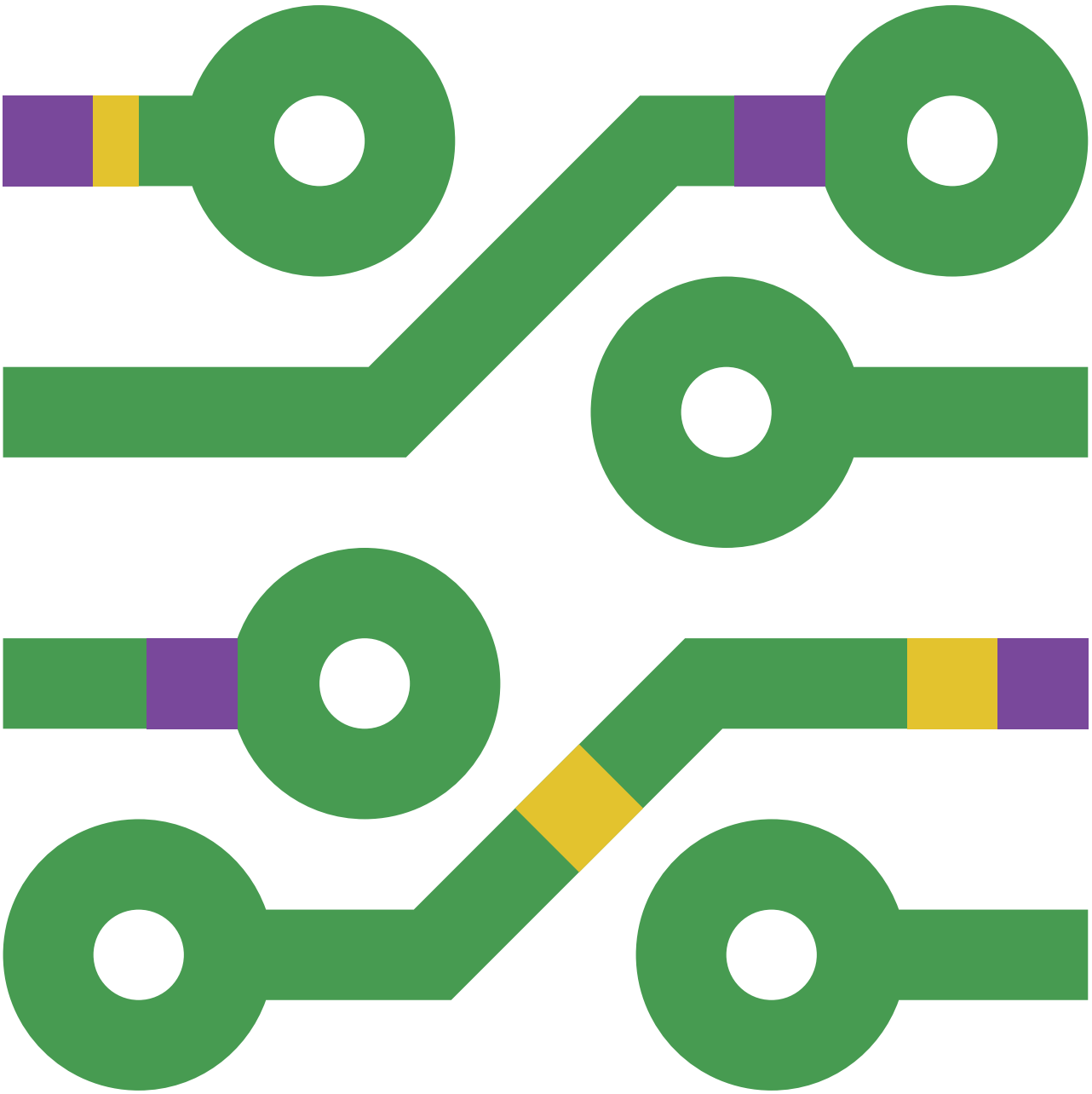


ARDUINO ЗА СВЕ

ПРИРУЧНИК ЗА НАСТАВНИКЕ

Основна школа

Мр сци. Муамер Халиловић,
дипл. инж. електротехнике

Подржано од:

< IT Girls >

УЈЕДИЊЕНЕ НАЦИЈЕ
БОСНА И ХЕРЦЕГОВИНА

ARDUINO ЗА СВЕ

ПРИРУЧНИК ЗА НАСТАВНИКЕ
Основна школа

Мр сци. Муамер Халиловић, дипл. инж. електротехнике

2019.

Садржај

Увод

У реду, вјерујте, ни мени није лако почети. Али као и свако путовање, и ово мора некако да почне. Прво што морам написати јесте да је важно схватити да је овај приручник писан једноставним рјечником, те да су неке ствари поједностављене да би их ученици лакше разумјели. Наравно, охрабрујем све оне који желе више истраживати и отићи много даље од овог приручника. Вјерујем да многи појмови који ће се овдје спомињати и нису тако непознати. Циљ је ово учинити разумљивим, ствари поједноставити на тај начин да знање стечено помоћу овог припручника буде довољно да читатељ добије самопоуздање да самостално настави истраживати свијет микрорачунара.

Наши животи се врте око мегабајта, мегахерца, чипова, процесора, AI, IoT, робота, ма, једноставно, читава збрка појмова и, признајете, некада се правимо да их разумијемо, а некада помислимо како би било баш добро користити и стварати, једноставно бити у том свијету, неки би рекли схватити матрицу (оп. а. MATRIX).

Док пишем ово, намјерно користим појам *стварати*, јер умјетници стварају и креирају, зар не? А програмери су, хтјели ми то признати или не, умјетници новог доба.

Одмах морамо направити разлику између класичних програмера и embedded програмера. Embedded или програмери микроконтролерских или SoC система нешто су као Ferrari у аутомобилској индустрији, прича за себе, па, ако могу бирати, бирам Ferrari.

Пројекат Arduino започет је 2003. године као програм за студенте на Interaction Designe Institute Ivrea, Ивреа, Италија. Циљ је био креирати јефтину и једноставну платформу за изучавање и креирање микроконтролерских система, како за почетнике тако и за професионалце. Платформа је осмишљена тако да на једноставан начин омогући креирање и тестирање прототипова уређаја који имају интеракцију с околином користећи сензоре и актуаторе.

Број продатих оригиналних Arduino платформи је милионски, а број клонираних, вјероватно, никада нећемо ни сазнати. Дакле, коначно смо у прилици да уђемо у један нови свијет у којем је дословно само небо граница.

Мр сци. Муамер Халиловић, дипл. инж. ел.

Arduino – Main chip

Прије него што уђемо у свијет микроконтролера, морамо направити поређење између „мозга” Arduino платформе и класичног CPU-а из десктоп или лаптоп рачунара.

Arduino платформа је израђена око ATMEL ATMEGA 328P контролера сљедећих карактеристика:

- 28 ножица
- Напајање 3 - 5 V
- 0,1 W
- Ради на 16 MHz
- 32 KB HDD – flash меморије
- 2 KB RAM меморије
- Цијена: око 2 $

Слика 1. ATMEGA 328P на Arduino платформи и изван ње

Модерни PC врло вјероватно има CPU 6, 7. или 8. генерације, а за поређење узећемо i5-6400.

Слика 2. CPU i5 6. генерације

- 1151 ножица
- Напајање 1,35 V
- 35 W
- Ради на 2,8 GHz
- Нема flash меморију, али већина рачунара данас има HDD с минимално 250 GB
- Нема интерни RAM, али већина рачунара данас има минимално 4 RAM-а
- Цијена: око 150 $

Наравно да је CPU далеко „моћнији" уређај, али замислите само уређај за мјерење тјелесне температуре величине slim-fit десктоп рачунара, који кошта неколико стотина конвертибилних марака. Једноставно, нема смисла.

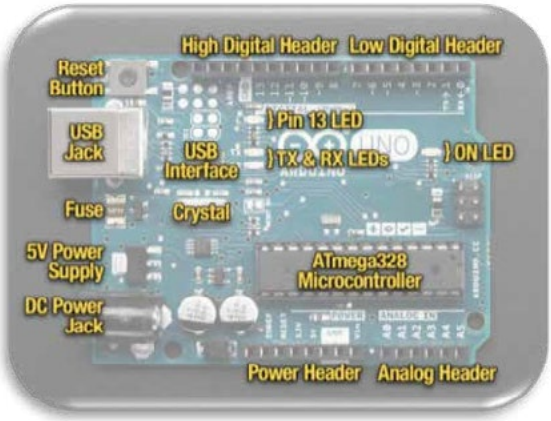
Мора се нагласити још једна основна разлика између PC-ја и микрорачунара, а то је да на PC-ју нема ограничења за инсталацију апликација. Кад нестане HDD простора, уради се *upgrade* и дода нови HDD, а код микрорачунара имамо само једну апликацију, а то је она за коју је микрорачунарски систем намијењен.

Arduino платформа

Слика 3. Golf MkII

Просјечан Босанац и Херцеговац без проблема би, вјероватно, могао навести све дијелове омиљеног VW аутомобила Golf MkII, познатијег на нашим просторима као „Golf 2".

Иако је „Golf 2" произведен у вишеструко мањој серији од Arduino платформе, вјероватно на прсте руке можемо да побројимо наше просјечне сугрђане који би исто могли урадити и за Arduino платформу, па идемо то мало промијенити.

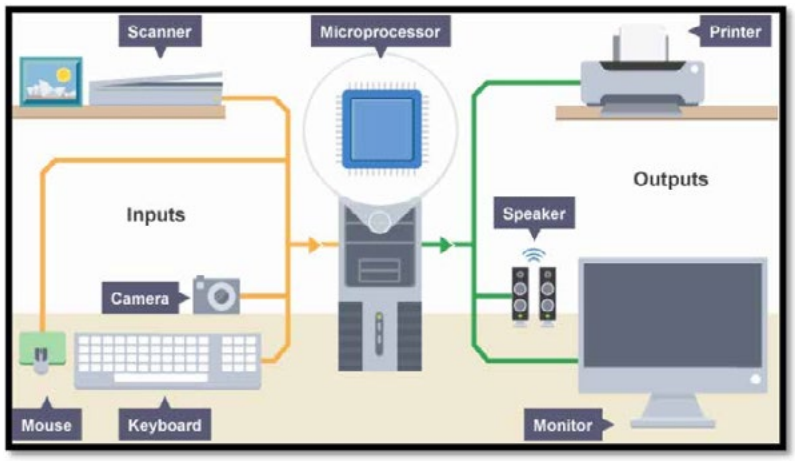


Слика 4. Arduino платформа

- *USB Jack* – служи за спајање Arduino микроконтролера с рачунаром, као интерфејс за програмирање и напајање приликом тестирања;
- *Reset Button* – ресетује микроконтролер, односно апликацију учитану у њега;
- *Fuse* – осигурач, морао вам је барем једном код куће искочити један, штити уређаје од струјних преоптерећења;
- *5V Power Supply* – напонски регулатор који лимитира напон са DC конектора на 5 V;
- *DC Power Jack* – екстерно напајање од 7 до 20 VDC;
- *Power Header* – сабирница за напајање сензора или актуатора;
- *Analog Header* – сабирница за аналогне сензоре;
- *ON Led* – сигнализација да је платформа под напоном;
- *Low & High Digital Header* – улази и излази за сензоре и актуаторе;
- Пин *13 LED* – свака микроконтролерска развојна платформа има бар једну LE диоду спојену на пин микроконтролера, да би се платформа могла тестирати;
- *USB Interface* – интерфејс (сучеље) који служи за комуникацију микроконтролера и PC-ја;
- *TX & RX LED* – диоде које нам омогућавају да видимо да постоји проток података – сигнала између PC-ја и микроконтролера и обратно;
- *Crystal* – електрични осцилатор који генерише сигнал тачне фреквенције – clock микрорачунара.

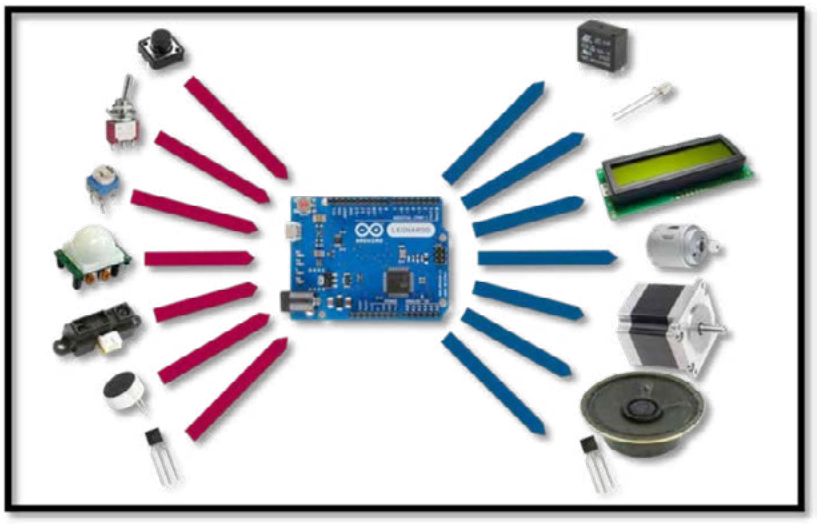
\

Сензори и актуатори



Слика 5. И/О (Улазни/излазни) уређаји код класичног PC-ја (рачунара)

Ми свакодневно имамо интеракцију са сензорима и актуаторима, можда је једноставније када кажемо улазним и излазним уређајима. Поново ћемо се вратити на PC. Као што је приказано на претходној слици, улазни уређаји су миш, скенер, тастатура и камера, а излазни уређаји су монитор, принтер и звучник.



Слика *6.* И/О уређаји код *Arduino* развојне платформе

Улазни уређаји у микроконтролерским системима су тастери, прекидачи, потенциометри, дигитални сензори или аналогни сензори. Излазни уређаји би били релеји, LED, LCD, мотори и звучници. Микроконтролерски систем прати промјене стања на својим улазима и спрам апликације ради промјене на својим излазима. Како будемо пролазили кроз практичне примјере, мало детаљније упознаћемо се са сваком од компоненти које ћемо користити.

Arduino IDE Software

Прије свега, прво морамо подесити свој PC на тај начин да одрадимо инсталацију **Arduino IDE Software-**а. Наравно, ријеч је о бесплатној апликацији коју користимо да бисмо креирали или размјењивали информације с Arduino микроконтролером. Апликација може да се преузме са сљедеће веб-локације:

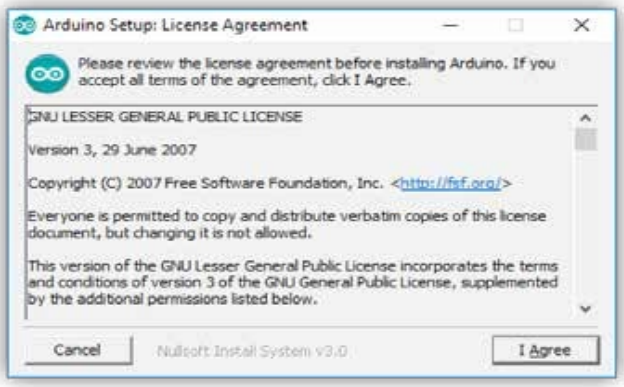
https://www.Arduino.cc/en/Main/Software

Слика 7. Arduino IDE одабир инсталационог фајла

Arduino software се стално ревидира. У тренутку писања овог приручника актуелна је верзија 1.8.9, али је врло могуће да, док овај приручник дође до крајњих корисника, новија верзија софтвера буде доступна за инсталацију.

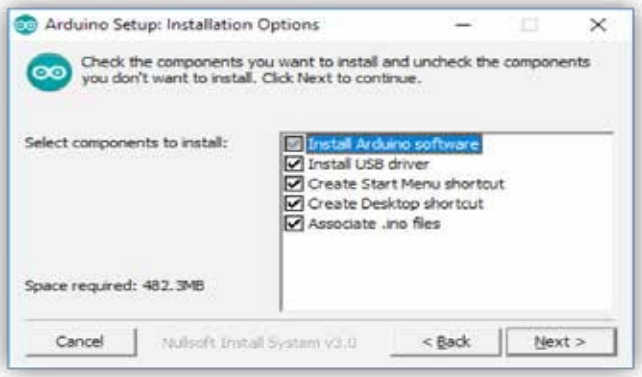
Проћи ћемо кроз кораке за подешавање и инсталацију Arduino IDE Software-а. Кликните на линк *Windows installer* за даунлоуд инсталационог фајла и потом кликните на новој страници *Just download*.

Морате да потврдите да се слажете с општим условима при употреби софтвера који користи тзв. *open source* лиценцу.

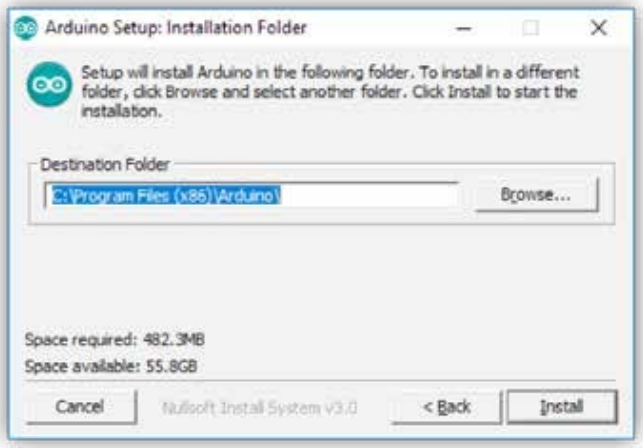


Слика 8. Arduino IDE потврда општег уговора за open source лиценцу

Одаберите инсталацију USB драјвера, те креирање икона и shortcut-а, а потом потврдите локацију за инсталирање апликације.

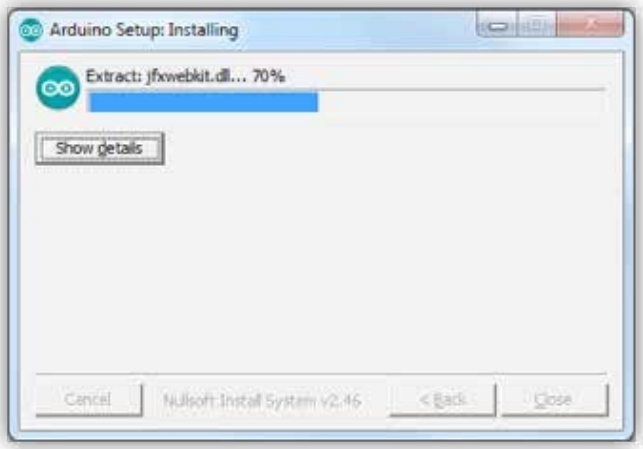


Слика 9. Arduino IDE одабири инсталације USB драјвера



Слика 10. Arduino IDE потврда локације инсталације

Процес инсталације у просјеку траје око минуте, што опет зависи од хардверске конфигурације рачунара на који се апликација инсталира.



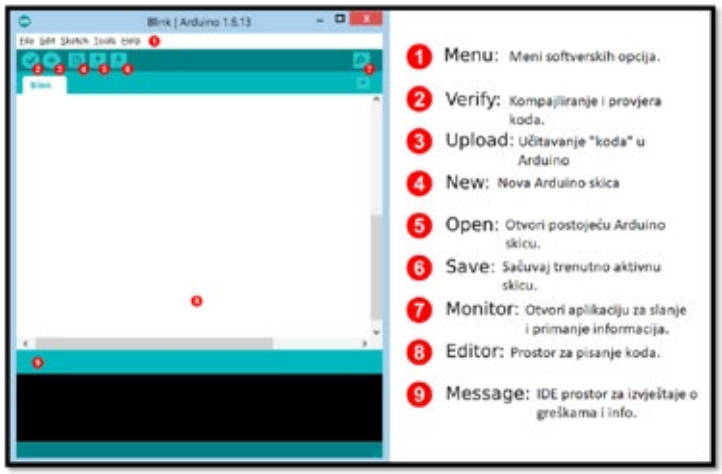
Слика 11. Arduino IDE процес инсталације

Коначно, још неколико ситних препрека и спремни смо за наше прве линије кода. Икона Arduino IDE је на десктопу. Дакле, све смо спремнији.

Слика 12. Arduino IDE икона

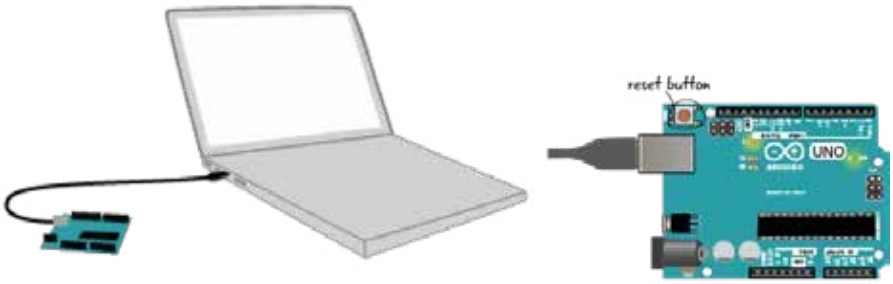
Arduino IDE окружење

Прије него што почнемо програмирање, појаснићемо Arduino IDE окружење.



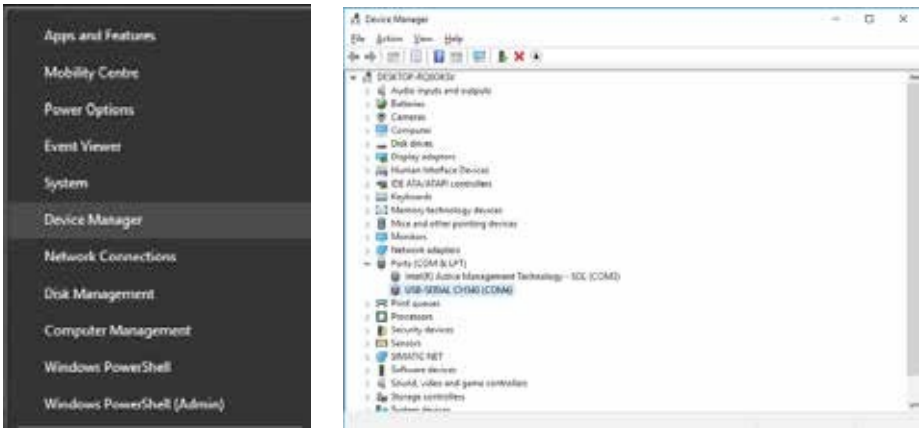
Слика 13. Arduino IDE опис

Прво је потребно, користећи USB A – USB Б кабл, спојити Arduino Uno на слободни USB порт PC-ја.



Слика 14. Спајање Arduin-а на PC

Добра рутина је у *Windows Device Menager-y* провјерити којем је COM (комуникационом интерфејсу) порту придружен Arduino. На Win10 оперативном систему потребно је десним кликом на Windows икони наћи *Device Menager* опцију и претражити комуникационе портове (Ports – COM & LPT).



Слика 15. Arduino на COM4

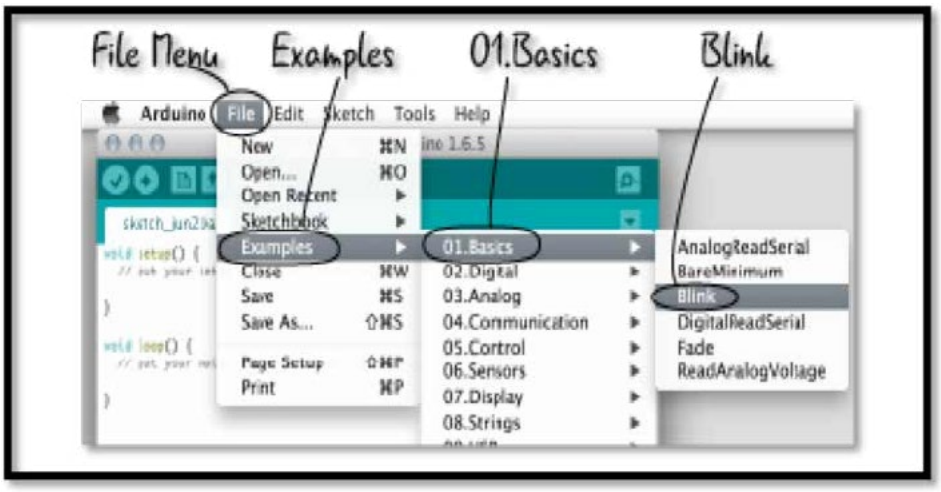
cout << „Hello, World!”;

„Hello, World!” или „Здраво, свијете!” је програм у C++ програмском језику. У свијету микрорачунарског програмирања то је апликација „Blink LE diode” на развојној плочи. Како ћемо у наредној вјежби видјети, на пин 13 Arduino платформе биће спојена LE диода, што ће значити да та апликација, уствари, мијења стање LE диоде на пину 13 из укљученог у искључено стање.

Мора се нагласити да је програмски језик који се користи у Arduino IDE поједностављени C програмски језик и сматра се да је такав начин уласка у свијет програмирања, гдје корисник види резултат свог кода у физичком свијету, идеалан за почетнике.

Наравно, основну синтаксу програмског језика учићемо кроз примјере. Не треба бити у заблуди да ће овај „hands-on” приручник бити довољан да корисник савлада све технике програмирања, али ће бити довољан да самостално ријеши задатке који се пред њега постављају.

Сљедећи корак би био отварање *example Blink* апликације, њено компајлирање и учитавање у Arduino, те детаљна анализа кода.



Слика 16. Отварање Arduino Blink скице

```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */


int ledPin = 13;                // LED спојена на digital пин.

void setup()                    // Једном се покреће када се скица стартује.
{
  pinMode(ledPin, OUTPUT);      // Проглашавање пина 13 ИЗЛАЗОМ.
}


void loop()                     // Стално се извршава.
{
  digitalWrite(ledPin, HIGH);   //Укључи LED.
  delay(1000);                  //Чекај секунду.
  digitalWrite(ledPin, LOW);    //Искључи LED.
  delay(1000);                  //Чекај секунду.
}
```

Па кренимо редом. Коментари су дијелови кода које добар програмер увијек оставља ради себе или других који ће читати код, те га на тај начин брже и лакше разумјети. Arduino коментаре третира баш као што је и речено, као коментаре, дакле, не узима их у обзир приликом извршавања кода.

Коментари се појављују у два облика:

```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED on for one second,
 * then off for one second, and so on...  We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

// Коментар
/*
Коментар
*/
```

Даље ћемо видјети једну класичну изјаву у C програмском језику. Послије изјаве слиједи коментар који нам је потпуно јасан, почиње великим словом, има смисао и завршава тачком. С друге стране, изјава на лијевој страни није ништа друго до програмска реченица која каже да је варијабла имена ledPin тип података ***int*** (интегер – „модерни назив за цијели број"), којој је додијељена вриједност 13.

```
int ledPin = 13;                    // LED спојена на дигитал пин 13.
```

Па овај поједностављени C програмски језик и није тако страшан. Само ради вјежбе, напишите двије изјаве које вас описују.

```
                                                    / моја висина је ___cm
                                                    / моја тежина је ___kg
```

Сада постаје интересантно. Сваки микроконтролерски уређај мора да има двије *default* специјалне процедуре **„setup()"** и **„loop()"**. Имена су им интуитивна и лако се да наслутити њихова улога.

```
void setup()                        // Једном се покреће кад се скица старта.
starts
{
  pinMode(ledPin, OUTPUT);          // Проглашавамо пин 13 ИЗЛАЗОМ.
}
```

Наиме, микроконтролер на Arduino развојној платформи има 13 пинова који су декларисани као *Digital inputs/outputs*. Исправно их је назвати GPIO (*general purpose input output* – или улази и излази опште намјене). У *setup* процедури ми контролеру дамо упуту да ћемо неке пинове користити као INPUTS, а неке као OUTPUTS. Тај дио кода извршава се само једном, када се платформа стави под напон.

У нашем конкретном случају, како је на пин 13 спојена LE диода (која спада у групу излазних елемената), логично је да позовемо функцију ***pinMode,*** која проглашава пин 13 излазним пином. Дакле, функција ***pinMode*** има два аргумента која корисник треба да подеси, а то су који пин желимо користити и у којем моду. И да поновимо, не смијемо заборавити да ставимо тачку на крају програмске изјаве.

Сада мало вјежбе. Погледајте *setup* процедуру у којој ће се пин 2 прогласити улазним пином, а пин 8 излазним.

```
                    pinMode(2, INPUT);          / Проглашавамо пин 2 УЛАЗНИМ пином.

                    pinMode(8, OUTPUT);         / Проглашавамо пин 8 ИЗЛАЗНИМ
                    пином.
```

```
void loop()                          // Стално се „врти".
{
  digitalWrite(ledPin, HIGH);        // Укључи LED на пину 13.
  delay(1000);                       // Задржи стање на 1
  digitalWrite(ledPin, LOW);         секунду.
  delay(1000);                       // Искључи LED на пину 13.
}
```

Процедура ***loop()*** се циклично извршава и, уствари, представља апликацију коју корисник у коначници види. У нашем примјеру унутар *loop* процедуре користимо нове двије функције ***digitalWrite()*** и ***delay()***. *DigitalWrite* функција има два аргумента пин ***pin*** и ***value***. Дакле, у нашем примјеру дајемо упуту контролеру да LE диоду на пину 13 укључи или искључи.

Функција *delay* зауставља извршавање програма за дефинисани број милисекунди.

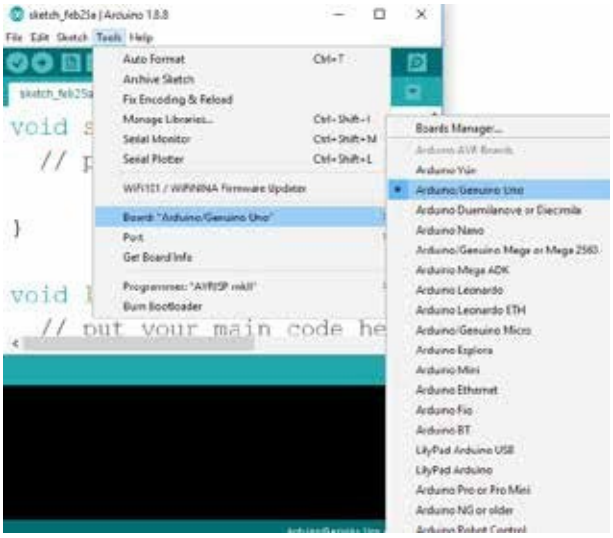
Да не заборавимо, процедуре су скупови програмских изјава. Да бисте боље разумјели процедуре, погледајте сљедећи примјер.

```
clean cat wash the cat(dirty cat)   // Процедуре за прање прљаве мачке.
{
      Нађите мачку.
      Ухватите је.
      Одврните чесму.
      Ставите мачку испод чесме.
      Добро оперите мачку.          // Док не буде чиста.
      Пустите мачку да настави безбрижан живот.
}
```

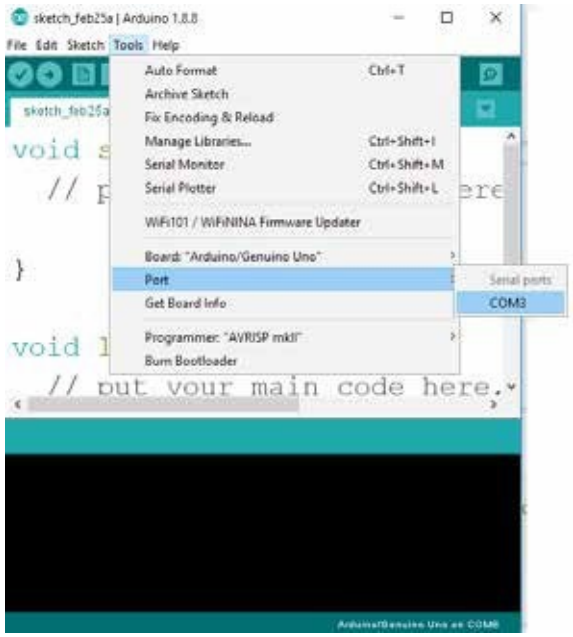
Име процедуре је *оперите мачку.* Процедура има један аргумент, прљаву мачку, која након успјешно извршених програмских изјава враћа на крају процедуре чисту мачку. Једноставно, зар не!?

Верификација и учитавање кода

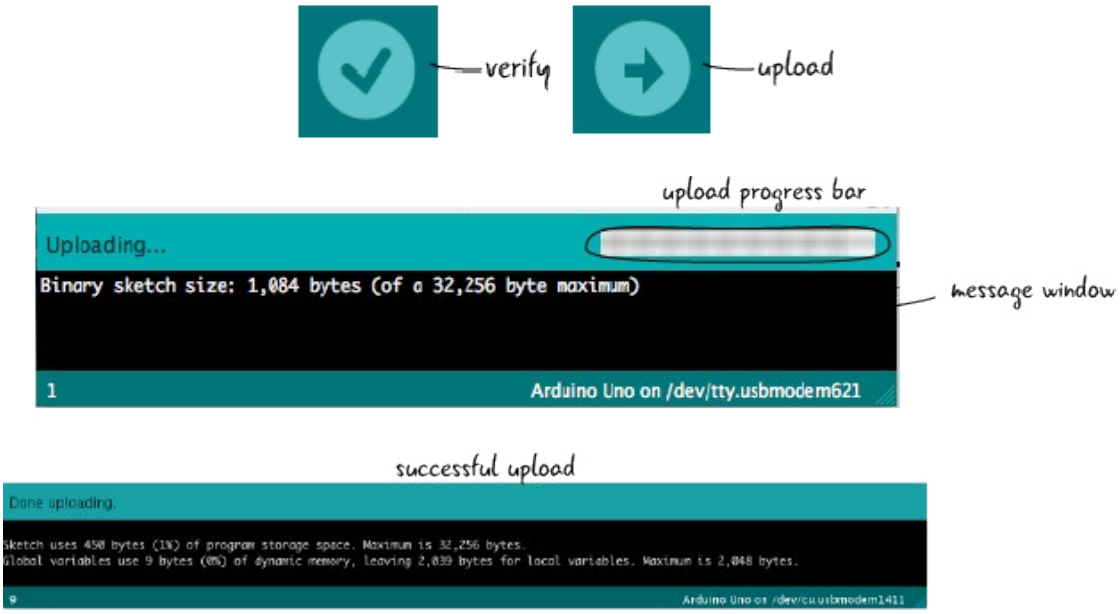
Прије саме процедуре учитавања наше апликације у микроконтролер потребно је у менију *Tools* одабрати одговарајућу развојну плочицу и припадајући COM порт.



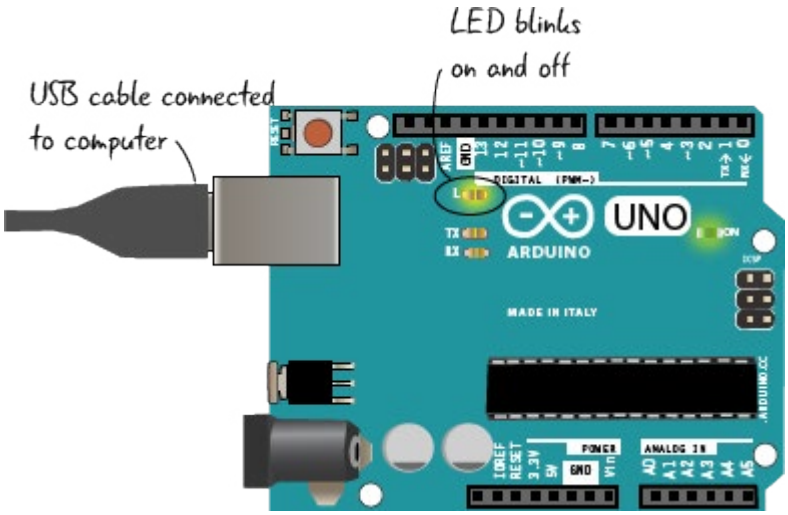
Слика 17. Одабир Arduino развојне плоче



Слика 18. Одабир припадајућег порта



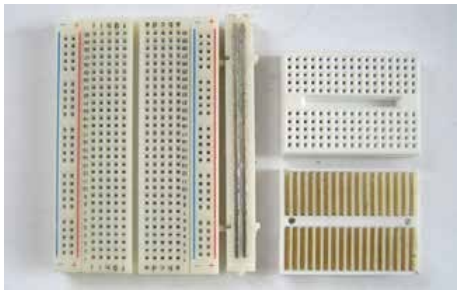
Слика 19. Процедура за верификацију и учитавање кода



Слика 20. LE диода на пину 13 укључује се и искључује

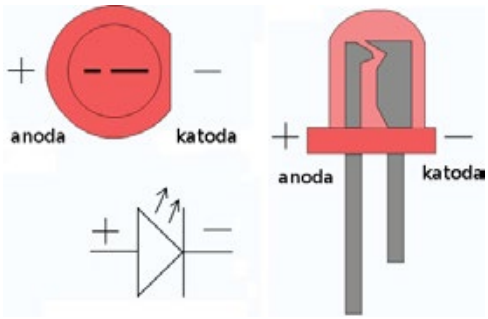
Основне електронске компоненте и помагала

У наредним корацима упознаћемо се с основним електронским компонентама и помагалима, те покушати да самостално креирамо мало комплексније апликације. За реализацију вјежбе потребна нам је матадор плоча. На наредној слици приказана је организација матадор плоче. Дакле, уздужне сабирнице означене са „+" и „-"користе се за дистрибуцију напона, а хоризонталне служе за спајање компоненти.



Слика 21. Матадор плоча, поглед одозго и одоздо

Наредни елемент који ћемо користити да бисмо реализовали вјежбу је LE диода или свијетлећа диода, а то је специјална врста диоде која струју усљед протицања претвара у свјетлост.



Слика 22. LE диода, изглед и симбол

Отпорник је неопходан да би се ограничила јачина струје кроз LED. Препоручена јачина струје у случају 5 mm црвене LED је око 20 mA. Када је излаз микроконтролера активан, он на свом излазу даје 5 V па бисмо по Омовом закону добили сљедећу препоручену вриједност отпорника:

$$II = \frac{\#}{\$} \rightarrow RR = \frac{\#}{\cdot} = \frac{((*)}{,-(-/)} = \frac{((*)}{-.-,(/)} = 250\ (\Omega)$$

Наравно, увијек је добра пракса обезбиједити електронску опрему па је одабрана вриједност отпора 330 Ω.

ИСХОДИ УЧЕЊА

(вјежбе 1 - 5)

ИСХОДИ УЧЕЊА	НИВО ПОСТИГНУЋА
– Ученик користи програмско окружење за Arduino – Ученик описује улогу основних дијелова на Arduin-у – Ученик разликује основне електронске компоненте (матадор плочу, отпорнике, LE диоде) које су коришћене у вјежбама – Ученик препознаје и описује улогу електронских компоненти – Ученик примјењује своје знање из електронике за повезивање Arduin-а и електронских компоненти – Ученик користи електричну шему споја за реализацију вјежбе – Ученик распознаје разлике између основних алгоритамских структура које су коришћене унутар вјежби – Ученик повезује Arduino с матадор плочом и осталим компонентама – Ученик верификује и извршава програм	**Минимална постигнућа** Ученик именује све потребне компоненте коришћене у вјежби Ученик уз помоћ наставника повезује дијелове с Arduin-ом, верификује и извршава програм **Довољна постигнућа** Ученик објашњава улогу основних електронских компоненти коришћених у вјежби Ученик повезује основне електронске компоненте према шеми споја Ученик посједује основна знања о писању програма за претходне вјежбе Ученик самостално верификује и извршава програм **Висока постигнућа** Ученик самостално повезује компоненте (Arduino, матадор плочу, отпорнике, LE диоде) коришћене у вјежбама према шеми споја Ученик распознаје разлике између основних алгоритамских структура, те примјењује своје знање за писање кода Ученик самостално верификује и извршава програм, те по потреби успјешно отклања настали буг

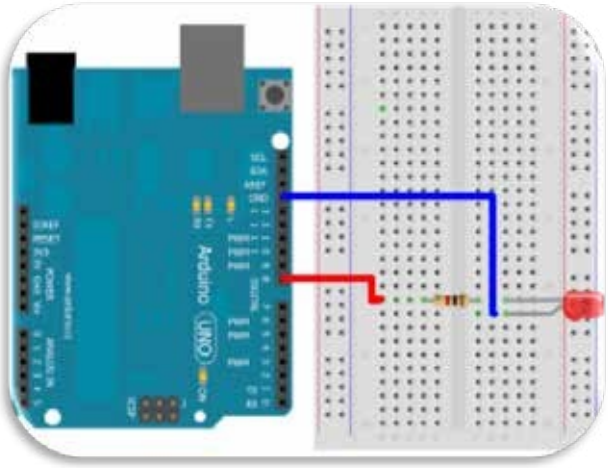
Вјежба 1.
НЕКА БУДЕ СВЈЕТЛО

Спојити LE диоду на дигитални излаз Arduino UNO плочице, те је палити и гасити у размаку од двије секунде.

Потребне компоненте:

- Arduino UNO плочица и USB 1 ком
- Матадор плочица 1 ком
- LE диоде 1 ком
- Отпорник 220 Ω 1 ком

Слика 23. Потребне компоненте



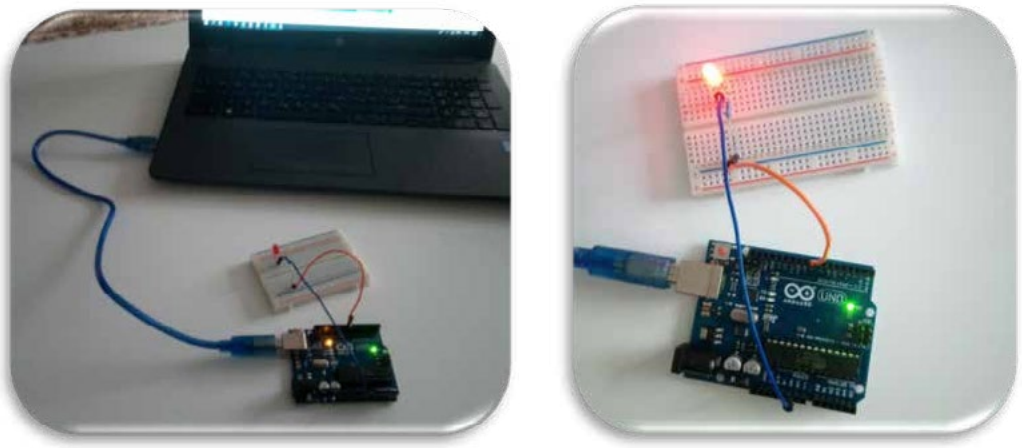
Слика 24. Шема споја

Кораци за реализацију вјежбе:

1. Повежите Arduino пин 8 с LE диодом на основу приложене шеме споја.
2. Креирајте *Sketch* којим ћете управљати радом LE диоде на пину 8:

Код:

```
int izlaz = 8;
void setup()
{
  // put your setup code here, to run once:
  pinMode(izlaz, OUTPUT);
}
void loop()
{
  // put your main code here, to run repeatedly:
  digitalWrite(izlaz, HIGH);
  delay(2000);
  digitalWrite(izlaz, LOW);
  delay(2000);
}
```



Слика 25. Верификација и тестирање

Примјер 1.

Модификујете код да је LED укључена на 100 ms, а искључена на 900 ms.

Примјер 2.

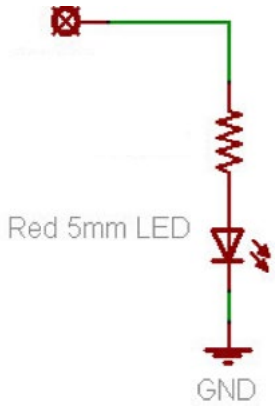
Модификујте код на тај начин да је LE диода укључена на 50 ms и искључена на 50 ms. Опишите како се понаша LE диода.

Добијемо на LE диоди тзв. STROBE ефекат, односно „титрање”!

Примјер 3.

Модификујте код на тај начин да је LE диода укључена и искључена у интервалима по 10 ms. Опишите појаву. Покушајте махати Arduino развојном платформом напријед–назад у замраченој просторији. Шта се дешава?

LE диоде остављају траг у зраку. На тај начин око је „преварено" и не може да види промјену стања на LE диоди. Махањем добијемо ефекат линије у зраку.

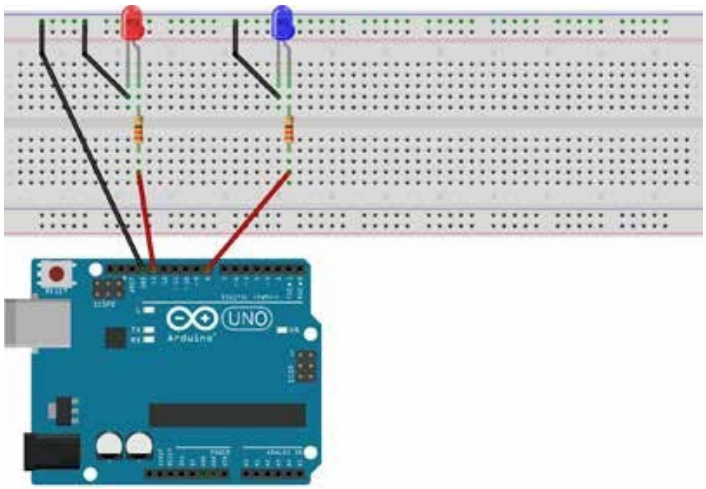


Слика 26. Електрична шема кола за спајање LE диода на пин 8

Вјежба 2.
ТОПЛО И ХЛАДНО!

Креирајте апликацију која ће наизмјенично палити и гасити LE диоду на пину 13 и пину 8. Потребне компоненте:

- Arduino UNO плочица и USB 1 ком
- Матадор плочица 1 ком
- LE диоде 2 ком
- Отпорник 220 Ω 2 ком



Слика 27. Шема споја

Код:

```
int led1 = 13;                // LED спојена на digital пин 13.
int led2 = 8;                 // LED спојена на digital пин 8.

void setup()
{
  pinMode(led1, OUTPUT);      // Проглашавање пина 13 ИЗЛАЗОМ.
  pinMode(led2, OUTPUT);      // Проглашавање пина 8 ИЗЛАЗОМ.

}

void loop()                   //Стално се извршава.
{
  digitalWrite(led1, HIGH);   //Укључи LED1.
  digitalWrite(led2, LOW);    //Искључи LED2.

  delay(1000);                //Чекај секунду.
  digitalWrite(led1, LOW);    //Искључи LED1.
  digitalWrite(led2, HIGH);   //Укључи LED2.
  delay(1000);                //Чекај секунду.
}
```

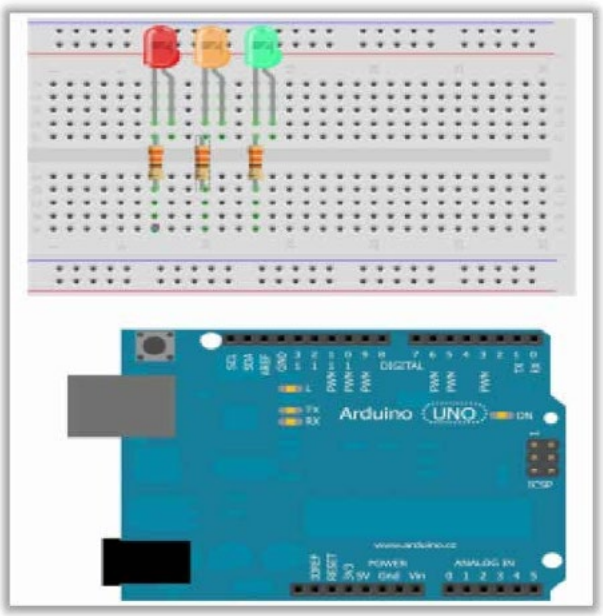
Вјежба 3.
СЕМАФОР

Креирати *Sketch* којим ћете симулирати рад семафора. Нека црвено свијетли 5 секунди, затим црвено и жуто 3 секунде, а на крају зелено 8 секунди и жуто 3 секунде.

Потребне компоненте за вјежбу и шема споја:

- Arduino UNO 1 ком
- LE диода 3 ком
- Отпорник 330 Ω 3 ком

Слика 28. Примјер изгледа семафора



Слика 29. Шема споја

Код:

```
void setup()
 {
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(13,OUTPUT);
}
void loop()
{
 digitalWrite(13,1);        //Црвено.
 digitalWrite(12,0);
 digitalWrite(11,0);
 delay(3000);                     //3 секунде.

 digitalWrite(12,1);        //Црвено и жуто.
 delay(3000);

 digitalWrite(13,0);
 digitalWrite(12,0);
 digitalWrite(11,1);        //Зелено.
 delay(8000);

 digitalWrite(12,1);              //Жуто.
 digitalWrite(11,0);
 delay(3000);
 }
```

Вјежба 4.
ДИСКО СВЈЕТЛА

Диско свјетла – 5 лампица и 4 ефекта

Креирати *Sketch* за контролу рада LE диоде. Овај задатак може да се ријеши на више начина. Први је примијенити логику коју смо до сада примјењивали и слиједно редати програмске исказе спрам жељене логике.

```
        digitalWrite(led1, HIGH);
          delay(500);
        digitalWrite(led2, HIGH);
          delay(500);
        digitalWrite(led3, HIGH);
          delay(500);
        digitalWrite(led4, HIGH);
          delay(500);
        digitalWrite(led5, HIGH);
          delay(500);
```

На тај начин добићемо ефекат да се LED пале једна иза друге са задршком од пола секунде. Међутим, како будемо развијали нове ефекте, то ће се број линија кода драстично повећавати.

Да бисмо то избјегли, можемо да урадимо сљедеће. LED спојимо у низ на сљедећи начин:

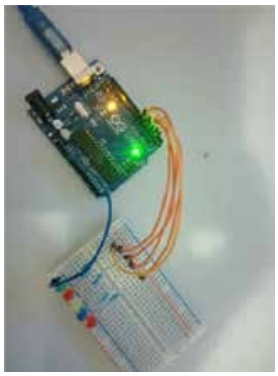
LED 1 – пин 2

LED 2 – пин 3

LED – пин 4

LED – пин 5

LED – пин 6



Слика 30. LED спојене у низ

Дакле, како имамо низ, можемо да искористимо и ***for*** петљу која се употребљава за понављање програмских изјава унутар витичасте заграде, при чему се користе инкрементални или декрементални бројачи за пролазак од почетног до крајњег услова за понављање ***for*** петље.

Заглавље *for* петље има три аргумента.

```
for (inicijalizacija; uslov; inkrement)
{
Programske_izjave(inkrement);
}
```

Код из посљедњег примјера с *for* петљом изгледао би овако:

```
for(int i=2;i<=6;i++)
{
    digitalWrite(i, HIGH);
    delay(500);
}
```

Преведено, за вриједност броја i = 2 до броја i = 6 с инкрементом од 1 изврши програмску изјаву са задршком од пола секунде:

за и = 2 извршиће се `digitalWrite (2, HIGH) ;`
за и = 3 извршиће се `digitalWrite (3, HIGH) ;`
за и = 4 извршиће се `digitalWrite (4, HIGH) ;`
за и = 5 извршиће се `digitalWrite (5, HIGH) ;`
за и = 6 извршиће се `digitalWrite (6, HIGH) ;`

при чему је наредба за инкремент

i++

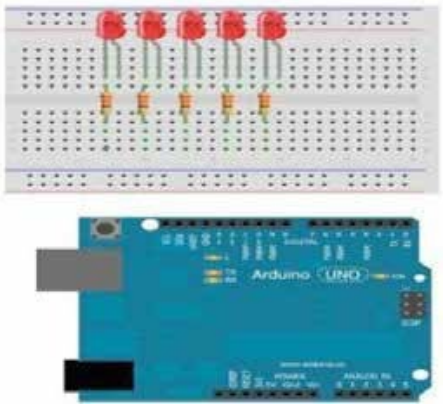
само краће паисан математички израз

i = i + 1

Пинове 0 и 1 треба да избјегавамо користити јер те пинове означене као RX и TX користи и интерфејс за програмирање.

Потребне компоненте за вјежбу:

1. Arduino UNO 1 ком
2. LE диода 5 ком
3. Отпорник 330 Ω 5 ком



Слика 31. Шема споја

Кораци за реализацију вјежбе:

1. Креирајте шему споја користећи претходно стечена знања.
2. Креирајте *Sketch* којим ћете управљати радом LE диоде креирајући сљедеће ефекте:
   - диоде се укључе са задршком од пола секунде, на матадор плочицу у портове од 2 до 6,
   - диоде се гасе са задршком од пола секунде, на матадор плочицу у портове од 2 до 6,

- диода се укључи, задржи стање пола секунде, угаси се па се процес понови и за остале диоде,
- све диоде се укључе и задрже стање пола секунде, искључе се, те се процес понови пет пута.

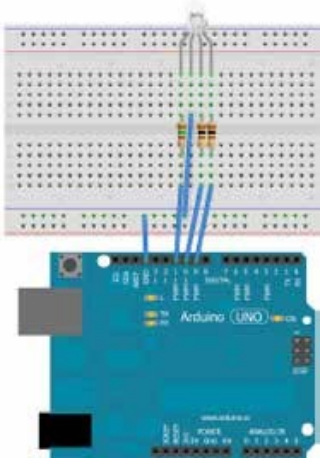
```
void setup()
 {
for(int i=2;i<=6;i++)
 {
      pinMode(i,OUTPUT); //Пинови од 2 до 6
      излазни.
 }
 }
 void loop()
 for(int i=2;i<=6;i++)      //Ефекат 1.
 {
   digitalWrite(i, HIGH);
    delay(500);
 }
 for(int i=2;i<=6;i++)      // Ефекат
 {                          2
    digitalWrite(i, LOW);
    delay(500);
 }
for(int i=2;i<=6;i++) // Ефекат 3.
 {
    digitalWrite(i, HIGH);
    delay(500);
    digitalWrite(i, LOW);
 }
for(int i =0;i<=5;i++)      // Ефекат
4.
 {
     for(int j=2;j<=6;j++)
         {
           digitalWrite(j,HIGH);
        }
    delay(500);
    for(int j=2;j<=6;j++)
         {
            digitalWrite(j,LOW);
         }
 }
```

Вјежба 5.
RGB LED

Креирати *Sketch* за управљање радом RGB LED. .

Потребне компоненте за вјежбу:

1. Arduino UNO 1 ком
2. RGB LED 1 ком
3. Отпорник 150 Ω 1 ком
4. Отпорник 100 Ω 2 ком



Слика 32. Шема споја

Кораци за реализацију вјежбе:

1. Креирајте тестни систем користећи шему споја.
2. Креирајте *Sketch* којим ћете управљати радом LE диоде на тај начин да се у једнаким временским интервалима мијењају комбинације боја.
3. Од папира направите кућиште те унутра убаците згужвану марамицу и RGB LED.

Слика 33. Израда кућишта за RGB LED

ИСХОДИ УЧЕЊА

(вјежбе 6 - 8)

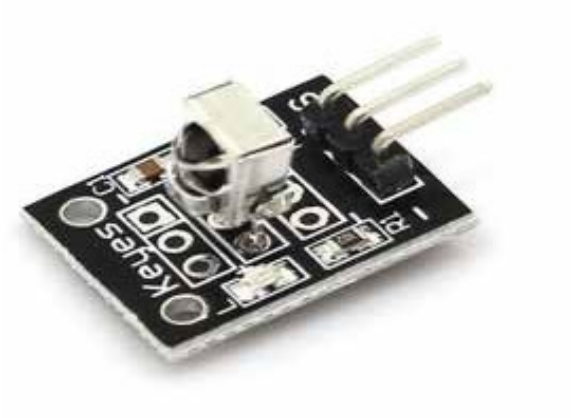
ИСХОДИ УЧЕЊА	НИВО ПОСТИГНУЋА
- Ученик користи програмско окружење за Arduino - Ученик разликује електронске компоненте које су коришћене у вјежбама - Ученик описује улогу електронских компоненти - Ученик дефинише својства сигналних уређаја (сензора) примијењених у вјежбама - Ученик дефинише начин рада седам-сегментног дисплеја - Ученик примјењује своје знање из електронике за повезивање Arduin-а и електронских компоненти - Ученик користи електричну шему споја за реализацију вјежбе - Ученик примјењује знање из програмирања за реализацију одређеног задатка - Ученик користи одговарајуће библиотеке унутар програма за реализацију вјежбе - Ученик користи шему за повезивање Arduino уређаја с матадор плочицом и осталим компонентама (сензори, седам-сегментни дисплеј, тастер) - Ученик верификује и извршава програм	**Минимална постигнућа** Ученик именује све потребне компоненте коришћене у вјежби Ученик уз помоћ наставника повезује дијелове с Arduin-ом, верификује и извршава програм **Довољна постигнућа** Ученик објашњава улогу електронских компоненти коришћених у вјежби Ученик повезује електронске компоненте према шеми споја Ученик посједује основна знања о писању програма за претходне вјежбе Ученик користи одговарајуће библиотеке унутар програма Ученик самостално верификује и извршава програм **Висока постигнућа** Ученик самостално повезује компоненте (Arduino, матадор плочу, отпорнике, LEдиоде, сензоре, седам-сегментни дисплеј, тастере) коришћене у вјежбама према шеми споја Ученик примјењује своје знање из програмирања за писање кода Ученик самостално верификује и извршава програм, те по потреби успјешно отклања настали буг

Вјежба 6.
ИНФРАЦРВЕНИ СЕНЗОР И УПРАВЉАЧ

Дешифровати сигнале које прима IR сензор притиском на дугмад управљача те програмирати Arduino UNO тако да се притиском на тастере 1, 2 и 3 пале одговарајуће диоде (зелена, црвена и жута).

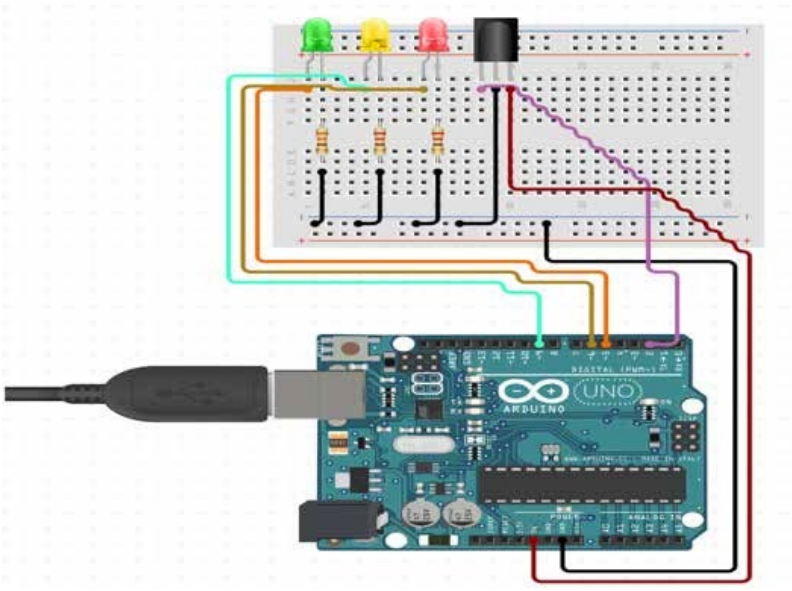
Потребне компоненте:

- Arduino UNO и USB
- Матадор плочица
- Црвена, зелена и жута LED с одговарајућим отпорницима
- IR receiver
- Даљински управљач



Слика 34. IR receiver

* Обратити пажњу на пинове IR сензора! Y - сигнал, G - ground, R - 5 V



Слика 35. Шема споја

За овај задатак потребно је користити библиотеку *IRreceiver.h* која у себи садржи основне функције за коришћење IR сензора.

Најбитније функције су:

креирање инстанце IR сензора
```
IRrecv irrecv (broj pina na koji je priključen Y pin senzora);
```

декларисање варијабле у којој ћемо спремати примљени сигнал од даљинског управљача
```
decode_results rezultat;
```

покретање IR сензора, тј. покретање декодирања улазног сигнала
```
irrecv.enableIRIn();
```

провјера је ли примљен неки сигнал
```
irrecv.decode (&резултат)
```

читање сљедећег сигнала
```
irrecv.resume();
```

Код:

```
#include "IRremote.h"
int receiverPIN = 3;

IRrecv irrecv(receiverPIN);
decode_results rezultat;

void setup()
{
  Serial.begin(9600);
  Serial.println("Primljeni signal: ");
  irrecv.enableIRIn();
}
void loop()
{
  if (irrecv.decode(&rezultat))   {
      Serial.println(rezultat.value);
      delay(500);
      irrecv.resume();
  }
}
```

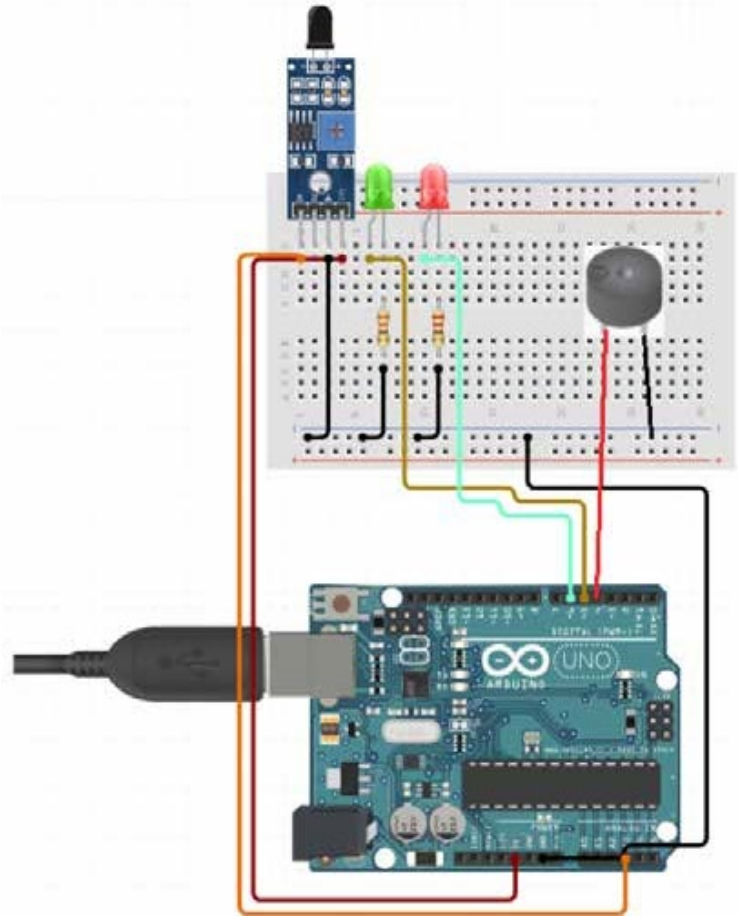
Вјежба 7.
СЕНЗОР ЗА ДЕТЕКЦИЈУ ПЛАМЕНА

Програмирати Arduino UNO тако да се при детекцији пламена оглашава buzzer и пали црвена LED. Уколико пламен није детектован, упалити зелену LED, а у случају да је детектован пламен, али се не налази у близини, упалити само црвену LED без buzzer-а.

* За овај пројекат биће битна функција `map(value, fromLow, fromHigh, toLow, toHigh)`

Потребне компоненте:

- Arduino UNO плочица и USB
- Матадор плочица
- Сензор за детекцију пламена
- 1 x зелена LED
- 1 x црвена LED
- 2 x отпорник 220 Ω



Слика 36. Шема споја

Код:

```
int buzzer = 4;
int zelenaLED = 5;
int crvenaLED = 6;

const int senzorMin = 0;
const int senzorMax = 1023;

void setup() {
   pinMode(buzzer, OUTPUT);
   pinMode(zelenaLED, OUTPUT);
   pinMode(crvenaLED, OUTPUT);
}
void loop() {
    int ocitanjeSenzora = analogRead(A0);

    int raspon = map(ocitanjeSenzora, senzorMin, senzorMax, 0, 2);

  switch (range) {
  case 0:
    digitalWrite(crvenaLED, HIGH);
    digitalWrite(zelenaLED, LOW);
    digitalWrite(buzzer, HIGH);
    break;
  case 1:
    digitalWrite(crvenaLED, HIGH);
    digitalWrite(zelenaLED, LOW);
    digitalWrite(buzzer, LOW);
    break;
  case 2:
    digitalWrite(crvenaLED, LOW);
    digitalWrite(zelenaLED, HIGH);
    digitalWrite(buzzer, LOW);
    Break;
  }
  delay(1);  // delay between reads
}
```

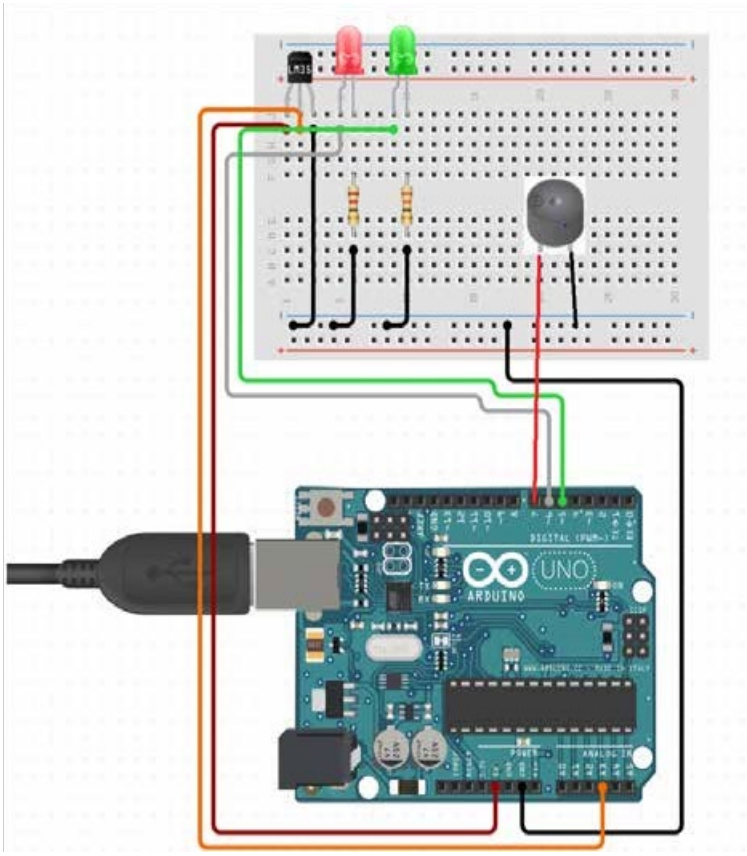
Вјежба 8.
МЈЕРЕЊЕ ТЕМПЕРАТУРЕ

Помоћу сензора TMP36 мјерити температуру те упалити зелену LED уколико је температура изнад 23° C. Када је температура нижа од 23° C, упалити црвену LED и buzzer.

Потребне компоненте:

- Arduino UNO плочица и USB
- Матадор плочица
- Црвена LED
- Зелена LED
- 2 x отпорник 220 Ω
- Сензор TMP 36
- Buzzer

Спојити компоненте као на слици:



Слика 37. Шема споја

Код:

```
int zelenaLED = 5;
int crvenaLED = 6;
int senzorPin = A3;
int buzzer = 7;

const float sobnaTemperatura = 28.0;

void setup() {
  Serial.begin(9600);
  // put your setup code here, to run once:
  pinMode(zelenaLED, OUTPUT);
  pinMode(crvenaLED, OUTPUT);
  pinMode(buzzer,OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  int vrijednostSenzora = analogRead(senzorPin);
  float napon = (vrijednostSenzora / 1024.0) * 5.0;
  float temperatura = (napon - .5) * 100;
  Serial.print("stepeni C: ");
  Serial.println(temperatura);

  if (temperatura < sobnaTemperatura - 2) {
    digitalWrite(zelenaLED, LOW);
    digitalWrite(crvenaLED, HIGH);
    digitalWrite(buzzer,HIGH);
  }
  else if (temperatura > sobnaTemperatura - 2 && temperatura < sob-
naTemperatura + 2) {
    digitalWrite(zelenaLED, HIGH);
    digitalWrite(crvenaLED, LOW);
    digitalWrite(buzzer,LOW);
  }
  else if (temperatura > sobnaTemperatura + 2) {
    digitalWrite(zelenaLED, LOW);
    digitalWrite(crvenaLED, HIGH);
    digitalWrite(buzzer,HIGH);
  }
}
```

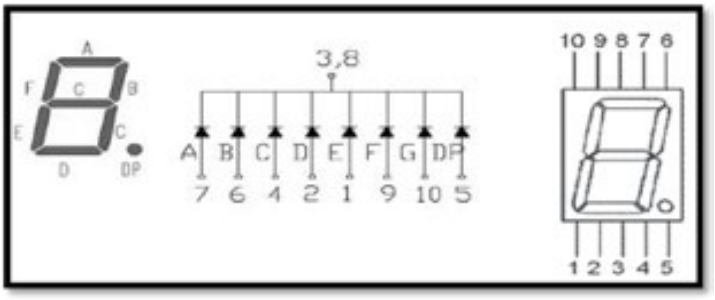
ИСХОДИ УЧЕЊА

ИСХОДИ УЧЕЊА	НИВО ПОСТИГНУЋА
- Ученик разликује електронске компоненте које су коришћене у вјежбама - Ученик описује улогу електронских компоненти коришћених у вјежбама - Ученик дефинише својства сензора за температуру примијењених у вјежбама - Ученик дефинише начин рада потенциометра - Ученик користи електричну шему споја за повезивање Arduin-а и електронских компоненти - Ученик примјењује знање о употреби функција из програмирања за реализацију задатака - Ученик примјењује одговарајућу врсту података (инт, флоат, доубле...) - Ученик користи одговарајуће библиотеке унутар програма за реализацију вјежбе - Ученик користи шему за повезивање Arduino уређаја с матадор плочицом и осталим компонентама (сензори, седам-сегментни дисплеј, тастер) - Ученик користи апликацију *Serial Monitor* приликом извршавања вјежбе - Ученик примјењује процедуре из библиотеке *Serial* за израду задатка - Ученик верификује и извршава програм	**Минимална постигнућа** Ученик именује све потребне компоненте коришћене у вјежби Ученик уз помоћ наставника повезује дијелове с Arduin-ом, верификује и извршава програм **Довољна постигнућа** Ученик објашњава улогу електронских компоненти коришћених у вјежби Ученик повезује електронске компоненте према шеми споја Ученик посједује основна знања о писању програма за претходне вјежбе Ученик користи одговарајуће библиотеке унутар програма Ученик самостално верификује и извршава програм те по потреби уз помоћ наставника отклања настали буг **Висока постигнућа** Ученик самостално повезује компоненте коришћене у вјежбама према шеми споја Ученик примјењује своје знање из програмирања за писање кода Ученик уочава примјену Arduino уређаја у другим областима (физика, математика, хемија...) Ученик самостално верификује и извршава програм, те по потреби успјешно отклања настали буг

Седам-сегментни дисплеј

Седам-сегментни дисплеј представља излазни уређај на којем је најлакше људима приказати информације. Може да прикаже све бројеве, али и нека слова. Долази у двије конфигурације и то са заједничком катодом или са заједничком анодом. На сљедећој слици приказана је конфигурација са заједничком катодом.

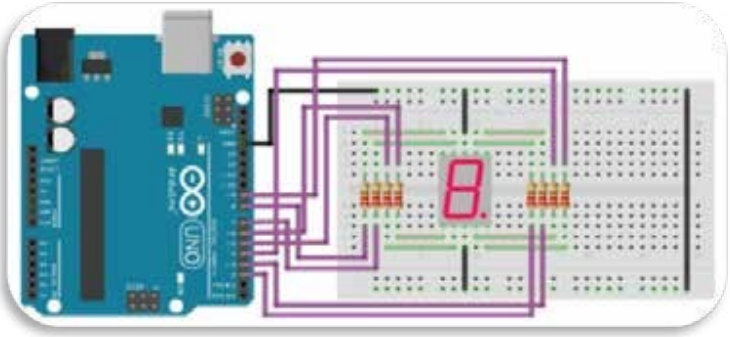
Дакле, аноде LE диода се спајају на пинове микроконтролера преко предотпора, а заједнички пин на *masu-gnd* на Arduino развојној платформи. HIGH сигнал на пину микроконтролера активира један сегмент на дисплеју. На тај начин комбинацијом активних пинова добијемо приказ бројева на седам-сегментном дисплеју.



Слика 38. Pinout за седам-сегментни дисплеј

Потребне компоненте за вјежбу:

1. Arduino UNO 1 ком
2. Седам-сегментни дисплеј 1 ком
3. Отпорник 330 Ω 7 ком



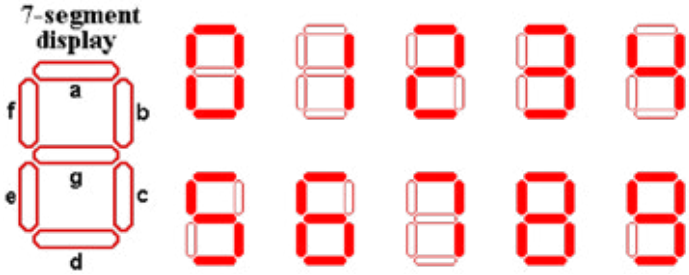
Слика 39. Шема споја

Кораци за реализацију вјежбе:

1. Креирајте шему споја користећи *Fritzing* skicu.
2. Креирајте *Sketch* којим ћете управљати радом седам-сегментног дисплеја.

Више је начина да се тај проблем ријеши. Један од њих је да поновно без употребе других програмских структура ријешимо проблем. Али прво требамо дефинисати pinout који ће нам олакшати само програмирање и рјешавање проблема:

PIN2 -> a
PIN3 -> b
PIN4 -> c
PIN5 -> d
PIN6 -> e
PIN7 -> f
PIN8 -> g
PIN9 -> dp



Слика 40. Приказ бројева на сегментном дисплеју

Према претходној слици, код за приказ броја „1" на нашем дисплеју био би сљедећи:

```
digitalWrite(a, LOW);
digitalWrite(b, HIGH);
digitalWrite(c, HIGH);
digitalWrite(d, LOW);
digitalWrite(e, LOW);
digitalWrite(f, LOW);
digitalWrite(g, LOW);
```

Апликација за приказ броја „1” на седам-сегментном дисплеју била би:

```
int a=2;
int b=3;
int c=4;
int d=5;
int e=6;
int f=7;
int g=8;

void setup ()
{
   for(int i=2;i<=9;i++)
    {
              pinMode(i,OUTPUT);//Пинови од 2 до 9
                                OUTPUT.
    }
}

void loop()
{
digitalWrite(a, LOW);
digitalWrite(b, HIGH);
digitalWrite(c, HIGH);
digitalWrite(d, LOW);
digitalWrite(e, LOW);
digitalWrite(f, LOW);
digitalWrite(g, LOW);
}
```

Наравно, једноставнији начин био би да се за сваки број креира *custom* функција за приказ броја „1” те да се она позове у *loop* петљу. То је могуће урадити на сљедећи начин:

```
vovoid jedan() //Креирање функције за приказ броја један.
{
    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
}
```

Односно, наш комплетан код за приказ броја „1” изгледао би на сљедећи начин:

```
int a=2;
int b=3;
int c=4;
int d=5;
int e=6;
int f=7;
int g=8;

void jedan()//Креирање функције за приказ броја „1”.
{
digitalWrite(a, LOW);
digitalWrite(b, HIGH);
digitalWrite(c, HIGH);
digitalWrite(d, LOW);
digitalWrite(e, LOW);
digitalWrite(f, LOW);
digitalWrite(g, LOW);

}
void setup ()
{
   for(int i=2;i<=9;i++)
    {
              pinMode(i,OUTPUT);//Пинови од 2 до 9
                                OUTPUT.
    }
}

void loop()
{
    jedan();  //Позивање функције за приказ броја „1”.
}
```

Ако бисмо жељели позвати функцију за приказ броја „2”, прије позивања функције потребно је „угасити” дисплеј; функција за гашење дисплеја изгледала би овако:

```
void gasi() //Креирање функције за „искључење” дисплеја.
{
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
```

```
void dva() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}
void tri() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}

void cetiri() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}

void pet() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}
```

```
void sest() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}
void sedam() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}

void osam() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}

void devet() //
{
      digitalWrite(a, _____);
      digitalWrite(b, _____);
      digitalWrite(c, _____);
      digitalWrite(d, _____);
      digitalWrite(e, _____);
      digitalWrite(f, _____);
      digitalWrite(g, _____);
}
```

Потребно је горње изразе допунити и креирати апликацију која ће сваке секунде приказати други број на дисплеју. Дакле, правимо бројач од 0 до 9.

Ово је, можда, најбољи примјер да проширимо своје знање из програмирања, јер се овај задатак може да ријеши на више различитих начина. Један од начина је употреба ***if*** теста.

```
if (uslov ispunjen)

{

//uradi nešto

}
```

If тест се користи с компарацијским операторима.

x == y (x jednako y)
x != y (x nije jednako y)
x < y (x manje od y)
x > y (x veće od y)
x <= y (x manje ili jednako od y)
x >= y (x veće ili jednako y)

Један типични *if* тест би гласио

if(ocjenaUcenika < 2) upaliAlarm();

Како нам if тест може да помогне да свој бројач ријешимо на љепши начин? Као прво, потребна нам је декларација једне глобалне варијабле бројач. Глобалне варијабле се декларишу изван тијела функције и видљиве су свим функцијама у програму. Локалне варијабле се декларишу унутар неке функције и само та функција види ту варијаблу.

```
int brojac=0;   //Глобално декларисана варијабла, сви је „виде”.

void setup ()

{

     int i=0;    //Локално декларисана варијабла, само setup је „види”.

 }
```

Покушајмо искористити новостечена знања на практичном примјеру.

```
      int a=2;
      int b=3;
      int c=4;
      int d=5;
      int e=6;
      int f=7;
      int g=8;
      int brojac = 0;
      void jedan()// Креирање функције за приказ броја „1”.
      {
      digitalWrite(a, LOW);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);

      }
      void setup ()
      {
         for(int i=2;i<=9;i++)
          {
               pinMode(i,OUTPUT);// Пинови од 2 do 9 OUTPUT.
          }
      }

      void loop()
      {
      if (brojac==0)nula();//Ако је стање бројача 0, позови функцију за
приказ
                    //броја 0, уколико се послије if теста имамо само један

                     //програмски исказ, нису потребне  витичасте заграде.

          brojac++;     //За сваки пролаз кроз loop уради инкремент.
бројача
          delay(1000);  // Сачекај секунду.

          if (brojac ==9) // Када избројимо до 9,
          {
                 brojac=0;    // вратимо стање бројача на 0 да поново
бројимо.
          }
      }
```

Додајте линије кода да стање на седам-сегментном дисплеју одговара стању глобалног бројача. Компајлирајте и тестирајте. Креирајте додатну опцију да бројач броји уназад.

```
      int a=2;
      int b=3;
      int c=4;
      int d=5;
      int e=6;
      int f=7;
      int g=8;

      int brojac = 0;

      void jedan()//Креирање функције за приказ броја „1”.
      {
      digitalWrite(a, LOW);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);

      }
      void setup ()
      {
         for(int i=2;i<=9;i++)
          {
                pinMode(i,OUTPUT);//Пинови од 2 до 9 OUTPUT.
          }
      }

      void loop()
      {
      if (brojac==0)nula();//Ако је стање бројача 0, позови функцију за
приказ
                           //броја 0, уколико послије if теста имамо само
један
                           //програмски исказ,нису потребне витичасте
заграде

          brojac++;     //За сваки пролаз кроз loop уради инкремент.
бројача
          delay(1000);  //Сачекај секунду.

          if (brojac ==9) //Када избројимо до 9,
          {
                 brojac=0;    //вратимо стање бројача на 0 да поново
бројимо.
          }
      }
```

Други начин да се овај задатак ријеши јесте коришћење *switch ... case structure.*

```
int brojac=0;
void loop ()
{
switch (brojac)
{
    case 1:
      jedan();      // Када је стање бројача 1, уђи у case 1.
      break;
    case 2:
      dva();        // Када је стање бројача 2, уђи у case 2
      break;
    default:
      nula();     // Када је стање бројача 0, уђи у default.

    break;
  }

}
```

Покушајте ријешити апликацију бројања и на овај начин!

Дигитални улази

Већ смо на почетку писали о улазима у микроконтролер. Када говоримо о дигиталним улазима, мислимо на улазе који могу да имају само два стања, стање логичке 1 и логичке 0, односно, ако је ријеч, рецимо, о прекидачу, онда кажемо да можемо имати два стања.

prekidač zatvoren -> logička „1“
otvoren -> logička „0“.

Прије свега морамо нашем микроконтролеру рећи да више нећемо имати интеракцију само с излазима, већ да ће на један пин бити спојен један input, тако да сада за тај input на који је спојен тастер, прекидач или неки сензор *pinMode* функција гласи:

```
int inPin = 2;
int outPin = 13;
void setup()
{
 pinMode(inPin, INPUT);           // Пин 2 је input
 pinMode(outPin, OUTPUT);         // Пин 13 је output
}
```

Провјеру стања на input пину 2 радимо помоћу функције

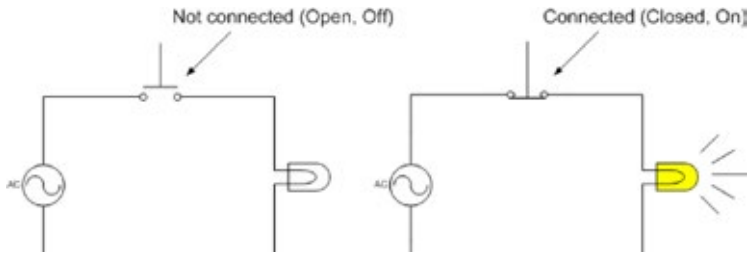
digitalRead(pin)

Функција има само један аргумент „пин” с којег читамо стање.

```
    /*
    Праћење стања на дигиталном улазу те промјена стања на LED на пину 13
    спрам стања на дигиталном улазу-> ако је корисник притиснуо тастер,
упали LED; ако тастер није притиснут, угаси LED.
    */
    int inPin = 2;
    int outPin = 13;
    int inPinstate = 0; //Варијабла у коју ћемо спремити стање на пину
    2.
    void setup()
    {
     pinMode(inPin, INPUT);           //Пин 2 је input.
     pinMode(outPin, OUTPUT);         //Пин 13 је
     output.
    }


    void loop()
    {
       inPinstate = digitalREad(inPin);//Спреми стање на пину 2 у
       варијаблу.

       if(inPinstate == 1)
             {
             digitalWrite(outPin, HIGH);
                     }
       else
                     {
             digitalWrite(outPin, LOW);
                     }

    }
```

Тиме смо обрадили софтверску страну проблема, идемо сада да видимо како ћемо ријешити хардвер за дигитални улаз.

Ми стално имамо интеракцију с прекидачима и тастерима: код куће, у стубишту зграде, лифту, разним кућанским апаратима... Кратко ћемо описати понашање класичног прекидача за сијалицу. Тај прекидач је једноставан уређај који има двије позиције и то укључено и искључено. То је сликовито приказано на сљедећој слици.

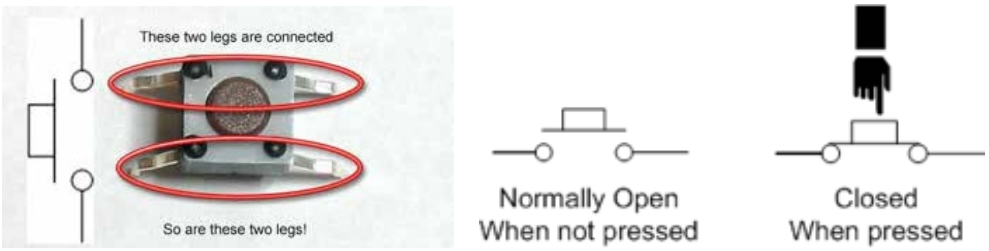


Слика 41. Тастер активан и тастер неактиван

Наравно, прекидачи које користимо у својим домовима су веома поуздани, али једноставно су габаритима велики за примјену у микроконтролерским системима. Ми ћемо за своје експерименте користити микротастер, 6 mm.

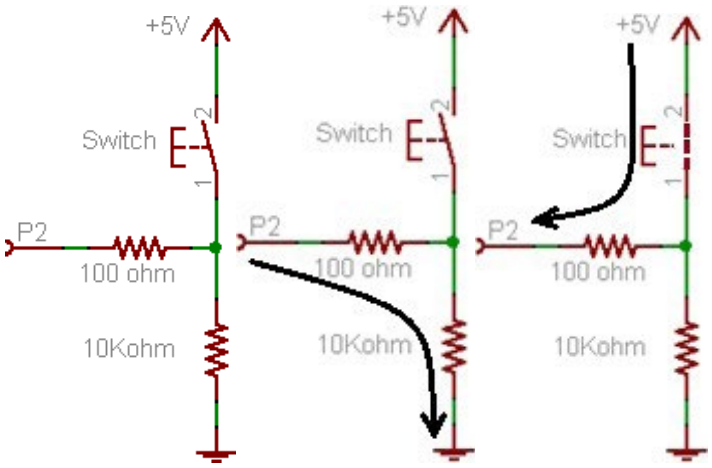
Слика 42. Микротастер, 6 mm

Ови мали тастери су јефтини, практични за употребу на матадор плочама, 4-пински су, с тим да су по два пина спојена заједно тако да су, без обзира на четири ножице, ово двожични прекидачи.



Слика 43. Микротастер, 6 mm, унутрашња структура

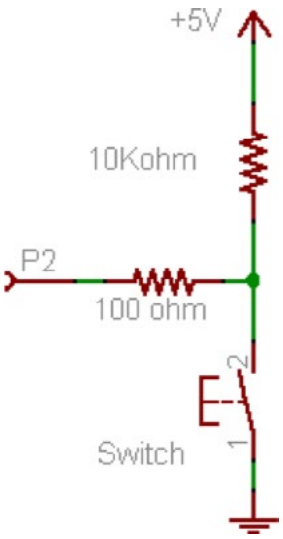
Добра пракса је исправити ножице јер се тако лакше позиционирају на матадор плочи. Али како формирати електрично коло на исправан начин? Наиме, двије су комбинације како спојити тастер на улаз микроконтролера. То су *pull-down* и *pull-up* конфигурација. *Pull-down* конфигурација приказана је на сљедећој слици.



Слика 44. Pull-down конфигурација

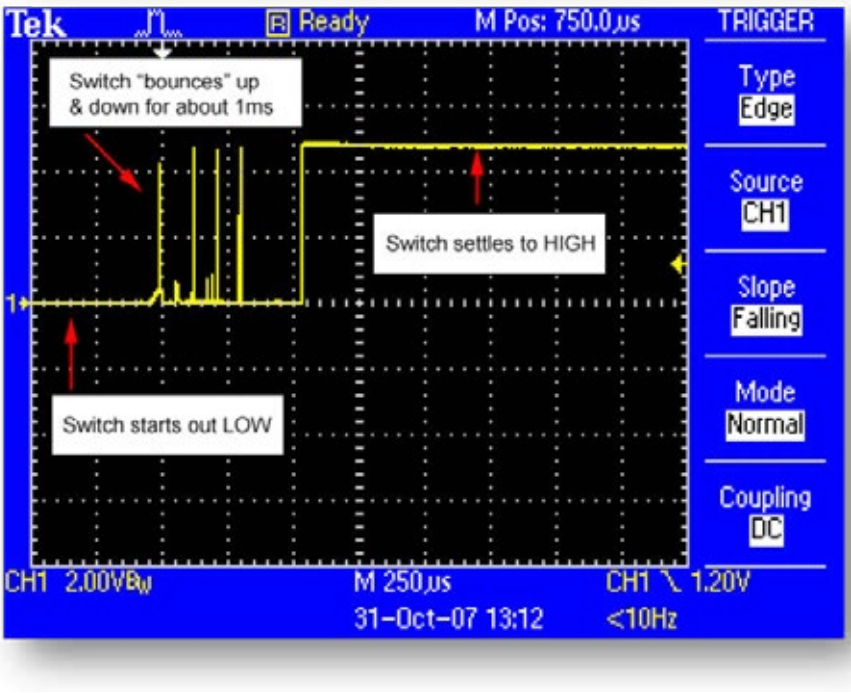
Објаснићемо претходну слику. Када тастер није притиснут, пин 2 микроконтролера је преко 100-омског и 10-килоомског отпора спојен на масу. Стога контролер то види као логичку 0 (нулу). С друге стране, када се тастер притисне, имамо ситуацију да на улаз пина 2 микроконтролера преко 100-омског отпора долази 5 V, што микроконтролер види као логичку 1 (јединицу).

Друга конфигурација је тзв. *pull-up* конфигурација. Концепт је сличан, а логика обратна. Опишите је.



Слика 45. Pull-up конфигурација

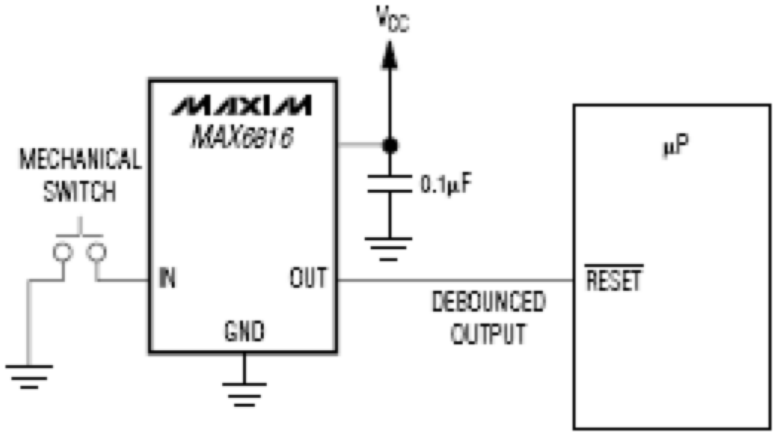
Други проблем код дигиталних улаза с механичким прекидачима је тзв. ***Debouncing*** или одскакање контаката, које се јавља приликом притиска на тастер. Ефекат одскакања контаката најбоље се види снимањем сигнала осцилоскопом.



Слика 46. Debouncing – одскакање контаката

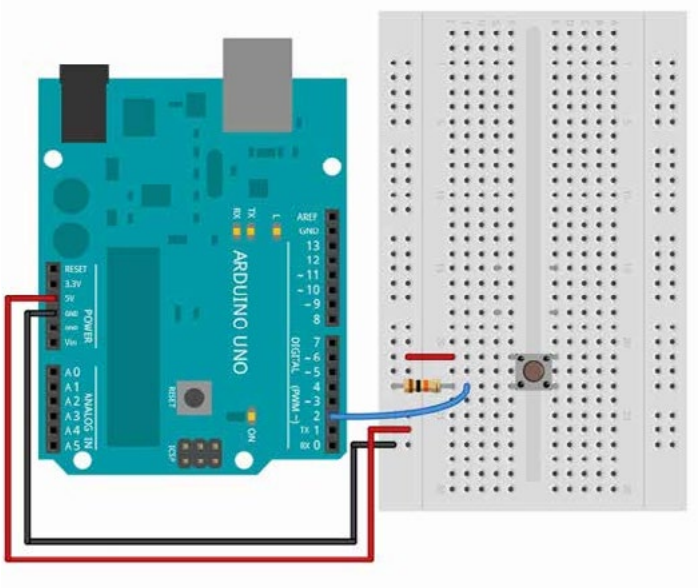
Сада је јасно да ће микроконтролер у случају са слике, умјесто да једном очита стање логичке јединице, то урадити пет пута, тако да, уколико бисте имали апликацију која броји притиске корисника на тастер, сваки пут бисте имали погрешно бројање.

Више је начина да се ријеши тај проблем. Један од њих је употреба *debouncing* интегралног кола, као на сљедећој слици.



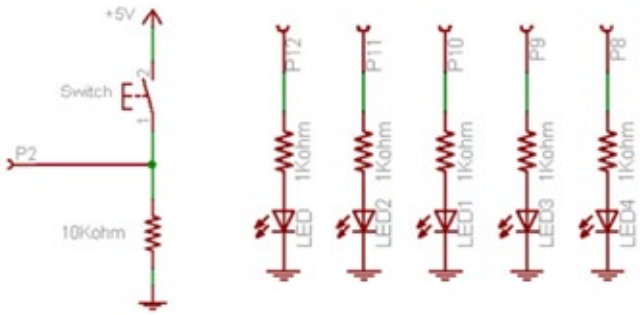
Слика 47. Debouncing интегрално коло

Наравно, то би било неко професионално рјешење. Ми можемо проблем ријешити тако што ћемо дати времена да се контакти смире, те ћемо послије притиска на тастер сачекати до 10 секунди. Наравно, постоје и друга рјешења, али ми ћемо овдје искористити најједноставније. Па идемо пробати.



Слика 48. Поједностављена шема спајања тастера на пин 2

Прво је потребно формирати спајање за контролу LED на матадор плочи према сљедећој електрошеми.



Слика 49. Шема спајања за контролу LED

```
/*
    Праћење стања на дигиталном улазу те промјена стања на LED на
пиновима од 12 до 8. Бројање колико пута је корисник притиснуо тастер,
повећавање вриједности глобалне варијабле counter те укључење одгова-
рајуће LED диоде спрам стања бројача.
*/
int inPin = 2;
int ledOut1 = 12;
int ledOut2 = 11;
int ledOut3 = 10;
int ledOut4 = 9;
int ledOut5 = 8;
int counter = 0;
int inPinstate = 0;               // Варијабла стања пина 2

void gasi()                       // Функција за гашење свих ЛЕД
{
    for(int i=8; i<=12; i++)
    {
          digitalWrite(i, LOW);         //Угаси LED od 8 do 12
    }
}
void setup()
{
 pinMode(inPin, INPUT);                 //Пин 2 је input
    for(int i=8; i<=12; i++)
    {
          pinMode(i, OUTPUT);     //Пинови od 8 do 12 OUTPUT
    }
}

void loop()
{
    if(counter == 0) gasi();      // Гаси све за count =0

    inPinstate = digitalREad(inPin);  //Спреми стање на pin2 у
варијаблу

    if(inPinstate == 1) counter++;// Повећај стање    када
    delay (250);                  // Сачекајмо мало

    if(counter == 1) digitalWrite(ledOut1, HIGH);
        if(counter == 2) digitalWrite(ledOut2, HIGH);
        if(counter == 3) digitalWrite(ledOut3, HIGH);
        if(counter == 4) digitalWrite(ledOut4, HIGH);
        if(counter == 5) digitalWrite(ledOut5, HIGH);
    if(counter >5) counter = 0;
}
```

Искористите седам-сегментни дисплеј те прикажите вриједност варијабле counter на седам-сегментном дисплеју.

```
/*
*/
int inPin = 2;
int ledOut1 = 12;
int ledOut2 = 11;
int ledOut3 = 10;
int ledOut4 = 9;
int ledOut5 = 8;
int counter = 0;
int inPinstate = 0;                //Варијабла стања пина 2.

void gasi()                        // Функција за гашење свих LED.
{
    for(int i=8; i<=12; i++)
    {
          digitalWrite(i, LOW);          //Угаси LED од 8 до 12.
    }
}
void setup()
{
 pinMode(inPin, INPUT);                  //Пин 2 је
    INPUT.  for(int i=8; i<=12; i++)
    {
          pinMode(i, OUTPUT);      //Пинови од 8 do 12 OUTPUT.
    }
}

void loop()
{
    if(counter == 0) gasi();       // Гаси све за count =0.

    inPinstate = digitalREad(inPin);  //Спреми стање на пин 2 у
варијаблу.

    if(inPinstate == 1) counter++;// Повећај стање  када
    delay (250);                   // Сачекајмо мало.

    if(counter == 1) digitalWrite(ledOut1, HIGH);
        if(counter == 2) digitalWrite(ledOut2, HIGH);
        if(counter == 3) digitalWrite(ledOut3, HIGH);
        if(counter == 4) digitalWrite(ledOut4, HIGH);
        if(counter == 5) digitalWrite(ledOut5, HIGH);
    if(counter >5) counter = 0;
}
```

Изазов за наставнике

Циљ вјежбе:

Употреба *digitalRead*() функције.

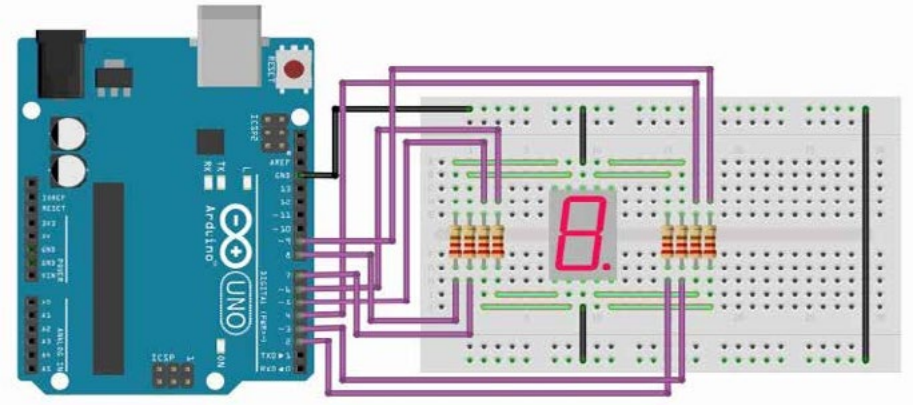
Задатак вјежбе:

Креирати *Sketch* за контролу рада седам-сегментног дисплеја -> *If uslovi!*

Потребне компоненте за вјежбу:

1. Arduino UNO 1 ком
2. Седам-сегментни дисплеј 1 ком
3. Отпорник 10 кΩ 1 ком
4. Отпорник 330 Ω 1 ком
5. Тастер 1 ком

Шема споја:



+Кораци за реализацију вјежбе:

1. Креирајте склоп користећи *Fritzing* скицу.
2. Креирајте *Sketch* којим ћете управљати радом седам-сегментног дисплеја на тај начин да се сваким притиском на тастер мијења приказ на њему од 0 до 9.
3. Уписати рјешење у за то предвиђен простор.
4. Користите примјере и готове функције искодиране у примјерима за *digital output.*

Написати код:

Циљ вјежбе:

Употреба *digitalRead()* функције.

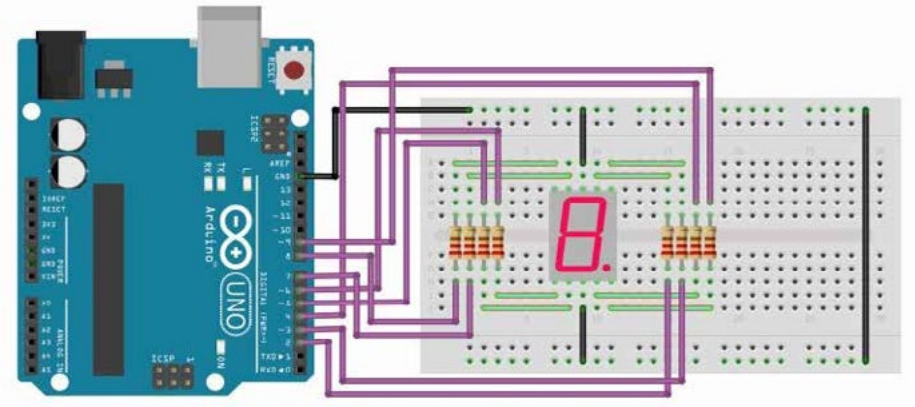
Задатак вјежбе:

Креирати *Sketch* за контролу рада седам-сегментног дисплеја (***switch case***).

Потребне компоненте за вјежбу:

1. Arduino UNO 1 ком
2. Седам-сегментни дисплеј 1 ком
3. Отпорник 10 кΩ 1 ком
4. Отпорник 330 Ω 1 ком
5. Тастер 1 ком

Шема споја:



Кораци за реализацију вјежбе

1. Креирајте склоп користећи *Fritzing* скицу.
2. Креирајте *Sketch* којим ћете управљати радом седам-сегментног дисплеја на тај начин да се сваким притиском на тастер мијења приказ на њему од 0 до 9.
3. Уписати рјешење у за то предвиђен простор.
4. Користите примјере и готове функције искодиране у примјерима за *digital output*.

Написати код:

Циљ вјежбе:

Употреба *digitalRead()* функције.

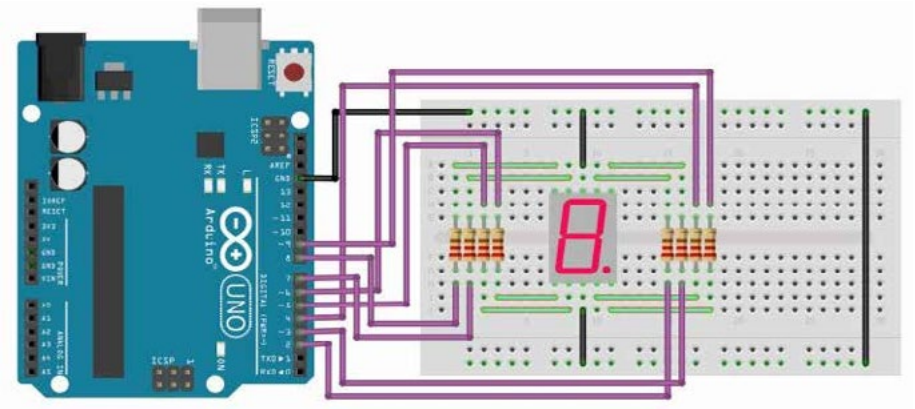
Задатак вјежбе:

Креирати *Sketch* за контролу рада седам-сегментног дисплеја.-> up/down бројач!

Потребне компоненте за вјежбу*:*

1. Arduino UNO 1 ком
2. Седам-сегментни дисплеј 1 ком
3. Отпорник 10 кΩ 2 ком
4. Отпорник 330 Ω 1 ком
5. Тастер 2 ком

Шема споја:



Кораци за реализацију вјежбе

1. Креирајте склоп користећи *Fritzing* скицу.
2. Креирајте *Sketch* којим ћете управљати радом седам-сегментног дисплеја на тај начин да се сваким притиском на тастер за бројање нагоре и надоље мијења стање бројача које је приказано на седам-сегментном дисплеју.
3. Уписати рјешење у за то предвиђен простор.
4. Користите примјере и готове функције искодиране у примјерима за *digital output*.

Написати код:

Serial библиотека – software debugging

Једна од мана Arduino развојне платформе јесте то што нема хардверски debugger, који омогућава онлајн праћење и извршавање програмских инструкција, тако да је лакше пронаћи грешке у коду. Креатори Arduin-а су стога креирали скуп процедура које омогућавају да имамо извјештавање шта се тренутно дешава с нашим кодом.

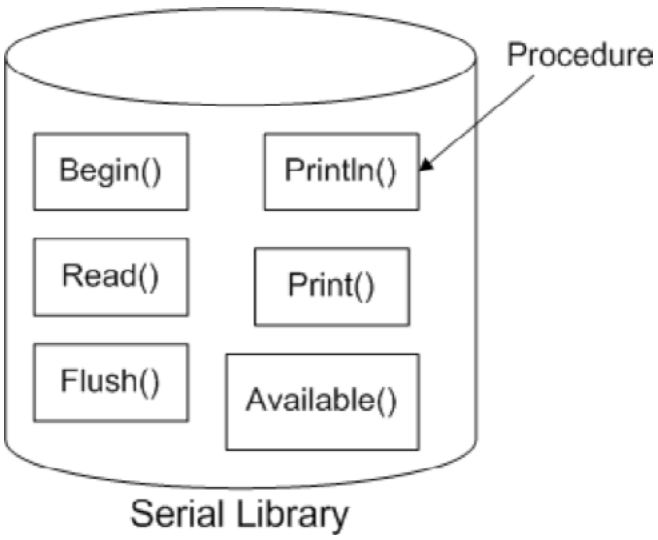
Тај скуп процедура зове се библиотека, у нашем конкретном случају ријеч је о Serial библиотеци која омогућава да PC апликација комуницира с Arduino кроз USB порт.

Прије него што наставимо, рећи ћемо неколико ријечи о библиотекама. Рецимо, када будете имали захтјев да направите апликацију за контролу stepper или серво мотора, вјероватно ћете требати *servo.h* библиотеку или *stepper.h* библиотеку. Позивање библиотеке ради се на сљедећи начин:

#include <servo.h>

На тај начин позивамо и користимо све процедуре из библиотеке серво те тако олакшавамо себи рјешавање проблема, односно избјегавамо писање својих процедура за контролу серво мотора.

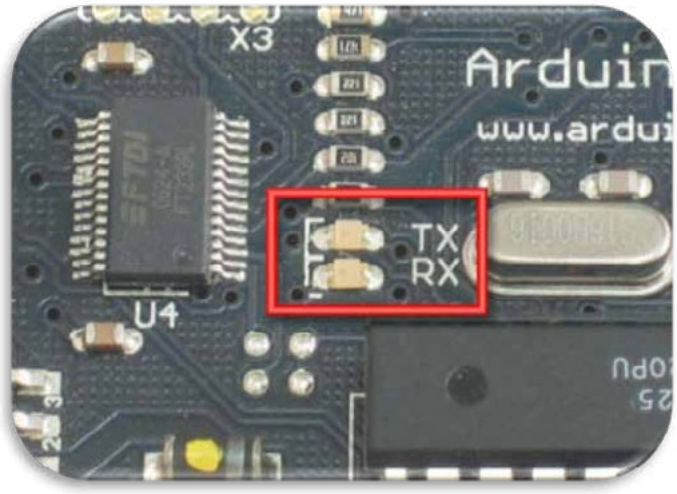
Библиотека коју прву треба да научимо користити јесте *Serial библиотека*. На сљедећој слици приказане су основне процедуре које ћемо користити у наредним примјерима.



Слика 50. Процедуре из Serial библиотеке

Ми смо већ користили серијску комуникацију, јер сваки пут када радимо **Compile/Verify** наше скице, код претварамо у бинарни запис (нуле и јединице), а када урадимо **Upload**, у ствари, шаљемо низ тих истих нула и јединица ка Arduin-у које се онда похрањују на простор предвиђен за апликацију.

Примијетили сте да сваки пут када радимо *upload* кода на Arduino двије LED означене са RX и TX блинкају када имамо проток информација, односно нула и јединица. RX линија блинка када имамо процес примања података на контролер, а TX линија кад имамо процес слања података од контролера ка PC-ју.

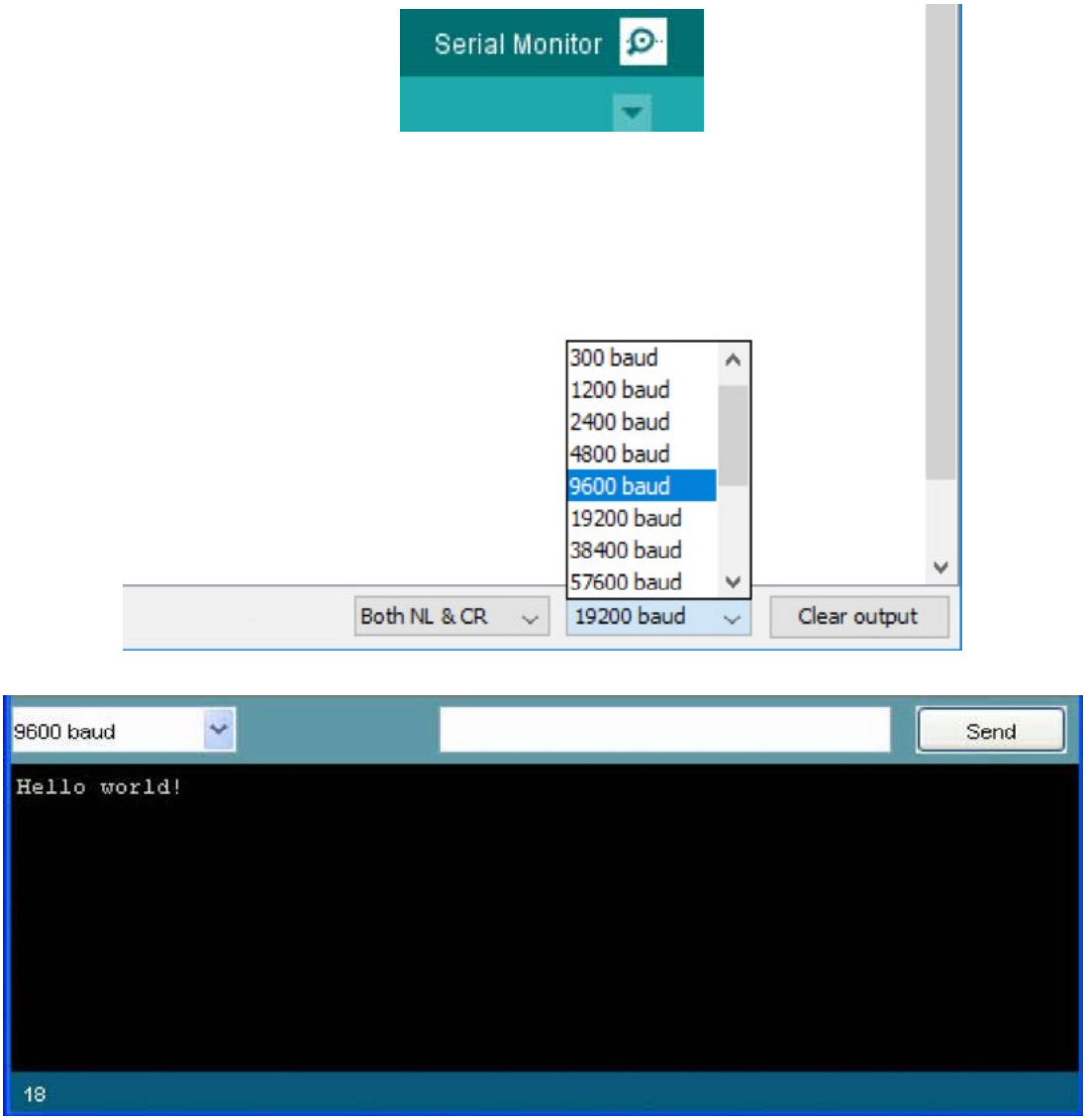


Слика 51. RX и TX LED

Идемо направити још једну „Здраво, свијете!" апликацију, али овог пута комуникациону „Здраво, свијете!" апликацију. Како све више постајемо програмери, значење процедура учићемо из коментара.

```
    /*
    „Здраво, свијете!" aplikacija u kojoj ćemo vidjeti kako Arduino
шаље информације на PC. Нећемо морати урадити #include <Serial.h> јер је
ова библиотека buil-in, дакле, већ уграђена.
    */
    void setup()
    {
     Serial.begin(9600);              //Користи комуникацију при
брзини од 9600 bps
     Serial.println("Zdravo, svijete!");// Пошаљи на PC „Здраво,
свијете!"
    }

    void loop()
    {
    }
```

Након што сте компајлирали и учитали апликацију у Arduino, на Arduino IDE нађите скраћеницу за апликацију *Serial Monitor,* подесите брзину као у апликацији на 9600 bps. По потреби ресетујте контролер на тастеру *reset* на развојној платформи.

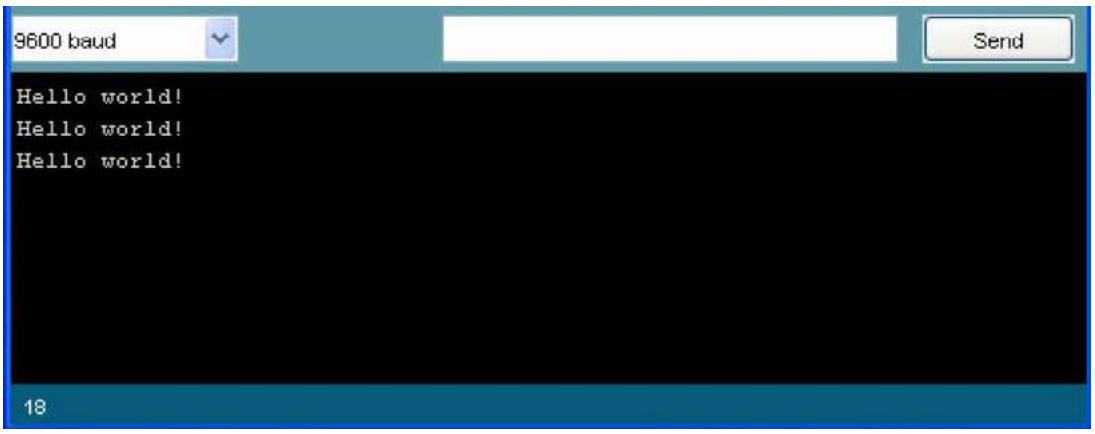


Слика 52. „Hello, World!" апликација за комуникацију

Како смо процедуру *println()* позвали у *setup* петљи, која се извршава само једанпут, то ће на серијском монитору бити исписано да је string „Здраво, свијете*!*" послан на PC само једном. Идемо то пробати у *loop* петљи.

```
     /*
     „Здраво, свијете!” апликација у којој ћемо видјети како Arduino
шаље информације на PC.
     */

     void setup()
     {
      Serial.begin(9600);              //Користи комуникацију при
од 9600 bps.                          брзини

     }

     void loop()
     {
         Serial.println(“Zdravo, svijete!”);// Пошаљи на PC „Здраво,
свијете!”
         delay(1000);                  // Чекај 1 секунду.
     }
```

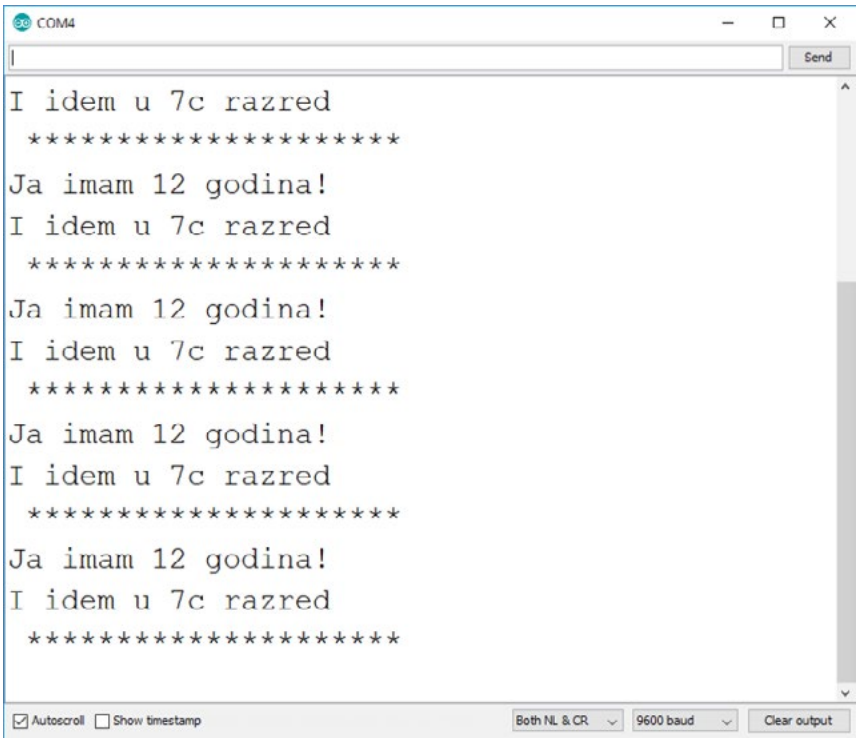
Резултат овог кода приказан је на сљедећој слици, дакле, сваке секунде микроконтролер на PC пошаље поруку „Hello, World!”.



Слика 53. „Hello, World!”

Види се да функција ***println*** шаље на PC поруку сваки пут у новом реду. Уколико желимо креирати неку комплекснију поруку, можемо да користимо и процедуру ***print***, која поруку исписује у истом реду, па идемо то тестирати.

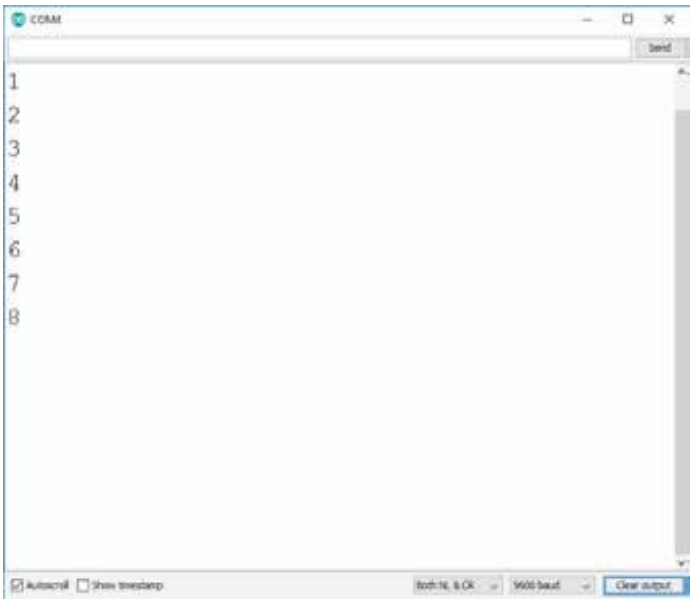
```
/*
„Здраво, свијете!” апликација у којој ћемо видјети како Arduino
шаље информације на PC
*/

void setup()
{
 Serial.begin(9600);              //Користи комуникацију при брзини
од 9600 bps.

}

void loop()
{
    Serial.print(“Ja imam”);
        Serial.print(“  “);       // Писати године.
        Serial.println(“godina!”);
        Serial.print(“I idem u”);
        Serial.print(“  “);       // Уписати
        разред. Serial.println(“razred”);
    Serial.println(“ ********************  “);
    delay(1000);                  // Чекај 1 секунду.
}
```



Слика 54. Можете написати и писмену задаћу у Arduin-у

Дакле, из претходног примјера се види да је могуће креирати и мало комплексније информације које се могу слати на PC. Прије него што покушамо радити комплексну математику у Arduin-у, идемо да видимо како можемо да користимо *Serial Monitor* као *debugger*, односно како можемо да пратимо стање неке варијабле у апликацији.

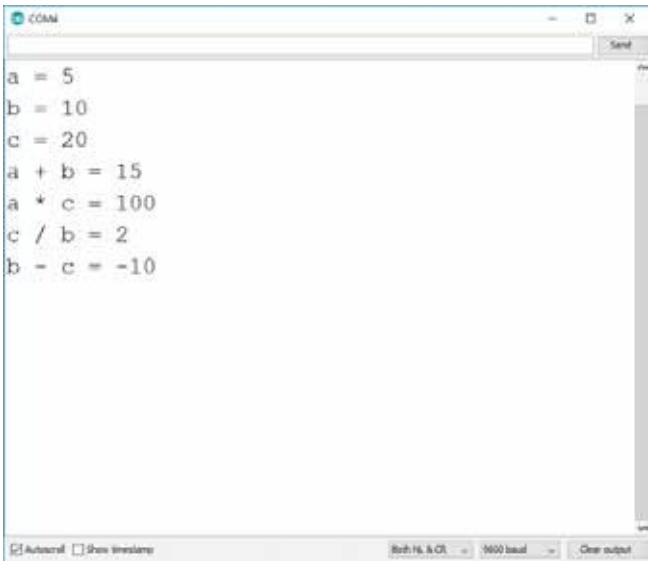
```
/*
„Здраво, свијете!” апликација у којој ћемо видјети како Arduino
шаље информације на PC-ју.
*/
int i=0;

void setup()
{
 Serial.begin(9600);              //Користи комуникацију при брзини
од 9600 bps.

}

void loop()
{
    i++;              // Ради инкримент.
    Serial.println(i);

    delay(1000);             // Чекај 1 секунду.
}
```



Слика 55. Приказ стања бојача у Serial Monitor-у

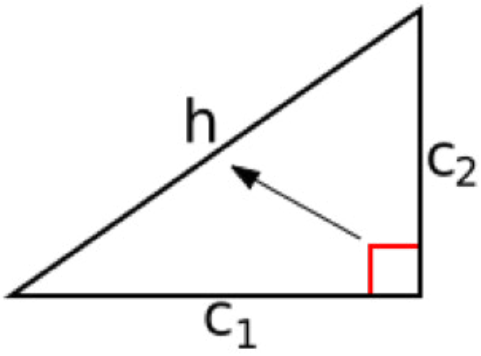
Идемо покушати радити математику користећи Arduino.

```
/*
 Домаћа задаћа из математике на мало другачији начин!
 */

int a = 5;
int b = 10;
int c = 20;

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

  Serial.println("Evo malo matematike: ");

  Serial.print("a = ");
  Serial.println(a);
  Serial.print("b = ");
  Serial.println(b);
  Serial.print("c = ");
  Serial.println(c);

  Serial.print("a + b = ");       // Сабирање.
  Serial.println(a + b);

  Serial.print("a * c = ");       // Множење.
  Serial.println(a * c);

  Serial.print("c / b = ");       // Дијељење.
  Serial.println(c / b);

  Serial.print("b - c = ");       // Одузимање.
  Serial.println(b - c);
}

void loop()
{
}
```



Слика 56. Рјешење у Serial Monitor-у

DIY – serial



Слика 57. Питагорина теорема

Израчунајте хипотенузу користећи Питагорину теорему и припадајуће формуле. За израчунавање коријена користите ***sqrt*** функцију.

$$a2 + b\,2 = h$$

$$2\,h = \sqrt{(a2 + b2)}$$

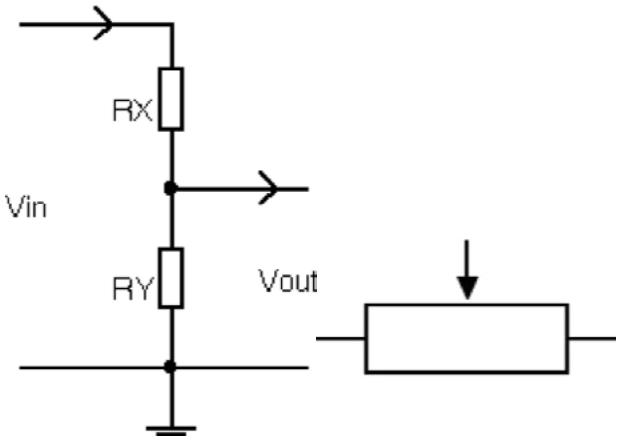
Преписати рјешење:

```
/*
 * Math is fun!
 */



int a = 3;
int b = 4;
int h;

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

  Serial.println("Proracun hipotenuze:");

  Serial.print("a = ");
  Serial.println(a);

  Serial.print("b = ");
  Serial.println(b);

  h = sqrt( a*a + b*b );

  Serial.print("h = ");
  Serial.println(h);
}

void loop()
{
}
```

AnalogRead

На Arduino платформи имамо 6 напонских аналогних улаза. За разлику од дигиталних улаза на које се могу спојити уређаји који могу да имају само двије вриједности „0" или „1", на аналогни улаз можемо да спојимо сензоре или електронске компоненте које на свом излазу могу да дају напон од 0 од 5 V. Најједноставнија електронска компонента коју можемо да спојимо на аналогни улаз микроконтролера јесте потенциометар.

Потенциометар је тропински промјењиви отпорник, који има клизни или ротирајући контакт, тако да се у колу, када је спојен као на сљедећој слици, понаша као дјелитељ напона, тј. ако на његове крајеве доведемо напон ***Vin***, напон ***Vout*** биће дупло мањи од ***Vin*** ако је ***Rx=Ry***.



Слика 58. Дјелитељ напона и симбол потенциометра

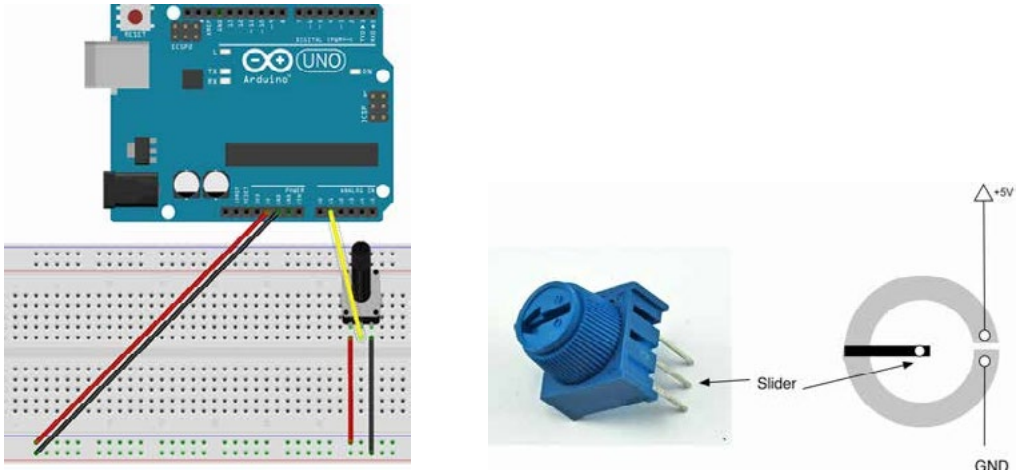
Тако, ако је напон на његовим крајевима 5 V, на клизном контакту можемо да имамо вриједности од 0 до 5 V, те овај елемент као такав представља идеалан електронски елеменат за тестирање аналогног улаза микроконтролера.

Наравно, микроконтролер не види напон на свом улазу, већ преко свог интерног ADC (*analog to digital* конвертера) кола претвара вриједност напона у дигиталну вриједност типа интегер.

Па пошто је резолуција/прецизност нашег аналогног улаза 10-битна, вриједности које функција ***analogRead*** враћа крећу се у границама од 0 до 1023.

Дакле, Arduino ће мапирати улазни напон потенциометра од 0 до 5 V у интегер вриједност од 0 до 1023. То значи вриједност „1" интегера у напону је 0,0049 V или 4,9 mV, а то је најмања промјена коју микроконтролер може да осјети.

Тестираћемо све до сада речено. Користићемо *Serial* библиотеку за софтверски *debug-ging*. Спојите шему на матадору према сљедећој слици.



Слика 59. Потенциометар на матадор плочи и као елеменат

```
      /*
      Употреба функције analogRead за читање вриједности сирових података
на аналогном улазу микроконтролера.
      */
      int analogPin=1;
      int val=0;
      void setup()
      {
       Serial.begin(9600);               //Користи комуникацију при
брзини од 9600 bps.

      }

      void loop()
      {
          val = analogRead(analogPin);// Читај стање s A1.

          Serial.println(val);         // Пошаљи податке на Serial Monitor.

      }
```

Покушајмо сада да добијемо и вриједност напона на нашем аналогном улазу. Формула за претварање сирових података на аналогном улазу у напон је једноставна и гласи:

float voltage = val(5.0/1023.0);*

Овдје користимо нови тип података *float*. То је тип података намијењен за приказ децималних бројева.

```
     /*
     Употреба функције analogRead за читање вриједности сирових података
на аналогном улазу микроконтролера.
     */
     int analogPin=1;
     int val=0;
     void setup()
     {
      Serial.begin(9600);              //Користи комуникацију при
брзини од 9600 bps.

     }

     void loop()
     {
         val = analogRead(analogPin);// Читај стање s A1.

         Serial.println(val);        // Пошаљи податке на Serial Monitor.

     }
```

Допишите формулу и дио кода за слање вриједности напона на серијски монитор сваких 250 милисекунди у сљедећем формату:

Raw data: 512 Voltage: 2.5 (V)

DiY – analogRead

Циљ вјежбе:

Употреба *analogRead(pin)* функције.

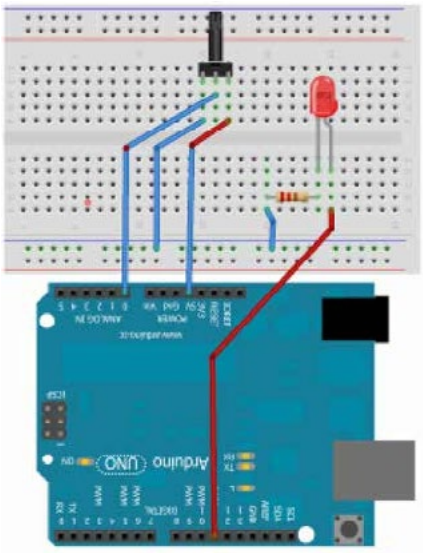
Задатак вјежбе:

Креирати *Sketch* за управљање бројем укључених LE диода спрам вриједности сирових података на аналогном улазу.

0-127 **LED 1 ON**
128-255 **LED 2 ON**
256-383 **LED 3 ON**
384-511 **LED 4 ON**
512-639 **LED 5 ON**
640-766 **LED 6 ON**
767-895 **LED 7 ON**
896-1023 **LED 8 ON**

Потребне компоненте за вјежбу:

1. Arduino UNO 1 ком
2. Потенциометар 10 kΩ 1 ком
3. LE диоде 8 ком
4. Отпорник 330 Ω 8 ком



Слика 60. Шема споја

Кораци за реализацију вјежбе:

1. Креирајте тестни систем користећи *Fritzing* скицу, водећи рачуна да је потребно спојити 8 LE диода.

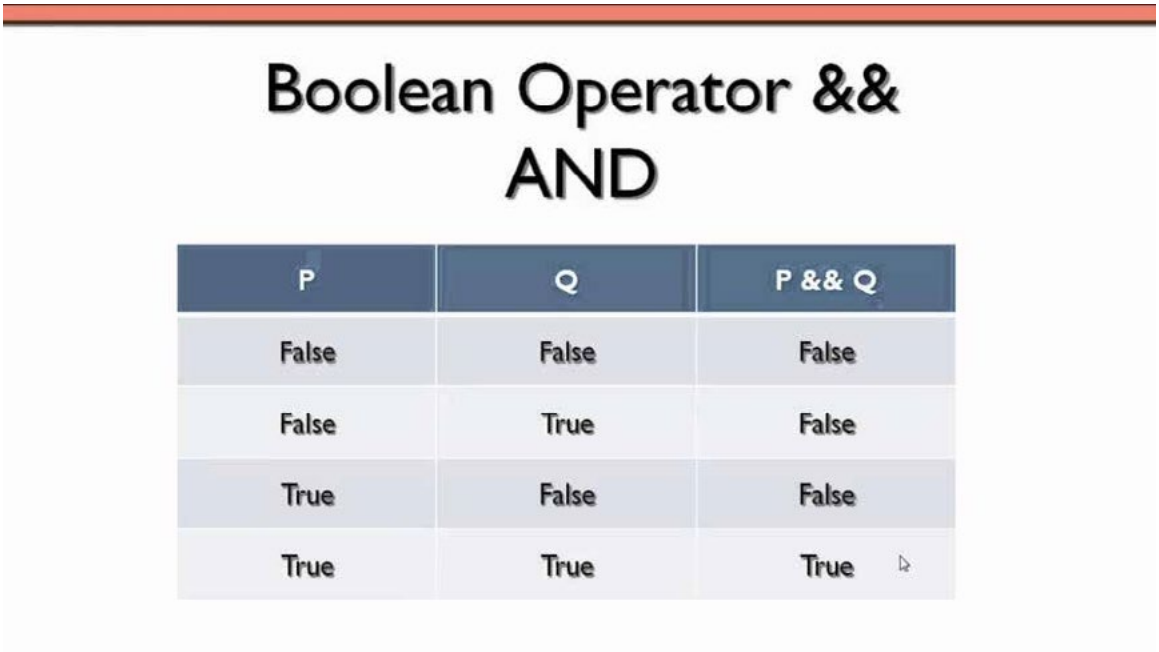
2. Креирајте *Sketch* којим ћете читати аналогну вриједност с аналогног пина 0, те вриједност приказати у *Serial Monitor*-у и спрам задатка вјежбе упалити одговарајуће LE диоде.

3. Задатак може да се ријеши *if* упитима, користећи операторе просљеђивања; употријебите *Serial Monitor* за праћење вриједности аналогног улаза.

   x == y (x jednako y)
   x != y (x nije jednako y)
   x < y (x manje od y)
   x > y (x veće od y)
   x <= y (x manje ili jednako od y)
   x >= y (x veće ili jednako y)

4. Такође, за рјешење задатка треба да користите и *Boolean* или логичке операторе.

&& (logical and)

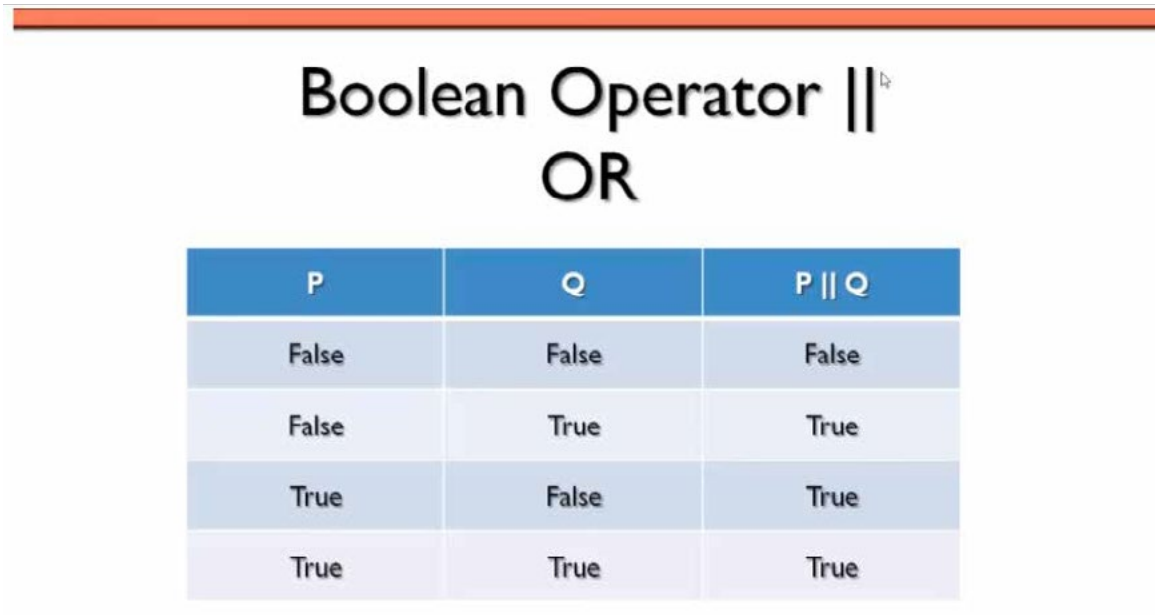


Логички AND или логички „i ” оператор захтијева да оба исказа буду истинита да би резултат упита био истинит. Например:

```
if (val1 == 100  && val2 == 200)
    {
        //uradi nešto
    }
```

Дакле, услов је испуњен само ако је вриједност варијабле val1 = 100 и варијабле val2 = 200. За све остале услове нећемо извршити оно што је унутар *if* услова.

|| (logical or)



Логички OR или логички „ili ” оператор захтијева да било који исказ или оба исказа буду истинити да би резултат упита био истинит. Например:

```
if (val1 == 100  ||  val2 == 200)
        {
                //uradi nešto
        }
```

Дакле, услов је испуњен ако је вриједност варијабле val1 = 100 или варијабле val2 = 200 или ако су обје варијабле тачне.

Препиши рјешење:

```
        int analogPin=1;
        int val=0;
        void setup()
        {
        for(int i =2;i<=10,i++)
        {
            pinMode(i, OUTPUT);
        }
         Serial.begin(9600);              //Користи комуникацију при
    од 9600 bps.                          брзини

        }
        void loop()
        {
            val = analogRead(analogPin);// Читај стање с A1.

            Serial.println(val);        // Пошаљи податке на Serial Monitor.

            if ((val>=0)&&(val<=127))
            {
                  digitalWrite(2, HIGH);
            }
            else
            {
                  digitalWrite(2, LOW);
            }


        }
```

Циљ вјежбе:

Употреба аналогРеад*()* и *serialPrintln()* функције.

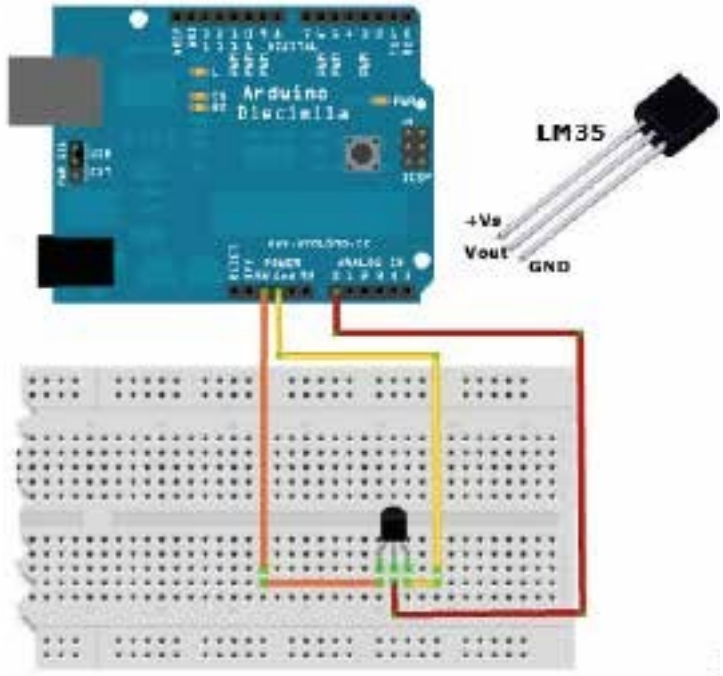
Задатак вјежбе:

Креирати *Sketch* за прорачун температуре.

Потребне компоненте за вјежбу:

1. Arduino UNO 1 ком
2. LM35 1 ком

Шема споја:



Кораци за реализацију вјежбе:

1. Креирајте тестни систем користећи претходну скицу.

2. Креирајте *Sketch* којим ћете читати аналогну вриједност с аналогног пина 0, те вриједност приказати у *Serial Monitor*-у. Користећи једначину за прорачун температуре, израчунати температуре у °C, K и F те их приказати у *Serial Monitor*-у.

3. Исписати температуре у *Serial Monitor*у у сљедећем формату:

Temperatura °C -> 23.5
Temperatura K -> 296.65
Temperatura °F -> 74.3

Основне теоријске поставке:

Температура температурног сензора LM35 може да се израчуна према сљедећем изразу:

T = (Vcc * ADC * 100.0) / 1024;
Celsius = Kelvin - 273.15
Celsius = 5/9 x (Fahrenheit -32)

Препиши рјешење:

```
int analogPin=0;
int val=0;
void setup()
{
Serial.begin(9600);              //Користи комуникацију при брзини
9600 bps.

}
void loop()
{
    val = analogRead(analogPin);// Читај стање с A1.

    Serial.println(val);        // Пошаљи податке на Serial Monitor
// Примијенити формуле за температуру и исписати
```

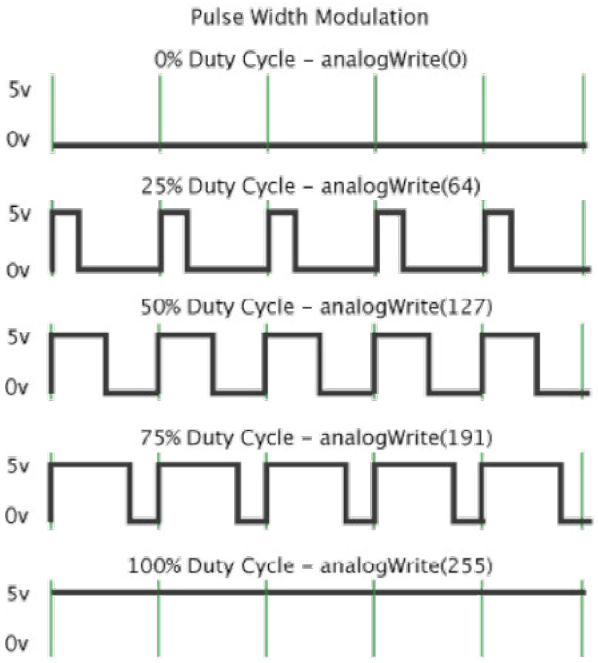
PWM – analogWrite

Вјероватно сте имали прилику да на разним расвјетним тијелима видите ефекат звани „fading" или постепено гашење. Тај ефекат можемо да реализујемо и с Arduin-ом јер он има 6 PWM пинова (PWM је скраћеница од *Pulse Width Modulation*), а то је техника за добијање аналогних резултата уз помоћ дигиталног сигнала.

Наиме, PWM подразумијева да на излазу микроконтролера генеришемо правоугаони импулс, те на тај начин можемо да симулирамо напоне од 0 до 5 V.

Вријеме „On time" сигнала зове се ширина импулса. Да бисмо добили различите аналогне вриједности, на излазу микроконтролера мијењамо ширину тог импулса, односно његово трајање. Ако такав сигнал понављамо довољно брзо те га примијенимо на спојену LED, резултат ће бити промјена интензитета свјетлости на LED.

На графикону је приказан PWM сигнал. Период (T) је означен зеленом бојом и траје 2 милисекунде, па је фреквенција f = 1/T, 500 Hz.



Слика 61. PWM сигнал

За нас је довољно да знамо да се PWM сигнал може генерисати функцијом

analogWrite(pin, value);

Веома је важно напоменути да је ту функцију могуће користити само на посебно означеним пиновима Arduin-а и то са сљедећим ознакама „~" , „#" или само „PWM".

Функција има два аргумента. Пин на којем желимо да генеришемо PWM сигнал и *value*, која може да буде у интервалу од 0 до 255. Наш PWM сигнал има 8-битну резолуцију па је $2^8 = 256$.

Тестираћемо до сада научено на сљедећем примјеру. Изаберите жељени PWM пин, спојите предотпор и LED те тестирајте наредни код.

```
/*
Употреба функције analogWrite за креирање PWM сигнала
*/
int pwmPin = 5;

void setup()
{
 Serial.begin(9600);              //Користи комуникацију при
од 9600 bps.                      брзини

}

void loop()
{
    analogWrite(pwmPin, 0);   //Угаси LED.
    delay(1000);

    analogWrite(pwmPin,64);   //25% интензитета.
    delay(1000);

    analogWrite(pwmPin,128);        //50% интензитета.

        delay(1000);

    analogWrite(pwmPin,192);        //75% интензитета.
        delay(1000);

    analogWrite(pwmPin,255);        // 100% интензитета

        delay(1000);
```

Пратите промјену на LED, она постепено појачава интензитет свјетлости коју исијава. Идемо мало вјежбати.

DiY – analogWrite

Циљ вјежбе:

Употреба *analogWrite()* функције.

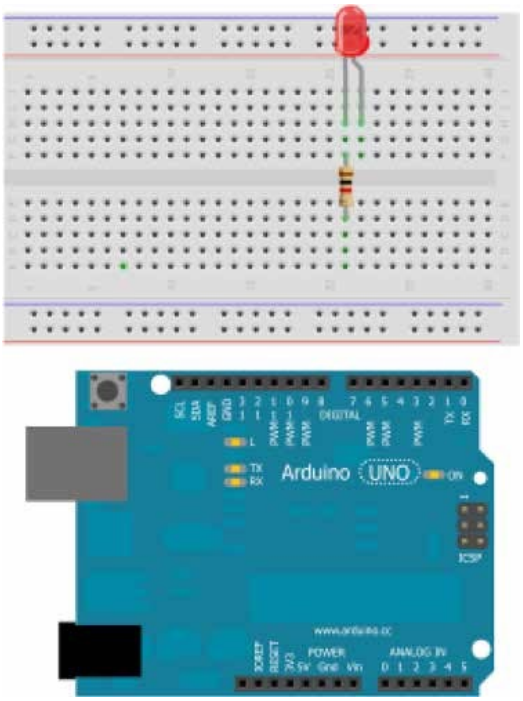
Задатак вјежбе:

Креирати *Sketch* за управљање радом LE диоде са *fade in u fade out* ефектом.

Потребне компоненте за вјежбу:

1. Arduino UNO 1 ком
2. LE диоде 1 ком
3. Отпорник 220 Ω 1 ком

Шема споја:



Кораци за реализацију вјежбе:

1. Креирајте шему споја користећи *Fritzing* скицу.

2. Креирајте *Sketch* којим ћете управљати радом LE диоде на тај начин да ће се у једнаким временским интервалима појачавати интензитет свјетлости LE диоде до максималне вриједности (255) и *vice versa* до (0), све с инкрементом и декрементом од 1.

3. Задатак је могуће ријешити употребом for петље. Обавезно унутар for петље позвати *delay()* функцију.

```
for(int i=0;i<=255;i++)
        {
        analogWrite(pin, Value);
        delay(50);
        }
```

Преписати рјешење:

```
/*
Употреба функције analogWrite за креирање PWM сигнала.
*/
int pwmPin = 5;

void setup()
{

}

void loop()
{
```

Циљ вјежбе:

Употреба *analogWrite()* функције.

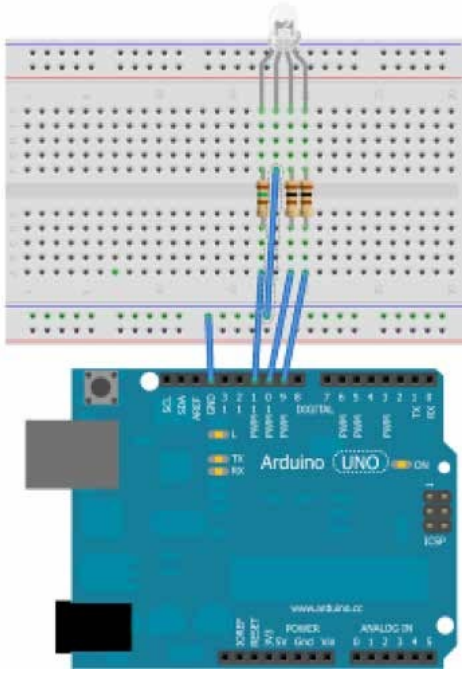
Задатак вјежбе:

Креирати *Sketch* за управљање радом RGB LED с *fade in* и *fade out* ефектом..

Потребне компоненте за вјежбу:

1. Arduino UNO 1 ком
2. RGB LED 1 ком
3. Отпорник 150 Ω 1 ком
4. Отпорник 100 Ω 2 ком

Шема споја:



Кораци за реализацију вјежбе:

1. Креирајте тестни систем користећи *Fritzing* шему споја.

2. Креирајте *Sketch* којим ћете управљати радом LE диоде на тај начин да ће се у једнаким временским интервалима појачавати интензитет свјетлости RGB LED појединачно до максималне вриједности (255) и *vice versa* до (0).

Преписати рјешење:

```
/*
Употреба функције analogWrite за креирање PWM сигнала.
*/
int pwmPin = 5;

void setup()
{

}

void loop()
{
```

Закључак

Искрено се надамо да вам је приручник разумљив. Ако сте помно пратили овај приручник, полако ћете у свакодневном животу почети примјећивати микроконтролерске системе, а, можда, чак и разумјети како они функционишу.

Надам се да ћете покушати направити и неки свој микроконтролерски систем, јер крај овог приручника је у суштини само нови почетак. Не журите да одмах правите „свемирску" апликацију, крените лаганим корацима, за почетак покушајте научити све сензоре из Arduino сета.

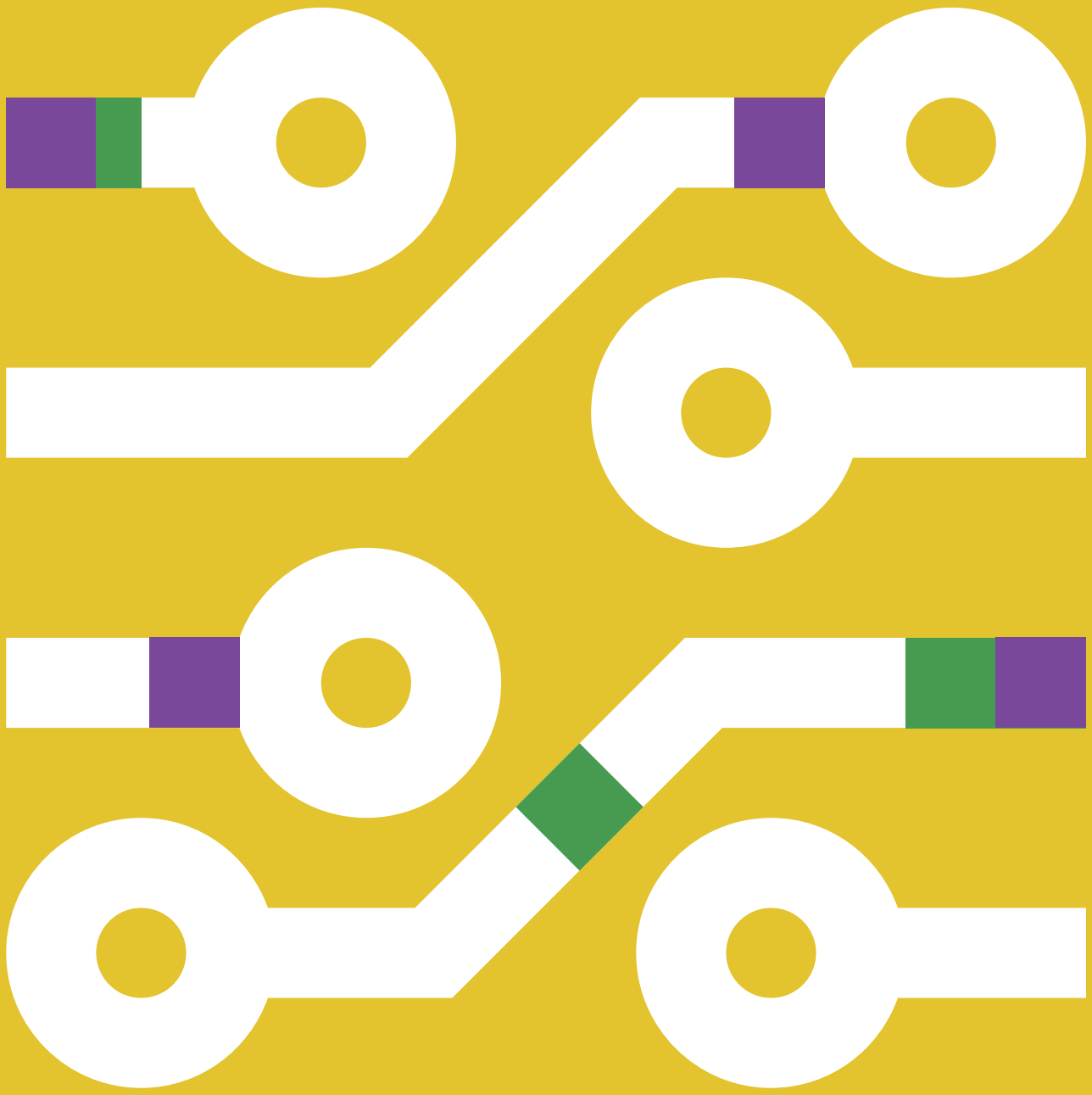
Покушајте увидјети с којим проблемом се сусрећете свакодневно. Размишљајте о томе је ли могуће за рјешење ваших проблема осмислити неки микроконтролерски систем.

Покушајте, па немали број људи се управо тако и обогатио. Наравно, не треба вам бити само то водиља. Само рјешавање проблема и послије безброј покушаја пронађено рјешење награда је само за себе. И, наравно, не заборавите уживати у свему томе.

Мр сци. Муамер Халиловић, дипл. инж. ел.

Литература

1. https://learn.adafruit.com/
2. https://www.Arduino.cc/en/Tutorial/HomePage
3. https://www.instructables.com/technology/Arduino/
4. https://www.makerspaces.com/Arduino-uno-tutorial-beginners/



ARDUINO ЗА СВЕ

ДОДАТАК ПРИРУЧНИКУ ЗА НАСТАВНИКЕ

Основна школа

Подржано од:

< IT Girls >

УЈЕДИЊЕНЕ НАЦИЈЕ
БОСНА И ХЕРЦЕГОВИНА

ЛАБОРАТОРИЈСКА ВЈЕЖБА БР. 1

ARDUINO UNO – LM35

УЧЕНИК/ЦА: .....

РАЗРЕД: .....

ДАТУМ: .....

ВЈЕЖБУ ПРЕГЛЕДАО/ЛА: .....

ДАТУМ: .....

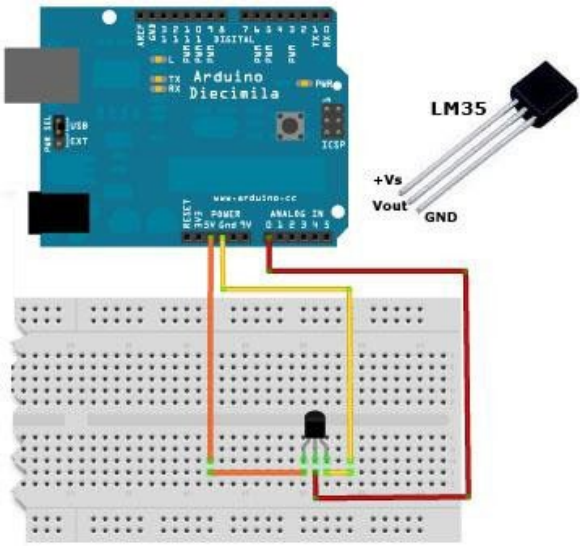
Циљ вјежбе:
Употреба analogRead(), analogWrite() , serialBegin() i serialPrintln() функције.

Задатак вјежбе:
Креирати *Sketch* за прорачун температуре с аналогног излаза LM35 сензора.

Потребни елементи за вјежбу:

1. Arduino Uno 1 ком
2. LM35 1 ком
3. Матадор плоча 1 ком

Шема споја:



Кораци за реализацију вјежбе:

1. Креирајте тестни систем користећи Fritzing скицу.

2. Креирајте *Sketch* којим ћете читати аналогну вриједност с аналогног пина 0 те вриједност приказати у *Serial Monitor*-у. Користећи једначину за прорачун температуре израчунати температуру у ℃, K i F те их приказати у Serial Monitor-у.

Основне теоријске поставке:

Температура температурног сензора LM35 може да се израчуна према сљедећем изразу:

$$T = (Vcc * ADC * 100.0) / 1024;$$
$$Celsius = Kelvin - 273.15 \quad Celsius = 5/9 \times (Fahrenheit - 32)$$

analogRead()

Опис
Чита вриједности с дефинисаног аналогног пина (10-битни A/D конвертер). Улазни напон 0 - 5 V мапира се у интегер вриједност 0 - 1023.

Синтакса
analogRead(pin)
Параметри
pin: број аналогног пина (0 - 5)
Враћа (return)
int (0 - 1023)

IT Girls - Arduino за све

Рјешење:

```
/*
 * Задатак ове вјежбе је употреба analogRead() функције те употреба тзв.
   arduino софтверског debuger-а
 * за очитање и приказ температуре просторије.
 */

int val=0; //Декларација integer типа варијабле.
           //у коју ћемо спремити сирове податке с
           //аналогног улаза.
float T=0; //Варијабла за спремање вриједности температуре.

void setup()
{
  Serial.begin(9600);//Користићемо Serial monitor app за провјеру.
температуре
  Serial.println("IT-Girls->2019");//Пошаљи Serial monitor app.
string "IT -Girls"
}

void loop()
{
  val = analogRead(A0);// Прочитај сирови податак с пина A0.
                       // Могуће вриједности су: 0-1023.

 T=(5*val*100)/1023;  //Формула за прорачун температуре за LM35
сензор.

Serial.print("Trenutna temperatura: ")
 Serial.print(T);
  Serial.println("°C");
}
```

Закључак:

ЛАБОРАТОРИЈСКА ВЈЕЖБА БР. 2

ARDUINO UNO – 2 x 16 LCD дисплеј

УЧЕНИК/ЦА: .....

РАЗРЕД: .....

ДАТУМ: .....

ВЈЕЖБУ ПРЕГЛЕДАО/ЛА: .....

ДАТУМ: .....

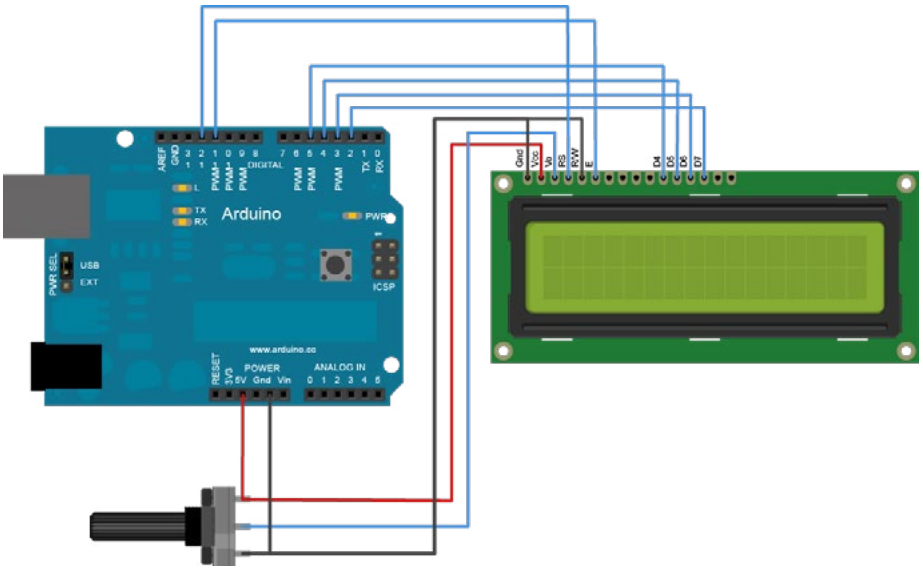
Циљ вјежбе:
Употреба 2 x 16 LCD дисплеја, начин спајања.

Задатак вјежбе:
Креирати *Sketch* за приказ текста.

Потребни елементи за вјежбу:

1.	Arduino Uno	1 ком
2.	2 x 16 LCD дисплеј	1 ком
3.	Потенциометар 10 KΩ	1 ком
4.	Матадор плоча	1 ком

Шема споја:



1. Креирајте тестни систем користећи Fritzing скицу.

2. Креирајте Sketch у којем ћете позвати библиотеку LiquidCrystal.h наредбом include.h i користећи HELP упознати се с функцијама потребним за испис текста на lcd дисплеју.

3. У првом реду написати своје име, а у другом име школе или разред и одјељење.

IT Girls - Arduino за све

Рјешење:

```
/*
 * Примјер кориштења 2 x 16 LCD дисплеја
 * за приказ информација кориснику.
 */

//Позив библиотеке за LCD

#include <LiquidCrystal.h>

//Креирање новог LCD објекта -> с конфигурацијом као у аргументима
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  //Наш LCD има 16 колона и два реда.
  lcd.begin(16,2);

  //Постави курсор на прво поље.
  lcd.setCursor(0,0);

  //printaj IT Girls
  lcd.print("IT - Girls :D");
  delay(1000);
}

void loop()
{
  lcd.setCursor(0,1);
  lcd.print("Sarajevo 2019");
  }
```

Закључак:

ЛАБОРАТОРИЈСКА ВЈЕЖБА БР. 3

ARDUINO UNO – 2 x 16 LCD дисплеј i LM35

УЧЕНИК/ЦА: .....

РАЗРЕД: .....

ДАТУМ: .....

ВЈЕЖБУ ПРЕГЛЕДАО/ЛА: .....

ДАТУМ: .....

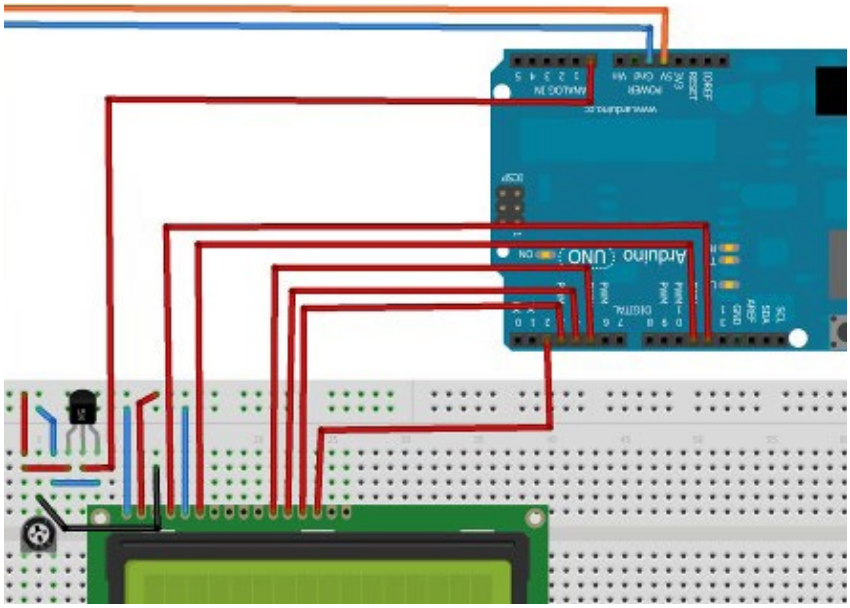
Циљ вјежбе:
Употреба 2 x 16 LCD дисплеја, начин спајања.

Задатак вјежбе:
Креирати *Sketch* за приказ температуре.

Потребни елементи за вјежбу:

1. Arduino Uno 1 ком
2. 2 x 16 LCD дисплеј 1 ком
3. Потенциометар 10 KΩ 1 ком
4. LM35 1 ком
5. Матадор плоча 1 ком

Шема споја:



Кораци за реализацију вјежбе:

1. Креирајте тестни систем користећи Fritzing скицу.
2. Креирајте *Sketch* у којем ћете позвати библиотеку LiquidCrystal.h наредбом include.h
3. Креирати апликацију за приказ температуре у ^{0}C у првом реду, а у другом вриједности напона на аналогном излазу A0. Креирати табелу вриједности напона и одговарајуће температуре.

IT Girls - Arduino за све

Рјешење:

```
                                                /*
 * Примјер кориштења 2 x 16 LCD дисплеја
 * за приказ информација кориснику.
 */

//Позив библиотеке за LCD

#include <LiquidCrystal.h>
int val=0;
float T=0;
//Креирање новог LCD објекта -> с конфигурацијом као у аргументима
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
{
  //Наш LCD има 16 колона и два реда
  .lcd.begin(16,2);

  //Постави курсор на прво поље.
  lcd.setCursor(0,0);

  //printaj IT Girls
  lcd.print("IT - Girls :D");
  delay(1000);
}

void loop()
{
   val = analogRead(A0);

   T=(5*val*100)/1023;

  lcd.setCursor(0,1);
  lcd.print("Temp:");

  lcd.setCursor(5,1);
  lcd.print(T);

  lcd.setCursor(12,1);
  lcd.print("°C");

  delay(1000);

}
```

Закључак:

ЛАБОРАТОРИЈСКА ВЈЕЖБА БР. 4

ON-OFF СЕНЗОРИ

УЧЕНИК/ЦА: .....

РАЗРЕД: .....

ДАТУМ: .....

ВЈЕЖБУ ПРЕГЛЕДАО/ЛА: .....

ДАТУМ: .....

Једноставни ON-OFF сензори су уређаји који мјере неку физичку величину и као излаз дају дигитално стање – укључено или искључено.

Сензор звука активира се када детектује јачину звука већу од оне подешене помоћу потенциометра на њему.

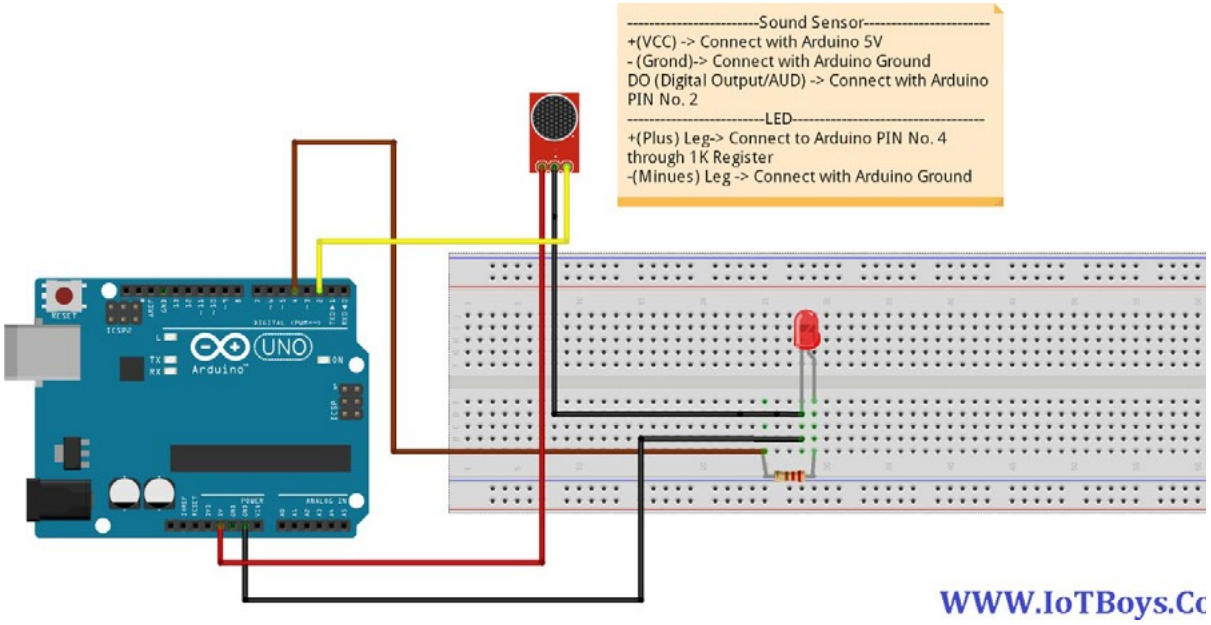
Задатак

Направити склоп за укључивање и искључивање свијетлеће диоде звуком. Сваки пут кад поред сензора пљеснете нпр. рукама нека диода се укључи ако је била искључена, односно искључи ако је била укључена.

Потребне компоненте:

- Arduino UNO плочица iUSB
- Матадор плочица
- Сензор звука
- 1 x црвена LED
- 1 x отпорник 220 Ω

Приказ спајања

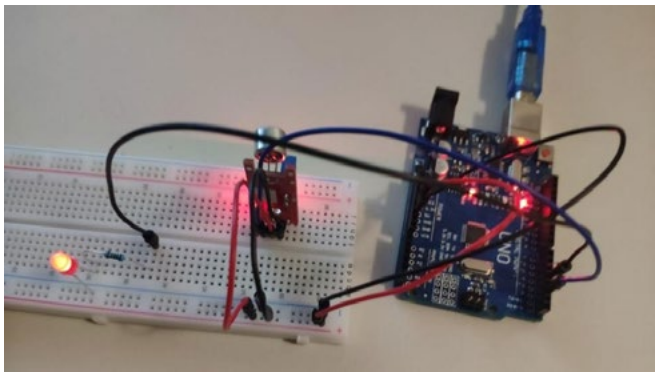


IT Girls - Arduino за све

Рјешење:

```
  int soundSensor=2;
int LED=4;
boolean LEDStatus=false;

void setup()
{
 pinMode(soundSensor,INPUT);                //Постави извод SenzorZvuk
kao ulazni                                  (2)
pinMode(LED,OUTPUT);                          //Постави извод LedCrvena
kao izlazni                                   (4).
}
void loop()
{
  int SensorData=digitalRead(soundSensor);
  if(SensorData==1)
  {                                         //Уколико је детектован звук.
    if(LEDStatus==false)
    {
       LEDStatus=true;
       digitalWrite(LED,HIGH);
    }
    else
    {
       LEDStatus=false;
       digitalWrite(LED,LOW);
    }
  }
 }
```

Закључак:



ЛАБОРАТОРИЈСКА ВЈЕЖБА БР. 5

ON-OFF СЕНЗОРИ

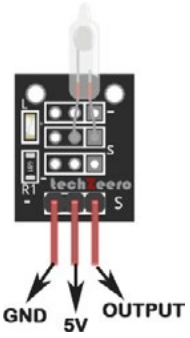
УЧЕНИК/ЦА: .....

РАЗРЕД: .....

ДАТУМ: .....

ВЈЕЖБУ ПРЕГЛЕДАО/ЛА: .....

ДАТУМ: .....



Сензор нагиба ради на принципу контакта два контакта и можемо грубо да одредимо је ли објект постављен усправно или хоризонтално на земљу или није.

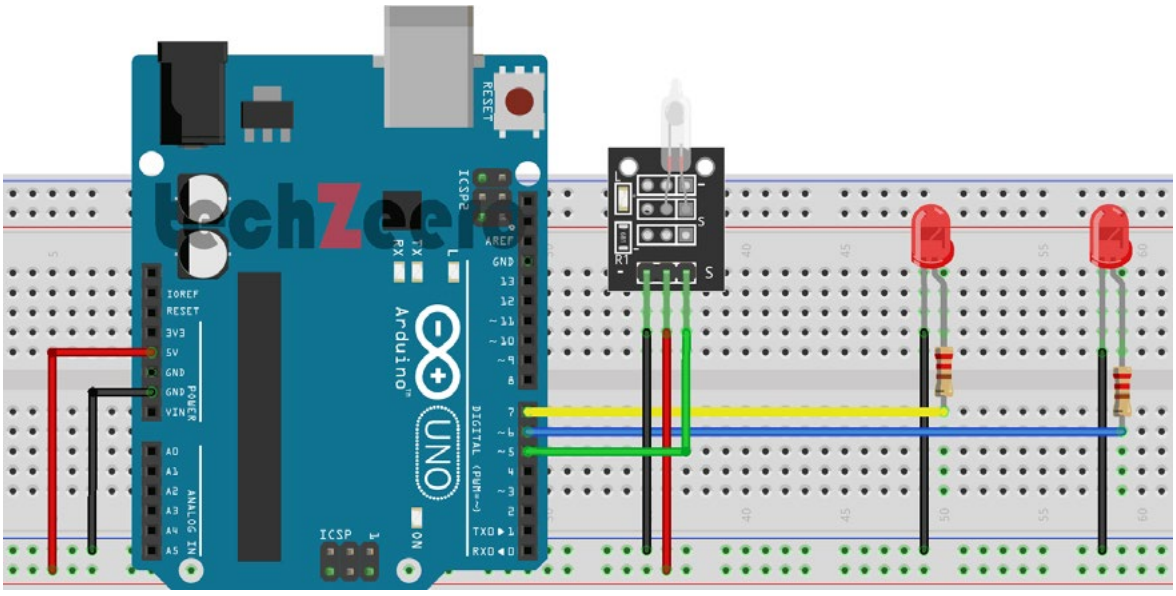
Задатак

Направити склоп за укључивање и искључивање свијетлећих диода зависно од положаја куглице унутар tilt сензора. Нека оба свјетла буду укључена када се куглица налази у положају равнотеже. Када је положај tilt сензора усправно, укључује се једно свјетло, а када је хоризонтално, укључује се друго свјетло.

Потребне компоненте:

- Arduino UNO плочица и USB
- Матадор плочица
- Сензор нагиба
- 1 x црвена LED
- 1 x зелена LED
- 2 x отпорник 220 Ω

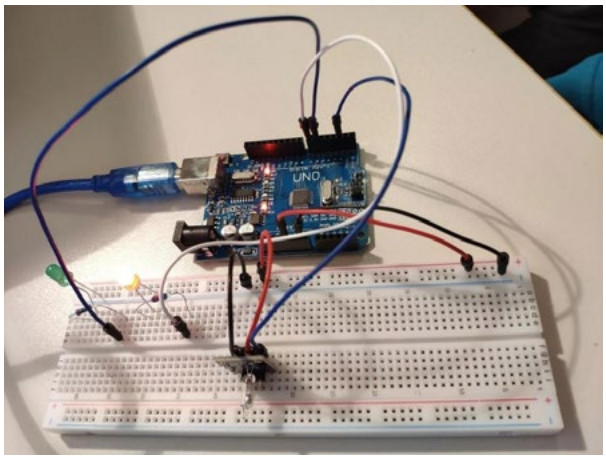
Приказ спајања

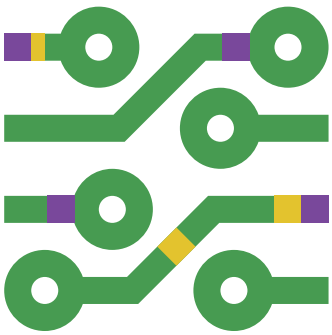


Рјешење:

```
  int tiltSensorPin = 2;            //Пин за сензор. nagiba
int notTiltLED = 7;
int TiltLED = 6;

void setup()

{
    pinMode(tiltSensorPin, INPUT);

    digitalWrite(tiltSensorPin, HIGH);

    pinMode(notTiltLED, OUTPUT);
    pinMode(TiltLED, OUTPUT);
}
void loop()
{
  if(digitalRead(tiltSensorPin))
  {

    digitalWrite(notTiltLED, HIGH);
    digitalWrite(TiltLED, LOW);
  }
  else
  {
    digitalWrite(notTiltLED, LOW);
    digitalWrite(TiltLED, HIGH);
  }
}
```

Закључак:





2019.