Arduino UNO

შესავალი

არდუინო წარმოადგენს ელექტრონულ კონსტრუქტორს და მოხერხებულ პლატფორმას სხვადასხვა სახის ელექტრონული მოწყობილობის შესაქმნელად. ის პოპულარულია გამოყენების ფართო არეალის, დაპროგრამების ენის სიმარტივის, ღია კოდისა და არქიტექტურის გამო. არდუინო საშუალებას აძლევს კომპიუტერს, გავიდეს ვირტუალური სამყაროს საზღვრებს გარეთ და გადავიდეს ფიზიკურში, მასთან ურთიერთქმედების მიზნით. არდუინოს ბაზაზე შექმნილ მოწყობილობებს შეუძლიათ მიიღონ ინფორმაცია გარემომცველ გარემოზე სხვადასხვა სენსორის მეშვეობით, ასევე შეუძლიათ სხვადასხვა სახის მმართველი ზემოქმედების განხორციელება.

არდუინოს საშუალებით შესაძლებელია საინტერესო, ინტერაქტიური პროექტების შექმნა. მაგალითად, როგორიცაა მოჩვენებების დეტექტორი, ჯოისტიკით კონტროლირებადი ლაზერი, ელექტრონული კუბი, ჭკვიანი სახლი და მრავალი სხვა. ყველა ეს პროექტი მარტივი გასაკეთებელია და აქვთ ერთი საერთო - ისინი იყენებენ არდუინოს.

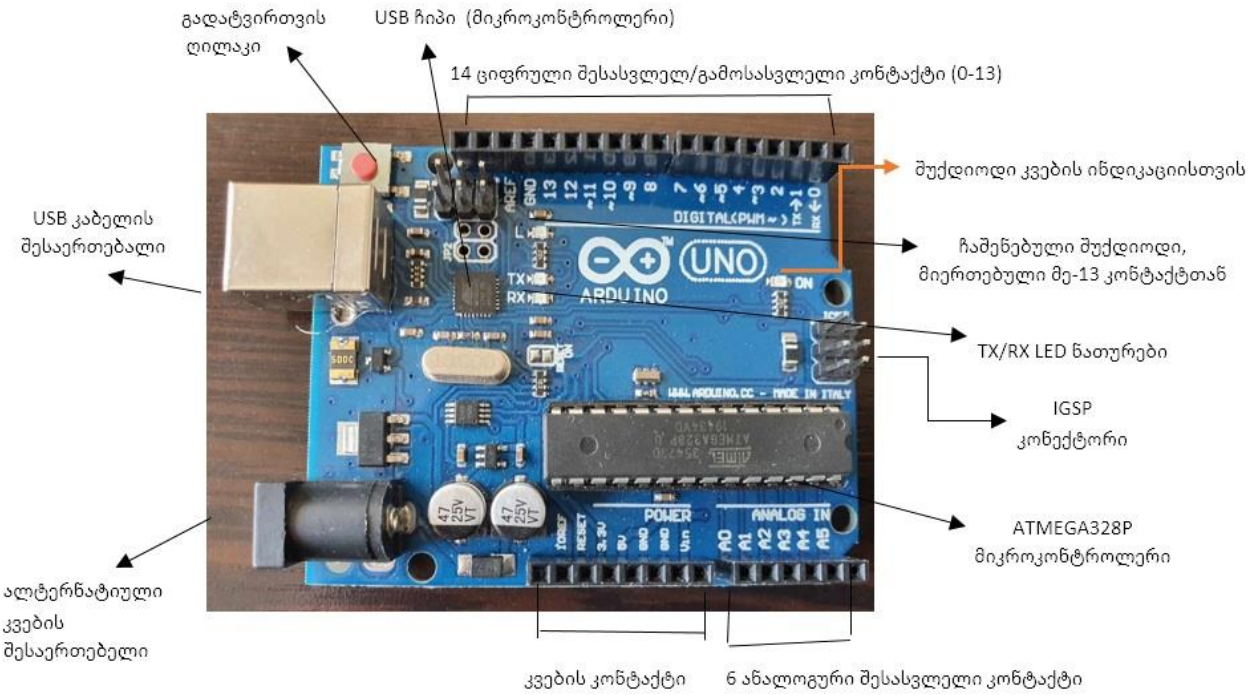
არდუინოს პოპულარობა მისი გამოყენების სიმარტივემ და დაბალმა ფასმაც განაპირობა. მოწყობილობა მუშაობს —plug-and-play‖ პრინციპით, რაც ნიშნავს იმას, რომ მომხმარებელს მისი კომპიუტერთან შეერთების მარტივი პროცესის შემდეგ დაუყოვნებლივ შეუძლია შეუდგეს მუშაობას.

არდუინოს პლატფორმის რამდენიმე ვერსია არსებობს. დღეისათვის ყველაზე გავრცელებულია Arduino UNO. მას უმეტესწილად ირჩევენ ადამიანები, რომლებიც მიკროკონტროლერების დაპროგრამების შესწავლას იწყებენ. არდუინო წარმოადგენს დაფას, მასზე განთავსებული კომპონენტებით, რომელთა შორის ყველაზე მნიშვნელოვანია მიკროკონტროლერი. Arduino UNO აგებულია ATmega328P-ს ბაზაზე. მიკროკონტროლერი ამ პლატფორმის ძირითადი გამოთვლითი სისტემაა. სწორედ მისთვის იქმნება პროგრამული უზრუნველყოფა, რომლის საშუალებითაც ის ურთიერთქმედებს გარე სამყაროსთან მონაცემთა შეტანა/გამოტანის სპეციალური პორტებით. მიკროკონტროლერი არდუინოს „ტვინია“, პატარა კომპიუტერი, რომელიც შეიცავს მიკროპროცესორს, ასრულებს ინსტრუქციებს, გააჩნია სხვადასხვა სახის მეხსიერება მონაცემებისა და ბრძანებების შესანახად, ასევე შესასვლელები და გამოსასვლელები მონაცემების შეტანა/გამოტანისთვის. Arduino UNO-ს აქვს 20 ციფრული და ანალოგური (6 ანალოგური შესასვლელი, 14 - ციფრული შესასვლელ-გამოსასვლელი) საკონტაქტო გამომყვანი.

მოწყობილობა

განვიხილოთ Arduino UNO-ს პლატა და მისი შესაძლებლობების ის ნაწილი, რომლებსაც თითქმის ყველა პროექტზე მუშაობისას გამოვიყენებთ.

სურათი 1: არდუინოს პლატა/პლატფორმა



არდუინო მართავს კომპონენტებს, რომლებსაც თქვენ მას მიუერთებთ, მაგალითად, ძრავები ან შუქდიოდები. შუქდიოდების მართვა ხდება მათზე ინფორმაციის გადაცემით გამომავალი სიგნალის სახით (ინფორმაცია, გამომავალი არდუინოდან). მონაცემებს, რომლებსაც არდუინო კითხულობს მრიცხველიდან, წარმოადგენენ შემავალ ინფორმაციას (ინფორმაცია, შემავალი არდუინოში). არსებობს 14 ციფრული შემავალ-გამომავალი პინი (0-13). თითოეული მათგანი შეიძლება იყოს გამოყენებული როგორც შემავალ, ისე გამომავალ პინად.

არდუინოს დაფაზე, ზემოთ აღნიშნული გამომყვანების ქვემოთ, მოთავსებულია შუქდიოდები. ისინი ანათებენ, როცა მათში დენი გადის. არდუინოს დაფაზე ოთხი შუქდიოდია: ერთი, რომელსაც აწერია ON, დაფაზე მარჯვენა მხარესაა მოთავსებული და მასზე მიერთებული ელექტროკვების ინდიკატორს წარმოადგენს, დანარჩენი სამი მარცხენა მხარეს, ერთმანეთის მიყოლებითაა განთავსებული.

- TX/RX LED ნათურები - მიმდევრობითი კომუნიკაციის შუქდიოდები
- Flash LED - L შუქდიოდი (ჩაშენებული შუქდიოდი, მიერთებული მე-13 კონტაქტთან)
- Power ON LED - შუქდიოდი ელექტროკვების ინდიკაციისთვის

RX და TX შუქდიოდები იმ შემთხვევაში ინთება, როდესაც ხდება მონაცემების გაცვლა არდუინოს დაფასა და მიმდევრობითი პორტისა და USB-ს მეშვეობით ჩართულ მოწყობილობებს შორის. L შუქდიოდი მიერთებულია მე-13 საკონტაქტო გამომყვანთან. ამ შუქდიოდებისგან მარცხნივ მოთავსებული მცირე ზომის შავი კვადრატი მიკროკონტროლერია, რომელიც USB ინტერფეისს მართავს. სწორედ ეს მიკროკონტროლერი აძლევს საშუალებას არდუინოს დაფას გააგზავნოს მონაცემები კომპიუტერზე ან მიიღოს ისინი კომპიუტერიდან.

როგორც ეს ზოგჯერ ხდება კომპიუტერების შემთხვევაში, არდუინოს დაფასაც შეიძლება დასჭირდეს გადატვირთვა. Reset ღილაკი სწორედ ამისთვისაა საჭირო.

კვება

Arduino UNO-ს კვების მიღება შეუძლია კომპიუტერზე მიერთებული USB კაბელიდან ან/და გარე კვების წყაროდან. გარე კვება (არა USB-დან) შეიძლება მიწოდებულ იქნას ან ძაბვის გარდამქმნელიდან AC/DC (კვების ბლოკი) ან აკუმულატორული ბატარეიდან. ძაბვის გარდამქმნელი მიუერთდება 2.1 მმ-იანი გასართის მეშვეობით ცენტრალური დადებითი პოლუსით. ბატარეიდან წამოსული გამტარების მიერთება ხდება კვების გასართის Gnd და Vin გამომყვანებთან. დაფის მიერ კვების წყაროს არჩევა ხდება ავტომატურად.

- სამუშაო ძაბვა 5 ვ;
- შესასვლელი ძაბვა (რეკომენდებული) 7 – 12 ვ;

ძაბვის გამომყვანები:

• VIN. - ეს შესასვლელი გამოიყენება კვების მიწოდების დროს გარე წყაროდან (5 ვოლტის არ არსებობის შემთხვევაში USB გასართიდან ანდა სხვა რეგულირებადი კვების წყაროდან). კვების ძაბვის მიწოდება ხდება ამ აღნიშნული გამომყვანის გავლით;

• 5V. - რეგულირებადი კვების ძაბვა, რომელიც გამოიყენება მიკროკონტროლერისა და დაფის სხვა კომპონენტების კვებისთვის. კვება შეიძლება მიწოდებულ იქნას VIN გამომყვანის გავლით ან USB გასართიდან, ან/და სხვა რეგულირებადი 5 ვ-იანი ძაბვის წყაროდან;

• 3V3. - 3.3 ვოლტის ტოლი ძაბვა გამომყვანზე. მისი გენერირება ხდება პლატაზე ჩაშენებული რეგულატორის მიერ. მაქსიმალური მოხმარების დენი 50 მა.

• GND. - დამიწების გამომყვანები.

დაპროგრამების მერე შეგვიძლია არდუინო გამოვაერთოთ კომპიუტერიდან და კვების წყაროდ გამოვიყენოთ ელემენტები.

სურათი 2: 9 ვოლტიანი ბატარეა არდუინოსთვის კვების მისაწოდებლად

რატომ არდუინო?

მსოფლიოში არსებობს მიკროკონტროლერებისა და მათი პლატფორმების დიდი რაოდენობა. ბევრი მათგანი მეტად ამარტივებს მიკროკონტროლერის (მკკ) დაპროგრამების და ამუშავების პროცესს. არდუინოს მიზანი და ძირითადი ხიბლის ზუსტად ეს არის: მუშაობის გამარტივება მასწავლებელი, მოსწავლე და მოყვარულებისთვის. მისი ძირითადი უპირატესობა შემდეგშია:

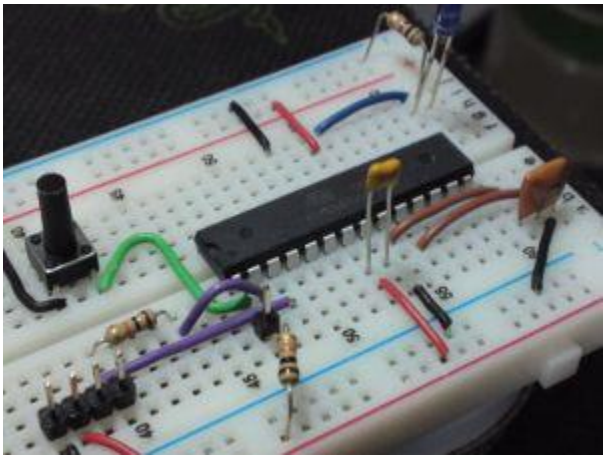
დაბალფასიანი – სხვებთან შედარებით, არდუინოს პლატფორმა საკმაოდ იაფია. ასევე, ღიად ხელმისაწვდომია დაფის ნახაზები და ყოველგვარი ინფორმაცია. ადამიანს შეუძლია თავისივე ხელით ააწყოს და აამუშაოს ეს დაფა.

არდუინოს პროგრამული უზრუნველყოფა მუშაობს როგორც Linux-ზე, ასევე Windows-ზე.

არდუინოს ყველა პროგრამული ნაწილის კოდის წყარო თავისუფალ წვდომაშია ყველა მსურველისთვის. ენის შესაძლებლობა შეიძლება გაფართოვდეს C++ ბიბლიოთეკების მეშვეობით.

ისიც კი შესაძლებელია, რომ მცირე გამოცდილების მქონე ადამიანმა აიღოს სამაკეტო დაფა და მასზე თვითონ ააწყოს იაფფასიანი არდუინო სისტემა (იხ. სურათი 3), ეს სასარგებლოა მისი მუშაობის პრინციპების შესასწავლად.

სურათი 3.



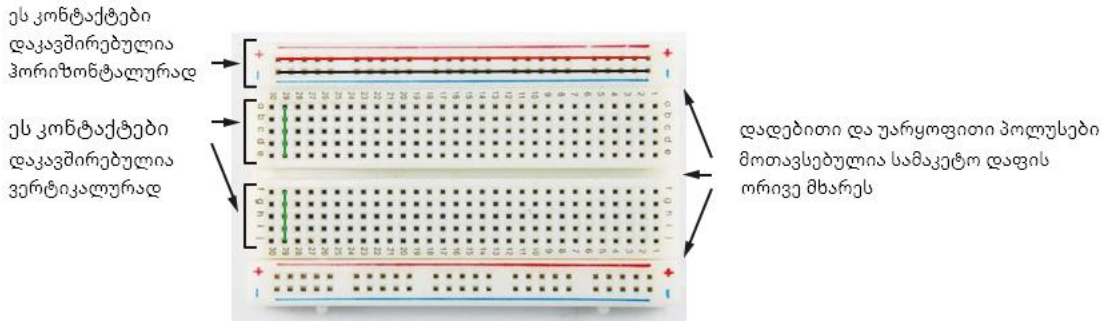
სამაკეტო დაფა

სამაკეტო დაფა (მას ფიჭურ დაფასაც უწოდებენ) წარმოადგენს კონსტრუქციულ საფუძველს ელექტრონიკის პროტოტიპების შესაქმნელად. აქ წარმოდგენილი ყველა პროექტისთვის მირჩილვის ნაცვლად ვიყენებთ სამაკეტო დაფას.

სახელწოდება „სამაკეტო დაფა“ ჯერ კიდევ იმ დროიდან არსებობს, როცა პროექტები ელექტრონიკის მიმართულებით იქმნებოდა ხის დაფებზე. ხეში ამაგრებდნენ პატარა ლურსმნებს, რომლებზეც კომპონენტების მისაერთებლად ახვევდნენ მავთულს, მირჩილვის გარეშე. თანამედროვე სამაკეტო დაფა, ისეთი, როგორიც ნაჩვენებია მეოთხე სურათზე, დამზადებულია პლასტიკისგან, წინასწარ დატანილი ხვრელებით (ე.წ. დასამაგრებელი

წერტილებით). ამ ხვრელებში, ასევე წინასწარ დატანილი მომჭერებით, მაგრდება კომპონენტები ან სამაკეტო მავთულები (იგივე გამტარები, ჯამფერები). ეს წერტილები ერთმანეთთან დაკავშირებულია დაფის ქვეშ დატანილი გამტარი ზოლებით.

სურათი 4: სამაკეტო დაფა. კავშირები

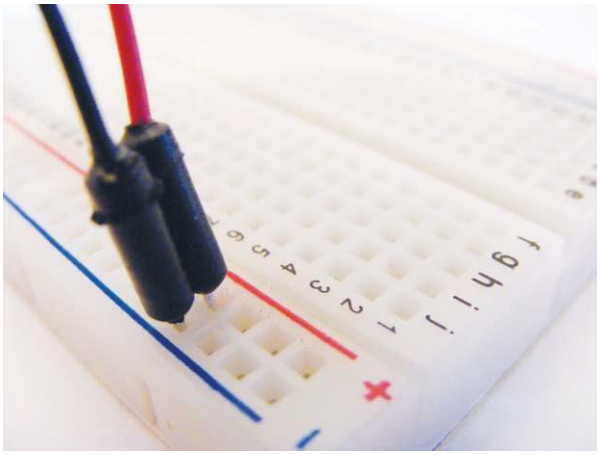


სამაკეტო დაფა არსებობს სხვადასხვა ზომის. ჩვენ მიერ შემოთავაზებული პროექტების შესაქმნელად თქვენ დაგჭირდებათ ოთხი სამაკეტო დაფა: ორი - 830 ხვრელით, ერთი - 420 და ერთიც, პატარა - 170 ხვრელით. დიდი, 830-ხვრელიანი სამაკეტო დაფა იდეალურია ისეთი პროექტების შესაქმნელად, რომლებშიც გამოიყენება LCD ეკრანი ან დიდი რაოდენობის კომპონენტები. ხოლო 420 და 170 ხვრელიანი დაფები უკეთესია პატარა პროექტებისთვის. ჩვენ მიერ შემოთავაზებული პროექტებისთვის რეკომენდაციას ვუწევთ მე-4 სურათზე ნაჩვენებ სამაკეტო დაფას წითელი და ლურჯი ზოლებით და ცენტრალური გამყოფით.

სამაკეტო დაფის მთავარ სივრცეს აქვს ერთმანეთთან ვერტიკალურად დაკავშირებული მიერთების წერტილების (ხვრელების) 30 სვეტი. სამაკეტო დაფის ცენტრში არის გამყოფი ზოლი.

დაფის ქვემოთ და ზემოთ არსებული წითელი და ლურჯი ზოლებიდან ხდება დაფის ცენტრალურ/ძირითად სივრცეზე განთავსებული კომპონენტებისთვის კვების მიწოდება (იხილეთ სურ. 5). კვების წითელი და ლურჯი ზოლები ყველა ხვრელს ჰორიზონტალურად აკავშირებს ერთმანეთთან. წითელი ზოლი განკუთვნილია დადებითი „+“ პოლუსისთვის, ლურჯი - „-„ პოლუსისთვის (ე.წ. დამიწებისთვის, GND).

სურათი 5.



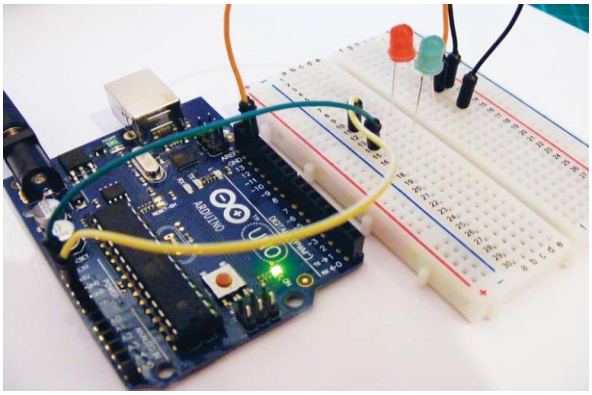
რჩევა

ჩვეულებრივ, + პოლუსისთვის
(სამაკეტო დაფაში რომ შევიყვანოთ „+“),
იოლად რომ განვასხვაოთ ერთმანეთისგან,
გამოვიყენოთ წითელი ფერის გამტარი,
ხოლო დამიწებისთვის („-„ პოლუსისთვის, GND) - შავი.
დანარჩენი გამტარების ფერები შეგიძლიათ
შეარჩიოთ თქვენი სურვილისამებრ.

სამაკეტო დაფაზე შეერთებების შესაქმნელად ვიყენებთ ჯამფერებს/გამტარებს ორივე ბოლოზე პლასტმასის თავებით, რაც აიოლებს დაფის ხვრელებში მის ჩამაგრებასა და ამოძრობას. თქვენ შეგიძლიათ ჩვეულებრივი გამტარების გამოყენებაც. თუმცა, ამ შემთხვევაში, გამტარი უნდა იყოს ერთწვერიანი, რომ არ გაგიჭირდეთ მისი სამაკეტო დაფის ხვრელში ჩამაგრება.

როცა თქვენ სამაკეტო დაფის ხვრელში ამაგრებთ ჯამფერს, მას იჭერს დაფის ქვეშ დატანილი ზამბარული მომჭერი. ამით თქვენ ქმნით ელექტრულ შეერთებას/კავშირს რიგში, რომელიც, ჩვეულებრივ, ხუთი ხვრელისგან შედგება. შემდეგ, მეზობელ ხვრელში შეგიძლიათ მოათავსოთ კომპონენტი სქემის შესაქმნელად, როგორც ეს ნაჩვენებია მე-6 სურათზე.

სურათი 6: სამაკეტო დაფის სქემის მაგალითი



არდუინოს პროგრამირება

თუ გვინდა, რომ ჩვენმა პროექტმა შეასრულოს ჩვენ მიერ მიცემული დავალება, აუცილებელია, დაიწეროს პროგრამა, რომელიც ინსტრუქციას მისცემს არდუინოს. ამისათვის არსებობს ინსტრუმენტი, სახელწოდებით, არდუინოს პროგრამირების მოდული (IDE). Arduino IDE შესაძლებელია ჩამოვტვირთოთ ვებგვერდიდან: https://www.arduino.cc/. ეს საიტი ხელმისაწვდომია როგორც Windows-თვის, ისე OS X и Linux-თვის. ეს საშუალებას გვაძლევს, დავწეროთ კომპიუტერული პროგრამები (ნაბიჯ-ნაბიჯ ინსტრუქციების ნაკრები, რომელსაც არდუინოს სამყაროში კოდებს/Sketch უწოდებენ), რომლებიც არდუინოში ჩაიტვირთება USB კაბელის მეშვეობით. არდუინო შეასრულებს ინსტრუქციებს გარე სამყაროსთან ურთიერთქმედების საფუძველზე.

შენიშვნა

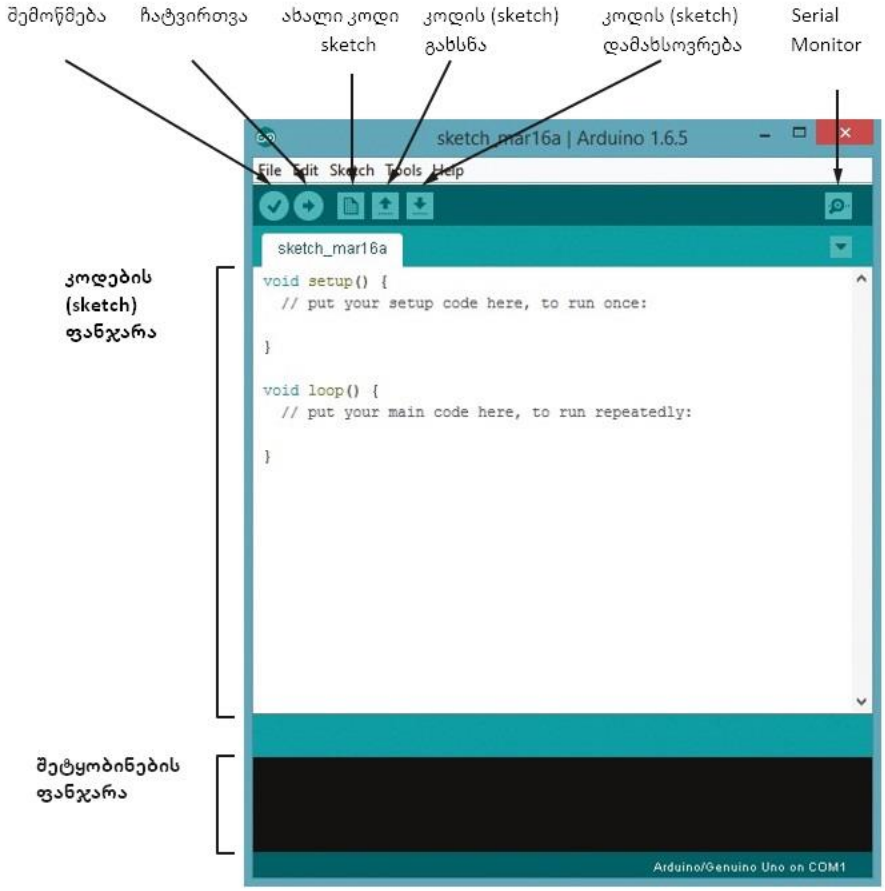
ვინაიდან IDE ვერსიები საკმაოდ სწრაფად იცვლება,
აქ არ შემოგთავაზებთ მისი დაყენების ინსტრუქციას.
თქვენ თავად უნდა მოძებნოთ
მისი მარტივი ვერსიის დაყენების საშუალება.
IDE ყველა ვერსია და მისი დაყენების სრული ინფორმაცია
თქვენი ოპერაციული სისტემისთვის ხელმისაწვდომია
საიტზე: https://www.arduino.cc/.

Arduino (IDE) - პროგრამირების გარემო
IDE ინტერფეისი

გახსნილი პროგრამირების გარემო (Arduino IDE) უნდა გამოიყურებოდეს დაახლოებით ისე, როგორც ეს მე-7 სურათზეა ნაჩვენები.

პროგრამირების გარემოს ზედა ნაწილში მოცემულია ორი ზოლი - მენიუ და ინსტრუმენტების პანელი, ყველაზე ხშირად გამოსაყენებელი ფუნქციებით; შემდეგ, ცენტრში (დიდი თეთრი ფანჯარა), პროგრამული კოდის (sketch) ფანჯარა; ქვედა ზოლში კი აისახება არდუინოს კომპიუტერთან დაკავშირების ინფორმაცია და შეცდომები - გიჩვენებთ, სწორად არის თუ არა შედგენილი თქვენი პროგრამული კოდი - sketch.

სურათი 7: პროგრამირების გარემო (IDE)



ღილაკების განმარტება:

Verify - ამოწმებს პროგრამული კოდების სისწორეს არდუინოში ჩატვირთვამდე.

Upload - ჩატვირთავს sketch-ის ფანჯარაში არსებულ კოდს არდუინოში. შემდგომი გაუგებრობების თავიდან აცილების მიზნით, ჩატვირთვამდე დარწმუნდით, რომ არდუინოს ტიპი სწორად გაქვთ შერჩეული.

New - ხსნის ახალ "ფურცელს" ახალი sketch-ის შესაყვანად. IDE შეგეკითხებათ, რა უნდა დაერქვას ახალ sketch-ს და სად იქნება ის შენახული.

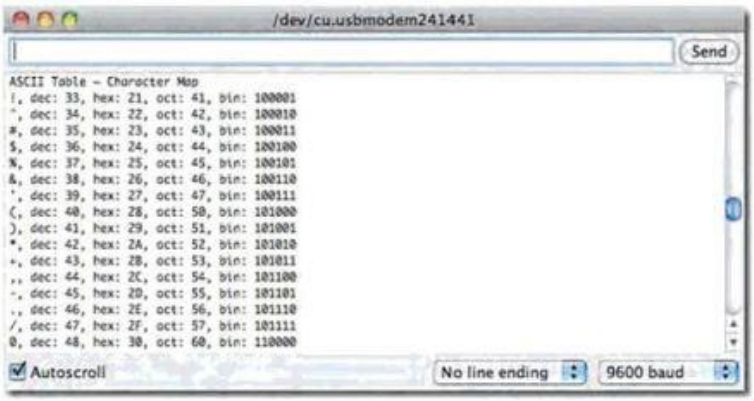
Open - გახსნის შენახულ sketches-ს.

Save - შეინახავს sketch-ის ფანჯარაში შეყვანილ კოდს sketch ფაილში. შენახვის დამთავრების შემდეგ კოდების ფანჯარაში გამოვა შეტყობინება "Done Saving".

ასევე, წესად შემოიღეთ ყოველი sketch-ის შემოწმება შენახვის წინ და შეინახეთ არდუინოში ჩატვირთვამდე.

Serial Monitor - საჭირო ინსტრუმენტია კოდებზე სამუშაოდ და შეცდომების გასასწორებლად. მისი საშუალებით შესაძლებელია არდუინოდან გამოგზავნილი მონაცემების კონტროლი. ასევე, მისი საშუალებით შესაძლებელია ინფორმაციის გაგზავნა არდუინოზე. Serial Monitor-ის მუშაობის მაგალითი იხ. მე-8 სურათზე. ქვედა მარჯვენა კუთხეში მოცემულია არდუინოსა და კომპიუტერს შორის გადაცემა/მიღების სიჩქარე ბიტი წამში (ბიტი/წმ). სტანდარტული სიჩქარეა 9600baud ანუ 9600ბიტი/წმ. რადგან ერთი სიმბოლო იკავებს 8 ბიტს, ამიტომ 9600ბიტი/წმ ტოლია 1200 სიმბოლო/წმ ან/და 1200 ბაიტი/წმ.

სურათი 8: serial window მუშაობის დროს



ზედა ნაწილში განლაგებულია ფანჯარა, სადაც შეიძლება ტექსტის შეყვანა და Send ღილაკით არდუინოზე გაგზავნა.

IDE-ს ქვედა ნაწილში განთავსებულია ფანჯარა, სადაც შეიძლება შეცდომების ნახვა (წითლად მონიშნული ტექსტი) მაშინ, როდესაც IDE ცდილობს არდუინოსთან შეერთებას, ან კოდების ჩატვირთვას, ან კოდების სისწორის შემოწმებას. ამ ფანჯრის მარცხენა მხარეს გამოისახება

ნომერი, რომელიც შეესაბამება შეცდომის ადგილმდებარეობას, რაც აადვილებს ამ შეცდომების მოძებნას და გასწორებას.

არდუინოს კოდები (sketches)

როგორც ნებისმიერი კომპიუტერული პროგრამა, კოდებიც წარმოადგენს ინსტრუქციების მკაცრად განსაზღვრულ ნაკრებს და ძალიან მგრძნობიარეა შეცდომების მიმართ. იმაში დასარწმუნებლად, რომ კოდი სწორად ჩასვით კოდის ფანჯარაში, დააჭირეთ ღილაკს „შემოწმება“ (პროგრამირების გარემოს ზედა მარჯვენა კუთხე). იმ შემთხვევაში, თუ კოდი შეცდომით არის ჩაწერილი, ინფორმაცია გამოგიჩნდებათ პროგრამირების გარემოს ქვედა ზოლში (შეტყობინების ფანჯარა) და გაცნობებთ, სწორად არის თუ არა შედგენილი კოდი.

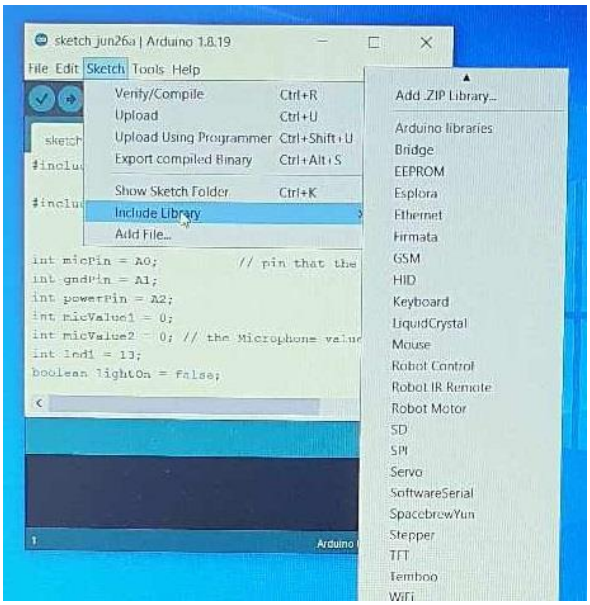
ბიბლიოთეკები

ბიბლიოთეკა პროგრამირების გარემოში შეგიძლიათ ნახოთ მენიუ Sketch-ში, როგორც ეს ნაჩვენებია მე-9 სურათზე. მენიუ Sketch-ზე დაწკაპუნებისას ჩამოიშლება პატარა ფანჯარა, სადაც ნახავთ Include Library-ს, რომელიც გიჩვენებთ ხელმისაწვდომ ბიბლიოთეკებს. მაგალითად, ბიბლიოთეკა LiquidCrystal საშუალებას მოგცემთ იმუშაოთ LCD ეკრანთან.

Library არის კოდების კრებული, რომლებიც საჭიროებისამებრ შეიძლება ჩართული იყოს თქვენს sketch-ში, რომ გაფართოვდეს, გაძლიერდეს თქვენი პროექტის შესაძლებლობები. ამით დაზოგავთ დროს და შეძლებთ თქვენ მიერ ან სხვისი შექმნილი კოდის გამოყენებას სხვადასხვა პროექტებში.

ჩვენ მიერ შემოთავაზებული პროექტების შესაქმნელად დაგჭირდებათ შემდეგი ბიბლიოთეკების იმპორტირება: RFID, Tone, Pitches, Keypad, Password, Ultrasonic, NewPing, IRRemote, and DHT. ყველა საჭირო ბიბლიოთეკის ჩამოტვირთვის ინსტრუქციას შემოგთავაზებთ კონკრეტულ პროექტზე მუშაობის პარალელურად. ბიბლიოთეკების ჩამოტვირთვის შემდეგ საჭიროა მათი ინსტალაცია Arduino UNO 1.8.19 ვერსიაში.

სურათი 9: ბიბლიოთეკები



არდუინოს შემოწმება

მოციმციმე ინდიკატორი და LED ნათურა

არდუინოსთან მუშაობის დაწყებამდე, გთავაზობთ რამდენიმე საჭირო ინფორმაციას, რომელიც აუცილებლად უნდა იცოდეთ. მოდით, შევხედოთ მოწყობილობას და პროგრამულ უზრუნველყოფას, რომელიც დაგჭირდებათ ამ სახელმძღვანელოსთვის. საჭიროა, ჯერ შეამოწმოთ არდუინო მარტივი შუქდიოდის საშუალებით, გამოიმუშაოთ უნარ-ჩვევები, რომლებიც დაგეხმარებათ დარჩილვასა და სასარგებლო კოდების ბიბლიოთეკის ჩატვირთვაში.

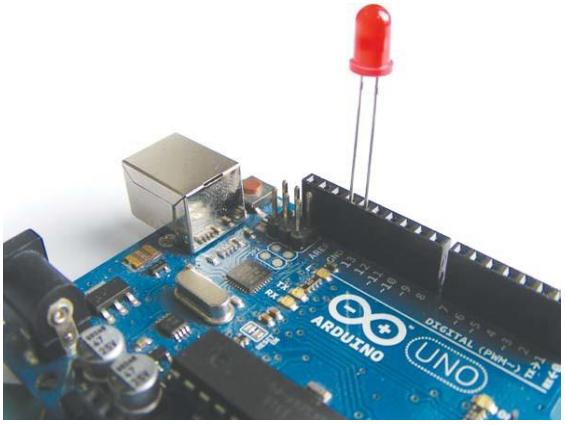
ჩვენ უკვე გავეცანით არდუინოს პროგრამირების გარემოს (IDE), მის ინტერფეისს. ახლა დროა, გავაკეთოთ მისი პირველი, კლასიკური პროექტი - შუქდიოდის ციმციმი. ეს საშუალებას მოგვცემს, არა მარტო შევამოწმოთ, როგორ მუშაობს ჩვენი არდუინო, არამედ გავეცნოთ მარტივ კოდს/sketch-ს. Sketch - ეს არის მხოლოდ ინსტრუქციების სერია. არდუინოს ერთდროულად შეუძლია მხოლოდ ერთი კოდის/sketch-ის დამახსოვრება. ამიტომ, მისი დამახსოვრების შემდეგ, რამდენჯერაც აამუშავებთ არდუინოს, შესრულდება ეს sketch/კოდი, ვიდრე ახალს არ ჩაწერთ.

ამჯერად ჩვენ მაგალითად ავიღებთ არდუინო IDE-ს მიერ შემოთავაზებულ საბაზო პროექტს Blink - შუქდიოდის ციმციმი. ამისთვის დაგვჭირდება ერთი LED ნათურა, რომელიც ერთწამიანი ინტერვალით აინთება და ჩაქრება. **შუქდიოდებს ესაჭიროებათ წინაღობა, სხვა შემთხვევაში ისინი გადაიწვება. ასეთი წინაღობა არდუინოში ჩაშენებულია მე-13 პინში.**

მაშ, ასე, გავყვეთ ნაბიჯ-ნაბიჯ:

1. LED ნათურის გრძელი ფეხი (დადებითი პოლუსი, +5 ვ., იგივე ანოდი) მიუერთეთ მე-13 პინს, ხოლო მეორე ფეხი (უარყოფით პოლუსი, იგივე კათოდი) – GND-ს (ე.წ. დამიწება), როგორც ეს ნაჩვენებია მე-10 სურათზე;

სურათი 10: პროექტი Blink



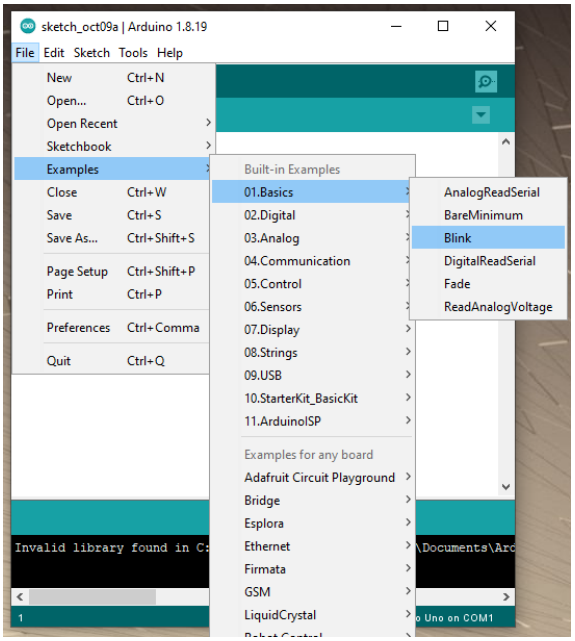
გაფრთხილება!

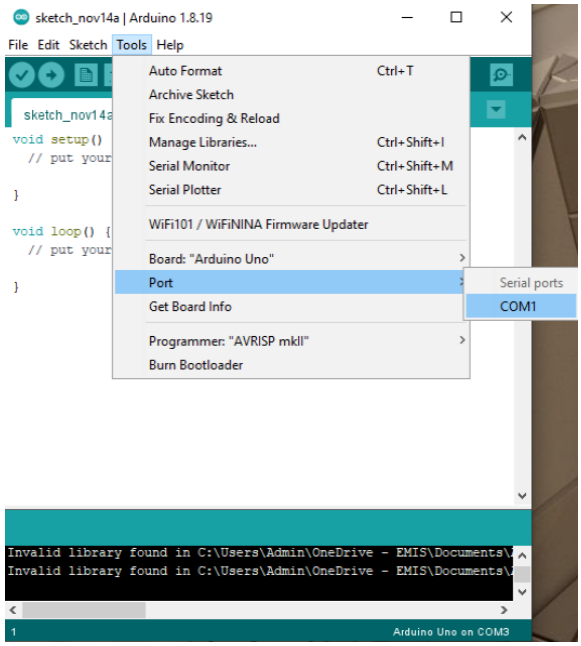
ყოველი პროექტის აწყობისა და დაშლის დროს არდუინო უნდა იყოს გამორთულ მდგომარეობაში.

2. დაუკავშირეთ არდუინო USB კაბელით თქვენს კომპიუტერს;
3. შეამოწმეთ, დაკავშირებულია თუ არა არდუინო პორტთან (იხ. მე-12 სურათი);
4. ჩასვით პროგრამული კოდი - sketch არდუინო IDE-ს კოდის/sketch-ის ფანჯარაში

როგორც ნაჩვენებია მე-11 სურათზე (File-Examples-Basics-Blink);

სურათი 11: როგორ მოვძებნოთ კოდი Blink არდუინოში. **სურათი 12:** არდუინოს დაკავშირება პორტთან.





Sketch/კოდი Blink:

1. // Blinking LED Project
2. int led = 13;
3. void setup() {
4. pinMode(led, OUTPUT);
   }
5. void loop() {
6. digitalWrite(led, HIGH);
7. delay(1000);
8. digitalWrite(led, LOW);
9. delay(1000);
10. }

---

5. დააწკაპუნეთ ღილაკზე **Verify**, რომ დარწმუნდეთ კოდის/sketch-ის მუშაობაში (სწორია თუ არა კოდი);
6. დააწკაპუნეთ ღილაკზე **Upload**, რომ კოდი/sketch ჩაიტვირთოს არდუინოში.

Sketch-ის განმარტება

პირველი სტრიქონი არის კომენტარი. ნებისმიერი სტრიქონი, რომელიც თქვენს პროგრამაში იწყება ნიშნით //, განკუთვნილია მხოლოდ მომხმარებლისთვის და მას არდუინო არ კითხულობს. ის გამოიყენება შენიშვნებისა და კოდის აღწერისთვის (კოდის კომენტირება). თუ კომენტარი ერთ სტრიქონზე მეტია, მაშინ პირველი სტრიქონი იწყება სიმბოლოთი /* და სრულდება */ სიმბოლოთი. მთელი ტექსტი, რაც ამ სიმბოლოთა შორის იქნება, არდუინოს მიერ იგნორირდება.

მეორე სტრიქონი გვეუბნება, რომ ამ კოდში მე-13 გამომყვანმა პინმა მიიღო სახელწოდება LED. რამდენჯერაც Sketch-ში ნახსენები იქნება LED, ის ყოველთვის შეეხება მე-13 პინს.

void setup (მოთავსდება პარამეტრების დასაყენებელი კოდი) ნიშნავს, რომ კოდი/Sketch, მოქცეული ფიგურულ ფრჩხილებში {}, რომელიც მოსდევს შეტყობინებას, პროგრამის გაშვებისას შესრულდება მხოლოდ ერთხელ. სიმბოლოთი { იწყება კოდის/Sketch-ის წერა.

PinMode ფუნქციით განვსაზღვრავთ საკონტაქტო გამომყვანს, რომელთანაც ვმუშაობთ და მის მდგომარეობას (INPUT ან OUTPUT). pinMode ფუნქციის არგუმენტებია 13 და OUTPUT. თუ ვკითხულობთ ინფორმაციას საკონტაქტო გამომყვანიდან, ეს იქნება INPUT, თუ ვწერთ ინფორმაციას საკონტაქტო გამომყვანში, მაშინ გვექნება OUTPUT. რადგანაც ჩვენ ვწერთ (ვაგზავნით) მე-13 გამომყვანში ინფორმაციას, ამიტომაც pinMode ფუნქციის არგუმენტი იქნება OUTPUT. **pinMode(LED, OUTPUT)** ბრძანებით საკონტაქტო გამომყვანის კონფიგურირება ხდება, როგორც გამოსასვლელის. ე.ი. ის ატყობინებს არდუინოს, რომ მე-13 პინი გამომავალია, რაც ნიშნავს, რომ ჩვენ გვინდა კვება მივაწოდოთ LED ნათურას (შუქდიოდს). სიმბოლო } ამთავრებს კოდს.

void loop (აქ მოთავსდება ძირითადი კოდი) ქმნის ციკლს. ფიგურულ ფრჩხილებში მოქცეული ინფორმაცია პროგრამის გაშვებისას განმეორდება ციკლურად, დაუსრულებლად, ვიდრე არ გავთიშავთ არდუინოს.

loop() ფუნქციის შიგნით ვწერთ **digitalWrite** ფუნქციას. მისი საშუალებით შუქდიოდის ჩართვა ან გამორთვა შეგვიძლია. ჩვენ ვაგზავნით, ვთქვათ 5V სიგნალს მე-13 საკონტაქტო გამომყვანზე, მაშინ არგუმენტად ვწერთ HIGH-ს. ეს ნიშნავს რომ ამ დროს შუქდიოდი ინთება.

შემდეგ პროგრამაში ვრთავთ **delay** ფუნქციას, რომელიც ON და OFF მდგომარეობას შორის დაყოვნებას უზრუნველყოფს. ამ ფუნქციის პარამეტრი მილიწამებშია მოცემული. ჩვენ შემთხვევაში, დაყოვნების დრო იქნება 1000 მილიწამი (1 წამი).

მერვე სტრიქონში ვცვლით **digitalWrite** ფუნქციას და მე-13 საკონტაქტო გამომყვანზე არგუმენტად ვწერთ LOW-ს. ეს ნიშნავს, რომ ამ დროს შუქდიოდს კვება აღარ მიეწოდება და ის ჩაქრება.

მეცხრე სტრიქონში კვლავ ვუთითებთ delay ფუნქციას და დაყოვნების დრო იქნება 1000 მილიწამი (1 წამი).

მეათე სტრიქონის ეს სიმბოლო } ნიშნავს ციკლის დასრულებას.

კოდის წერისას განსაკუთრებული მნიშვნელობა ენიჭება ყველა ნიუანსს: ასომთავრულ ასოებს, წერტილ-მძიმეს, კოდების მოთავსებას სიმბოლოებში {}... ნებისმიერი გადაცდომა არდუინოს მიერ აღიქმება, როგორც შეცდომა და პროგრამა არ იმუშავებს.

პროგრამული კოდი Blink

```
// Blinking LED Project
int led = 13;
void setup() {
pinMode(led, OUTPUT);
}
void loop() {
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
```

როგორც პროგრამის კოდიდან ჩანს, ჯერ უნდა მოხდეს შუქდიოდის ანთება, შემდეგ 1 წამის განმავლობაში ხდება დაყოვნება ანუ შუქდიოდი რჩება ანთებულ მდგომარეობაში, ამის შემდეგ შუქდიოდი ჩაქრება და მისი ჩამქრალ მდგომარეობაში ყოფნის დრო ისევ 1 წამია, შემდეგ ისევ აინთება შუქდიოდი და ა.შ. ეს პროცესი უსასრულოდ, ციკლურად მეორდება. დავაჭიროთ upload ღილაკს. დავინახავთ, როგორ ციმციმებს შუქდიოდი.

თქვენი სურვილის შემთხვევაში, შესაძლებელია პროექტის მცირედით გართულება: შუქდიოდის დამატება ან/და დაყოვნების დროის ცვლილება. ამ შემთხვევაში თქვენ მოგიწევთ კოდის შეცვლა (იხ. Sketc-ს განმარტება)

გამოყენებული ლიტერატურა:

1. Mark Geddes – Arduino Project Handbook. 2016
2. ზ. ტაბატაძე, თ. თოდუა - არდუინო. პრაქტიკული სახელმძღვანელო დამწყებთათვის. თბილისი, 2019
3. ჯ. გრიგალაშვილი - Arduino-ს ვიზუალური დაპროგრამება FLProg გარემოში. საქართველოს ტექნიკური უნივერსიტეტი. თბილისი, 2015