრძანება. პირველი, Serial.begin(9600);, რომელიც რთავს ჩვენს „ჩატის ფანჯარას“ და ამზადებს მას კომუნიკაციისთვის.
- მეორე ბრძანება Serial.println() ტექსტის დასაბეჭდად უნდა იყოს განკუთვნილი. println ნიშნავს, რომ დაბეჭდვის შემდეგ კურსორი ავტომატურად გადავა ახალ ხაზზე.
- თავის მხრივ, `void loop()` ფუნქცია პასუხისმგებელია კოდის მუდმივ გამეორებაზე. რადგან დაბეჭდვა მხოლოდ ერთხელ გვინდა ეს ფუნქცია ცარიელი რჩება.

```
void setup() {
  // დაწერე მთლიანი კოდი აქ.
}
void loop() {
  // ეს ფუნქცია ამ ეტაპზე ცარიელი რჩება.
}
```

სწორი პასუხი (პროგრამული კოდი სრულად):

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

  Serial.println("hello! i am nino");
```

```
  Serial.println("my remote control car is STEAM”);
}

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": "first_println",
          "name": "First Serial.println command",
          "kind": "require_regex",
          "pattern":
"Serial\\.println\\s*\\(\\s*\"[^\"]+\"\\s*\\)",
          "flags": "iu",
          "must_pass": true,
```

```
          "source": "stripped",
          "msg_fail": "დაამატე პირველი Serial.println ბრძანება
ტექსტით.",
          "msg_pass": "პირველი Serial.println ბრძანება
დამატებულია."
        },
        {
          "id": "two_println_commands",
          "name": "Two Serial.println commands",
          "kind": "require_min_matches",
          "pattern":
"Serial\\.println\\s*\\(\\s*\"[^\"]+\"\\s*\\)",
          "min_matches": 2,
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "დაამატე ორი Serial.println ბრძანება (ჯამში
setup()-ში უნდა იყოს 2 Serial.println).",
          "msg_pass": "ორივე Serial.println ბრძანება
დამატებულია."
        },
        {
          "id": "serial_before_prints",
          "name": "Serial.begin before prints",
          "kind": "require_ordered_regex",
          "patterns": [
            "Serial\\.begin\\s*\\(\\s*9600\\s*\\)",
            "Serial\\.println\\s*\\("
          ],
          "flags": "iu",
          "must_pass": true,
          "source": "stripped",
          "msg_fail": "Serial.begin(9600) უნდა იყოს
Serial.println-მდე.",
          "msg_pass": "თანმიმდევრობა სწორია: Serial.begin →
Serial.println."
        }
      ]
    }
  ]
}
```