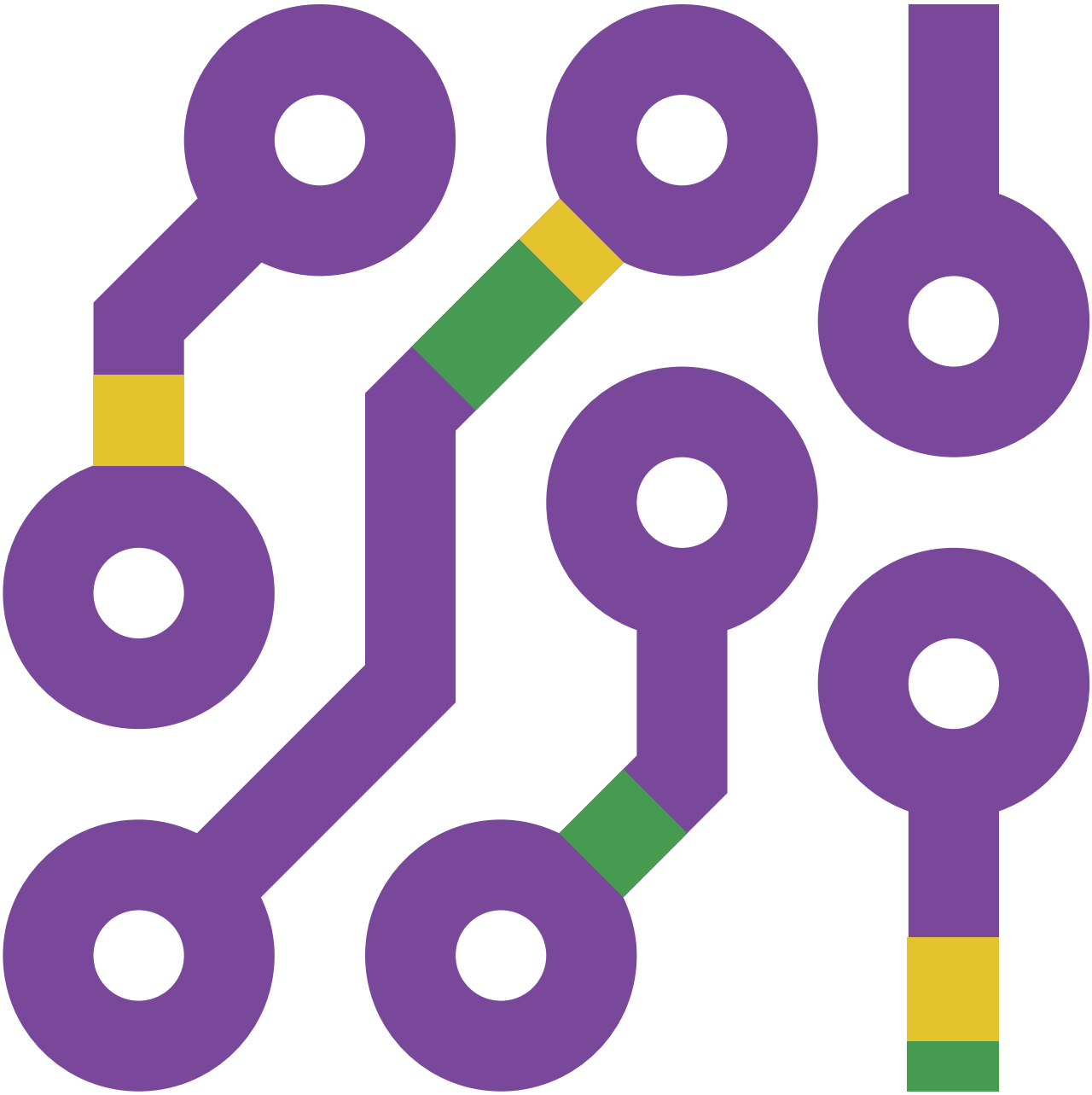


ARDUINO ЗА СВЕ

ПРИРУЧНИК ЗА НАСТАВНИКЕ

Средња школа

Мр сци. Муамер Халиловић,
дипл. инж. електротехнике

Мр сци. Ајла Халиловић,
дипл. математичарка и
информатичарка

IT Girls

Подржано од:
УЈЕДИЊЕНЕ НАЦИЈЕ
БОСНА И ХЕРЦЕГОВИНА

ARDUINO ЗА СВЕ

ПРИРУЧНИК ЗА НАСТАВНИКЕ

Средња школа

Мр сци. Муамер Халиловић, дипл. инж. електротехнике

Мр сци. Ајла Халиловић, дипл. математичарка и информатичарка

2021.

Садржај

Увод

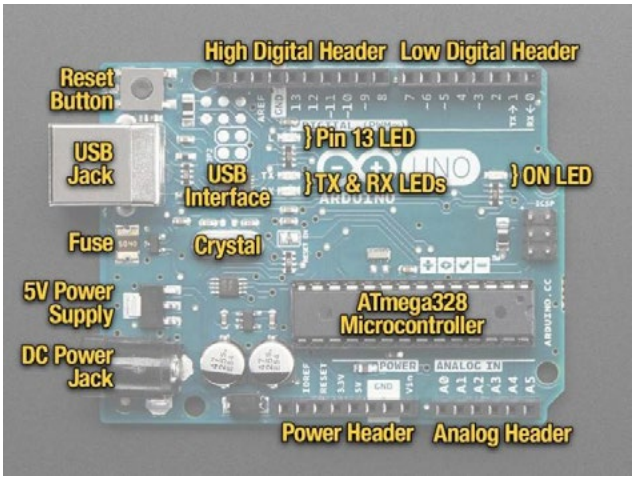
Наши животи се врте око мегабајта, мегахерца, чипова, процесора, AI, IoT, робота, ма једноставно читава збрка појмова и, признајете, некада се правимо да их разумијемо, а некада помислимо како би било баш добро користити и стварати, једноставно бити у том свијету, неки би рекли схватити матрицу (оп. а. MATRIX). Већина појмова који ће се овдје спомињати и нису тако непознати, али је неопходно направити увод у свијет кроз који ћемо заједно крочити с Arduin-ом.

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.

Arduino платформа

Упознајмо прво Arduino платформу:



- → 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 V DC;
- → 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-а;
- → TX & RX LED – диоде које нам омогућавају да видимо да постоји проток података – сигнала између PC-а и микроконтролера и обратно;
- → Crystal – електрични осцилатор који генерише сигнал тачне фреквенције – цлоцк микрорачунара.

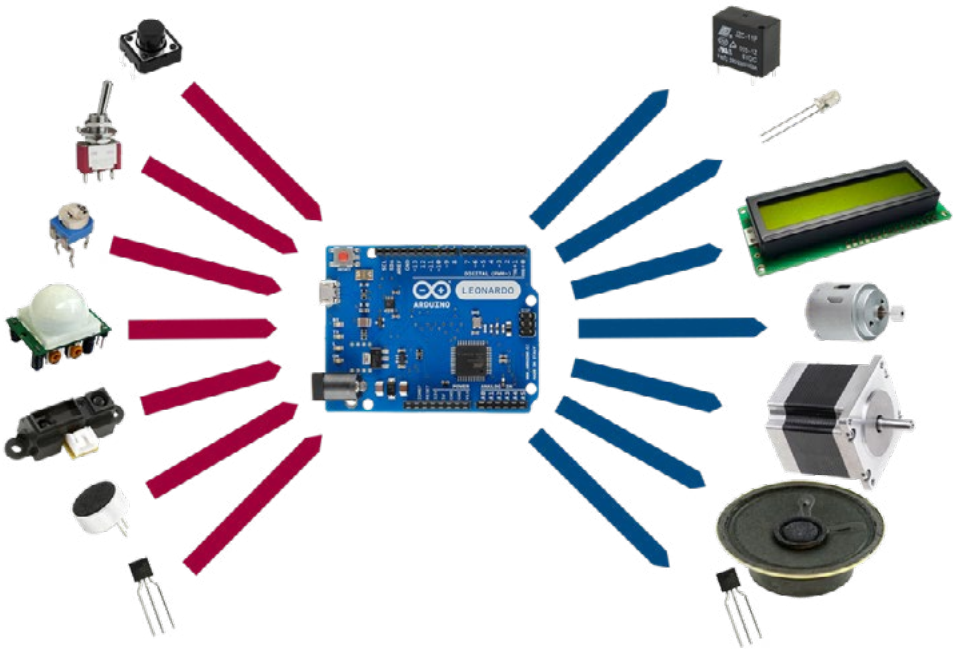
За рад с овом платформом није неопходно велико предзнање из електротехнике, мада се одмах тако и не чини.

С укупно 14 дигиталних улазно/излазних пинова и 6 аналогних 10-битних улаза, Arduino Uno или нека друга Arduino плоча сасвим је довољна да креирамо врло корисне пројекте.

Сензори и актуатори

Компоненте које вежемо за платформу можемо подијелити у двије групе: сензоре и актуаторе.

Ми свакодневно имамо интеракцију са сензорима и актуаторима, можда је једноставније када кажемо улазним и излазним уређајима.



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

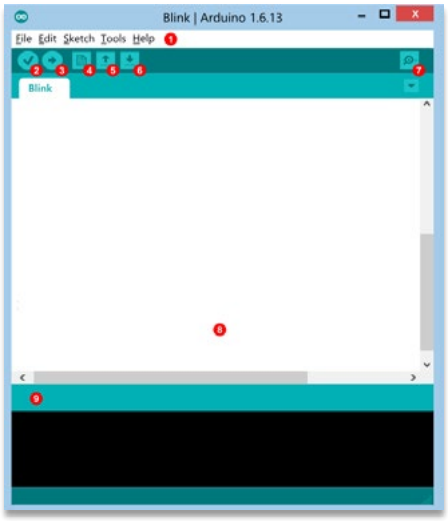
Arduino IDE Software

Упознали смо се с платформом и компонентама, преостаје нам да инсталирамо Arduino IDE Software. 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/software

Arduino software стално се ревидира. У тренутку писања овог приручника актуелна је верзија 1.8.15, али је врло могуће да, док овај приручник дође до крајњих корисника, новија верзија софтвера буде доступна за инсталацију. Потребно је одабрати инсталацију према понуђеним опцијама и проћи кроз краке инсталације.

Прије него што почнемо с програмирањем, погледајмо Arduino IDE окружење.



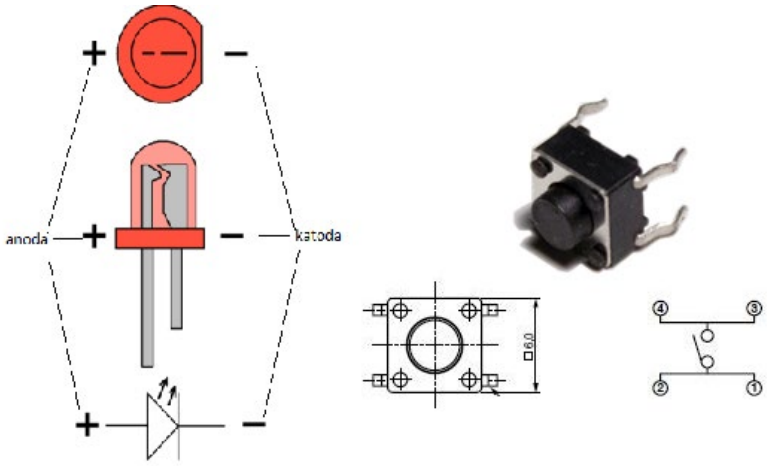
1. Menu: Мени софтверских опција
2. Verify: Компајлирање и провјера кода
3. Upload: Учитавање „кода“ у Arduino
4. New: Нова Arduino скица
5. Open: Отвори постојећу Arduino скицу
6. Save: Сачувај тренутно активну скицу
7. Monitor: Отвори апликацију за слање и примање информација
8. Editor: Простор за писање кода
9. Message: IDE простор за извјештаје о грешкама и инфо.

Користећи USB кабл Arduino спајамо на слободни USB порт. Добра рутина је у Windows Device Manager-у провјерити којем COM (комуникационом интерфејсу) порту је придружен Arduino. (Halilović, 2019).

За почетак смо припремили једноставније вјежбе које ћете моћи даље да развијате и примјењујете кроз различите пројекте.

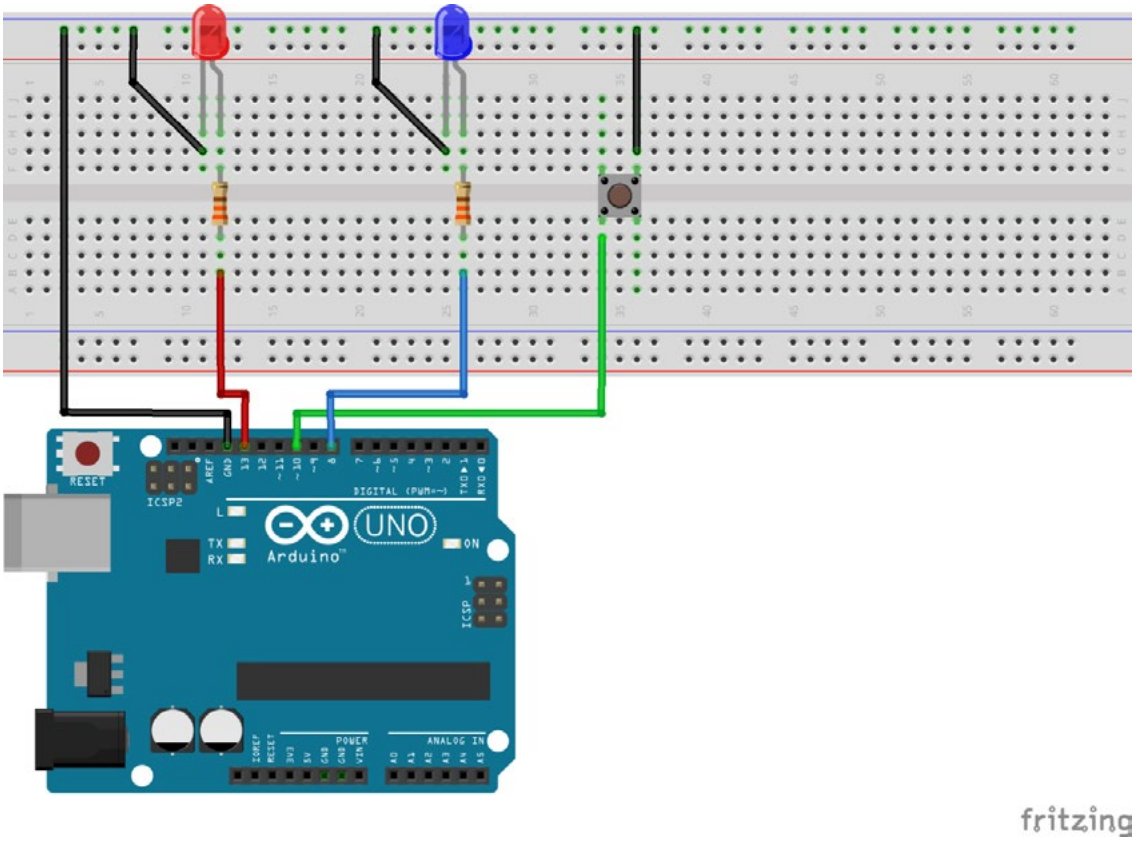
Вјежба 1. Топло и хладно!

За оне који су се одважили и први пут крећу у програмирање Arduina ова апликација је идеална као прва вјежба: На Arduino треба спојити двије LE диоде и један тастер. Када се тастер притисне, треба да свијетли црвена, а иначе је укључена плава LE диода.



Слика 1.1. LE диода и тастер - изглед и симбол

1. Потребни елементи за реализацију вјежбе:
2. Arduino UNO плочица и USB 1 ком
3. Матадор плочица 1 ком
4. LE диоде 2 ком
5. Отпорник 220 Ω 2 ком
6. Тастер 1 ком
7. Каблићи



Слика 1.2. Шема споја

Arduino код:

Све Arduino скице садрже двије основне фунцкије: setup и loop.

Setup функција је функција у којој се према слиједу извршавају програмске инструкције, али само једном у току извршавања апликације. Дакле, приликом сваког покретања апликације код у setup функцији изврши се само једном. Само име функције каже да се унутар ње ради подешавање микроконтролера, дефинише се који ће пинови бити улазни, а који излазни, подешавају се или активирају неке специјалне функције самог микроконтролера.

У loop() функцији налазе се све наредбе које Arduino треба да изврши. За разлику од setup() наредбе, loop() наредба извршава се бесконачно много пута, докле год је Arduino укључен. Loop се примјењује након дефинисања почетних вриједности у функцији setup. (Vuković, 2017).

У наредном примјеру упознаћемо се с неколико основних фунцкија у Arduino скицама.

Унутар ***setup*** дијела функције имамо позвану функцију:

```
pinMode(LedCrvena, OUTPUT);
```

Функција има два аргумента. Први подразумијева број пина којем желимо дати специфичну функцију, а с другим аргументом дефинишемо да ли ће неки пин бити INPUT (на пин спојен тастер, дигитални сензор или прекидач), OUTPUT (на пин спојен LED, релеј или транзистор за укључење већих потрошача типа мотора, гријача итд.).

Унутар ***loop*** дијела функције имамо позване двије функције:

```
digitalRead(Taster);
```

```
digitalWrite(LedCrvena, HIGH);
```

Прва функција прати промјену стања на пину који је проглашен INPUT-ом и, ако је „тастер" активиран, функција враћа „1", у супротном фунцкија враћа вриједност „0". Функција нам омогућава детекцију промјене стања на пину који је проглашен дигиталним улазом.

Друга функција има два аргумента. Првим дефинишемо пин којем желимо мијењати напонско стање, а другим вриједност на том пину. Како је ријеч о дигиталним излазима који могу да имају два стања, укључено или искључено, аргумент HIGH значи „укључи" пин, а LOW „искључи" пин. Ако говоримо о напону HIGH, значи 5 V, а LOW 0 V.

```
int LedCrvena = 13;
int LedPlava = 8;
int Taster = 10;

int StanjeTastera;

void setup() {
  pinMode(LedCrvena, OUTPUT);
  pinMode(LedPlava, OUTPUT);
  pinMode(Taster, INPUT_PULLUP);
}

void loop() {
  StanjeTastera = digitalRead(Taster);
  if (StanjeTastera == LOW) {
    digitalWrite(LedCrvena, HIGH);
    digitalWrite(LedPlava, LOW);
  } else {
    digitalWrite(LedCrvena, LOW);
    digitalWrite(LedPlava, HIGH);
  }
}
```

Ова вјежба може се и модификовати, надопунити и проширити уз примјену неких других сензора и актуатора како бисмо направили сложенији пројекат. На примјер, уколико користимо сензор звука и ултразвучни сензор, можемо да направимо алармни систем.

Дигитални и аналогни сензори

У Arduino сету имамо различите сензоре које можемо да разврстамо у двије групе према сигналу који дају на излазу.

Дигитални сензори на излазу дају сигнал који може да поприми само одређене вриједности амплитуде и доступан је у само одређеним временским интервалима. Дигитални сензори имају слово Д које се налази исписано на компонентама. Неки од дигиталних сензора су:

- Digital Push Button – тастер
- Touch Sensor – сензор додира
- Tilt Sensor – сензор нагиба
- Vibration Sensor – сензор вибрација
- Magnetic Sensor – магнетни сензор.

У претходној вјежби користили смо дигитални сензор – тастер. На исти начин можете да изаберете неки други дигитални сензор и да тестирате рад апликације.

Аналогни сензори на излазу дају величину која је континуирана у времену и по амплитуди. Аналогни сензори имају слово А које се налази исписано на компонентама. У аналогне сензоре спадају:

- Ultrasonic Sensor – ултразвучни сензор
- Rotation Sensor – потенциометар
- Moisture Sensor – сензор влажности
- Steam Sensor – сензор присуства паре
- Sound Sensor – сензор звука
- Ambient Light Sensor – сензор амбијенталног осветљења
- Grayscale Sensor – свјетлосни сензор
- Piezo Disk Vibration Sensor – сензор вибрације.

Вјежба 2. Влажност тла

Креирати апликацију коју ћемо користити за мјерење нивоа влажности тла. У овој вјежби користићемо једну LE диоду која ће се укључити уколико је неопходно повећати влажност тла.

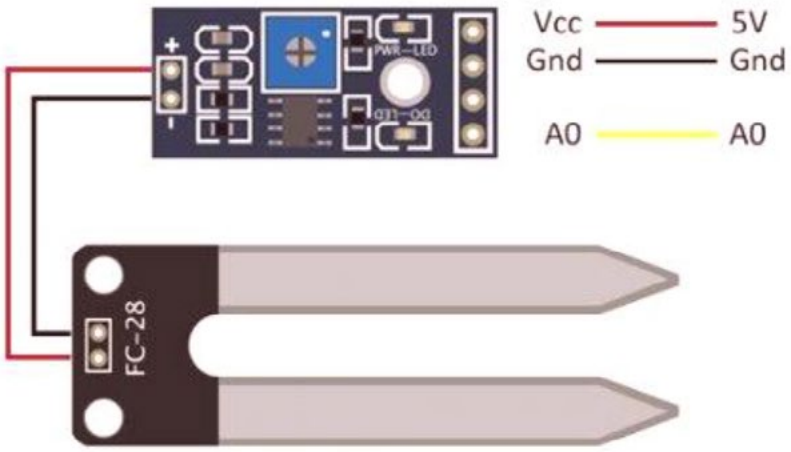
Уколико имамо собно биље, онда ову апликацију можемо да примијенимо у периоду када смо одсутни. Неће нам требати услуга сусједа за заливањем цвијећа. Помоћу сензора за влажност можемо да направимо аутоматски систем наводњавања. А уз овај сензор може да се направи много различитих и корисних пројеката.

Сензор влажности има двије сонде које проводе струју кроз земљу. Ако је тло влажно, тада је отпор између сонди мањи. У сухом земљишту отпор је већи. Arduino прихвата ове вриједности, упоређује их и, ако је потребно, укључује. На примјер, пумпу. Сензор се може користити на начин да се споји на аналогни улаз.

За повезивање преко аналогног улаза користимо A0 улаз. Сензор влаге у тлу Arduino узима вриједности од 0 до 1023. У вјежби смо поставили услов (`StanjeSenzora <400)` за укључивање црвене LE диоде, а то може да буде промјењиво у зависности од тога која је влажност земљишта одговарајућа за одређену биљку.

Сензор се спаја на сљедећи начин:

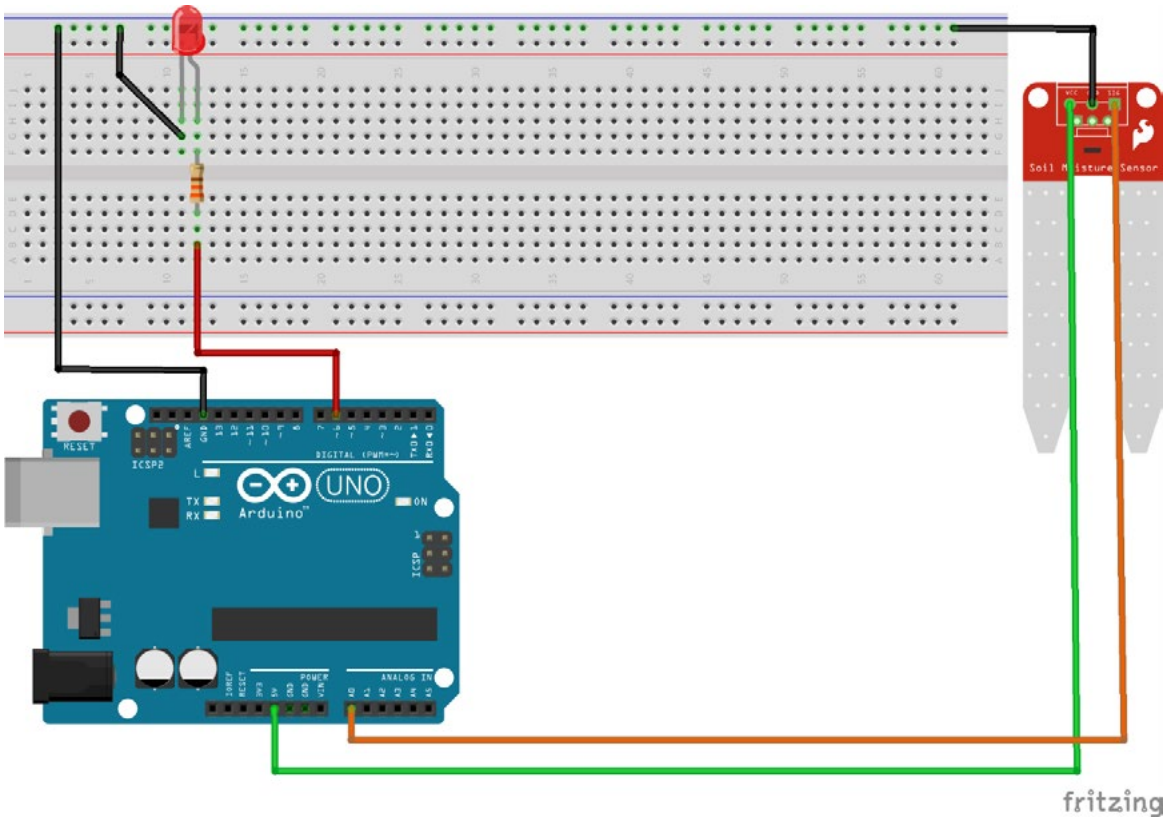
- Vcc се повезује с 5V на Arduino
- Gnd на сензор на Gnd на Arduino плочи
- A0 се повезује с A0 на Arduin-у.



Слика 2.1. Moisture Sensor – сензор влажности

Потребни елементи за реализацију вјежбе:

- Arduino UNO плочица и USB 1 ком
- Матадор плочица 1 ком
- LE диоде 1 ком
- Отпорник 220 Ω 1 ком
- Сензор за влажност 1 ком
- Каблићи



Слика 2.2. Шема споја

Arduino код:

Унутар ***setup*** дијела функције имамо позвану функцију:

Serial.begin(9600);

Serial.begin(9600) користимо за подешавање микроконтролера да активира серијску комуникацију на брзини 9600 bit/sec. Серијска комуникација омогућава нам да кроз само два вода означена с PX (вод за примање података) и TX (вод за слање података) можемо послати или примити комплексне информације тима string-a. Осим тога, активирањем и подешавањем серијске комуникације добијамо опцију тзв. софтверског debugging-a. На примјер:

Serial.println („Сада сам у дијелу кода за одлучивање!”);

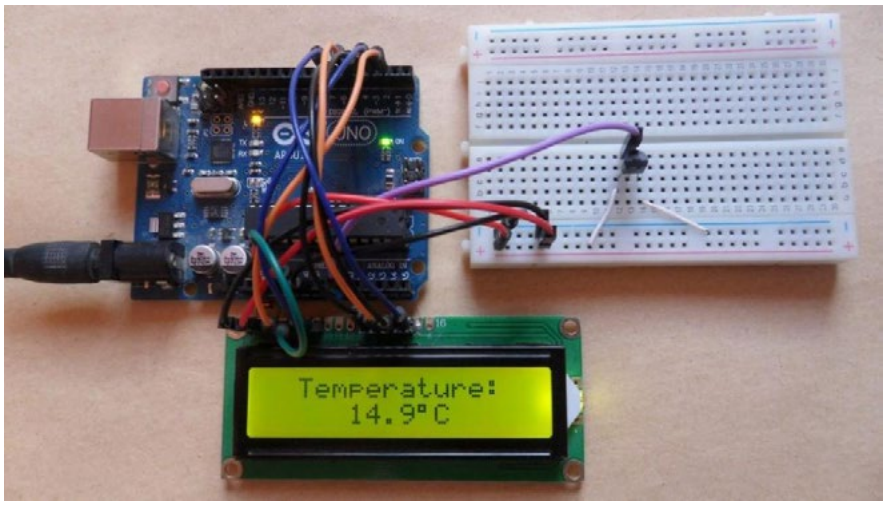
```
int StanjeSenzora;
int LedCrvena = 6;

void setup() {
  Serial.begin(9600);
  pinMode(LedCrvena, OUTPUT);
}

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

  if (StanjeSenzora <400) {
    digitalWrite(LedCrvena, HIGH);
  } else {
    digitalWrite(LedCrvena, LOW);
  }
  Serial.println(StanjeSenzora);
  delay(100);
}
```

ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца:	**Ученик/ца:**
- описује улогу електронских компоненти које су коришћене у вјежбама	- дефинише основну улогу дигиталних и аналогних сензора - описује примјену LE диоде у апликацијама - користи аналогне и дигиталне сензоре у апликацијама - описује улогу сензора за влажност тла - користи тастере у апликацијама
- користи електричну шему споја за повезивање Arduin-a и електронских компоненти	- повезује електронске компоненте на Arduino према шеми споја - препознаје пинове на коришћеним компонентама како би их исправно повезао
- примјењује стечено знање из програмирања за израду пројекта	- креира програм за дати задатак уз адекватне смјернице наставника/це - по потреби модификује креиране кодове
- верификује и извршава програм	- анализира програмски код - извршава отклањање грешке по потреби и према смјерницама наставника/це
- тестира пројектни задатак	- учитава код на Arduino платформу - анализира пројектни задатак - развија сопствене идеје за примјену аналогних и дигиталних сензора који су коришћени у вјежбама

С обзиром на то да смо се упознали с основама Arduin-a, спремни смо за креирање и сложенијих пројеката.

Пројекат број 1.
LM35

Као први озбиљнији пројекат који може да нам послужи у много корисних апликација јесте мјерење температуре помоћу аналогног сензора темературе LM35. Аналогни улази на Arduino развојној платформи су 10-битне резолуције. То нам говори да су вриједности од 0-1023 (2^10 = 1024), које може дати било који сензор када је спојен на аналогни улаз. Тај податак зовемо „сирови“ податак и ми људи га не разумијемо. Да бисмо добили разумљив податак, нпр. вриједност температуре, притиска или удаљености, потребно је користити математичке формуле које су специфичне за сваки сензор и добију се из описа производа произвођача (ор. а. datasheet). У овом конкретном пројекту потребно је реализовати једноставан дигитални термометар са сензором LM35 и приказом температуре на LCD дисплеју.



Слика ***1.1.*** Приказ температуре на ***LCD*** дисплеју

Можда вам тренутно пада на памет да мјерите температуру у просторији у којој боравите, али ево неколико идеја које можете да искористите за неки upgrade овог почетног пројекта:

1. мјерење температуре кућишта PC-а и PCB-а графичке картице
2. мјерење спољашње и унутрашње температуре пластеника
3. мјерење температуре воде.

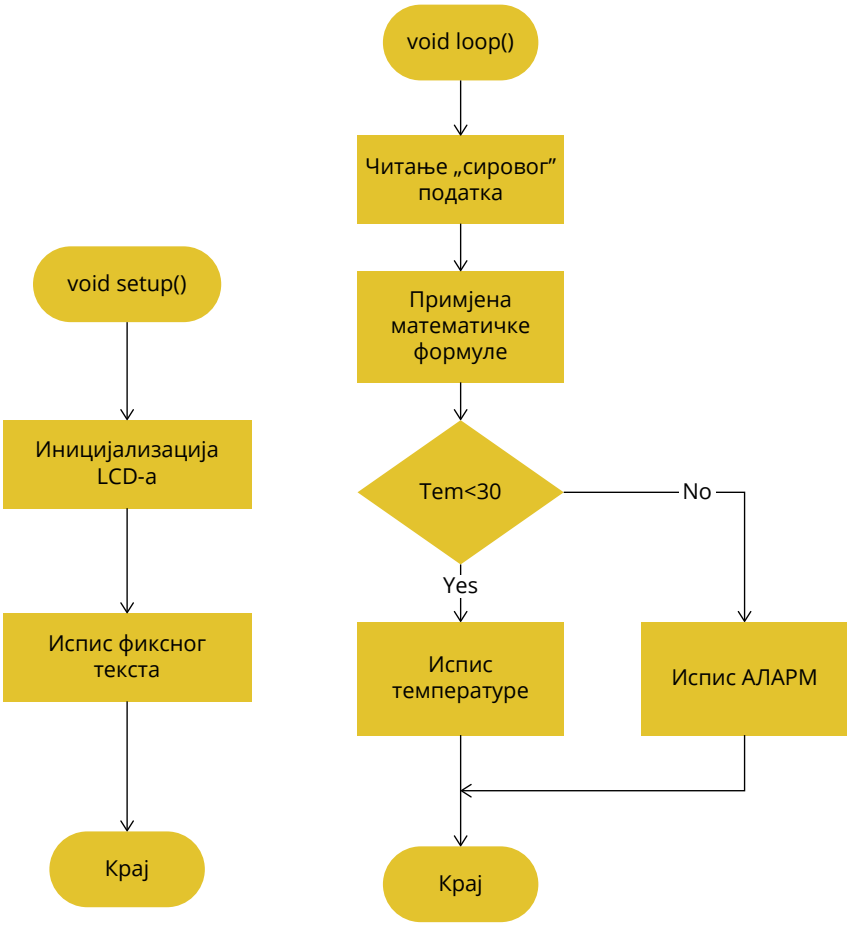
Сваки од поменутих приједлога можете да надоградите уводећи додатне компоненте. На примјер, осим мјерења температуре пластеника за узгој биљака неопходно је и мјерење влажности тла.

Када проширите своје знање, онда неће бити тешко на Arduino платформу спојити Bluetooth модул или, што да не, WiFi модул, те прикупљене податке слати на неки мобилни уређај или веб. Али више о овоме у неком од наредних пројеката.

Потребни елементи за реализацију пројекта су:

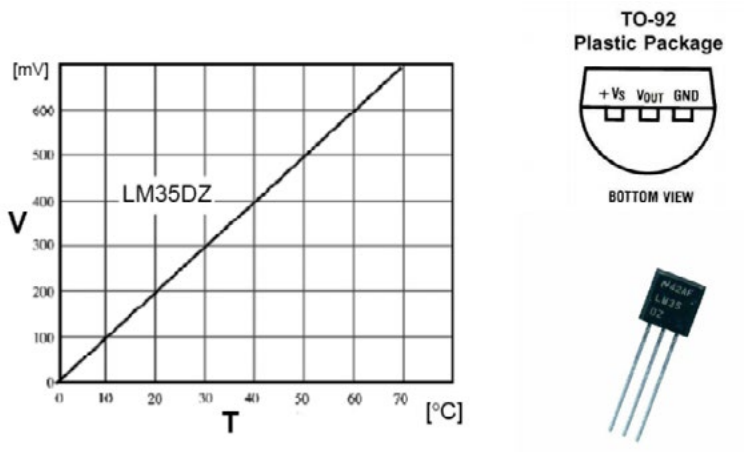
1. ARDUINO UNO
2. LM35 температурни сензор
3. 16 × 2 LCD дисплеј
4. 10 KΩ потенциометар
5. Матадор плоча
6. Каблићи
7. Отпорник 330 Ω

Било би сјајно када бисте покушали за сваки пројекат који замислите прво нацртати грубу алгоритамску структуру пројекта. У овом пројекту ми ћемо то урадити за вас, али надамо се да ћете стећи навику да, прије него почнете писати код, прво тај код покушате визуализирати и „нацртати“ га.



Слика ***1.2.*** Поједностављена алгоритамска структура апликације

Према каталошким подацима, сензор посједује осјетљивост од 10 mV/°C, а његова преносна функција је прилично линеарна и одступање се налази у границама од ±0,1°C.



Слика ***1.3.*** Карактеристика и ***pinout LM35***

Као што се види на претходној слици, сензор температуре LM35 има три пина (**V_S,** V_{out} и **GND**). Уколико бисте на крајеве сензора довели напон у границама од 3.3 до 30 V истосмјерно, а између крајева V_{out} и GND спојили волтметар, за измјерене вриједности напона могли бисте спрам карактеристике сензора закључити да сензор тренутно мјери сљедеће вриједности температуре:

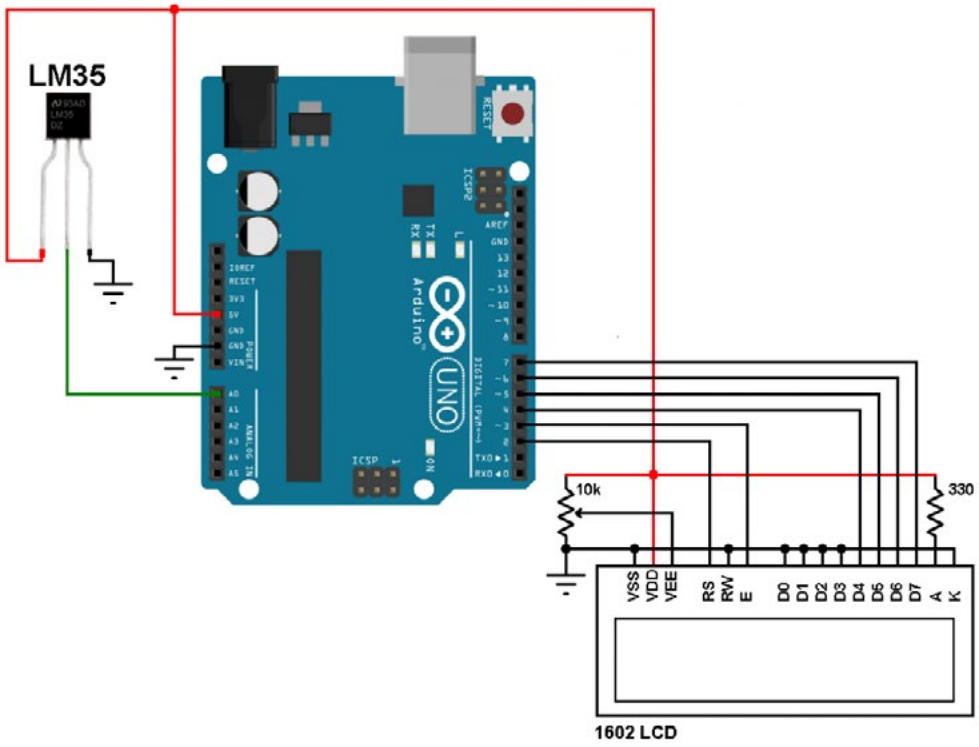
1. за излазни напон: 10 mV —> температура = 1° C
2. за излазни напон: 100 mV —> температура = 10° C
3. за излазни напон: 200 mV —> температура = 20° C
4. за излазни напон: 370 mV —> температура = 37° C

Оно што нам је као програмерима/кама најважније јесте математичка формула помоћу које можемо добити нама разумљив податак:

*T = (Vcc * ADC * 100.0) / 1024;*

Celsius = Kelvin - 273.15

Celsius = 5/9 x (Fahrenheit -32)



Слика ***1.4.*** Шема споја

Arduino код:

LyquidCrystal.h библиотеку позивамо у свој пројекат код програмирања апликација с LCD дисплејом. LCD је базиран на Hitachi HD44780 chipset-y. Библиотека садржи скуп функција чије коришћење олакшава подешавање LCD-а те слање и испис варијабли на њему.

```
#include <LiquideCrystal.h>
```

Укратко ћемо појаснити неке од функција из LyquidCrystal библиотеке.

```
const int rs = 2, en = 3, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
```

```
lcd(2, 3, 4, 5, 6, 7);
```

Аргументи функције LCD дефинишу како ће пинови LCD-а бити спојени на Arduino развојну плочу. LCD ради у тзв. 4-битном моду и пинови с индексом „д" служе за слање команди и података ка дисплеју. Пинови RS и EN служе за контролу дисплеја. Постоји још један пин са ознаком R/W, који се спаја на GND и у том моду LCD може само да прима податке, односно LCD је у тзв. read моду.

```
lcd.begin(16,2);
```

Функција lcd.begin служи за иницијализацију LCD -а, а прослијеђени аргументи дефинишу димензије дисплеја који се користи.

```
lcd.setCursor(3,1);
```

Функција lcd.setCursor поставља курсор на позицију прослијеђених аргумената, на примјеру изнад. Курсор на LCD-у биће подешен на 3. ред и 1. колону.

lcd.print(„Здраво, свијете”);

Функција lcd.print прослеђује податке које би требало исписати на LCD-у. Уколико подаци долазе у формату с двоструким апострофом ”ja sam string”, ријеч је о константном string-у. Могуће је послати на испис и варијабле као у примјеру за мјерење темепратуре наведеном у наставку.

```
// Примјер апликације
// Include LCD библиотеке
#include <LiquidCrystal.h>
// LM35 out pin је спојен на A0 pin Arduin-a
#define  LM35_pin  0
// Декларација глобалне варијабле
float temp;

// Креирање LCD објекта
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup() {
    // Иницијализација LCD и подешавање курсора
    lcd.begin(16, 2);
    lcd.setCursor(2, 0);
    // Принт фуксног текста
    lcd.print("Temperatura:");
     analogReference(INTERNAL);// За референтни напон користимо интерни
реф. напон на ADC-у
}

void loop() {
   // Аналогну вриједност напона претвара у °C ( 9.3 = 1023/(1.1*100))
   temp = analogRead(LM35_pin) / 9.3;
   lcd.setCursor(3, 1);
   lcd.print(temp);
   delay(1000);  // 1s за поновно учитавање
}
```

Овај дио кода не треба да посматрате као завршен пројекат, већ треба да га покушате надоградити већ предложеним идејама или да реализујете неке своје.

ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца:	**Ученик/ца:**
- описује улогу електронских компоненти које су коришћене у пројекту	- дефинише карактеристике сензора температуре LM35 - описује улогу 16 × 2 LCD дисплеја
- користи електричну шему споја за повезивање Arduin-а и електронских компоненти	- повезује 16 × 2 LCD дисплеј на Arduino према шеми споја - повезује сензор температуре LM35 на Arduino према шеми споја
- креира алгоритамску структуру споја	- процјењује предности и ограничења алгоритамског приступа у рјешавању проблема - визуализира пројектни задатак кроз алгоритамску шему
- креира програмски код за дигитални термометар са сензором LM35 и приказом температуре на LCD дисплеју	- примјењује знање о употреби функција из програмирања за реализацију задатака - користи математичке формуле које су специфичне за сензор - креира програм за дати задатак уз адекватне смјернице наставника/це - реконструише програмски код према сопственим идејама
- користи одговарајуће библиотеке унутар програма за реализацију пројектног задатка -	- проналази одговарајуће библиотеке за реализацију датог пројектног задатка - инсталира одговарајуће библиотеке - позива унутар програмског кода адекватну библиотеку
- верификује и извршава програм	- анализира програмски код - отклања грешке по потреби и према смјерницама наставника/це
- тестира пројектні задатак	- учитава код на Arduino платформу - мјери температуру помоћу аналогног сензора LM35 - учитава температуру са 16 × 2 LCD дисплеја - анализира пројектни задатак - допуњује пројектни задатак својим идејама

Простор за вјежбу:

```
 void setup() {

}
void loop() {

}
```

Пројекат број 2.
HC-SR04

Надам се да сви знамо како шишмиш (слијепи миш) лети у скученом простору, а не удара у препреке или како подморница може да детектује другу подморницу? Одговор је сонар (према енгл. Sound Navigation and Ranging: навигација и одређивање удаљености помоћу звука). Поједностављено, у оба случаја извор емитује звук и мјери вријеме које је прошло док се тај звук врати. Наравно, и систем детекције и код шимиша и код подморнице јесте комплексан, али то нас неће спријечити да покушамо направити нешто, у најмању руку, слично.

VCC: Напон напајања (+5 V)
Trig: Trigger input pin
Echo: Echo output pin
GND: Ground (masa) (0 V)

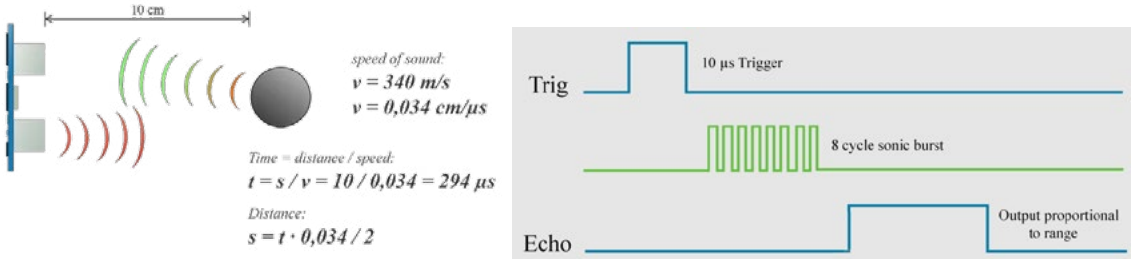
Слика ***2.3.*** Изглед и ***pinout HC-R04***

HC-SR04 – Ултразвучни модул је једноставан сензор који емитује ултразвучни талас при 40 000 Hz. Овај талас путује ваздухом те, ако на његовом путу постоји неки објекат или препрека, одбиће се до модула. Узимајући у обзир вријеме путовања и брзину звука, можемо израчунати удаљеност.

HC-SR04 посједује 4 пина: GND, Vcc, Trig и Echo. Пинови GND и Vcc треба да се повежу на Arduino пинове GND и 5V Arduino плоче, респективно. Пинови Trig и Echo се повезују на било који дигитални I/O пин Arduino плоче.

Како би генерисали ултразвучни талас и почели с прорачуном удаљености, потребно је **Trig** пин подићи у стање логичке јединице на период од 10 µs у виду импулса.Уколико се испред сензора налази објекат, **Echo** пин ће отићи у стање логичке јединице, и у зависности од удаљености предмета трајање стања логичке јединице ће варирати. Echo пин даје као излаз вријеме путовања звука у микросекундама (µs).

На примјер, ако је објекат удаљен 10 cm од сензора, знајући да је брзина звука 340 m/s (или 0.034 cm/µs), звучни талас прелазиће пут око 294 µs. Ово је вријеме за које талас пређе пут од извора (сензора) до препреке и врати се до извора (погледати слику испод).



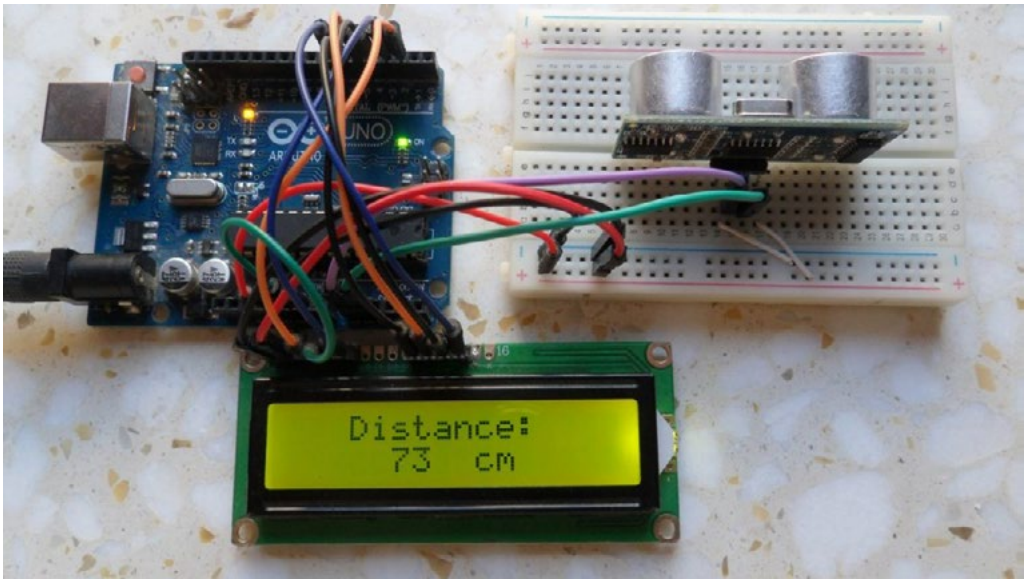
Извор: https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

Тако да, ако желимо добити тачну удаљеност у центиметрима (cm), морамо помножити измјерено вријеме на Echo пину с брзином (вриједност 0.034) и подијелити све с 2. Односно, можемо то изразити и као удаљеност = вријеме / (2/0.034) = вријеме / 58.8 ≈ вријеме / 58.

Као и у претходном пројекту, и овог пута можемо да користимо формулу за прорачун коју је дао произвођач сензора, а помоћу које се вријеме трајања импулса претвори у удаљеност:

Удаљеност у **cm = (Трајање HIGH стања Echo пина)/58**

Уколико се испред сензора не налази препрека, на Echo пину генерисаће се импулс трајања 38 ms, у супротном имаћемо импулсе трајања (125 µs – 25 ms) у зависности од удаљености.



Слика ***2.1.*** Приказ удаљености на ***LCD*** дисплеју

Можда вам тренутно пада на памет да мјерите удаљеност неког предмета на столу на којем се налази сензор, али ево неколико идеја које можете да искористите за унапређење овог почетног пројекта:

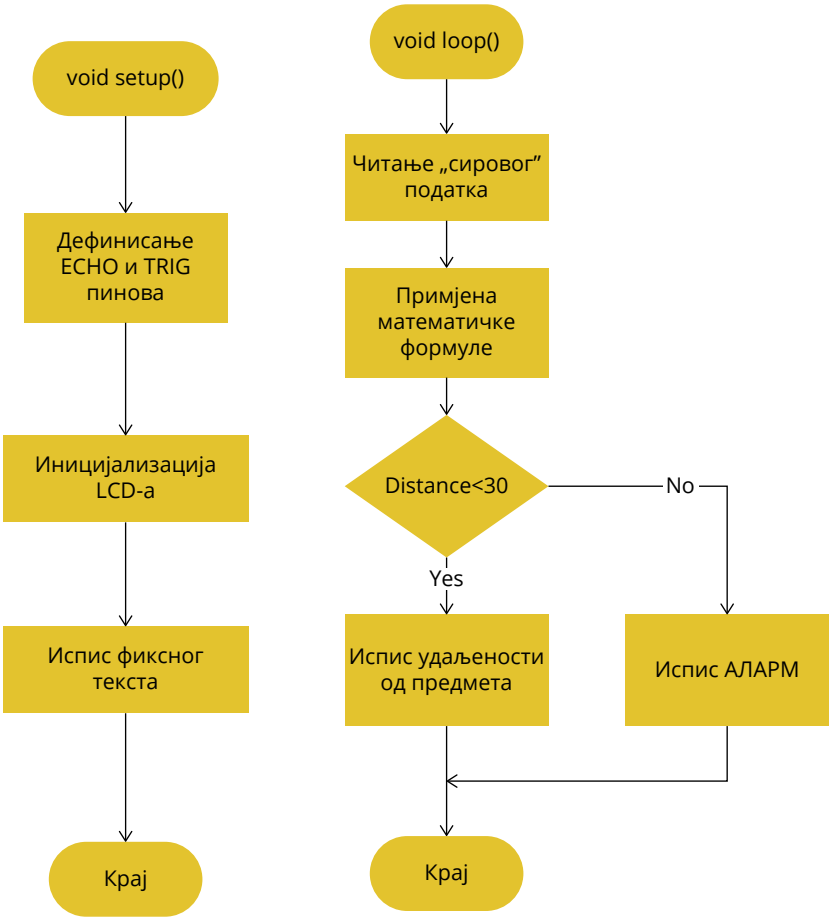
1. Мјерење висине ваших школских колега/ица
2. Користити га за мјерење скока у даљ на часу спорта
3. Паркинг-сензор за аутомобил ваших родитеља
4. Детекција прописно размакнутих предмета у простору

Када проширите своје знање, онда неће бити тешко на Arduino платформу спојити Bluetooth моду или, што да не, WiFi модул те прикупљене податке слати на неки мобилни уређај или веб. Али више о овоме у неком од наредних пројеката.

Потребни елементи за реализацију пројекта су:

1. ARDUINO UNO
2. HC-SR04 ултразвучни модул
3. 16 × 2 LCD дисплеј
4. 10 KΩ потенциометар
5. 330 Ω отпорник
6. Матадор плочица
7. Каблићи

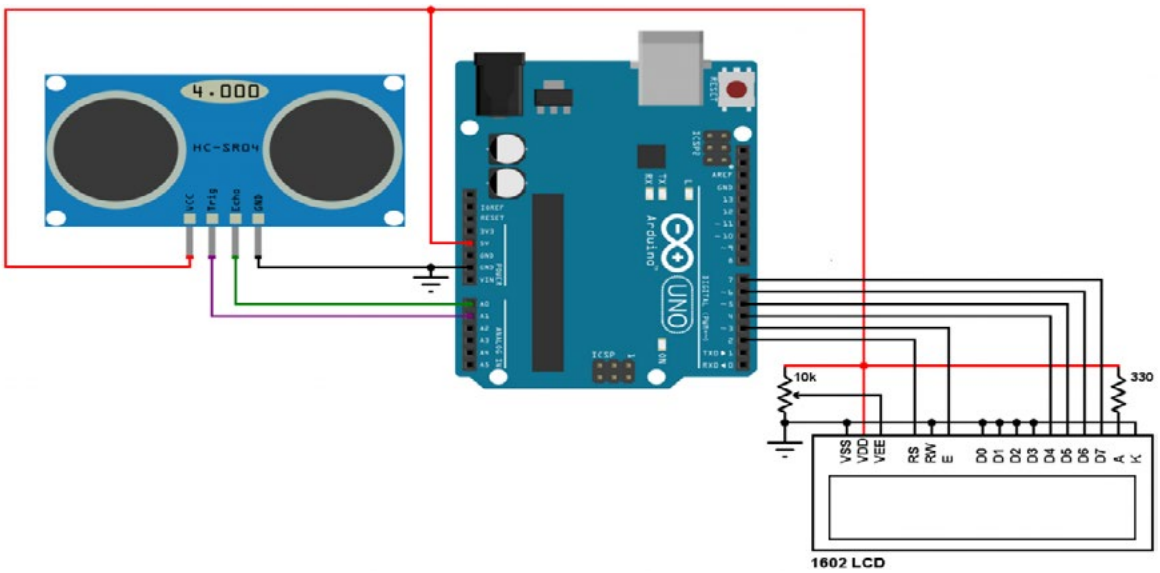
Било би сјајно када бисте покушали за сваки пројекат који замислите прво нацртати грубу алгоритамску структуру пројекта. У овом пројекту ми ћемо то урадити за вас, али надамо се да ћете стећи навику да, прије него почнете писати код, прво тај код пробате визуализовати и „нацртати“ га.



Слика 2.2. Поједностављена алгоритамска структура апликације

За добијање дужине трајања импулса користити ***pulsIn()*** функцију. То је функција која има два аргумента, пин на којем се у овом конкретном случају спаја Echo пин сензора, другим аргументом дефинишемо очекујемо ли HIGH или LOW сигнал на пину. На сљедећем примјеру можемо да уочимо да је пин 7, пин на којем се очекује промјена стања. Конкретно нас занима и вријеме које је потребно да се појави HIGH сигнал.

```
int pin = 7;
unsigned long duration;

void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}

void loop() {
  duration = pulseIn(pin, HIGH);
  Serial.println(duration);
}
```



Слика 2.4. Шема споја

Arduino код:

```
// Мјерење удаљености с Arduino UNO и HC-SR04
// ултразвучним сензором

// Укључујемо LCD библиотеку
#include <LiquidCrystal.h>
#define  trigger_pin  15   // HC-SR04 trigger пин је повезан на Arduino A1
#define  echo_pin     14   // HC-SR04 echo пин је повезан на Arduino A0


// Декларација глобалних варијабли
unsigned long duration;
unsigned int  distance;

// LCD објекат
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup() {
  pinMode(trigger_pin, OUTPUT);
  pinMode(echo_pin, INPUT);
  digitalWrite(trigger_pin, LOW);
  // inicijalizacija LCD-a
  lcd.begin(16, 2);
  lcd.setCursor(3, 0);
  lcd.print("Distance:");
   }
void loop() {

    // Пошаљи пулс у трајању од 10us на HC-SR04 Trig пин
    digitalWrite(trigger_pin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigger_pin, LOW);
```

```
  // Прочитај пулс који стиже на Echo пин
     duration = pulseIn(echo_pin, HIGH);              // Прочитај ширину
пулса с Echo пина
   if(duration == 0) {
// Претходна функција враћа вриједност нула у случају да је истекло
вријеме
     lcd.setCursor(2, 1);
       lcd.print("  Time Out  ");
     }
     else {
       distance = duration / 58;    // Израчунај удаљеност у cm
        if(distance > 400) {          // HC-SR04 модул мјери удаљеност и
до  400 cm
         lcd.setCursor(2, 1);
          lcd.print("Out of Range");
      }
       else {
         lcd.setCursor(2, 1);
         lcd.print("        cm    ");
               lcd.setCursor(5, 1);
        lcd.print(distance);
      }
     }
     delay(100);                     // Сачекај 100 ms између очитања
 }
}
```

Овај дио кода не треба да посматрате као завршен пројекат, већ треба да га покушате надоградити већ предложеним идејама или да реализујете неке своје.

ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца:	**Ученик/ца:**
- описује улогу електронских компоненти које су коришћене у пројекту	- објашњава начин рада HC-SR04 сензора за удаљеност - наводи улогу пинова на ултразвучном сензору
- користи електричну шему споја за повезивање Arduin-а и електронских компоненти	- повезује HC-SR04 сензор на Arduino према шеми споја - повезује преостале компоненте у пројекту према шеми споја
- креира алгоритамску структуру споја	- процјењује предности и ограничења алгоритамског приступа у рјешавању проблема - визуализира пројектни задатак кроз алгоритамску шему
- креира програмски код за ултразвучни сензор	- примјењује формулу помоћу које се вријеме трајања импулса претвори у удаљеност - креира програм за дати задатак уз адекватне смјернице наставника/це - реконструише програмски код према сопственим идејама
- користи одговарајуће библиотеке унутар програма за реализацију пројектног задатка -	- по потреби инсталира одговарајуће библиотеке - позива унутар програмског кода адекватну библиотеку
- верификује и извршава програм	- анализира програмски код - отклања грешке уколико су настале у коду према смјерницама наставника/це по потреби
- тестира пројектни задатак	- учитава код на Arduino платформу - учитава удаљеност која се приказује на дисплеју - анализира пројектни задатак - допуњује пројектни задатак датим приједлозима или неким својим идејама

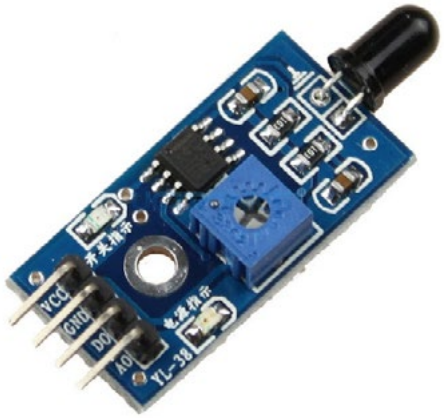
Простор за вјежбу:

```
 void setup() {

}
void loop() {

}
```

Пројекат број 3.
Flame Sensor

Један од врло корисних сензора у нашем сету је сензор за детекцију пламена - *Flame Sensor*. Користећи овај сензор у комбинацији с другим сензорима и компонентама можете да креирате веома комплексне пројекте. Свака установа треба да има сензоре за детекцију пламена, а ми то можемо да креирамо користећи Arduino.

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



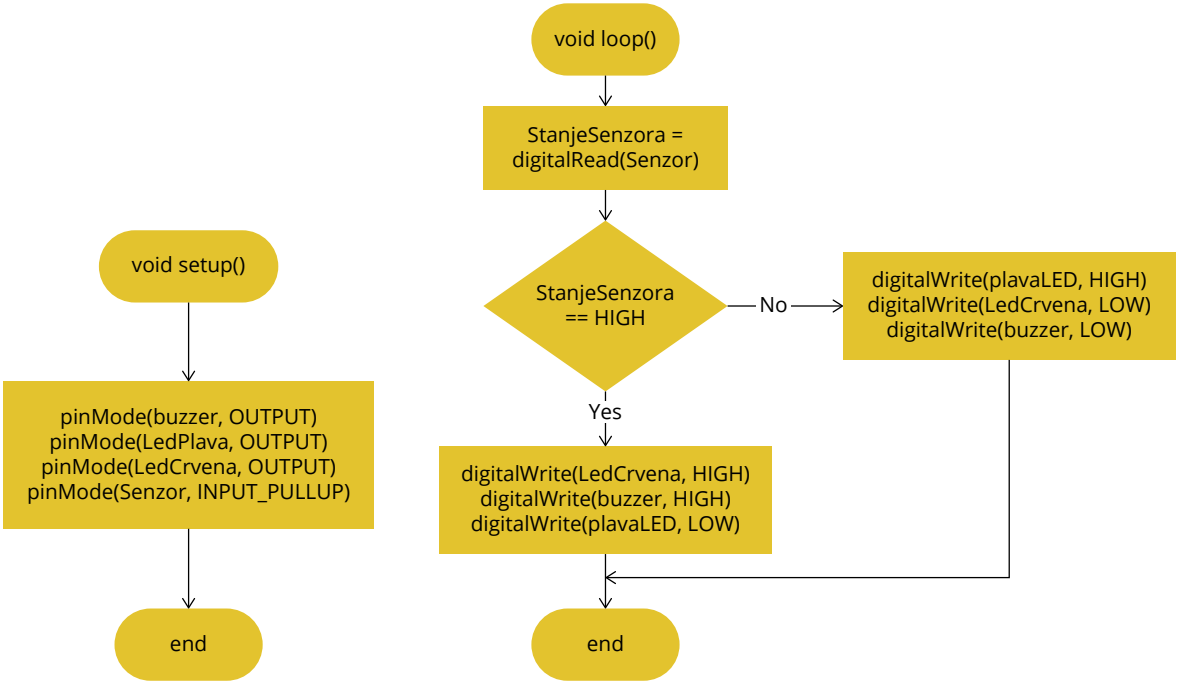
Слика ***3.1.*** **Приказ *Flame Sensor*-а**

Ова врста сензора користи се за детекцију пожара кратког домета и може се користити за праћење пројеката или као безбједносна мјера за искључивање/укључивање уређаја. Сензор пламена је врло осјетљив на инфрацрвену (IR) таласну дужину свјетлости (760 nm до 1100 nm). И овај сензор има аналогни и дигитални излаз. Уколико у пројекту користимо аналогни излаз, онда Vcc вежемо за пин 5 V, а ако користимо дигитални излаз, онда Vcc вежемо за пин од 3.3 V.

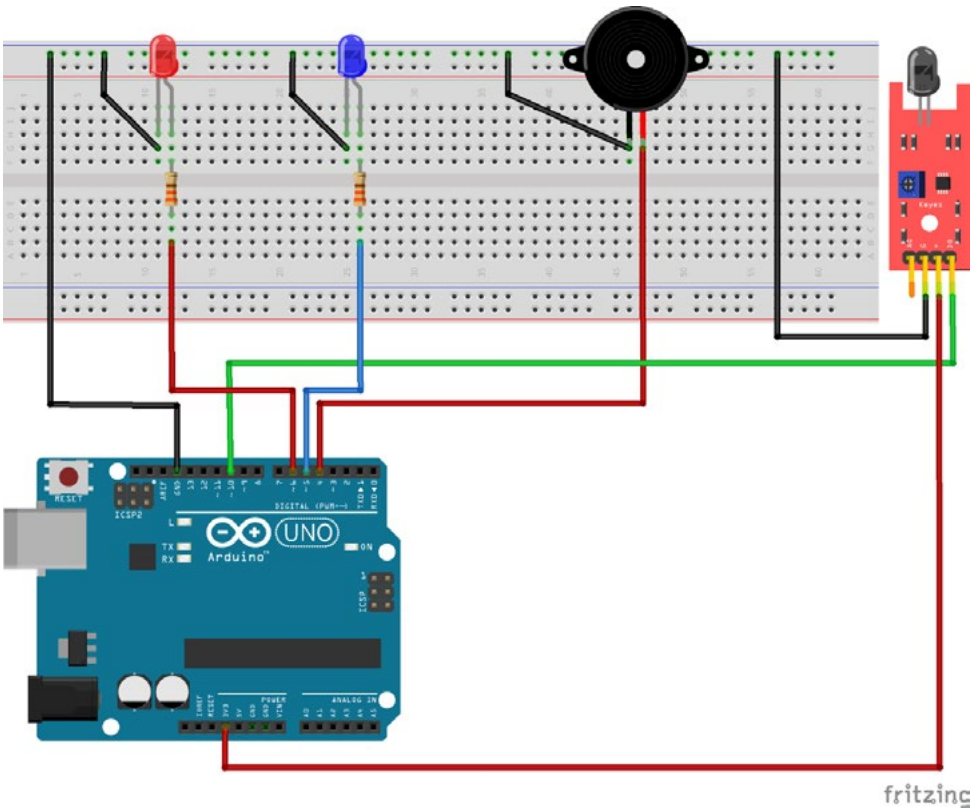
Потребни елементи за реализацију пројекта су:

1. Arduino UNO плочица и USB
2. Матадор плочица
3. Сензор за детекцију пламена
4. 1 x зелена LED
5. 1 x црвена LED
6. 2 x отпорник 220 Ω

Као и у претходним примјерима, нацртајмо грубу алгоритамску структуру пројекта.



Слика 2.2. Поједностављена алгоритамска структура апликације



Слика 2.4. Шема споја

Arduino код:

```
int LedCrvena = 6;
int LedPlava=5;
int Senzor = 10;
int buzzer=4;


int StanjeSenzora;

void setup() {
  pinMode(buzzer, OUTPUT);
  pinMode(LedPlava,OUTPUT);
   pinMode(LedCrvena, OUTPUT);      // Постави извод LedCrvena (6) као
излазни
   pinMode(Senzor, INPUT_PULLUP);   // Постави извод Senzor (10) као
улазни
}

void loop() {
  StanjeSenzora = digitalRead(Senzor); // Очитај стање извода и похрани
у StanjeSenzora
  if (StanjeSenzora == HIGH) {
    digitalWrite(LedCrvena, HIGH);
    digitalWrite(buzzer, HIGH);
    digitalWrite(LedPlava, LOW);
  } else {                            // Иначе је дигитална вриједност 0
    digitalWrite(LedPlava, HIGH);
    digitalWrite(LedCrvena, LOW);
    digitalWrite(buzzer, LOW);
  }
}
```

Овај дио кода не треба да посматрате као завршен пројекат, већ треба да га покушате надоградити користећи друге компоненте како бисте развили сопствене идеје.

НАПОМЕНА:

Будите опрезни приликом тестирања ове апликације. Неопходно је обезбиједити простор и уклонити запаљиве материјале како бисте тестирали сензор пламена.

ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца:	**Ученик/ца:**
- описује улогу електронских компоненти које су коришћене у пројекту	- дефинише карактеристике сензора за детекцију пламена - описује улогу Базера (енг. *buzzer*) - описује примјену LE диода у апликацији - објашњава разлику између аналогних и дигиталних сензора
- користи електричну шему споја за повезивање Arduin-a и електронских компоненти	- повезује сензор за детекцију пламена на Arduino према шеми споја - повезује остале компоненте апликације на Arduino према шеми споја
- креира алгоритамску структуру споја	- процјењује предности и ограничења алгоритамског приступа у рјешавању проблема - визуализира пројектни задатак кроз алгоритамску шему
- креира програмски код за Flame Sensor	- креира програм за дати задатак примјењујући стечено знање из разгранатих програмских структура - реконструише програмски код према сопственим идејама
- верификује и извршава програм	- анализира програмски код - извршава отклањање грешке по потреби уз смјернице наставника/це
- тестира пројектног задатка	- учитава код на Arduino платформу - тестира апликацију користећи упаљач или шибицу, водећи рачуна о безбједности - анализира пројектни задатак - допуњује пројектни задатак својим идејама

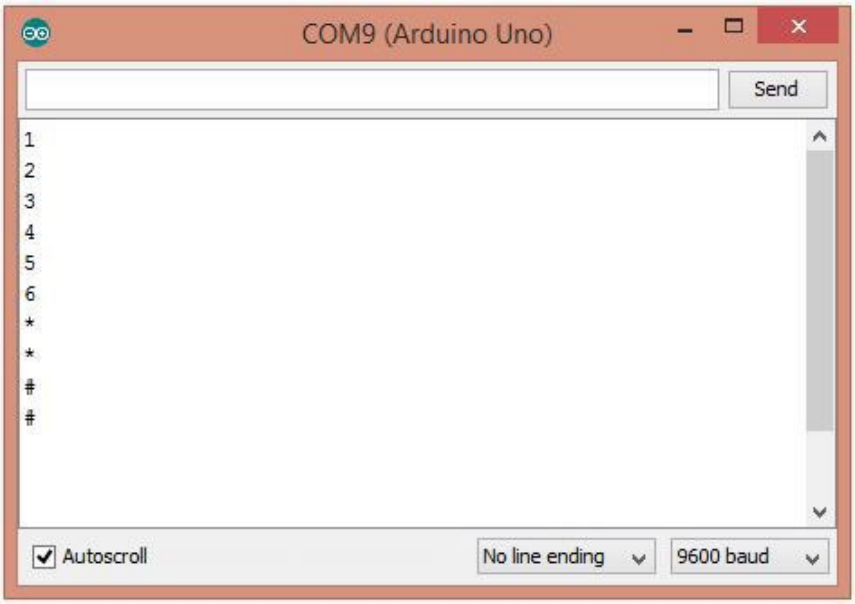
Простор за вјежбу:

```
 void setup() {

}
void loop() {

}
```

Пројекат број 4.
Arduino Keyboard

Још један интересантан *gadet* који можемо da користimo у нашим пројектима. Усудио бих се рећи сигурносни уређај. Користићемо мембранску тастатуру.

Мембранска тастатура диже сваки Arduino пројекат на један нови ниво. Рецимо, омогућава нам да у пројекат убацимо унос шифре, пина, ID корисника апликације или, што да не, унос броја телефона који желимо позвати. Мембранска тастатура састоји се од 12 или, у неким изведбама, 16 типки поређаних у колоне и редове. Она представља улазни уређај у микроконтролер и његова унутрашња структура представља тастере који су спојени у формату матрице, нпр. 4 x 4. Овог пута нећемо користити LCD дисплеј за приказ уноса с тастатуре, већ ћемо испис уноса радити кроз уграђено (енг. *built-in)* у Serial Monitor апликацију.



Слика 4.1. Приказ удаљености на LCD дисплеју

Вјерујем да и ви волите шпијунске или SciFi филмове и вјероватно вам неће недостајати идеја за употребу овог једноставног, али ефектног уређаја у неком од пројеката, али, за сваки случај, ево неколико примјера како би се он могао искористити:

1. PinLock брава – брава која се закључава/откључава уношењем неке шифре (пина)
2. Arduino GSM dialer - у комбинацији с GSM модулом креирајте Arduino телефон
3. Arduino тастатура

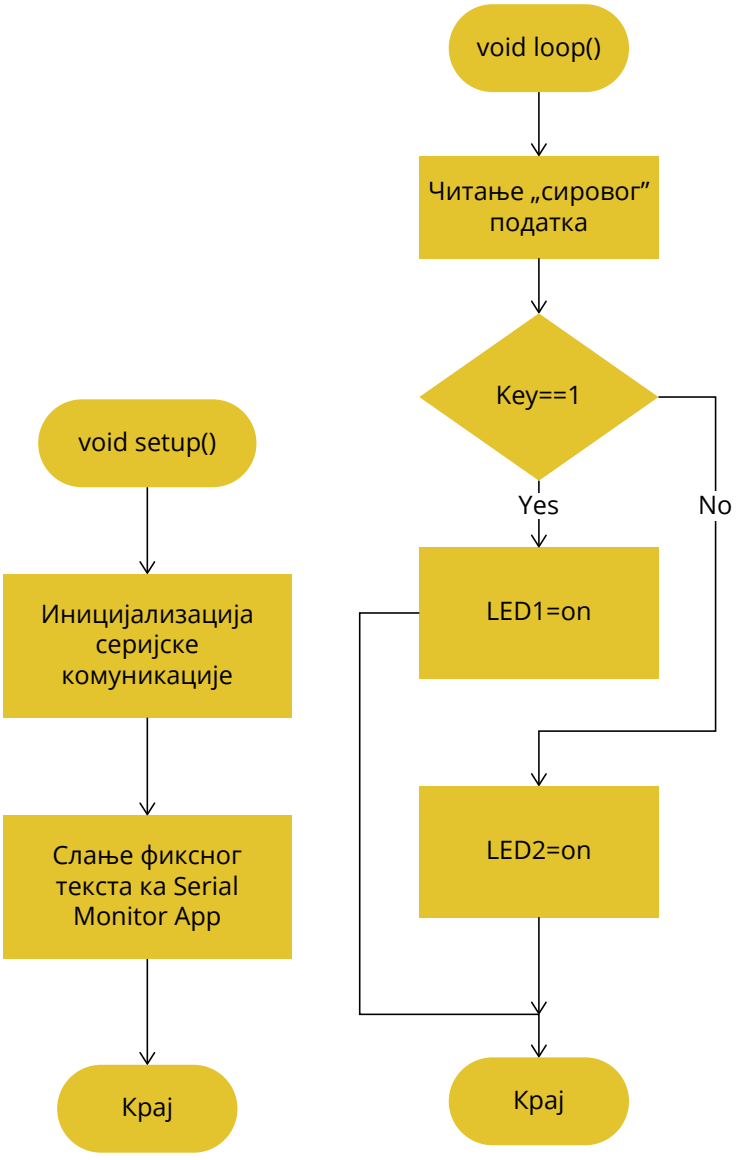
Опционо је могуће приликом пројектовања неке ваше апликације оставити простора и слободне пинове на контролеру те креирати *plug-in* конектор који ће се моћи с написаном апликацијом користити за евентуално „плаћено откључавање“ опција на апликацији.

Потребни елементи за реализацију пројекта су:

1. ARDUINO UNO
2. Тастатура

Након упознавања с хардвером и опцијом *Serial Monitor,* пројектни задатак је да креирате апликацију која ће, уколико је притиснут тастер 1, активирати LED спојен на ПИН2, ако је активиран неки други тастер, активирати LED спојен на ПИН3.

Нацртајмо прво грубу алгоритамску структуру пројекта, као што смо радили у претходним пројектима. Овог пута искористићемо и тастатуру за мало понављање IF...THEN...ELSE изјаве.

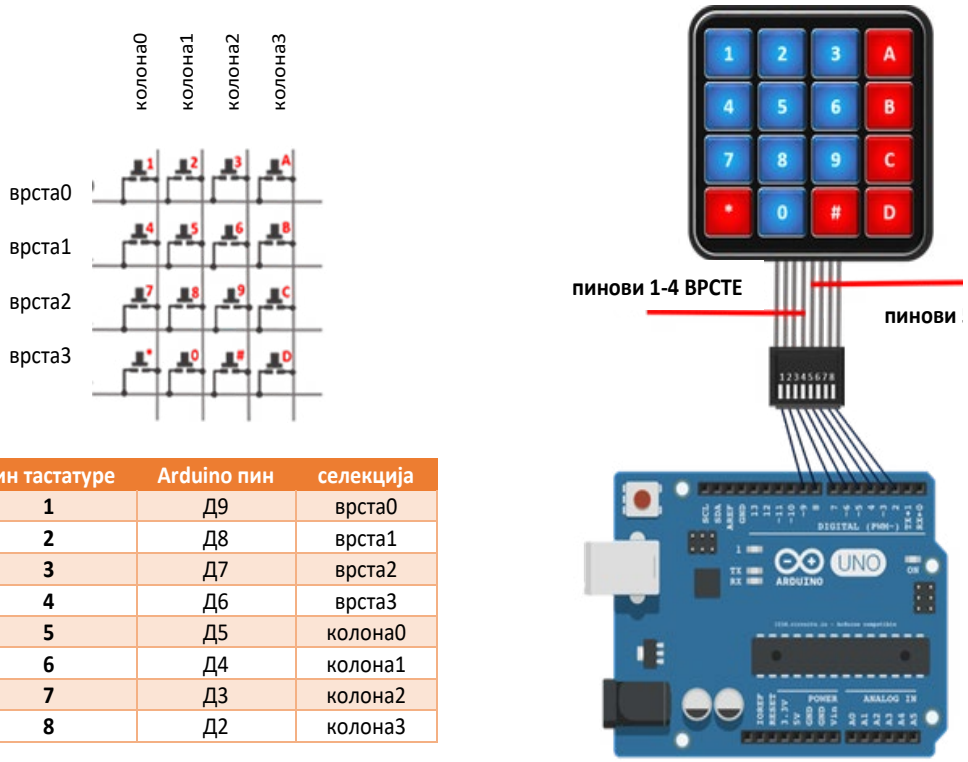


Слика 4.2. Поједностављена алгоритамска структура апликације

За овај конкретни уређај није нам потребна математичка формула, али се већ у самом алгоритму *sample* апликације поставља питање могућности „памћења“ комплекснијих информација. Могуће је креирати низ (*array*) типа података:

```
int enteredPIN[4];

int fixedPIN[] = {1, 2, 3, 4};
 enteredPIN[0]== fixedPIN[0];
     enteredPIN[0]== 1;
     enteredPIN[1]== 2;
```

елементи низа би се потом пунили тако да се провјерава унос с тастатуре с елементима „фиксног“ низа. Наравно, ово је само један од начина како бисте могли развити мало комплекснију апликацију.



пин тастатуре	Arduino пин	селекција
1	Д9	врста0
2	Д8	врста1
3	Д7	врста2
4	Д6	врста3
5	Д5	колона0
6	Д4	колона1
7	Д3	колона2
8	Д2	колона3

Слика 4.3. Изглед и pinout keyboard-a

Arduino код:

```
#include "Keypad.h"
const byte ROWS = 4; // Број редова
const byte COLS = 3; // Број колона
char keys[ROWS][COLS] = {
{'1','2','3'},
{'4','5','6'},
{'7','8','9'},
{'#','0','*'}
};
byte rowPins[ROWS] = {8, 7, 6, 5}; // Излазни пинови тастатуре по редовима R1 = D8, R2 = D7, R3 = D6, R4 = D5
byte colPins[COLS] = {4, 3, 2}; // Излазни пинови тастатуре по колонама C1 = D4, C2 = D3, C3 = D2
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  char key = keypad.getKey();
  if (key != NO_KEY)
    Serial.println(key);
}
```

Овај дио кода не треба да посматрате као завршен пројекат, већ треба да га покушате надоградити већ предложеним идејама или да реализујете неке своје идеје.

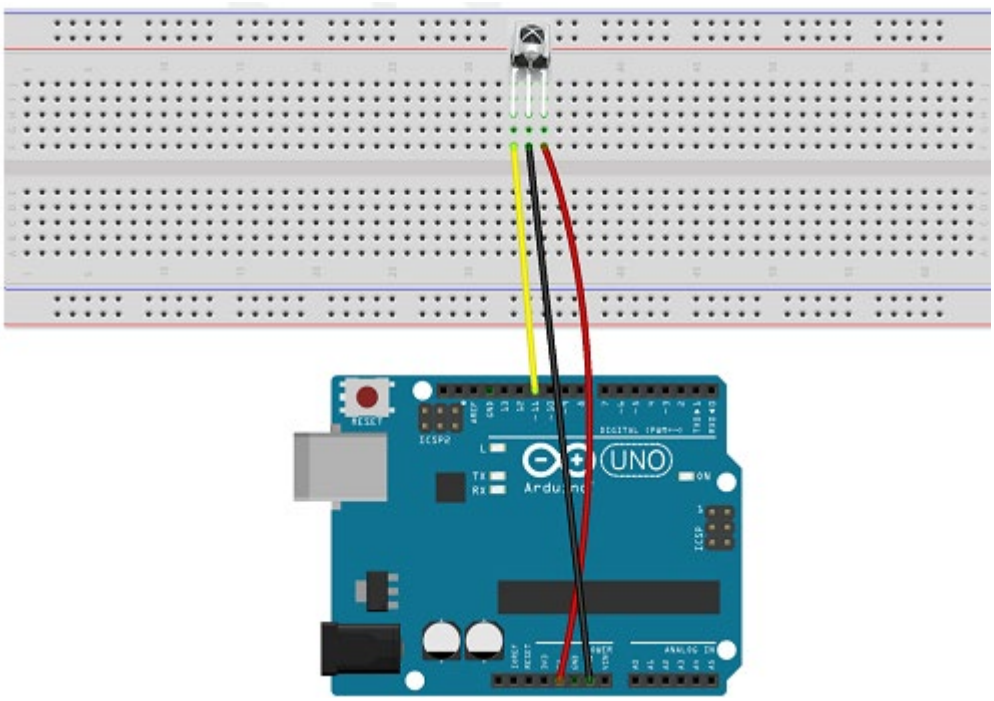
ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца:	**Ученик/ца:**
- описује улогу електронских компоненти које су коришћене у пројекту	- објашњава улогу мембранске тастатуре - користи Serial Monitor
- користи електричну шему споја за повезивање Arduin-а и електронских компоненти	- повезује мембранску тастатуру на Arduino према шеми споја - повезује преостале компоненте у пројекту према шеми споја
- креира алгоритамску структуру споја	- процјењује предности и ограничења алгоритамског приступа у рјешавању проблема - визуализира пројектни задатак кроз алгоритамску шему
- креира програмски код за примјену мембранске тастатуре	- примјењује стечено знање из низова и матрица - креира програмски код за дати задатак уз адекватне смјернице наставника/це - реконструише програмски код према сопственим идејама
- користи одговарајуће библиотеке унутар програма за реализацију пројектног задатка	- по потреби инсталира одговарајуће библиотеке - позива унутар програмског кода адекватну библиотеку
- верификује и извршава програм	- анализира програмски код - отклања грешке уколико су настале у коду према смјерницама наставника/це
- тестира пројектни задатак	- учитава код на Arduino платформу - користи Serial Monitor за тестирање пројектног задатка - анализира пројектни задатак - допуњује пројектни задатак с датим приједлозима или неким својим идејама

Простор за вјежбу:

```
 void setup() {

}
void loop() {

}
```

Пројекат 5.
IR сензор

Овај пројекат нам је посебно интересантан, јер смо од малих ногу окружени и фасцинирани даљинским управљачима. Ово ће уједно бити испуњење сна сваке програмерке и програмера, коначно улазимо у свијет даљинске контроле. Реализацијом овог пројекта отварамо безброј могућности за контролу уређаја на удаљености од четири до девет метара. Да, то и није баш много, али битно је направити први корак. Мора се нагласити да удаљеност с које можемо да контролишемо неку апликацију користећи овај сензор увелико зависи од окружења у којем се користи, као и евентуалним препрекама приликом његовог коришћења.



Слика 5.1. Начин спајања IR сензора

За почетак, прво ћемо покушати контролисати LE диоде, али ево неколико идеја које можете искористити за неки *upgrade* овог почетног пројекта.

1. Контрола собног вентилатора
2. Активација собне расвјете
3. Израда РГБ контролера

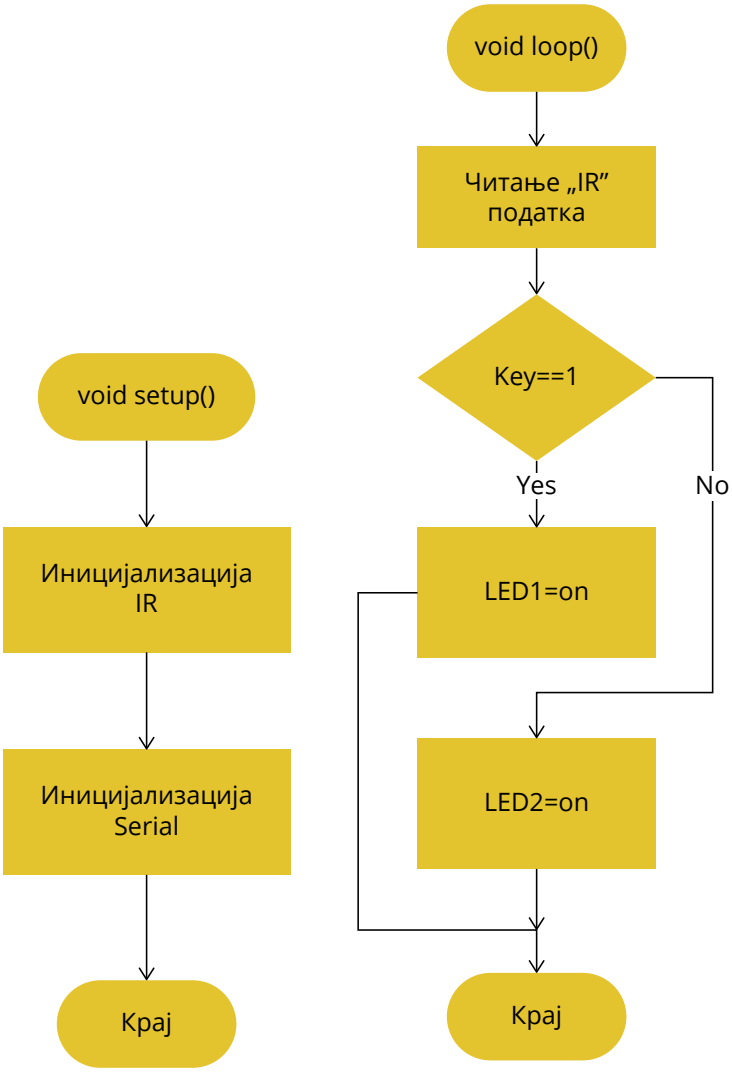
Свако од поменутих приједлога може да се надогради уводећи у пројекат нове или компоненте које су већ коришћене у овом приручнику. Једна од идеја која би могла да вам помогне јесте израда IR debuggera, за који можете да користите LCD дисплеј. Наиме, могуће је искористити LCD дисплеј за испис примљеног податка с IR предајника. На тај начин ваш LCD постаје live debugger и олакшаће вам писање апликације.

Како будете проширивали своје знање можда помоћу овог сензора направите опцију за одабир апликације спрам примљеног податка с даљинског управљача. Можете да напишете три различите апликације на свом Arduin-у. Одабир која апликација ће се користити може се урадити спрам податка добијеног с IR сензора. На овај начин наш Arduino, барем привидно, постаје мултифункционалан уређај.

Потребни елементи за вјежбу:

1. ARDUINO UNO
2. IR Receiver XX1838
3. Матадор
4. Даљински управљач
5. Каблићи

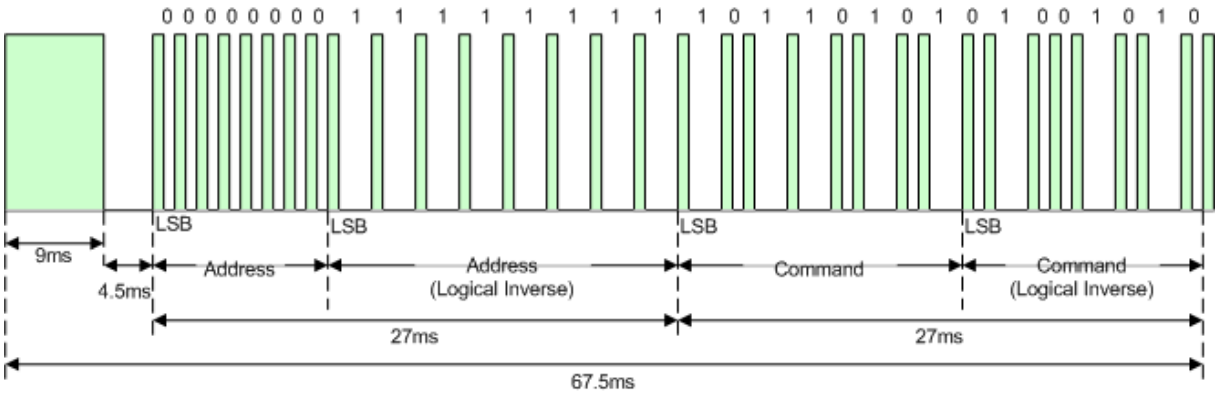
Наставићемо с праксом цртања грубе алгоритамске структуре пројекта и надамо се да ћете стећи навику да, прије него почнете писати код, прво тај код пробате визуализирати и „нацртати“. У овом случају не мора значити да ће алгоритамска структура одговарати стварној апликацији, али сугерише вам се да је најједноставније кренути од упоређивања запримљеног податка с неком константом те донијети одлуку на основу тога.



Слика 5.2. Поједностављена алгоритамска структура апликације

Иако ћемо у овом пројекту користити готову библиотеку, било би корисно научити или барем видјети како то IR предајник, у нашем случају IR даљински управљач, генерише поруку коју наш IR сензор мора „декодирати“. Сваки пут када се на даљинском управљачу притисне нека типка, пошаље се порука сљедећег формата:

- Сигнал логичке 1 трајања 9 ms
- Сигнал логичке 0 трајања 4.5 ms
- 8-битна адреса пријемника
- 8-битна инвертована адреса пријемника
- 8-битна команда
- Инвертована 8-битна команда
- Пулс трајања 562,5 µs који сигнализује крај поруке

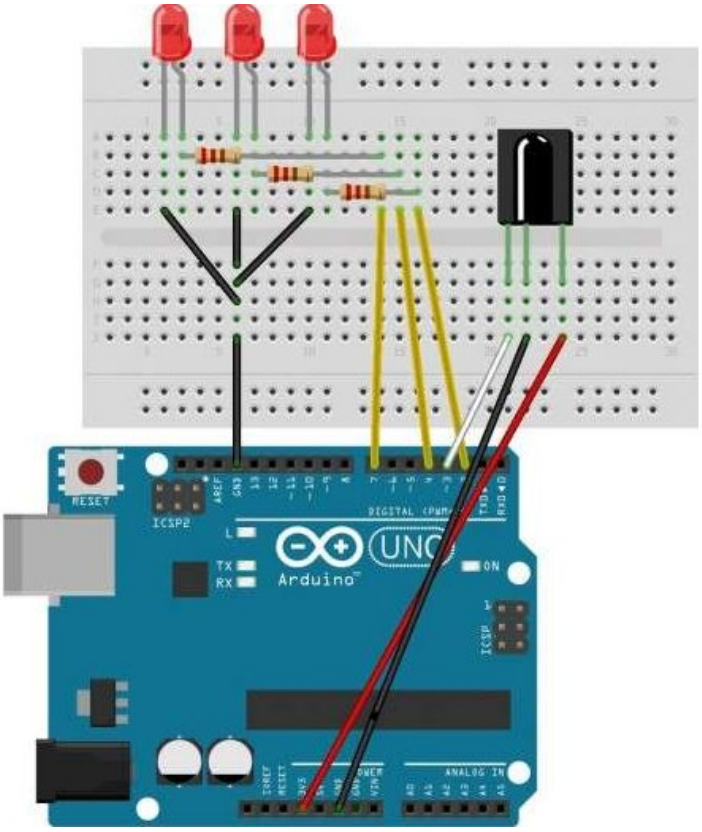


Слика 4.3. IR NEC протокол

Коришћењем библиотеке бићемо закинути за потпуно разумијевање протокола и процеса декодирања поруке, али на овај начин много брже ћемо да дођемо до готове апликације. Дакле, да поновимо, IR сензор XX1838 прима инфрацрвени сигнал који генерише даљински управљач, декодира сигнал који ћемо да искористимо у својој апликацији за одлучивање и има три пина (input, Vcc і GND).

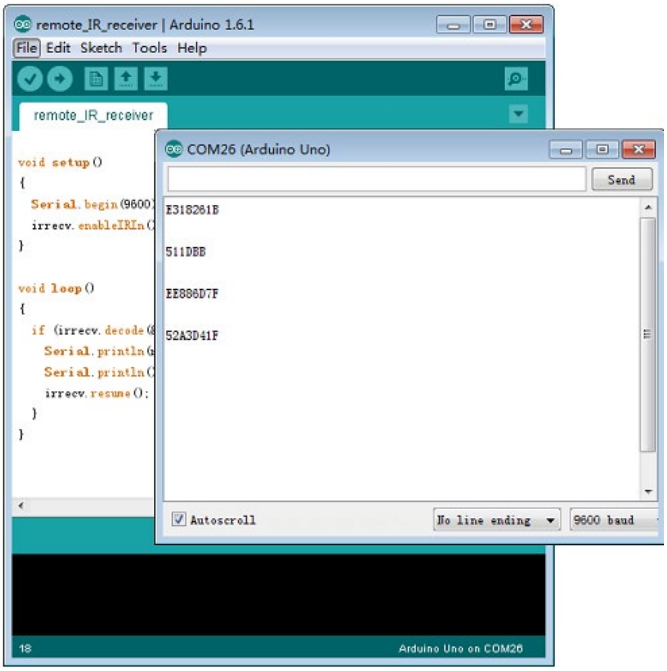
Слика 5.4. Pinout XX1838 и изглед даљинског управљача

У овом пројекту потребно је прво испрограмирати Arduino да прима инфрацрвени сигнал и да га пошаље на Serial Monitor. За ово нам је потребна библиотека Arduino-IR-remote-master.



Слика ***5.5.*** **Шема споја**

У тестној апликацији за софтверски debugging користићемо Serial Monitor апликацију. Препорука је направити табеларни испис свих декодираних наредби из Serial Monitor-а. На тај начин олакшаћете себи планирање апликације, а отвара вам се бржи пут за дефинисање константи у некој од ваших наредних апликација.



Слика 5.6. Изглед декодиране команде

Arduino код:

```
#include <IRremote.h>
int RECV_PIN = 11;// Декларација IR пријемника на пину 11
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
  Serial.begin(9600); // Отвори серијски порт
  irrecv.enableIRIn(); // Иницијализација IR пријемника
}
void loop()
{
 if (irrecv.decode(&results)) {
 Serial.println(results.value, HEX);// Испис у хексадецималном пријемном
коду
  Serial.println(); // Ради прегледа исписујемо једну празну линију.
  irrecv.resume(); // Пријем наредне вриједности
  }
  }
```

И овај дио кода не треба да посматрате као завршен пројекат, већ треба да га покушате надоградити већ предложеним идејама или да реализујете неке своје.

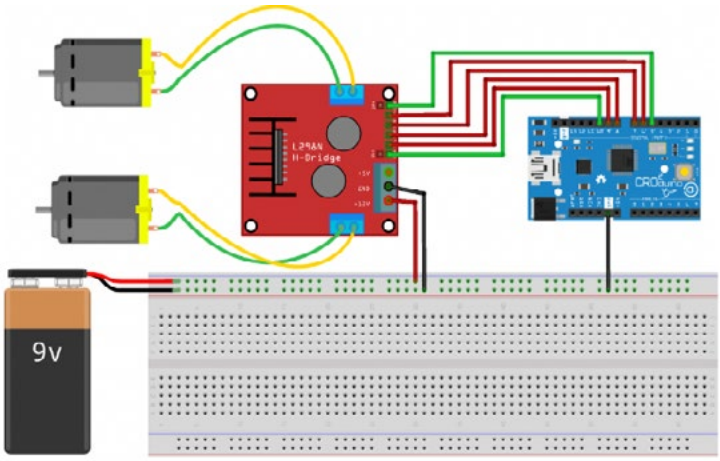
ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца:	**Ученик/ца:**
- описује улогу електронских компоненти које су коришћене у пројекту	- објашњава улогу IR Receiver XX1838 сензора - објашњава примјену IR предајника и начин одашиљања порука с даљинског управљача - описује везу између даљинског управљача и IR сензора - описује примјену LE диоде у пројекту - користи Serial Monitor за учитавање резултата
- користи електричну шему споја за повезивање Arduin-a и електронских компоненти	- повезује IR Receiver XX1838 сензор према шеми споја - повезује преостале компоненте у пројекту према шеми споја
- креира алгоритамску структуру споја	- процјењује предности и ограничења алгоритамског приступа у рјешавању проблема - визуализира пројектни задатак кроз алгоритамску шему
- креира програмски код за даљинско управљање	- програмира Arduino да прима инфрацрвени сигнал и да га шаље на Serial Monitor уз адекватне смјернице наставника/це - реконструише програмски код према сопственим идејама
- користи одговарајуће библиотеке унутар програма за реализацију пројектног задатка	- користи готову библиотеку за IR предајник, Arduino-IRremote-master library - позива унутар програмског кода адекватну библиотеку
- верификује и извршава програм	- анализира програмски код - отклања грешке уколико су настале у коду према смјерницама наставника/це
- тестира пројектни задатак	- учитава код на Arduino платформу - користи Serial Monitor за тестирање пројектног задатка - анализира пројектни задатак - примјењује IR Receiver XX1838 сензор за управљање на даљину у другим пројектним идејама - Развија спопствене идеје за примјену IR Receiver XX1838 сензора

Простор за вјежбу:

```
 void setup() {

}
void loop() {

}
```

Пројекат 6.
L298 драјвер

Већина робот или ЦНЦ ентузијаста крене у авантуру с микроконтролерским системима управо с овим драјвером. Наиме, због ограничења количине струје коју микроконтролер може дати на својим излазима, готово је немогуће директно на излаз микроконтролера спојити било какав озбиљнији мотор. Стога, ако није ријеч о hobby RC серво мотору, за покретање мотора веће снаге потребан нам је драјвер. Драјвер мотора, осим тога што обезбјеђује довољну количину струје за покретање, отвара нам и опцију за софтверску контролу броја обртаја и смјера вртње. Мотор драјвер захтијева и стабилно екстерно напајање за мотор, на тај начин штитимо и сам микроконтролер. Како се будете бавили овом проблематиком, наићете на различите изведбе драјвера с различитим принципом дјеловања, а у овом пројекту користићемо један од најједноставнијих, али у исто вријеме врло поуздан L298N драјвер. И у овом пројекту користићемо га за контролу и управљање истосмјерним (DC) моторима, уз напомену да истим драјвером можемо контролисати и stepper моторе. О њима ће бити више говора у неком од наредних пројеката.



Слика 6.1. Шема спајања L298 драјвера и DC мотора

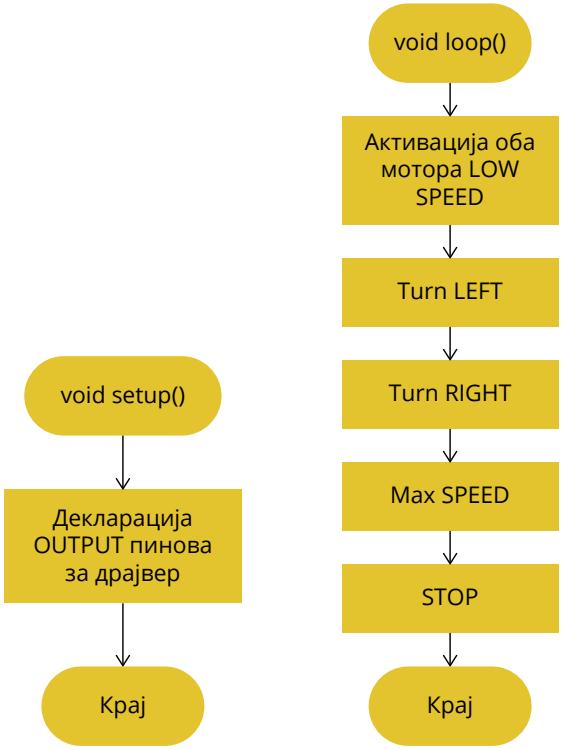
Према setup-у са слике 6.1, одмах се намеће идеја о аутономном возилу. Ако томе још додате и HC-SR04 и неко лијепо дизајнирано кућиште, мислим да би се Илон Маск (Elon Musk) морао мало забринути. Ово озбиљно схватите јер су сви велики људи данашњице кренули од малих пројеката.

Размислите о пројектним идејама у којима се користи L298 драјвер те о пројектима у којима се користи DC мотор. Мало истражујте!

Потребни елементи за вјежбу:

- ARDUINO UNO
- L298 драјвер
- Два истосмјерна мотора
- Матадор плочица
- Каблићи

Прво ћемо нацртати грубу алгоритамску структуру пројекта, као што смо радили и у претходним примјерима. Надамо се да ћете стећи навику да, прије него почнете писати код, прво пробате визуализирати тај код и „нацртати“ га.



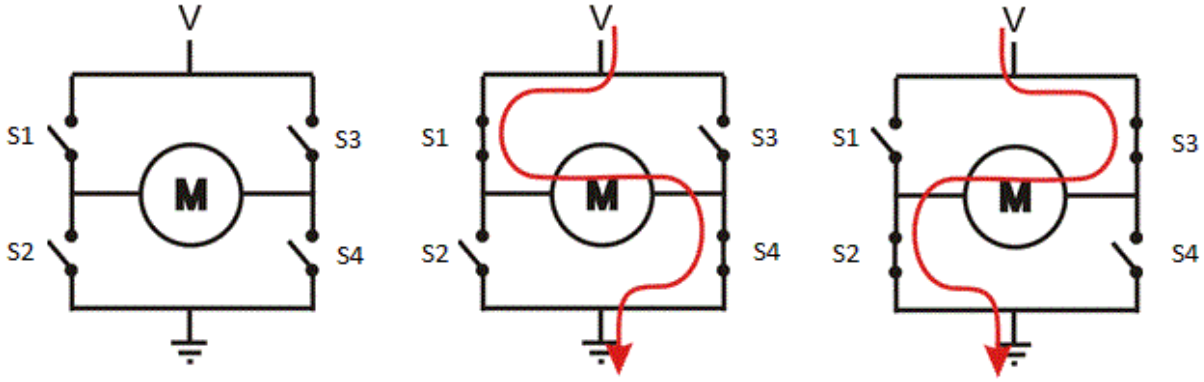
Слика 6.2. Поједностављена алгоритамска структура апликације

OUT1 – +/- Излаз за мотор А
OUT2 - +/- Излаз за мотор А
OUT3 - +/- Излаз за мотор Б
OUT4 – +/- Излаз за мотор Б
CON5 – Служи за укључивање 5V регулатора напона
EnA – Служи за контролу PWM-а за мотор А
EnB - Служи за контролу PWM-а за мотор Б

In1 - Служи за контролу смјера окретаја мотора А
In2 - Служи за контролу смјера окретаја мотора А.
In3 - Служи за контролу смјера окретаја мотора Б.
In4v - Служи за контролу смјера окретаја мотора Б
+5V –Улаз у који иде спољни извор напајања
GND – Ground
+12V – Улаз спољног извора напајања за моторе

Слика 6.3. Изглед и pinout L298 драјвера

Да бисте могли написати код за неки од својих будућих пројеката који би користили овај драјвер, требало би описати принцип рада L298N драјвера. Прва ствар коју треба споменути јесте да он користи такозвани Х-мост (*енгл. H-bridge*) интегрисани чип. Х-мост је струјни круг који се састоји од 4 прекидача спојена с мотором на начин приказан на сљедећој слици. Ако С2 и С3 прекидаче затворимо, а С1 и С4 прекидаче отворимо, струја ће тећи у одређеном смјеру. Ако замијенимо стања прекидача тако да су С2 и С3 отворени, а С1 и С4 затворени, струја ће кроз мотор протећи у супротном смјеру. Не смијемо затворити сва четири прекидача одједном или оба прекидача на једној страни Х-моста (нпр. С1 и С2), јер тиме изазивамо кратки спој.



Слика 6.4. Принцип рада L298 драјвера

Наредна табела приказује све могуће комбинације стања прекидача и њихове исходе.

C1	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*
C2	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*
C3	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*
C4	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
Резултат	Нема промјене	Нема промјене	Нема промјене	Кратки спој	Нема промјене	Режим кочења	Мотор се окреће у лијеву страну	Кратки спој	Нема промјене	Мотор се окреће у лијеву страну	Режим кочења	Кратки спој	Нема промјене	Кратки спој	Нема промјене	Кратки спој

Помоћу Х-моста врло је једноставно замијенити поларитет на трошилу. Најчешће се користи за замјену смјера вртње DC мотора. Х-мост се користи и у разним другим апликацијама попут DC/AC, AC/AC или DC/DC конвертера.

Иако је намијењен за моторе, може да се користи и за друге уређаје који за одређену потребу захтијевају замјену поларитета на својим улазним крајевима. Уз могућност замјене поларитета, L298N драјвер има још могућност контроле брзине окретаја мотора користећи PWM (енгл. Puls-Width Modulation) сигнале.

Овај пут за једноставну апликацију нећемо користити библиотеку, али је ово идеалан хардвер за који би на предмету програмирање могли написати једноставан „драјвер“. За сада је довољно контролом стања излаза помоћу digitalWrite() функције покренути моторе. За брзину окретаја мотора користи се analogWrite() функција.

Arduino код:

```
// Декларација пинова за први мотор
int enA = 10;
int in1 = 9;
int in2 = 8;
// Декларација пинова за други мотор
int enB = 5;
int in3 = 7;
int in4 = 6;

void setup()
{
  // Постави GPIO као излазе.
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
}

void loop()
{
  // Активирај први мотор.
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  // Постави брзину окретаја првог мотора 200 (max 255).
  analogWrite(enA, 200);
  // Активирај други мотор.
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  // Постави брзину окретаја другог мотора 200 200 (max 255).
  analogWrite(enB, 200);
  delay(2000);

  // Замијени смјерове окретаја мотора.
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  delay(2000);

  // Искључи оба мотора
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  delay(2000);
}
```

Овај дио кода не треба да посматрате као завршен пројекат, него треба да га покушате надоградити већ предложеним идејама или да реализујете неке своје.

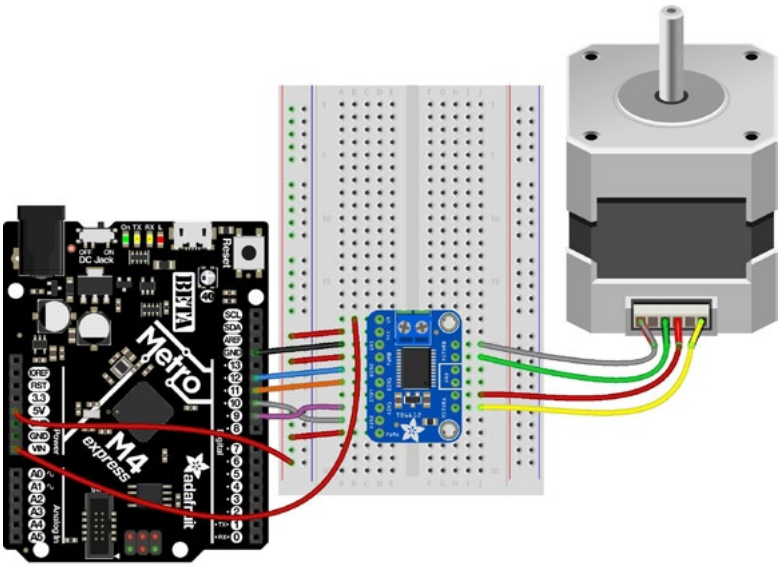
ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца:	**Ученик/ца:**
- описује улогу електронских компоненти које су коришћене у пројекту	- објашњава улогу драјвера мотора - описује везу између L298 драјвера и DC мотора - демонстрира улогу X-моста кроз шематски приказ - објашњава принцип рада L298 драјвера
- користи електричну шему споја за повезивање Arduin-a и електронских компоненти	- повезује L298 драјвер и DC мотор - према шеми споја
- креира алгоритамску структуру споја	- процјењује предности и ограничења алгоритамског приступа у рјешавању проблема - визуализира пројектни задатак кроз алгоритамску шему
- креира програмски код за L298 драјвер и DC мотор	- програмира Arduino декларушући прво пинове мотора - реконструише програмски код према сопственим идејама
- верификује и извршава програм	- анализира програмски код - отклања грешке уколико су настале у коду према смјерницама наставника/це
- тестира пројектни задатак	- учитава код на Arduino платформу - анализира пројектни задатак - надограђује програмски код сопственим идејама

Простор за вјежбу:

```
 void setup() {

}
void loop() {

}
```

Пројекат 7.
TB6612 драјвер - DC/Stepper motor

Овај пројекат је сличан претходном, с тим да се користи драјвер новије генерације. Као и L293 драјвер и TB6612 може контролисати два DC мотора мале снаге или један stepper мотор. Принцип рада је потпуно исти као и код L293 драјвера, TB6612 садржи два пуна X-моста (четири полу-X-моста). То значи да можете покретати 2-4 соленоида (само два могу да буду активна истовремено, на супротним мостовима). Поједностављено, можемо да управљамо или с два DC мотора двосмјерно, као у претходном пројекту, или с једним stepper мотором. Важно је да струјно оптерећење не прелази 1,2 A будући да је то горња граница струје овог драјвера. Мада он може да поднесе и струјни удар од 3 A, али само за кратко вријеме (20 милисекунди). Треба напоменути да TB6612 долази с уграђеним диодама за повратни удар, тако да се не морате бринути да ће индуктивни удар оштетити ваш Arduino.



Слика 7.1. Шема спајања TB6612 драјвера и stepper мотор
с Arduino кампатибилном Adafruit Metro развојном плочом

На претходној слици нису уцртани водичи за екстерни напон за драјвер који се мора обезбиједити са стабилног истосмјерног извора напајања (4.5-13-5 V).

Потребни елементи за вјежбу:

- ARDUINO UNO
- TB6612 драјвер
- Stepper мотор - могуће их је наћи у старим штампачима
- Матадор плочица
- Каблићи

Код овог драјвера постоје три групе пинова те ћемо их побројати и укратко објаснити њихове функције.

Пинови за напајање

Vmotor - пин за напајање мотора. Препоручено је да се напон креће у границама од 4.5 V до 13.5 V. Важно је напоменути, за неке ваше будуће пројекте, да је могуће стварање „сметњи" па, за случај да имате пројекат с аналогним сензорима или РФ читачима, потребно је да ово напајање буде одвојено од остатка кола или барем филтрирано.

Vcc - напајање логике драјвера. Напонски ниво требало би да буде за Arduino 5V.

GND - мотор ground.

Улазни сигнали на драјверу

INA1, INA2 - улазни пинови за Мотор А на Х-мосту

PWMA – PWM улаз за Мотор А на Х-мосту, уколико за пројекат није потребна PMV контрола, пин спојити на +5 V.

INB1, INB2 - улазни пинови за Мотор Б на Х-мосту

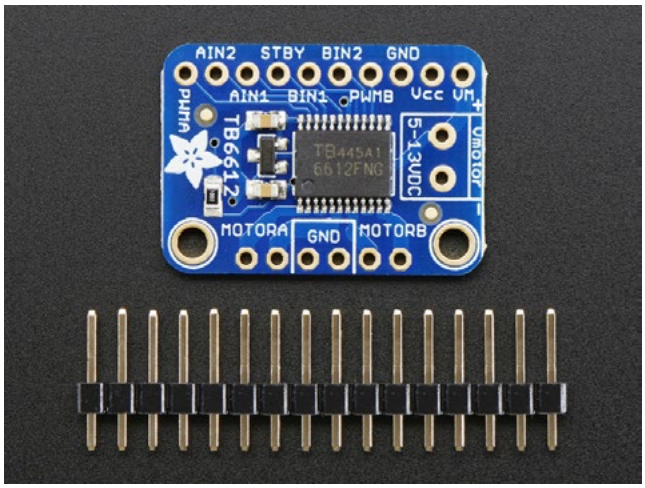
PWMB – PWM улаз за Мотор Б на Х-мосту, уколико за пројекат није потребна PWM контрола пин спојити на +5 V.

STBY - standbay пин за брзо искључивање мотора, pull up ка Vcc преко 10К отпор. Спојити на масу за искључење.

Излазни сигнали за мотор

Мотор А - излази за мотор А или А-намотај stepper мотора контролисан од INA1, INA2 и PWMA

Мотор Б - излази за мотор Б или Б-намотај stepper мотора контролисан од INB1, INB2 и PWMB



Слика 7.2. Изглед ***TB6612*** драјвера

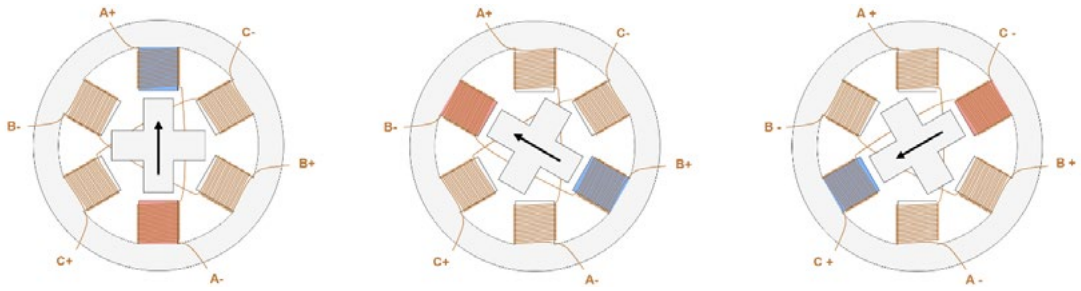
Како драјвер долази као на претходној слици, потребно га је припремити за употребу. Најлакша опција је искористити матадор плочу из starter kit-а те поставити драјвер и залемити пинове. Техника лемљења објашњена је кроз слике у пројекту број 8.

Stepper или корачни мотор јесте електрични мотор чија је главна особина да се његова осовина ротира корачно, односно помиче се за фиксни износ степени. Ова особина је постигнута захваљујући унутрашњој структури мотора и омогућује да се зна тачан угаони положај осовине једноставним бројањем колико је корака осовина мотора направила, без потребе за сензором.

Као и сви електромотори, stepper мотори имају стационарни дио (статор) и покретни дио (ротор). На статору се налазе зупци на којима су намотаји жице, док је ротор или трајни магнет или жељезна језгра промјењиве релуктанције.

Основни принцип рада stepper мотора је сљедећи: Довођењем под напон једне или више фаза статора, струја која тече у завојници ствара магнетско поље и ротор се поравнава с тим пољем. Довођењем под напон различитих намотаја у секвенци, ротор се може ротирати за тачно дефинисан угао како би се постигао жељени крајњи положај. Слика 7.3 приказује принцип рада.

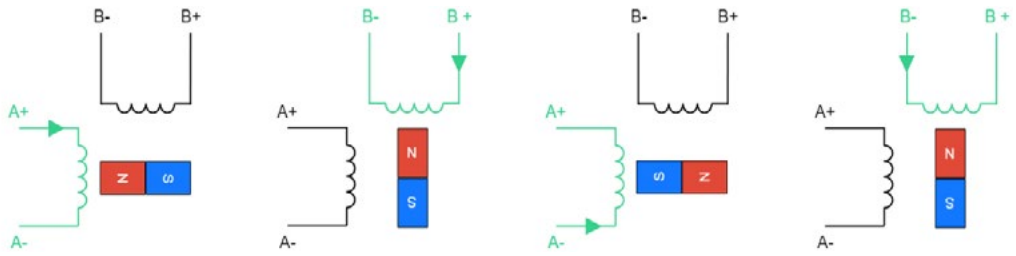
На почетку је завојница А под напоном и ротор је поравнат с магнетским пољем које производи. Када је завојница Б под напоном, ротор се ротира у смјеру казаљке на сату за 60° како би се ускладио с новим магнетским пољем. Исто се догађа када је завојница Ц под напоном. На сликама боје зубаца статора означавају смјер магнетског поља које ствара намотај статора.



Слика 7.3. Принцип рада stepper мотора

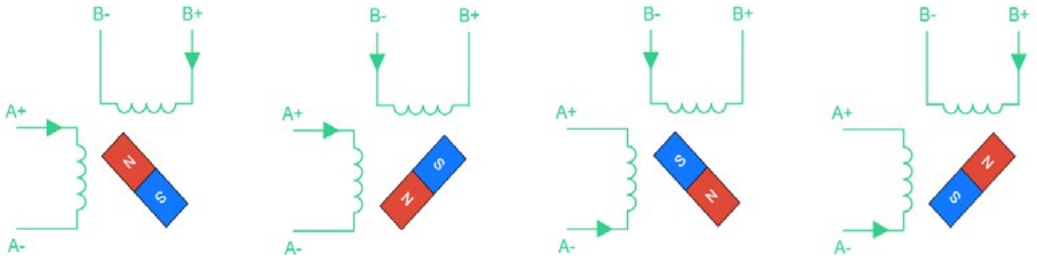
Постоје четири различите технике управљања stepper мотором:

У wave моду само једна фаза је под напоном. Ради једноставности, рећи ћемо да струја тече у позитивном смјеру од + до - (нпр. од А+ до А-), у супротном, смјер је негативан. Почевши с лијеве стране, струја тече само кроз А намотај у позитивном смјеру, а ротор, представљен магнетом, усклађен је с магнетским пољем које ствара. У сљедећем кораку струја тече само кроз намотај Б у позитивном смјеру, а ротор се окреће за 90° у смјеру казаљке на сату како би се ускладио с магнетским пољем које ствара намотај Б. Касније се намотај А поновно активира, али струја тече у негативном смјеру, а ротор се поновно окреће за 90°. У посљедњем кораку струја тече негативно у намотају Б и ротор се поновно окреће за 90°.



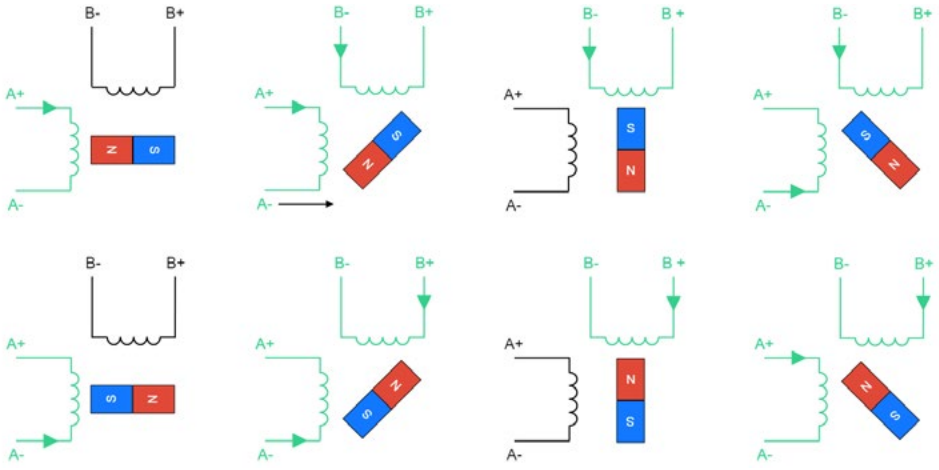
Слика 7.4. Wave мод

У **full-step** начину рада, два намотаја су увијек под напоном у исто вријеме. Наредна слика приказује различите кораке овог начина контроле. Кораци су слични онима у wave начину рада, а најзначајнија разлика је у томе што с овим начином рада мотор може да произведе већи моменат, јер у мотору тече више струје и ствара се јаче магнетско поље.



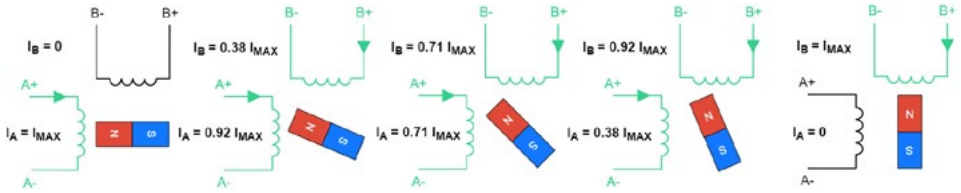
Слика 7.5. Full step мод

Half-step мод је комбинација претходна два начина контроле stepper мотора. Коришћење ове комбинације омогућава да се величина корака смањи за пола (у овом случају, 45° умјесто 90°). Једини недостатак је што моменат који производи мотор није константан, јер је већи када су оба намотаја под напоном, а слабији када је под напоном само један намотај.



Слика 7.6. Half-step мод

Microstepping се може посматрати као даљње побољшање half-step начина рада, јер омогућава још више смањење величине корака и константан излазни моменат. То се постиже контролисањем интензитета струје која тече у сваком намотају. Коришћење оваквог начина управљања захтијева комплекснији мотор драјвер.



Слика 7.7. Microstepiping

У даљем тексту навешћемо неке од предности stepper мотора. Наиме, због своје унутрашње структуре ови мотори не захтијевају сензор за детекцију положаја мотора. Будући да се мотор креће изводећи „кораке", једноставним бројањем ових корака можете да добијете положај мотора у одређеном тренутку.

Осим тога, управљање мотором прилично је једноставно. Мотору је потребан драјвер, али не треба сложене израчуне или подешавање да би исправно радио, односно контролни захтјеви су мањи у поређењу с другим моторима. С микрокораком може се постићи висока тачност положаја, до приближно 0,007°.

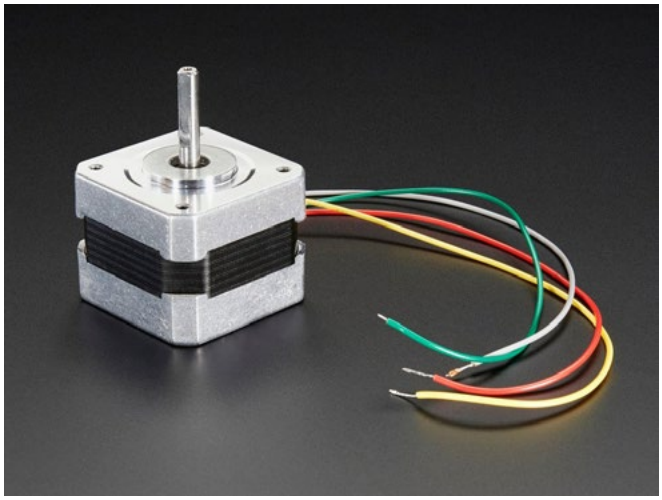
Ови мотори нуде добар закретни моменат при малим брзинама, изврсни су за држање положаја, а имају и дуг животни вијек.

Наравно, ови мотори и упркос свему имају и неколико недостатака. Као прво, могу да „пропусте" корак ако је моменат оптерећења превисок. То негативно утиче на управљање, јер не постоји начин да се сазна стварни положај мотора. Коришћење микрокорака повећава вјероватност да ће корачни мотори имати овај проблем.

Ови мотори увијек троше максималну струју чак и када су у празном ходу, што погоршава ефикасност и може да узрокује прегријавање, при томе имају и мали окретни моменат и постају прилично бучни при великим брзинама.

Укратко, stepper мотори су добри када требате јефтино рјешење које се лако контролише и када ефикасност и велики закретни моменат при великим брзинама нису потребни. Ово их чини идеалним рјешењем за хобисте, а одлични су и као први корак у смислу истраживања роботике.

Ми ћемо у нашем примјеру користити Биполарни stepper мотор јер такви мотори имају намотаје с два извода те захтијевају Х-мост за управљање. Униполарни stepper мотори имају централни извод на намотају и могу се препознати по томе што увијек имају више од четири водича на себи.



Слика 7.8. Биполарни stepper мотор

Спојићемо наш драјвер према шеми споја и то:

- Vmotor ***на*** 12 V
- Vcc ***на*** 5 V
- GND ***на масу***
- AIN ***у пин*** 10
- AIN1 ***у пин*** 9
- BIN1 ***у пин*** 11
- BIN2 ***у пин*** 12
- PWMA ***и*** PWMB ***на*** Vcc

Затим закачите један намотај мотора на мотор А прикључак (црвена и жута), а други намотај на мотор Б прикључак (зелена и сива/смеђа). Ако имате други мотор, мораћете мало експериментисати да схватите које су жице који намотај. Можете да користите мултиметар за мјерење између жица. Оне с малим отпором између њих су пар намотаја. Ако мотор вибрира, али се не окреће, провјерите јесу ли сви водичи спојени и покушајте окренути неколико водича или поново провјерите парове.

Arduino код:

```
#include <Stepper.h>
// Примијените ово на број корака на свом мотору #define STEPS 200.
// Креирај инстанцу stepper класе, спецификујући број корака мотора и
пинова (STEPS, 4, 5, 6, 7).
void setup(){
  Serial.begin(9600);
  Serial.println(„Stepper test!“);
  // Подесите брзину мотора на 30 обртаја у минути.
  stepper.setSpeed(60);}
void loop(){
  Serial.println(„Forward“);
  stepper.step(STEPS);
  Serial.println(„Backward“);
  stepper.step(-STEPS);}
```

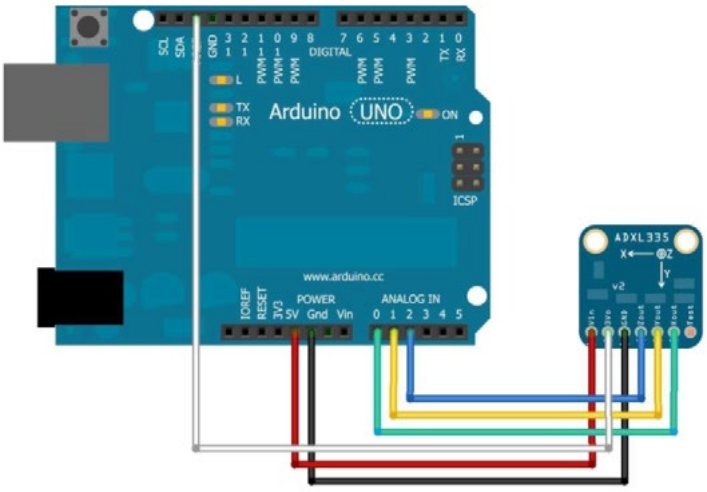
ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца:	**Ученик/ца:**
- објашњава улогу електронских компоненти које су коришћене у пројекту	- описује улогу електромотора - објашњава принцип рада stepper мотора - наводи улогу TB6612 драјвера - разврстава пинове драјвера у три групе - повезује стечена знања из физике, област електрицитет и магнетизам - Разликује Биполарни stepper од Униполарног steppera према њиховој улози и начину рада
- користи електричну шему споја за повезивање Arduin-a и електронских компоненти	- повезује stepper мотор и TB6612 драјвера с Arduino - користи лемилицу уз мјере опреза и надзор наставника/це
- креира програмски код за stepper мотор	- програмира Arduino уводећи Stepper.h библиотеку - реконструише програмски код према сопственим идејама
- верификује и извршава програм	- анализира програмски код - отклања грешке уколико су настале у коду према смјерницама наставника/це
- тестира пројектни задатак	- учитава код на Arduino платформу - анализира пројектни задатак - надограђује програмски код сопственим идејама

Простор за вјежбу:

```
.....
.....
.....
 void setup() {
.....
.....
.....
}
void loop() {
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
}
```

Пројекат 8.
ADXL335 троосни акцелерометар

У овом пројекту користићемо један MEMC уређај, а MEMC је скраћеница од микро-електро-механички систем. Сам сензор састоји се од комбинације микромашински обрађене структуре на силиконској плочици. Конструкција је подупрта полисилицијским опругама које јој омогућују скретање у моменту убрзања по x,y и z оси. Отклон узрокује промјену капацитивности између фиксних плоча и плоча причвршћених на висећу конструкцију. Ова промјена капацитивности на свакој оси претвара се у излазни напон пропорционалан убрзању по тој оси. Уреду, ово звучи баш компликовано, али ми свакодневно користимо добробити овакве технологије у својим smart телефонима. Осим овог интегрисаног кола на PCB акцелерометар се налази и један напонски регулатор, помоћу којег је олакшана интеграција овог уређаја с Arduino развојном платформом.



Слика 8.1. Шема спајања ADXL335 сензора

НАПОМЕНА:
Ако спојите пин EVO на пин AREF, морате аналогну референцу поставити на EXTERNAL прије позивања аналогРеад() у функцији setup(). У супротном ћете интерну референцу кратко спојити са спољном референцом, што би могло да оштети вашу Arduino плочу.

Надамо се да видите да ваши пројекти постају све интересантнији, те да вам одмах падају на памет неке нове идеје како бисте овај сензор могли да искористите у неком од претходних пројеката, а ево поново неколико идеја:

1. Контрола смјера вртње мотора
2. Праћење нагиба при кретању аутономног аутомобила
3. Интеграција 2 x 16 LCD дисплеја у пројекат

Потребни елементи за реализацију пројекта су:

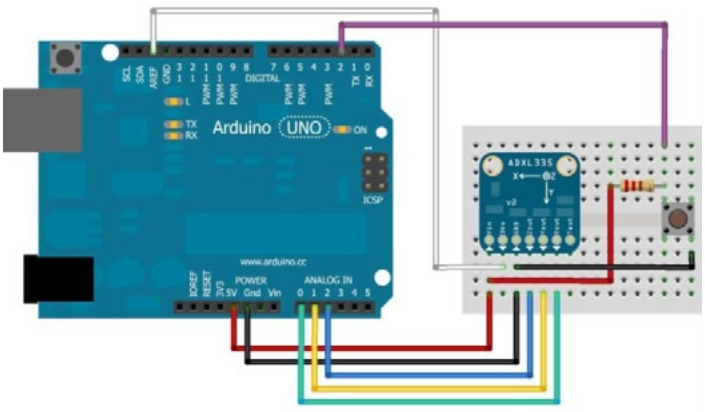
1. ARDUINO UNO
2. 220 Ω потенциометар
3. Матадор плоча
4. Тастер
5. Каблићи

Као и код већине сензора, постоје неке варијације код излазних вриједности са сензора. За неке некритичне апликације типа контролера за игре или детекције нагиба ове варијације нису тако „страшне“ и можемо да их занемаримо у изради своје апликације. Али за апликације када се мора стриктно водити рачуна о прецизности мјерења, калибрација сензора је најбоља метода.

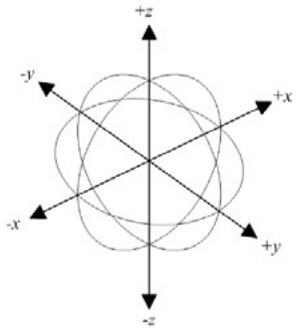
Убрзање код овог сензора мјери се у јединицама гравитационе силе, гдје 1 G представља гравитационо повлачење на површини. Пошто је гравитациона сила стабилна величина, то је чини добром референцом за калибрацију, осим ако не живите у облацима 😊. Само мала напомена прије него пробате калибрисати свој сензор, пропорционални излаз овог сензора значи да се излазни напон линеарно повећава с убрзањем у распону од 0 V при -3 G до 3.3V при +3 G.

За калибрацију сензора на гравитациону референцу потребно је одредити излаз сензора када је он прецизно позициониран спрам осе гравитационе силе. Наравно, у специјализованим лабораторијама користе се посебни стројеви или прихвати за ово, али ви ћете добити добре резултате и без тога.

Као прво, потребно је на стандардну шему спајања додати још два елемента као на наредној слици, додајући на матадор плочу 220 Ω отпора и један тастер.



Слика 8.2. Калибрисање ADXL335 сензора



Слика 8.3. Смјер дјеловања гравитационих сила

Процедура за калибрацију је сљедећа:

1. Учитајте код за калибрацију
2. Отворите апликацију Serial Monitor
3. Поставите матадор на равну површину
   - Стисните и држите тастер док се на Serial Monitor-у не испише „Calibrate“
   - На овај начин сте калибрисали минималну вриједност за **-**З осу
4. Поставите матадор на предњу ивицу и поновите поступак за калибрацију **+Y**
5. Поновите процедуру за остале три ивице матадор плоче за калибрацију **+X,-Y** и**-X**.
6. Окрените матадор наопако и урадите процедуру за калибрацију **+**З.

Након завршене калибрације на серијском монитору добићете сирове калибрационе податке за сваку осу. Ови подаци могу да се користе као константе у вашим апликацијама.

```
RawRanges: X: 408-616, Y: 398-610, Z: 422-625511, 511, 625::-0.01G, 0.07G, 1.00G
RawRanges: X: 408-616, Y: 398-610, Z: 422-625511, 511, 625::-0.01G, 0.07G, 1.00G
RawRanges: X: 408-616, Y: 398-610, Z: 422-625511, 511, 625::-0.01G, 0.07G, 1.00G
RawRanges: X: 408-616, Y: 398-610, Z: 422-625511, 511, 625::-0.01G, 0.07G, 1.00G
```

Arduino код:

```
const int xInput = A0;
const int yInput = A1;
const int zInput = A2;
const int buttonPin = 2;

// „Сирови“ подаци
// Иницијализација на „средину могућег распона података”
int xRawMin = 512;
int xRawMax = 512;

int yRawMin = 512;
int yRawMax = 512;

int zRawMin = 512;
int zRawMax = 512;

// Multisampling ради смањења сметњи
const int sampleSize = 10;

void setup()
{
  analogReference(EXTERNAL);
  Serial.begin(9600);
}

void loop()
{
  int xRaw = ReadAxis(xInput);
  int yRaw = ReadAxis(yInput);
  int zRaw = ReadAxis(zInput);

  if (digitalRead(buttonPin) == LOW)
  {
    AutoCalibrate(xRaw, yRaw, zRaw);
  }
  else
  {
    Serial.print("Raw Ranges: X: ");
    Serial.print(xRawMin);
    Serial.print("-");
    Serial.print(xRawMax);

    Serial.print(", Y: ");
    Serial.print(yRawMin);
    Serial.print("-");
    Serial.print(yRawMax);

    Serial.print(„, Z: „);
    Serial.print(zRawMin);
    Serial.print(„-");
    Serial.print(zRawMax);
    Serial.println();
    Serial.print(xRaw);
```

```
    Serial.print(„, „);
    Serial.print(yRaw);
    Serial.print(„, „);
    Serial.print(zRaw);

    // Претварање сирових података на ‘milli-G-s”
    long xScaled = map(xRaw, xRawMin, xRawMax, -1000, 1000);
    long yScaled = map(yRaw, yRawMin, yRawMax, -1000, 1000);
    long zScaled = map(zRaw, zRawMin, zRawMax, -1000, 1000);

    // Ре-скалирање на убрзање по осама
    float xAccel = xScaled / 1000.0;
    float yAccel = yScaled / 1000.0;
    float zAccel = zScaled / 1000.0;

    Serial.print(" :: ");
    Serial.print(xAccel);
    Serial.print("G, ");
    Serial.print(yAccel);
    Serial.print("G, ");
    Serial.print(zAccel);
    Serial.println("G");

  delay(500);
  }
}

//
// Читање "sampleSize" узорака и тражење средње вриједности
//
int ReadAxis(int axisPin)
{
  long reading = 0;
  analogRead(axisPin);
  delay(1);
  for (int i = 0; i < sampleSize; i++)
  {
    reading += analogRead(axisPin);
  }
  return reading/sampleSize;
}

//
// Тражење екстремних вриједности за сваку осу
//
void AutoCalibrate(int xRaw, int yRaw, int zRaw)
{
  Serial.println("Calibrate");
  if (xRaw < xRawMin)
  {
    xRawMin = xRaw;
  }
  if (xRaw > xRawMax)
  {
    xRawMax = xRaw;
```

```
  }

  if (yRaw < yRawMin)
  {
    yRawMin = yRaw;
  }
  if (yRaw > yRawMax)
  {
    yRawMax = yRaw;
  }

  if (zRaw < zRawMin)
  {
    zRawMin = zRaw;
  }
  if (zRaw > zRawMax)
  {
    zRawMax = zRaw;
  }
}
```

Као што видите, ова апликација за калибрацију мало је компликованија од свега што сте до сада радили, али то само тако изгледа. Двије су ствари које су нове и то су екстерне функције, при чему је једна типа **int**, а друга типа **void**. Прво ћемо објаснити функцију Auto Calibrate() која не враћа ништа (повратни тип је void). Наиме, та функција може да има аргументе као у примјеру:

```
void AutoCalibrate(int xRaw, int yRaw, int zRaw)
```

Дакле, код декларације функције унутар заграда имамо могуће аргументе, у овом случају три аргумента типа int. Код позива ове функције унутар loop петље ми само тој функцији прослиједимо три аргумента, коју она искористи за прорачун, али не врати нам никакву вриједност.

```
if (digitalRead(buttonPin) == LOW)
   {
    AutoCalibrate(xRaw, yRaw, zRaw);
   }
```

Други тип функције који видимо у претходној апликацији је функција која има „повратну“ вриједност, односно након прорачуна враћа вриједност int типа.

```
int ReadAxis(int axisPin)
{
  long reading = 0;
  analogRead(axisPin);
  delay(1);
  for (int i = 0; i < sampleSize; i++)
  {
    reading += analogRead(axisPin);
  }
  return reading/sampleSize;
}
```

Функцију декларишемо на начин да испред назива функције дефинишемо тип података који ће та функција након прорачуна вратити. У овом конкретном случају функција има један аргумент типа int

```
(int axisPin)
```

Након прорачуна вратиће:

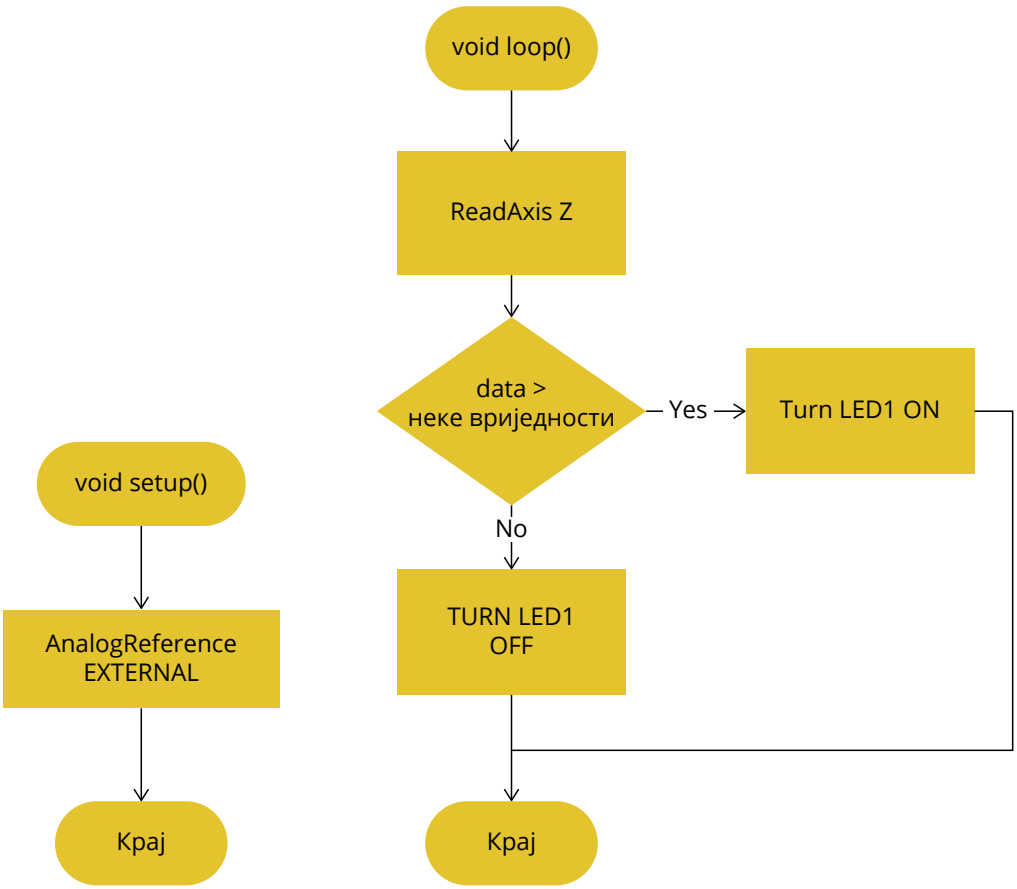
```
return reading/sampleSize;
```

Позив функције у loop петљи урађен је на сљедећи начин:

```
int xRaw = ReadAxis(xInput);
int yRaw = ReadAxis(yInput);
int zRaw = ReadAxis(zInput);
```

Па и није тако страшно, овако у наредним пројектима можете да пишете мало озбиљније апликације, јер можете сегменте апликације да издвојите из loop петље те да почнете писати функције које ће вам олакшати рјешавање проблема. Уз напомену: *keep it simple!*

За крај овог пројекта остали смо дужни поједностављену алгоритамску структуру за један једноставан пројекат са ADXL335.



Слика 8.4. Поједностављена алгоритамска структура апликације

ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца: - описује улогу електронских компоненти које су коришћене у пројекту - проширује стечена знања из области гравитације	**Ученик/ца:** - описује улогу MEMC уређаја - проширује стечена знања из области гравитације - објашњава улогу ADXL335 троосног брзиномјера - користи Serial Monitor за учитавање резултата
- користи електричну шему споја за повезивање Arduin-a и електронских компоненти	- извршава калибрацију ADXL335 сензора према шеми споја и процедури - повезује преостале компоненте у пројекту према шеми споја
- креира алгоритамску структуру споја	- процјењује предности и ограничења алгоритамског приступа у рјешавању проблема - визуализира пројектни задатак кроз алгоритамску шему
- креира програмски код за даљинско управљање	- програмира пројекат уз адекватне смјернице наставника/це - декларише функције и промјењиве у програмском коду - реконструише програмски код према сопственим идејама
- верификује и извршава програм	- анализира програмски код - отклања грешке уколико су настале у коду према смјерницама наставника/це
- тестира пројектни задатак	- учитава код на Arduino платформу - користи Serial Monitor за тестирање пројектног задатка - анализира пројектни задатак - Развија сопствене идеје за примјену MEMC уређаја

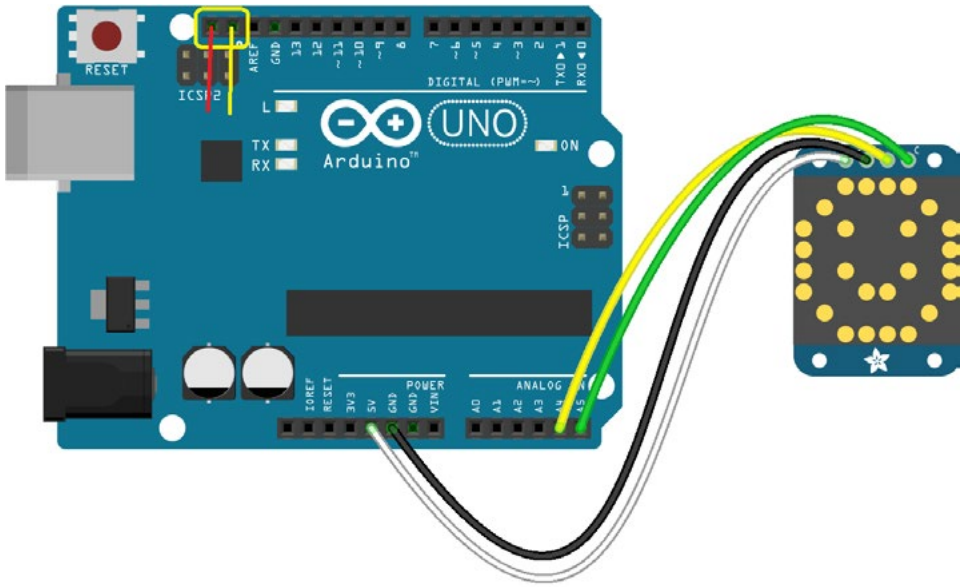
Простор за вјежбу:

```
 void setup() {

}
void loop() {

}
```

Пројекат број 9.
LED Matrix драјвер HT16K33/MAX7219

Поново LED, али само 64 комада истих. Одличан начин да направимо мали дисплеј који користи матрицу од 8 x 8 LED, којом управља XT16K33 чип, који има способност да управља мултиплексираном 16 x 8 матрицом, што представља 128 LED-а. Импресивно! Комуникациони протокол који чип користи је I²C, стога су нам, као и код серијске комуникације за трансфер података, довољна само два пина, при чему нећемо детаљно улазити у I²C тип комуникације, али се мора рећи да је то комуникациона сабирница базирана на комуникацији између *мастер* уређаја (Arduino) и *славе* уређаја, од којих сваки има јединствену адресу. Како LED Matrix драјвер долази у виду breakout board-а, на себи има опцију да можемо подесити осам различитих I²C адреса, те на тај начин можемо да контролишемо 1024 LED-а.

Важно је напоменути да чип има интегрисан модул за држање константне струје управљања теLEDматрица увијек има конзистентно исијавање свјетлости на LED-у. Такође је могуће мијењати интензитет свјетлости свих LE диода истовремено (није могуће појединачно) у матрици и то у 16 различитих нивоа.



Слика 9.1. Шема спајања LED матрице

Уреду, потпуно вас разумијем ако вољеној особи желите на овој матрици нацртати срце, и сами смо то прво урадили 😊. Али, као и увијек до сада, ево неколико идеја које можете да искористите за неки upgrade почетног пројекта:

1. Бец реклама
2. Игра балансирања с ADXL335
3. Scrolling реклама с више дисплеја

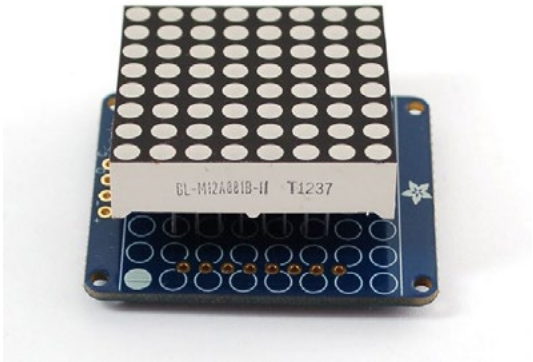
Наравно, не морате стати на овоме уколико добијете неку своју идеју, слободно је имплементирајте. На крају, истраживањем се највише учи. Сваки од наведених приједлога као и у свим другим пројектима можете да надоградите другим компонентама те тако учините пројекат занимљивим.

Потребни елементи за реализацију пројекта су:

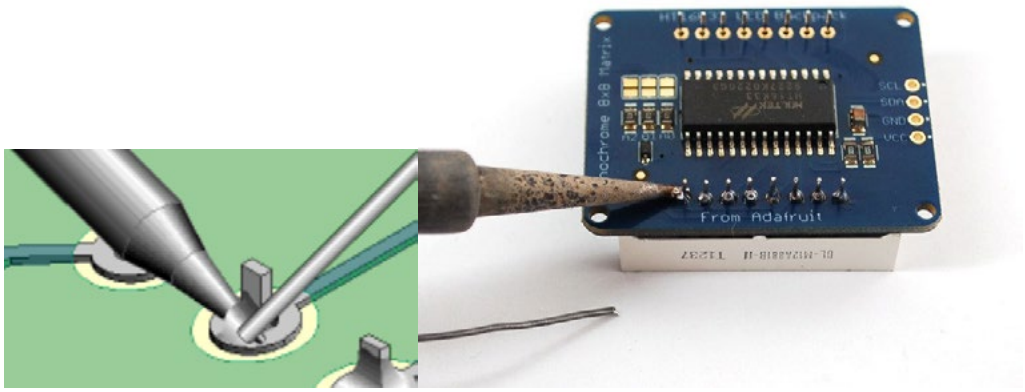
1. ARDUINO UNO
2. XT16K33 backpack
3. 8 x 8 LED матрица
4. Матадор плоча
5. Каблићи

Пошто у већини случајева овај драјвер и дисплеј буду достављени као DIY kit, приложено вам је и упутство за састављање прије него га почнете тестирати.

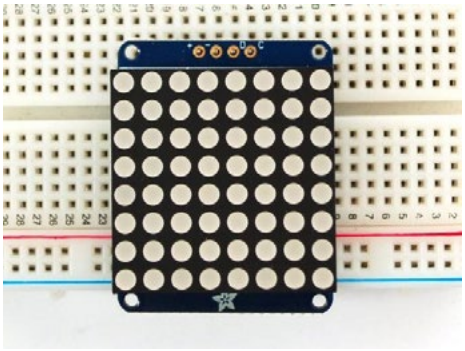
Корак 1: Подесите дисплеј као на сљедећој слици, текст типа дисплеја треба да стоји под 90° у односу на пинове комуникационе сабирнице.



Корак 2: Користећи лемилицу залемите пинове дисплеја за PCB. Напомена: тинол жицу додати на већ претходно загријано мјесто за лемљење и при извлачењу повући лемилицу уз пин дисплеја.

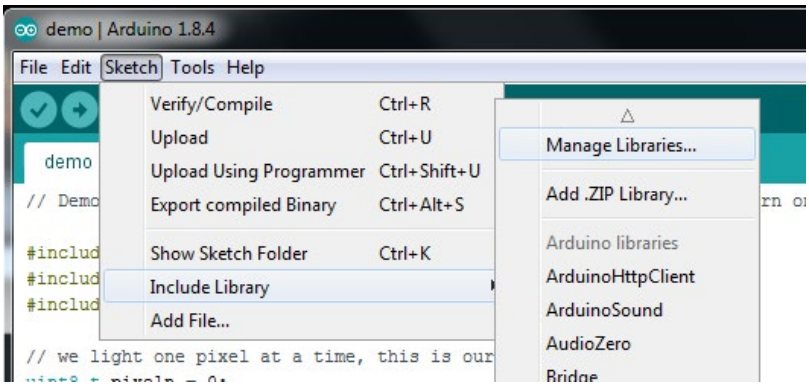


Корак 3: Убацити 4-пински конектор у матадор плочу и потом монтирати bacpack PCB као што је приказано на слици. Пинове залемити користећи исту технику као што је напоменуто.



Пошто ће се за овај пројекат користити готова библиотека, биће приказано како ју је могуће инсталирати користећи Library Manager. Осим библиотеке Adafruit LED Backpack, потребно је инсталирати и библиотеку Adafruit GFX.

Прво је потребно из менија Sketches изабрати опцију *Include Library* па потом изабрати опцију *Manage Libraries*.



Слика 9.2. Путања до опције Manage Libraries

У pop-up прозору у поље за тражење упишите Adafruit LED backpack и исту библиотеку инсталирајте, исто поновите и за библиотеку Adafruit GFX. Уколико користите старију верзију Arduino IDE од 1.8.10, потребно је да инсталирате и библиотеку Adafruit_DusIO.

Ако сте пажљиво пратили упутство, онда **ћете** у свом IDE-у на путањи **File->Examples->Adafruit_LEDBackpack->matrix88 наћи демо-скицу коју је потребно учитати. Пошто је скица интелектуална својина, оставићемо неизмијењене неке постојеће коментаре.**

Arduino код:

```
 This is a library for our I2C LED Backpacks

  Designed specifically to work with the Adafruit LED Matrix backpacks
  ----> http://www.adafruit.com/products/872
  ----> http://www.adafruit.com/products/871
  ----> http://www.adafruit.com/products/870

  These displays use I2C to communicate, 2 pins are required to
  interface. There are multiple selectable I2C addresses. For backpacks
  with 2 Address Select pins: 0x70, 0x71, 0x72 or 0x73. For backpacks
  with 3 Address Select pins: 0x70 thru 0x77

  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  BSD license, all text above must be included in any redistribution
 ****************************************************/

#include <Wire.h>
#include <Adafruit_GFX.h>
#include "Adafruit_LEDBackpack.h"

Adafruit_8x8matrix matrix = Adafruit_8x8matrix();

void setup() {
  Serial.begin(9600);
  Serial.println("8x8 LED Matrix Test");

  matrix.begin(0x70);  // Иницијализуј с наведеном i2c адресом.
}

static const uint8_t PROGMEM
  smile_bmp[] =
  { B00111100,
    B01000010,
    B10100101,
    B10000001,
    B10100101,
    B10011001,
    B01000010,
    B00111100 },
  neutral_bmp[] =
  { B00111100,
    B01000010,
    B10100101,
    B10000001,
    B10111101,
    B10000001,
    B01000010,
    B00111100 },
  frown_bmp[] =
```

```
  { B00111100,
    B01000010,
    B10100101,
    B10000001,
    B10011001,
    B10100101,
    B01000010,
    B00111100 };

void loop() {
  matrix.clear();
  matrix.drawBitmap(0, 0, smile_bmp, 8, 8, LED_ON);
  matrix.writeDisplay();
  delay(500);

  matrix.clear();
  matrix.drawBitmap(0, 0, neutral_bmp, 8, 8, LED_ON);
  matrix.writeDisplay();
  delay(500);

  matrix.clear();
  matrix.drawBitmap(0, 0, frown_bmp, 8, 8, LED_ON);
  matrix.writeDisplay();
  delay(500);

  matrix.clear();      // Очисти дисплеј.
  matrix.drawPixel(0, 0, LED_ON);
  matrix.writeDisplay();  // Прикажи на LED матрици.
  delay(500);

  matrix.clear();
  matrix.drawLine(0,0, 7,7, LED_ON);
  matrix.writeDisplay();  // Прикажи на LED матрици.
  delay(500);

  matrix.clear();
  matrix.drawRect(0,0, 8,8, LED_ON);
  matrix.fillRect(2,2, 4,4, LED_ON);
  matrix.writeDisplay();  // Прикажи на LED матрици.
  delay(500);

  matrix.clear();
  matrix.drawCircle(3,3, 3, LED_ON);
  matrix.writeDisplay();  // Прикажи на LED матрици.
  delay(500);

  matrix.setTextSize(1);
  matrix.setTextWrap(false);  // Не желимо wrap текст, већ скролан.
  matrix.setTextColor(LED_ON);
  for (int8_t x=0; x>=-36; x--) {
    matrix.clear();
    matrix.setCursor(x,0);
    matrix.print("Hello");
    matrix.writeDisplay();
    delay(100);
```

```
  }
  matrix.setRotation(3);
  for (int8_t x=7; x>=-36; x--) {
    matrix.clear();
    matrix.setCursor(x,0);
    matrix.print("World");
    matrix.writeDisplay();
    delay(100);
  }
  matrix.setRotation(0);
}
```

Сада када сте видјели како ваша LED матрица ради, можете почети да имплементирате своје идеје. Матрица 8 x 8 подржава све функције из библиотеке Adafruit GFX – цртање пиксела, линије, правоугаоника, као и малих битмапа.

Потребно је споменути двије функције које вам могу користити у пројектима, а то су:

- **setBrightness(*brightness*)** - омогућава вам подешавање интензитета свјетлости читавог дисплеја на скали од 0 до 15
- **blinkRate(*rate*)** – омогућава блинкање цијелог дисплеја на скали од 0 до 3.

LED Matrix је интересантан gadget и овог пута у пројекту ће бити изостављена алгоритамска структура којом сугеришемо задатак и пустићемо вашој машти на вољу да осмислите интересантан пројекат, не сумњамо да ћете успјети. Сретно!

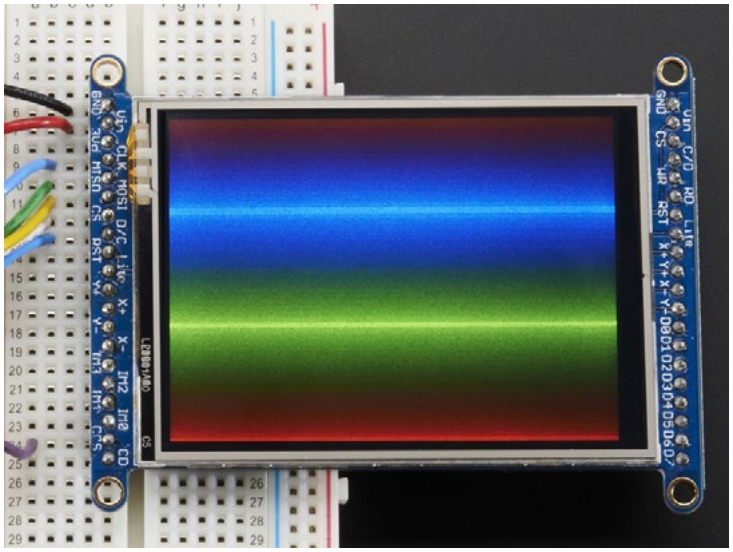
ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца: - описује улогу електронских компоненти које су коришћене у пројекту	**Ученик/ца:** - креира мали дисплеј који користити матрицу од 8 x 8 LE диода која је управљана HT16K33 чипом - објашњава улогу I^2C типа комуникације - описује основне карактеристике примјене HT16K33 чипа - наводи улоге setBrightness(brightness) и blinkRate(rate) функција
- користи електричну шему споја за повезивање Arduin-a и електронских компоненти	- саставља драјвер и дисплеј према приложеном упутству DIY kit-a - спаја пинове микроконтролера према шеми споја - повезује преостале компоненте у пројекту према шеми споја
- креира програмски код за даљинско управљање	- програмира Arduino за употребу LED Matrix драјвера HT16K33/MAX7219 - примјењује стечено знање из низова и матрица - учитава демо-скицу File->Examples->Adafruit_LEDBackpack->matrix88 - реконструише програмски код према сопственим идејама
- користи одговарајуће библиотеке унутар програма за реализацију пројектног задатка	- инсталира библиотеке Adafruit LED Backpack i Adafruit GFX из Library Manager-a - по потреби инсталира и библиотеку Adafruit_DusIO у зависности од верзије Arduino IDE - позива библиотеке унутар програмског кода
- верификује и извршава програм	- анализира програмски код - отклања грешке уколико су настале у коду према смјерницама наставника/це
- тестира пројектни задатак	- учитава код на Arduino платформу - извршава тестирање дисплеја - анализира пројектни задатак - користи дисплеј у другим пројектним идејама - развија сопствене идеје за примјену дисплеја

Простор за вјежбу:

```
 void setup() {

}
void loop() {

}
```

Пројекат број 10.
TFT LCD с екраном осјетљивим на додир

Постајемо дигитални. За разлику од нас, наши ученици и ученице потпуно су TFT генерација. Наиме, одмакнућемо се од матрица и 2 x 16 LCD дисплеја и крећемо у пројекат који је мало тежи, али је више забаван. TFT LCD је облик LCD дисплеја који користи технологију танког филм-транзистора, стога назив TFT (скраћеница од Thin-Film-Transistor). Дисплеј посједује и резистивни *touch,* тако да је могуће направити и интерактивну апликацију. Сам дисплеј посједује микроконтролер с RAM међуспремником, што олакшава његову контролу. Када смо већ код контроле, овај дисплеј може да ради у два мода, 8-битном и SPI моду. За 8-битни мод вам је потребно 12 дигиталних пинова, док SPI мод захтијева само 5 пинова за комуникацију и 4 пина за резистивни *touch*, при чему је ипак спорији од 8-битног мода. Дисплеј се може набавити у величинама од 2.8“ или 3.2“ с резолуцијом од 240 x 320 RGB пиксела.



Слика 10.1. Приказ боја на TFT дисплеју

Можда вам тренутно пада на памет да креирате апликацију која ће визуелно лијепо приказати температуру у просторији у којој боравите и уз то нацртати графикон промјене температуре у времену, али ево неколико идеја које можете да искористите за неки upgrade овог почетног пројекта:

1. Једноставна апликација за цртање
2. TFT џепна реклама
3. Log-on конзола

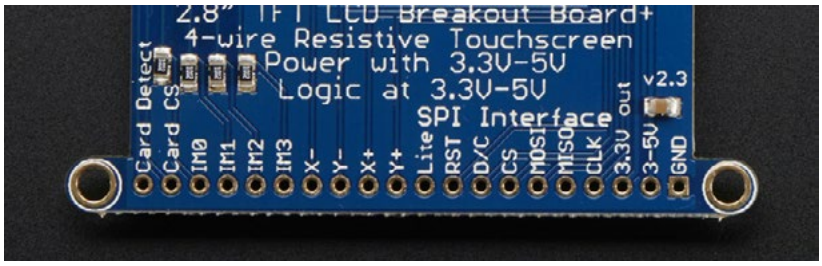
Сваки од поменутих приједлога можете да надоградите уводећи додатне компоненте.

Када проширите своје знање, онда неће бити тешко на Arduino платформу спојити Bluetooth модул или, што да не, WiFi модул те направити малу временску станицу.

Потребни елементи за реализацију пројекта су:

1. ARDUINO UNO
2. Матадор плоча
3. Каблићи
4. Adafruit ILI9341 TFT

SPI мод је популарнији те омогућава и коришћење SD картице и, иако је бржи два до три пута у односу на 8-битни мод, ову вјежбу реализоваћемо управо користећи тај специјални облик серијске комуникације. На тај начин смањићемо и могућност за грешку приликом ожичења дисплеја.

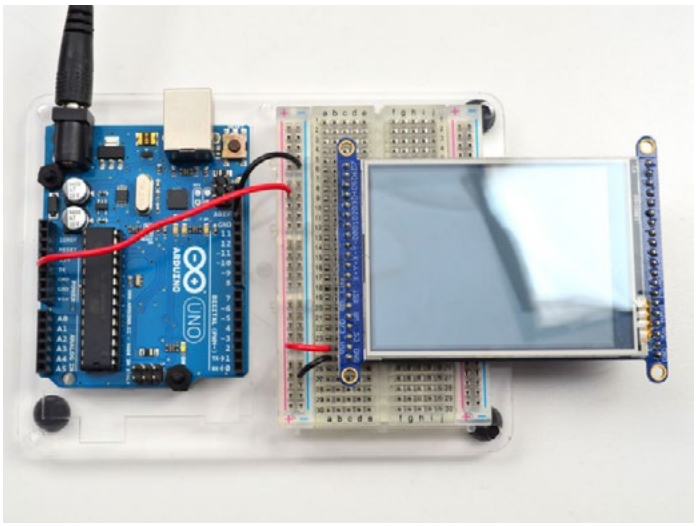


Слика 10.2 Pinout SPI интерфејса на TFT дисплеју

- **GND** - пин
- **3-5 V / Vin** – пин за напон 3 – 5 V DC, посједује заштиту од замјене поларитета!
- **3.3 Vout** – излаз с регулатора напона 3.3 V
- **CLK** - улазни SPI clock пин
- **MISO** – скраћено од *Microcontroller In Serial Out* пин, обично се користити за SD картицу или за дебугирање, није потребан за употребу TFT дисплеја у write-only моду
- **MOSI** – скраћено од SPI *Microcontroller Out Serial In* пин, користи се за слање података с микроконтролера на SD картицу или TFT дисплеј
- **CS** - TFT SPI chip select пин
- **D/C** - TFT SPI data ili *command selector* пин
- **RST** - TFT reset пин (спојити га с масом да ресетујете TFT)
- **Lite** - PWM улаз за контролу позадинског освјетљења
- **IM3, IM2, IM1, IM0** – jumperi за одабир мода
- **Card CS / CCS** - SD card chip select, користе се за читање са SD картице
- **Card Detect / CD** - SD card detect пин

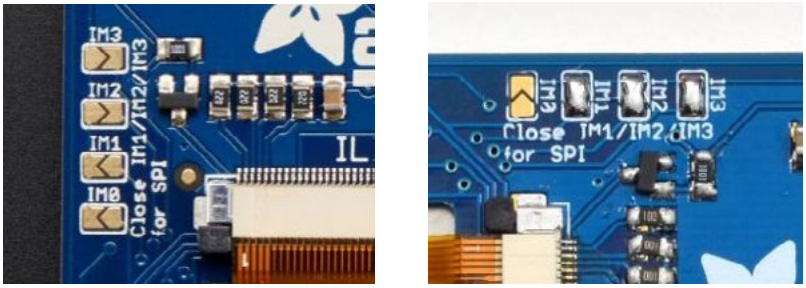
Уколико желите користити резистивни *touch*, пинови **Y+ X+ Y-X-** могу да се споје на аналогне улазе за одређивање положаја тачке додира.

Прије него кренемо у тестирање дисплеја, прво је потребно урадити тест дисплеја. Наиме, потребно је спојити 3 и 5V пинове на 5V а GND пинове на GND пин Arduin-а, како је приказано на сљедећој слици.



Слика 10.3. Тест TFT -а

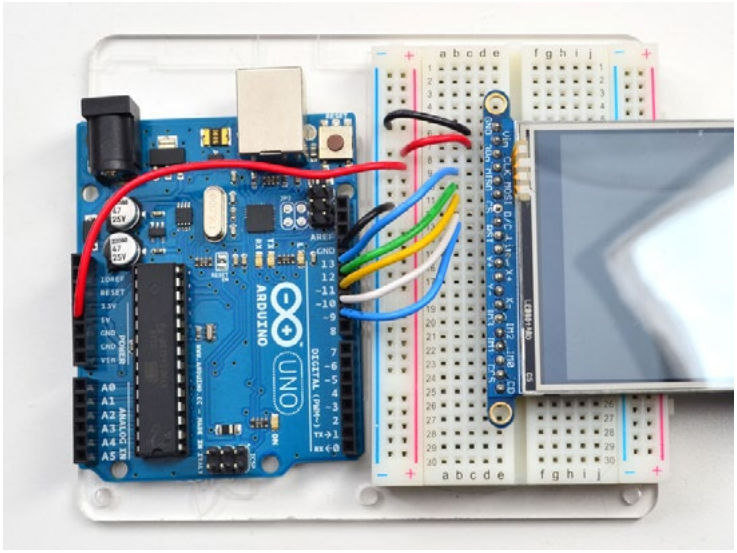
Уколико се укључи позадинско освјетљење, корак ближе смо успјешној реализацији пројекта. Потом је потребно дисплеј ставити у SPI мод. На сљедећој слици приказани су IMx jumperi које је потребно кратко спојити тинол жицом и лемилицом. Дакле, пинове IM1,IM2 і IM3 потребно је кратко спојити, а пин IM0 оставити у стању у каквом јесте.



Слика 10.4. IMx jumperi

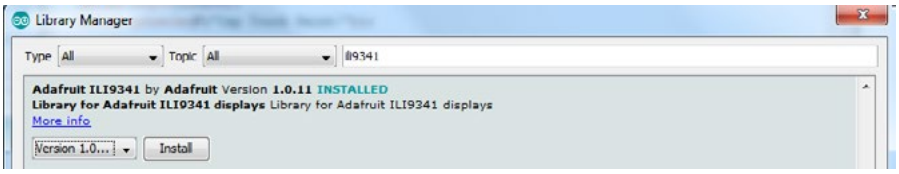
Сада је још потребно спојити пет каблића за SPI комуникацију.

- CLK -> Digital 13
- MISO -> Digital 12
- MOSI -> 11
- CS -> Digital 10
- D/C -> Digital 9



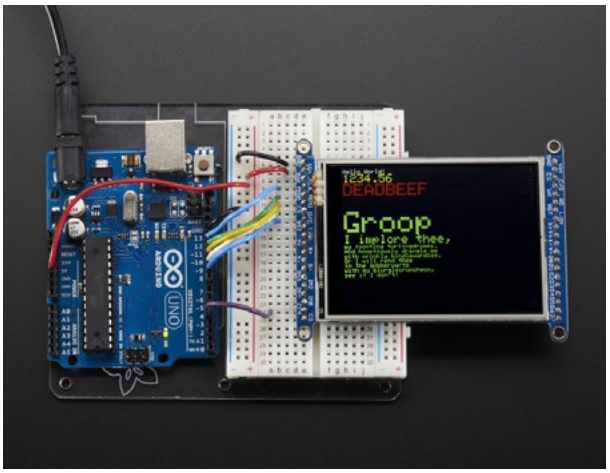
Слика 10.5. Шема спајања TFT-а

И овог пута инсталираћемо библиотеку која ће нам олакшати креирање наше апликације. На исти начин као и у претходном пројекту потребно је користећи Library Manager додати библиотеку Adfruit ILI9341 TFT. Уколико вам прије инсталирања библиотеке IDE нагласи да су потребне још неке библиотеке, будите слободни и потврдите да желите да их инсталирате.



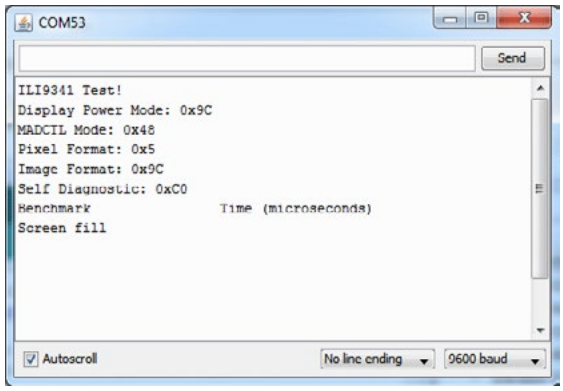
Слика 10.6. Библиотека ILI9341

Након инсталације библиотеке и ресета Arduino DE-а, потребно је у примјерима наћи апликацију **Graphictest** и учитати је. Резултат би требало да изгледа као на сљедећој слици.



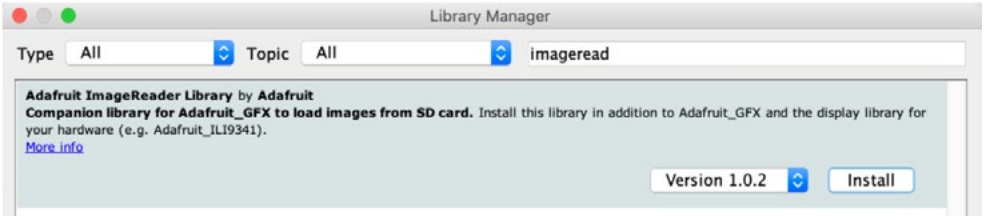
Слика 10.7. Тест TFT app

Уколико нисте успјели из првог покушаја добити овакав приказ на свом TFT-у, потребно је отворити апликацију Serial Monitor и провјерити јесте ли добили feedback као на сљедећој слици. Ако то није случај, не заборавите провјерити ожичење.



Слика 10.8. Тест TFT-а у апликацији Serial Monitor

Пошто је на TFT модулу монтирана и SD картица, то нам отвара могућност читања слика које су спремљене у 24-битном BMP формату и нису веће од 240 x 230 пиксела. Прије је потребно инсталирати још једну библиотеку кроз Library Manager и то **Imageread.**

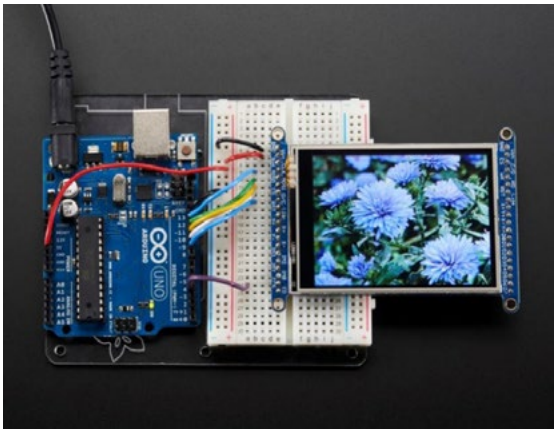


Слика 10.9. Библиотека Imageread

Након инсталације библиотеке потребно је спојити **CCS** пин на **Digital4** пин микроконтролера, потом наћи и припремити слику у препорученом формату, те је потом пребацити на SD картицу. Имена фајлова морају да имају мање од осам карактера, а позивамо их на сљедећи начин:

bmpDraw(bmpfilename, x, y);

Ако сте добро пратили упутство, резултат би требало да изгледа као на сљедећој слици.



Слика 10.10. SD bitmap read

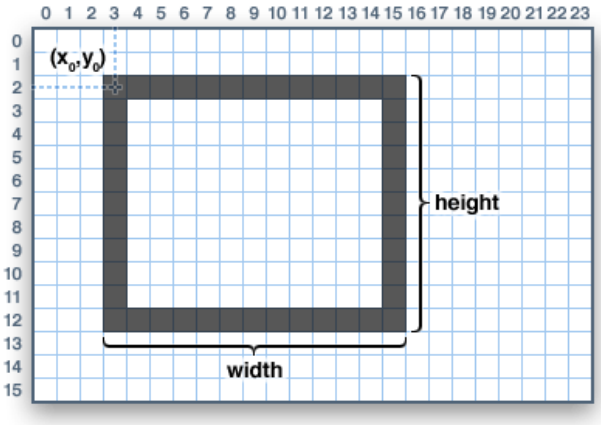
И за сами крај, gadget који користимо свакодневно и због којег нас много не зна више користити онај стари телефон с бројчаником. Иако овај модел TFT-а може да има и капацитивни touchscreen, због ниже цијене много чешће се сусреће резистивни touch. Потребно је прво инсталирати библиотеку **Adafruit TouchScreen**, те оживити Arduino на сљедећи начин:

- Y+ to Arduino **A2**
- X+ to Arduino **D9** (Same as D/C)
- Y- to Arduino **D8**
- X- to Arduino **A3**

Arduino код:

```
#include <stdint.h>
#include "TouchScreen.h"

#define YP A2  // Analogni pin 2!
#define XM A3  // Analogni pin 3!
#define YM 8   // digital pin 8
#define XP 9   // digital pin 9

// За боља очитања потребно је измјерити вриједност отпора између X+ //
и X- користећи Multimetar.
// За примјер је наведена вриједност 300 ома.
// Креирање нове инстанце класе ts.
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);

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

void loop(void) {
  // Point објекат „чува" вриједности координата.

  TSPoint p = ts.getPoint();

  // За сваки притисак већи од 0 рећи ћемо да је valid.

  if (p.z > ts.pressureThreshhold) {
     Serial.print("X = "); Serial.print(p.x);
     Serial.print("\tY = "); Serial.print(p.y);
     Serial.print("\tPressure = "); Serial.println(p.z);
  }

  delay(100);
}
```

Овај дио кода не треба да посматрате као завршен пројекат. За крај овог пројекта покушаћемо вам дати приједлог како да посљедњи код надоградите те реализујете једноставну апликацију која ће све одушевити.



Слика 10.11. Организација координатног система на TFT-у

Користећи претходну слику и функције за цртање једноставних геометријских облика из GFX библиотеке, можете да покушате направити два button-а на свом TFT дисплеју.

```
void drawRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color);

void fillRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color);
```

Види се да ове двије наведене функције имају по 5 аргумената, те се може направити повезница са сликом која им претходи. Прва два аргумента представљају почетну позицију правоугаоника, друга два дужину и ширину страница правоугаоника, а задњи аргумент дефинише која је боја линије која твори правоугаоник на TFT дисплеју. Друга функција има исте аргументе, али она испуњава правоугаоник дефинисаном бојом.

Уколико се детектује притисак на touch-у у пољу гдје се налази нацртани правоугаоник, промијените му позадинску боју. Када ово успијете, само небо је граница. Само напријед.

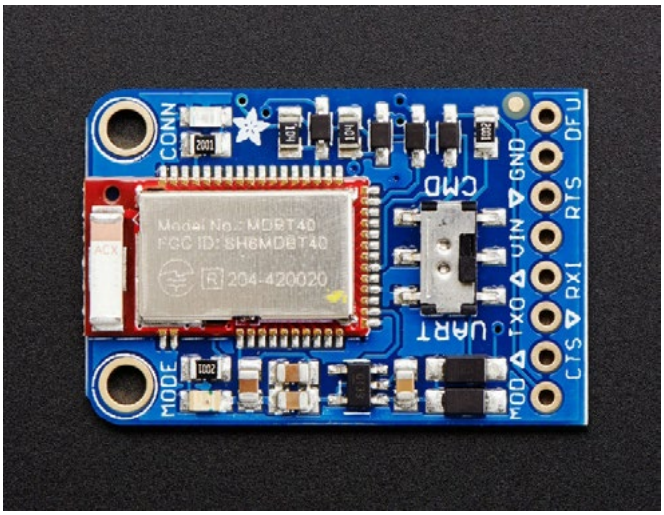
ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца:	**Ученик/ца:**
- описује улогу електронских компоненти које су коришћене у пројекту	- описује карактеристике TFT LCD-а с екраном осјетљивим на додир - препознаје разлику између модова код дисплеја - класификује пинове према 8-битном и SPI моду - наводи улогу пинова SPI интерфејса на TFT дисплеју, по потреби се користи Serial Monitor-ом за учитавање резултата
- користи електричну шему споја за повезивање Arduin-а и електронских компоненти	- спаја пинове микроконтролера према шеми споја - извршава тестирање дисплеја према шеми спајања или смјерницама наставника - повезује преостале компоненте у пројекту према шеми споја
- креира програмски код за даљинско управљање	- програмира Arduino за употребу TFT LCD дисплеја - реконструише програмски код према сопственим идејама
- користи одговарајуће библиотеке унутар програма за реализацију пројектног задатка	- користећи Libray Manager додаје Adfruit ILI9341 TFT библиотеку - проналази Graphictest апликацију ради учитавања - инсталира Imageread библиотеку кроз Libray Manager - по потреби инсталира Adafruit TouchScreen библиотеку - позива библиотеке унутар програмског кода
- верификује и извршава програм	- користи Serial Monitor апликацију ради тестирања дисплеја - извршава тестирање дисплеја према шеми спајања или смјерницама наставника - анализира програмски код - отклања грешке уколико су настале у коду према смјерницама наставника
- тестира пројектни задатак	- учитава код на Arduino платформу - анализира пројектни задатак - примјењује TFT LCD дисплеј у другим пројектним идејама - Развија сопствене идеје за примјену TFT дисплеја - користи TFT дисплеј за креирање button-а и слично

Простор за вјежбу:

```
 void setup() {

}
void loop() {

}
```

Пројекат број 11.
BLE - Bluetooth Low Energy

Вјероватно ћете у једном тренутку жељети на своју апликацију додати и Bluetooth Low Energy модул. Вјероватно знате да су ови BLE модули свеприсутни у нашим smart уређајима, лаптопима па чак и у аутомобилима. Оригиналан назив BLE модула је Bluefruit LE UART Friend, при чему под UART - Universal asynchronous receiver-transmitter подразумијевамо протокол који поједностављено називамо серијском комуникацијом. Помоћу овог модула једноставно је додати BT опцију у апликацију гдје год постоји софтверски или хардверски серијски порт. Дакле, може се спојити на Arduino или неку другу фамилију микроконтролера или, једноставно, на FTDI кабл за debugging и тестирање. Овај мали мултифункционални модул може много тога, али за већину просјечних корисника довољна је опција стандардног Nordic UART Rx/Tx конекцијског профила. Тада се BLE модул понаша као комуникациони тунел или канал који може да шаље и прима податке са iOS или Android апликације. Могуће је користити припадајућу апликацију коју можете да скинете са App Store-а или с Google Play-а.



Слика 11.1. Изглед BLE модула

Слика 11.2. Bluefruit Connect апликација

Потребни елементи за реализацију пројекта су:

1. ARDUINO UNO
2. Матадор плоча
3. Каблићи
4. BlueFruit Android/iOS app
5. Bluefruit LE UART Friend модул

Овај мали модул може много више од пуког бежичног слања string-ова. Он посједује једноставан AT сет команди помоћу којег можемо да контролишемо понашање, па чак и мијењамо начин на који је видљив другим BT уређајима. AT командни сет може да се искористи и за потраживање информација о температури, напону батерије, провјеру MAC адреса и слично, дакле, користимо само мали дио потенцијала овог малог, али моћног уређаја.

Први задатак је да урадите даунлоуд Bluefruit апликације на своје Apple ili Android уређаје. Унутар апликације могуће је одабрати четири главна мода рада и то:

- Color Picker
- Accelerometer Data
- GPS Data
- Control game pad

За почетак у нашем пројекту користићемо овај задњи мод, а ви, како будете напредовали са знањем, можете да искористите неки од других модова.

Да бисте разумјели зашто овај модул има овако велики број могућности коришћења, погледајте његову техничку спецификацију и схватићете да је он програмабилни уређај базиран на nRF51822 ultra low power SoC-y (System on a Chip). Без обзира на то што се један овакав уређај користи тек као tuneling уређај за прослеђивање података с неког smart уређаја, то не умањује тежину и важност овог пројекта. Дио основних техничких спецификација nRF51822 је:

- ARM Cortex M0 core – 16 MHz
- 256 KB flash меморија
- 32 KB SRAM
- 2.4 GHz Transceiver

Прије него опишемо pinout, потребно је навести да се с друге стране модула налази припрема за опциони батеријски прикључак, те опционо мјесто за монтажу 32 KHz осцилатора. Осим овога, на позадини модула су и сљедећи пинови:

SWC: SWC clock пин, 3 V логички ниво – хакерски пинови!

SWD: SWD data пин, 3 V логички ниво – хакерски пинови!

3Vo: Излаз с 3 V напонског регулатора

FCR: Пин за враћање на творничке поставке

Осим ових специјалних пинова на BLE модулу су и стандардни пинови које ћете користити у овом пројекту.

Пинови за напајање:

- **VIN**: пин за напајње напоном од 3.3 до 16 V.
- **GND**: заједнички пин за напајање и логику.

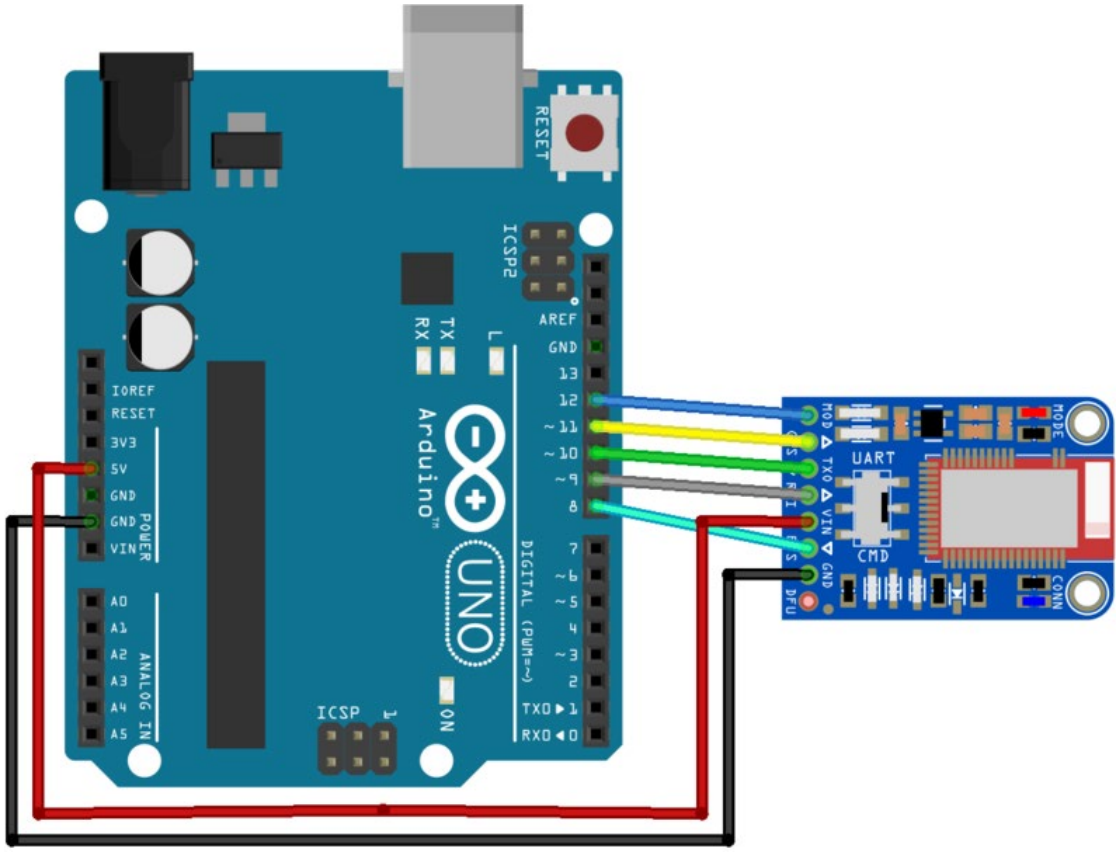
UART пинови:

- **TXO** - UART Transmit пин: пин за слање података (Bluefruit LE --> MCU), 3.3 V логички ниво
- **RXI** - UART Receive пин: пин за примање података (MCU --> Bluefruit LE), 3-5 V логички ниво
- **CTS** – Овај пин је по *дифолту* у стању логичке 0: да бисте обезбиједили трансфер података, потребно је овај пин ставити у стање логичке нуле, 3-5 V логички ниво
- **RTS** - Read to Send пин за контролу протока података

Екстра пинови:

- **МОД**: Селекција мода рада: Bluefruit има два мода рада, командни мод и дата мод
- **DFU**: Мод за update firmware-a OTA (Over the Air)

Иако се BLE модул може спојити на било који микроконтролер који има било хардверски или софтверски UART комуникациони канал, у овом пројекту искористићемо Software-Serial библиотеку и BLE модул спојити на тај софтверски UART порт.

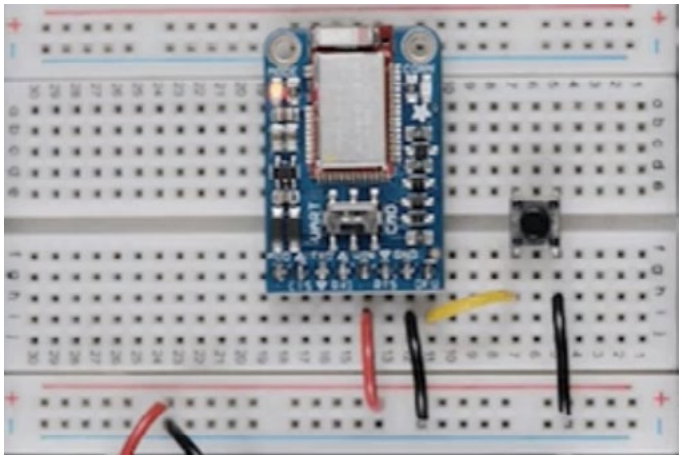


Слика 11.3. Bluefruit шема спајања са софтверским UART портом

Дакле, водећи се претходном сликом, потребно је пинове BLE модула спојити са Arduino UNO-ом на сљедећи начин:

- **МОД** то **Пин 12**
- **CTS** то **Пин 11**
- **TXO** то **Пин 10**
- **RXI** то **Пин 9**
- **VIN** то **5V**
- **RTS** то **Пин 8**
- **GND** то **GND**

Прије него почнемо с примјерима, потребно је да упознате макар један од начина на враћање на творничке поставке. BLE треба да вратите на творничке поставке уколико сте га, рецимо, погрешно конфигурисали. Користићемо, можда, најједноставнији начин, поготово ако је на BLE модул постављен DFU тастер. Наиме, довољно је тастер држати дуже од 5 секунди док је BLE под напоном. При томе ће прво заблинкати црвена LED па потом плава LED и то је знак да је BLE ресетован. Уколико DFU тастер није монтиран на BLE модул, потребно је оживити BLE као на сљедећој слици и провести процедуру ресетовања као и с монтираним DFU тастером.



Слика 11.4. Bluefruit Factory reset шема

Поново ћемо, користећи већ познату процедуру, инсталирати Bluefruit библиотеку помоћу Library Manager-а. Како је библиотека универзално написана за широки спектар уређаја, потребно је прије самог коришћења примјера из библиотеке подесити библиотеку за BLE модул.

Прије свега, прегледаћемо припадајући header фајл *BluefruitConfig.h* у којем можемо да мијењамо неке од стандардних поставки у апликацији. На примјер, ако вам је пин 12 заузет, можете да урадите сљедеће:

```
// Заједничка UART подешавања
// ------------------------------------------------------------------------
-----
// Опционо подешавање Mode пина, препоручује се, али није обавезно.
// ------------------------------------------------------------------------
-----
#define BLUEFRUIT_UART_MODE_PIN       3    // Поставите на -1 ако се пин
не користи.
```

Можете у *BluefruitConfig.h* пин 3 прогласити МОД пином у својој апликацији, испод је комплетан оригинални header фајл:

```
// СТАНДАРДНА ПОДЕШАВАЊА
// ------------------------------------------------------------------------
-----------------
// Подешавања за SW UART, HW UART i SPI
// ------------------------------------------------------------------------
-----------------
#define BUFSIZE                          128   // Величина buffer-a за долажење
податке
#define VERBOSE_MODE                    true  // True омогућава debug


// ПОДЕШАВАЊА СОФТВЕРСКОГ UART-А
// ------------------------------------------------------------------------
-----------------
// Сљедећи macro-и декларишу који ће пинови бити кориштени за SW серијску
комуникацију
//
// ------------------------------------------------------------------------
-----------------
#define BLUEFRUIT_SWUART_RXD_PIN        9    // rx sw serial пин
#define BLUEFRUIT_SWUART_TXD_PIN        10   // tx sw serila пин
#define BLUEFRUIT_UART_CTS_PIN          11   // cts sw serial пин
#define BLUEFRUIT_UART_RTS_PIN          -1    // Опционо, сетовати на -1 ако се
не користи


// ПОДЕШАВАЊА ХАРДВЕРСКОГ UART-А
// ------------------------------------------------------------------------
-----------------
// Сљедећи macro-и декларишу који ће пинови бити кориштени за HW серијску
комуникацију
//
// ------------------------------------------------------------------------
-----------------
#ifdef Serial1    // Резервисање Serial1 константе
  #define BLUEFRUIT_HWSERIAL_NAME      Serial1
#endif


// ДИЈЕЉЕНА UART ПОДЕШАВАЊА
// ------------------------------------------------------------------------
-----------------
// Опциони Mode пин
// ------------------------------------------------------------------------
-----------------
#define BLUEFRUIT_UART_MODE_PIN       12    // Поставити на -1 ако се не користи


// ДИЈЕЉЕНА SPI ПОДЕШАВАЊА
// ------------------------------------------------------------------------
-----------------
```

```
// Сљедећи macro-и декларишу пинове који се користе за HW i SW SPI комуникацију
// -------------------------------------------------------------------------
-----------------
#define BLUEFRUIT_SPI_CS               8
#define BLUEFRUIT_SPI_IRQ              7
#define BLUEFRUIT_SPI_RST              4    // Поставити на -1 ако се не користи

// СОФТВЕРСКА SPI ПОДЕШАВАЊА
// -------------------------------------------------------------------------
-----------------
// Сљедећи macro-и декларишу пинове који се користе за SW SPI комуникацију
// -------------------------------------------------------------------------
-----------------
#define BLUEFRUIT_SPI_SCK              13
#define BLUEFRUIT_SPI_MISO             12
#define BLUEFRUIT_SPI_MOSI             11
```

И дошли смо коначно до дијела који нас највише интересује. Наиме, у библиотеци има много примјера, али овдје ћемо објаснити само два, прво *Bleuart_cmdmod.* Надам се да се нећете препасти можда неких непознатих дијелова кода, примјери су ту да се из њих учи нешто ново.

```
This is an example for our nRF51822 based Bluefruit LE modules

 Pick one up today in the adafruit shop!

 Adafruit invests time and resources providing this open source code,
 please support Adafruit and open-source hardware by purchasing
 products from Adafruit!

 MIT license, check LICENSE for more information
 All text above, and the splash screen below must be included in
 any redistribution
*********************************************************************/
#include <Arduino.h>
#include <SPI.h>
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"

#include "BluefruitConfig.h"

#if SOFTWARE_SERIAL_AVAILABLE
  #include <SoftwareSerial.h>
#endif
```

```
/*=========================================================================
=====
    APPLICATION SETTINGS

    FACTORYRESET_ENABLE       // Ресет на фабричке поставке
     -----------------------------------------------------------------
-----*/
    #define FACTORYRESET_ENABLE         1
    #define MINIMUM_FIRMWARE_VERSION    "0.6.6"
    #define MODE_LED_BEHAVIOUR          "MODE"
/*=========================================================================
=====*/

// Креирање Bluefruit објекта

SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN,
BLUEFRUIT_SWUART_RXD_PIN);

Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
                      BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);


// error debugging
void error(const __FlashStringHelper*err) {
  Serial.println(err);
  while (1);
}

/**************************************************************************
*****/
/*!
    @brief  Подешавање HW  BLE модула (ова функција се позива аутоматски
приликом startup-а)
*/
/**************************************************************************
*****/
void setup(void)
{
  while (!Serial);  // За Flora & Micro развојне плоче
  delay(500);

  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit Command Mode Example"));
  Serial.println(F("---------------------------------------"));

  /* Иницијализација модула */
  Serial.print(F("Initialising the Bluefruit LE module: "));

  if ( !ble.begin(VERBOSE_MODE) )
  {
    error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode &
check wiring?"));
  }
  Serial.println( F("OK!") );
```

```
  if ( FACTORYRESET_ENABLE )
  {
/* Ресет на фабричке поставке */

    Serial.println(F("Performing a factory reset: "));
    if ( ! ble.factoryReset() ){
      error(F("Couldn't factory reset"));
    }
  }

  /* Онемогући eho опцију с Bluefruit-a */
  ble.echo(false);

  Serial.println("Requesting Bluefruit info:");
  /* Print Bluefruit info */
  ble.info();

  Serial.println(F("Please use Adafruit Bluefruit LE app to connect in
UART mode"));
  Serial.println(F("Then Enter characters to send to Bluefruit"));
  Serial.println();

  ble.verbose(false);  // Искључи debug info!

  /* Чекај конекцију */
  while (! ble.isConnected()) {
      delay(500);
  }

  // LED Activity command је подржана од fw верзије 0.6.6
  if ( ble.isVersionAtLeast(MINIMUM_FIRMWARE_VERSION) )
  {
    // Промијените Mode LED Activity
    Serial.println(F("******************************"));
    Serial.println(F("Change LED activity to " MODE_LED_BEHAVIOUR));
    ble.sendCommandCheckOK("AT+HWModeLED=" MODE_LED_BEHAVIOUR);
    Serial.println(F("******************************"));
  }
}

/**************************************************************************
*****/
/*!
@brief  Константно провјеравамо има ли нових команди или одзива
*/
/**************************************************************************
*****/
void loop(void)
{
  // Провјерите user инпута
  char inputs[BUFSIZE+1];

  if ( getUserInput(inputs, BUFSIZE) )
  {
    // Пошаљите карактере ка Bluefruit-у
```

```
    Serial.print("[Send] ");
    Serial.println(inputs);

    ble.print("AT+BLEUARTTX=");
    ble.println(inputs);

    // Провјерите статус одзива
    if (! ble.waitForOK() ) {
      Serial.println(F("Failed to send?"));
    }
  }

  // Провјерите долазне карактере с  Bluefruit-а
  ble.println("AT+BLEUARTRX");
  ble.readline();
  if (strcmp(ble.buffer, "OK") == 0) {
    // Ако нема података
    return;
  }
  // Имамо неке податке, у buffer-у су
  Serial.print(F("[Recv] "));
  Serial.println(ble.buffer);
  ble.waitForOK();
}

/**************************************************************************
*****/
/*!
    @brief  Провјера уноса од корисника(via Serial Monitor-а)
*/
/**************************************************************************
*****/
bool getUserInput(char buffer[], uint8_t maxSize)
{
  // Timeout  100 милисекунди
  TimeoutTimer timeout(100);

  memset(buffer, 0, maxSize);
  while( (!Serial.available()) && !timeout.expired() ) { delay(1); }

  if ( timeout.expired() ) return false;

  delay(2);
  uint8_t count=0;
  do
  {
    count += Serial.readBytes(buffer+count, maxSize);
    delay(2);
  } while( (count < maxSize) && (Serial.available()) );

  return true;
}
```

Као прво, неки коментари у вези с примјером остављени су у оригиналном облику ради ауторских права писца библиотеке. Није циљ објаснити сваку линију кода, јер за разумијевање апликације морате самостално да ишчитате документацију ове библиотеке. Видјећете да ће то дати одговоре на многа питања. То је једини исправан пут да не залутате у беспуће copy/paste рјешења.

Једна непознаница, вјероватно је сљедеће прво што сте примијетили:

```
Serial.print(F("[Recv] "));
```

Ово F у основи значи сљедеће:

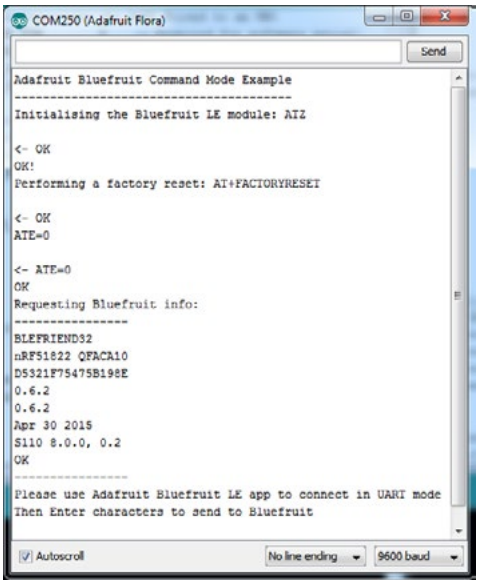
```
Serial.println(F(„Ја не користим RAM за ову инструкцију!!!"));

Serial.println(F(„Ја сам константа у FLASH меморији!!!"));

Serial.println(F(„Поједностављено, ја сам на HDD Arduin-у!!!"));
```

Можда и ово:

```
if (strcmp(ble.buffer, "OK") == 0)
```

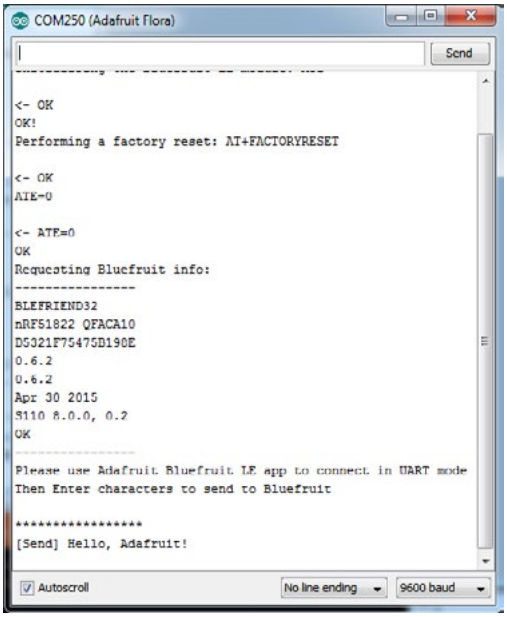
али само ако нисте имали основе програмирања, ријеч је о string compare инструкцији која упоређује карактер по карактер два string-а.

Већина осталих ствари је позната или их је врло лако разумјети. Сада је потребно видјети како је могуће тестирати апликацију. Након што сте компајлирали и учитали апп у Arduino, потребно је отворити Serial Monitor апп, подесити у доњем десном углу BaudRate на 115200. Уколико сте добро ожичили BLE модул, учитали example код са преправкама, онда би на Serial Monitor-у требало да добијете сљедеће:

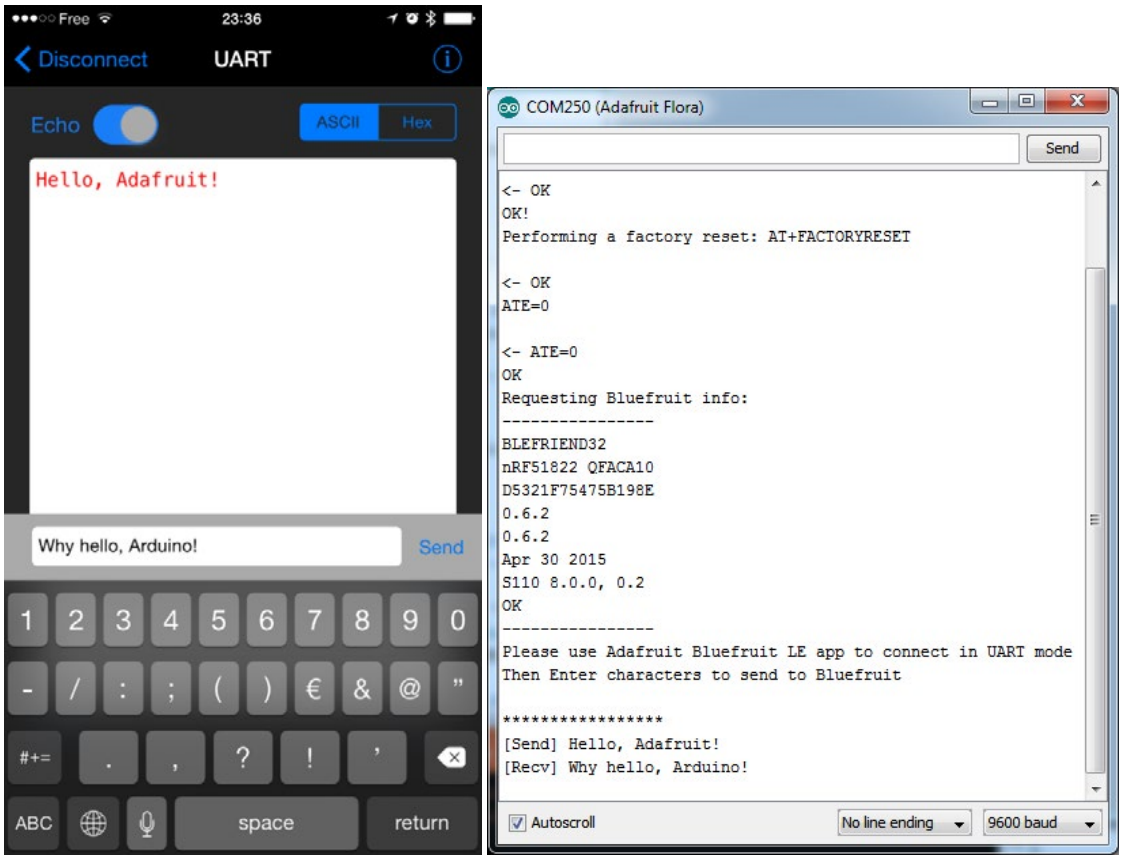


Слика 11.5. Serial Monitor одзив успјешно спојеног и програмираног example пројекта

Сада је потребно покренути инсталирану апликацију на телефону или неком другом уређају, скенирати и наћи Bluefruit LE модул те се конектовати у UART моду. Када сте то успјешно урадили, можете да шаљете текст ка телефону директно из Serial Monitor-а.



Слика 11.6. Слање string-а ка Bluefruit апликацији



Слика 11.6. Запримање string-а и одговор ка Arduin-у

Машине комуницирају. Дакле, два различита уређаја, један smart, други embedded, међусобно размјењују string-ове. Не знам како се то вама чини, али мислим да је довољно неколико if...else...endif инструкција да задивимо пријатеље и направимо неки једноставан ChatBot или сценаријем дефинисан привидно интелигентан разговор двију машина, само корак смо до Terminator мода 😊.

Наравно, не можемо се зауставити овдје, слиједи примјер апликације коју можемо да искористимо у комбинацији с неким од наших претходних пројеката.

У примјерима нађите *controller* и прегледајте добро примјер из приручника јер су неки сегменти измијењени да би одговарали нашој хардверској конфигурацији.

```
/*********************************************************************
 This is an example for our nRF51822 based Bluefruit LE modules

 Pick one up today in the adafruit shop!

 Adafruit invests time and resources providing this open source code,
 please support Adafruit and open-source hardware by purchasing
 products from Adafruit!

 MIT license, check LICENSE for more information
 All text above, and the splash screen below must be included in
 any redistribution
*********************************************************************/

#include <string.h>
#include <Arduino.h>
#include <SPI.h>
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"

#include "BluefruitConfig.h"

#if SOFTWARE_SERIAL_AVAILABLE
  #include <SoftwareSerial.h>
#endif

/*=========================================================================
    APPLICATION SETTINGS

    FACTORYRESET_ENABLE       Ресет на фабричке поставке
    -----------------------------------------------------------------------*/
    #define FACTORYRESET_ENABLE         1
    #define MINIMUM_FIRMWARE_VERSION    "0.6.6"
    #define MODE_LED_BEHAVIOUR          "MODE"
/*=========================================================================*/
```

```
// Креирање Bluefruit објекта

SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN,
BLUEFRUIT_SWUART_RXD_PIN);

Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
                      BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);


// error debugging
void error(const __FlashStringHelper*err) {
  Serial.println(err);
  while (1);
}

// Прототипи функција у packetparser.cpp
uint8_t readPacket(Adafruit_BLE *ble, uint16_t timeout);
float parsefloat(uint8_t *buffer);
void printHex(const uint8_t * data, const uint32_t numBytes);

// Пакет buffer
extern uint8_t packetbuffer[];


/**************************************************************************
*****/
/*!
    @brief  Подешавање HW  BLE модула (ова функција се позива аутоматски
приликом startup-а)

*/
/**************************************************************************
*****/
void setup(void)
{
  while (!Serial);  // За Flora & Micro развојне плоче
  delay(500);

  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit App Controller Example"));
  Serial.println(F("-----------------------------------------"));

  /* Иницијализација модула */
  Serial.print(F("Initialising the Bluefruit LE module: "));

  if ( !ble.begin(VERBOSE_MODE) )
  {
    error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode &
check wiring?"));
  }
  Serial.println( F("OK!") );

  if ( FACTORYRESET_ENABLE )
  {
/* Ресет на фабричке поставке*/
```

```
      Serial.println(F("Performing a factory reset: "));
    if ( ! ble.factoryReset() ){
      error(F("Couldn't factory reset"));
    }
  }


  /* Онемогући echo опцију са Bluefruit-a */
  ble.echo(false);

  Serial.println("Requesting Bluefruit info:");
  /* Print Bluefruit information */
  ble.info();

  Serial.println(F("Please use Adafruit Bluefruit LE app to connect in
Controller mode"));
   Serial.println(F("Then activate/use the sensors, color picker, game
controller, etc!"));
  Serial.println();

   ble.verbose(false);  // debug info is a little annoying after this
point!

  /* Чекај конекцију */
  while (! ble.isConnected()) {
      delay(500);
  }

  Serial.println(F("******************************"));

  // LED Activity command је подржана од fw верзије 0.6.6
  if ( ble.isVersionAtLeast(MINIMUM_FIRMWARE_VERSION) )
  {
    // Промијени Mode LED Activity
    Serial.println(F("Change LED activity to " MODE_LED_BEHAVIOUR));
    ble.sendCommandCheckOK("AT+HWModeLED=" MODE_LED_BEHAVIOUR);
  }

  // Постави Bluefruit u DATA mode
  Serial.println( F("Switching to DATA mode!") );
  ble.setMode(BLUEFRUIT_MODE_DATA);

  Serial.println(F("******************************"));

}

/**************************************************************************
*****/
/*!
    @brief  Константно провјеравамо има ли нових команди или одзива
*/
/**************************************************************************
```

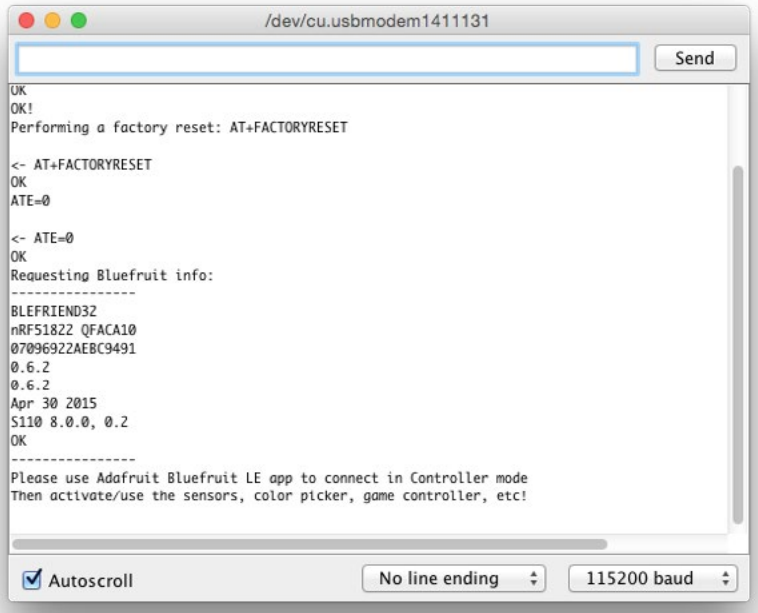
```
*****/
void loop(void)
{
  /* Чекање нових података */
  uint8_t len = readPacket(&ble, BLE_READPACKET_TIMEOUT);
  if (len == 0) return;

  /* Пакет запримљен! */
  // printHex(packetbuffer, len);

  // Боја
  if (packetbuffer[1] == 'C') {
    uint8_t red = packetbuffer[2];
    uint8_t green = packetbuffer[3];
    uint8_t blue = packetbuffer[4];
    Serial.print ("RGB #");
    if (red < 0x10) Serial.print("0");
    Serial.print(red, HEX);
    if (green < 0x10) Serial.print("0");
    Serial.print(green, HEX);
    if (blue < 0x10) Serial.print("0");
    Serial.println(blue, HEX);
  }

  // Тастер
  if (packetbuffer[1] == 'B') {
    uint8_t buttnum = packetbuffer[2] - '0';
    boolean pressed = packetbuffer[3] - '0';
    Serial.print ("Button "); Serial.print(buttnum);
    if (pressed) {
      Serial.println(" pressed");
    } else {
      Serial.println(" released");
    }
  }

  // GPS локација
  if (packetbuffer[1] == 'L') {
    float lat, lon, alt;
    lat = parsefloat(packetbuffer+2);
    lon = parsefloat(packetbuffer+6);
    alt = parsefloat(packetbuffer+10);
    Serial.print("GPS Location\t");
    Serial.print("Lat: "); Serial.print(lat, 4); // 4 децимале!
    Serial.print('\t');
    Serial.print("Lon: "); Serial.print(lon, 4); // 4 децимале!
    Serial.print('\t');
    Serial.print(alt, 4); Serial.println(" meters");
  }

  // Акцелерометар
  if (packetbuffer[1] == 'A') {
    float x, y, z;
    x = parsefloat(packetbuffer+2);
    y = parsefloat(packetbuffer+6);
```

```
    z = parsefloat(packetbuffer+10);
    Serial.print("Accel\t");
    Serial.print(x); Serial.print('\t');
    Serial.print(y); Serial.print('\t');
    Serial.print(z); Serial.println();
  }

  // Магнетометар
  if (packetbuffer[1] == 'M') {
    float x, y, z;
    x = parsefloat(packetbuffer+2);
    y = parsefloat(packetbuffer+6);
    z = parsefloat(packetbuffer+10);
    Serial.print("Mag\t");
    Serial.print(x); Serial.print('\t');
    Serial.print(y); Serial.print('\t');
    Serial.print(z); Serial.println();
  }

  // Жироскоп
  if (packetbuffer[1] == 'G') {
    float x, y, z;
    x = parsefloat(packetbuffer+2);
    y = parsefloat(packetbuffer+6);
    z = parsefloat(packetbuffer+10);
    Serial.print(„Gyro\t");
    Serial.print(x); Serial.print('\t');
    Serial.print(y); Serial.print('\t');
    Serial.print(z); Serial.println();
  }

  // Кватерниони
  if (packetbuffer[1] == 'Q') {
    float x, y, z, w;
    x = parsefloat(packetbuffer+2);
    y = parsefloat(packetbuffer+6);
    z = parsefloat(packetbuffer+10);
    w = parsefloat(packetbuffer+14);
    Serial.print("Quat\t");
    Serial.print(x); Serial.print('\t');
    Serial.print(y); Serial.print('\t');
    Serial.print(z); Serial.print(,\t');
    Serial.print(w); Serial.println();
  }
}
```

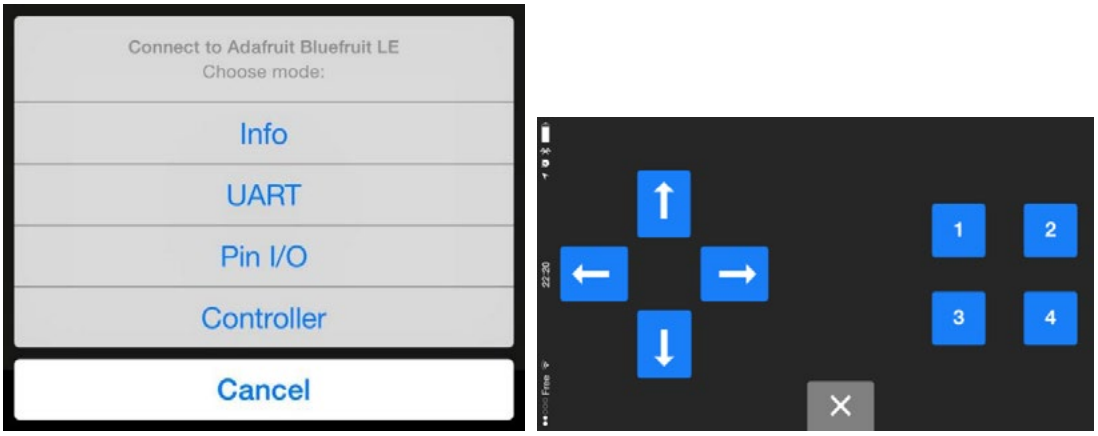
Овај пројекат садржи и додатни фајл *packetParser.cpp*, који садржи API за парсирање података који долазе с Bluefruit апликације.

Наведени примјер омогућава нам да претворимо iOS или Android уређај у даљински контролер или екстерни извор података о GPS локацији или подацима са сензора за детекцију убрзања интрегрисаног у телефон или таблет.

Уколико сте BLE оживили без грешака те учитали наведени примјер, у Serial Monitor-у требало би да добијете сљедећи одзив:



Слика 11.7. Успјешан одзив controler.ino у Serial Monitor-у



Слика 11.8. Успјешан одзив controler.ino у Serial Monitor-у

Притиском типке на Bluefruit апликацији у моду Control Pad регистроваће се на Serial Monitor апликацији с ИД бројем типке који је коришћен:

```
Button 8 pressed
Button 8 released
Button 3 pressed
Button 3 released
```

Мислим да је сада јасно да се уз помоћ провјере података који долазе с BLE апликације можемо интегрисати у сваки досадашњи пројекат, сада је само потребно препустити се својој машти.

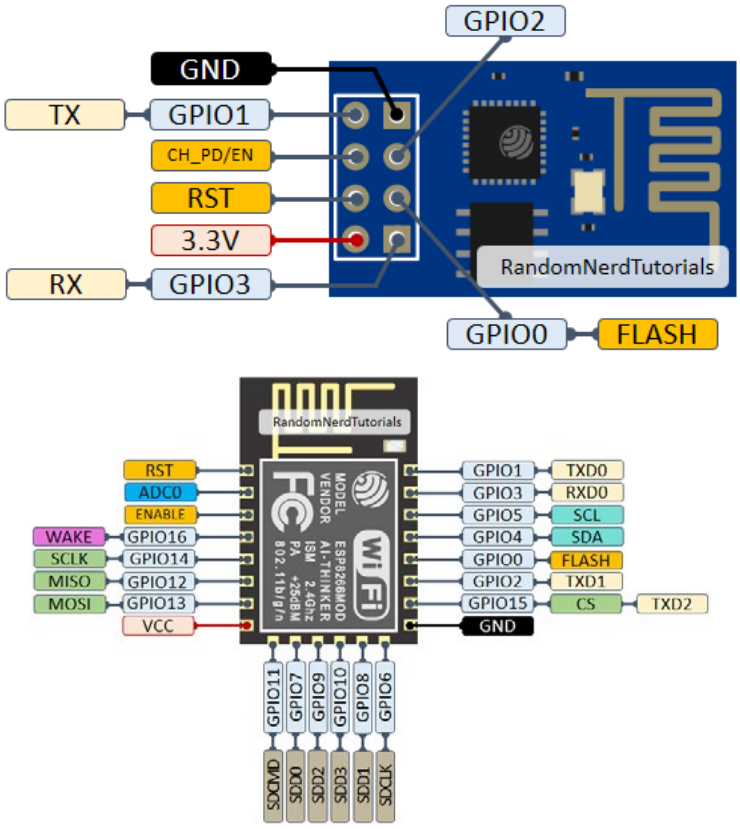
ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца:	**Ученик/ца:**
- описује улогу електронских компоненти које су коришћене у пројекту	- описује карактеристике BLE модула - наводи примјере примјене BLE модула - набраја модове унутар Bluefruit Connect апликације - користи Bluefruit Connect апликацију за повезивање
- користи електричну шему споја за повезивање Arduin-a и електронских компоненти	- повезује пинове BLE модула с Arduino UNO према шеми споја - извршава тестирање BLE модула - повезује преостале компоненте у пројекту према шеми споја
- креира програмски код за управљање користећи BLE (Bluetooth Low Energy) према упутама	- преузима, инсталира и примјењује одговарајућу Bluefruit Connect апликацију у зависности од оперативног система (iOS ili Android) - конфигурише BLE модул - враћа BLE модул на творничке поставке уколико је погрешно конфигурисан - користи готове примјере програмског кода како бисте научили нешто ново - реконструише програмски код према сопственим идејама
- користи одговарајуће библиотеке унутар програма за реализацију пројектног задатка	- инсталира SoftwareSerial библиотеку користећи Library Manager - управља са BluefruitConfig.h вршећи потребне измјене - позива библиотеке унутар програмског кода
- верификује и извршава програм	- користи Serial Monitor апликацију ради тестирања апликације - анализира програмски код - отклања грешке уколико су настале у коду према смјерницама наставника
- тестира пројектни задатак	- учитава код на Arduino платформу - примјењује Serial Monitor за тестирање апликације - повезује мобител или неки други уређај на Bluefruit LE модул - тестира слање текста путем Bluetooth-a - уочава улогу додатног фајла packetParser.cpp - анализира пројектни задатак - Развија сопствене идеје за примјену BLE

Простор за вјежбу:

```
 void setup() {

}
void loop() {

}
```

Пројекат број 12.
ESP8266 IoT

IoT или Internet of Things почео се рапидно развијати појавом ESP8266 WiFI SoC-а с потпуно интегрисаним TCP/IP протоколом. То је low-cost WiFi модул способан да ради хостинг веб-апликације или обезбиједи све мрежне функције микроконтролеру који иначе нема могућност приступа WiFi мрежи. Модул долази с програмираним firmwareom-ом за контролу помоћу AT командног сета, те се једноставно може прикључити на Arduino. Огроман је спектар апликација које се могу реализовати с овим модулом, али ми ћемо се базирати на нешто што је у посљедње вријеме постало веома популарно, јер сви желе да живе у Smart Home окружењу. Дакле, кренућемо у овај пројекат с једноставним задатком даљинске контроле уређаја помоћу WiFi мреже. Треба напоменути да је могуће пројекат реализовати и са ESP8266 SMD модулом чији је pinout приказан на сљедећој слици.



Слика 12.1. ESP8266 pinout и ESP8266 smd pinout

По општеприхваћеној дефиницији Smart Home технологија користи уређаје типа сензора или уређаја који су спојени на IoT те се могу даљински надзирати, контролисати или нуде опцију приступа за обезбјеђивање сервиса спрам захтјева корисника.

Кренућемо од поставке проблема и то тако да ћемо поставити себи задатак да креирамо једноставни веб базирани control panel за управљање уређајима који су спојени на

локалну WiFi мрежу. Ово одмах постаје мултидисциплинарни пројекат, јер сада осим ембеддед апликације имамо задатак креирати веб-апликације, те морамо обезбиједити начин да те двије апликације комуницирају.

За израду веб-апликације потребно је користити сљедеће технологије:

- HTML
- CSS
- Jquery

Веб-страница ће слати захтјеве ка ESP8266 који ће радити као веб-сервер спојен преко UART комуникационог канала с Arduin-ом. У зависности од података који долазе на Arduino, дешаваће се кореспондентне акције на његовим излазима, нпр. укључење или искључење неког од потрошача.

Потребни елементи за реализацију пројекта су:

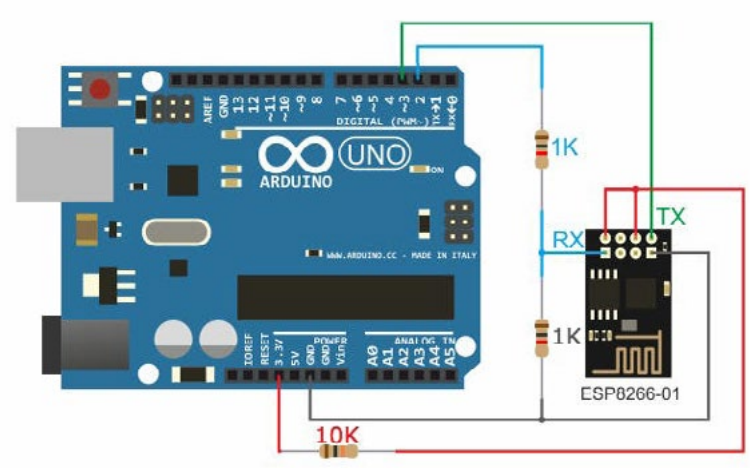
1. ARDUINO UNO
2. Матадор плоча
3. Каблићи
4. 1 x 10K отпорник
5. 2 x 1K отпорник
6. ESP8266 модул

Детаљни pinout WiFi модула морамо навести с посебним нагласком да модул нема интегрисан напонски регулатор те да се мора спојити на напон од 3.3 V.

- **VCC**: +3.3 V: спајање на 5 V *извор оштетиће модул*
- **GND**: заједнички пин
- **CH_PD**: Chip enable пин: потребно је пин спојити 3.3 V да би се модул *bootao*.
- **RST**: Chip Reset пин: спајање на GND ресетује модул
- **GPIO0, GPIO2**: Пинови опште намјене
- **Tx**: пин за слање података
- **Rx**: пин за примање података

Као што смо већ навели, ESP8266 модул комуницира с Arduino модулом користећи серијску комуникацију, што значи да би требало искористити хардверске пинове УАРТ модула пин 0 и пин 1, али на тај начин нисмо у могућности користити Serial Monitor апликацију за debugging софтвера и troubleshooting. Овај проблем ријешићемо

коришћењем библиотеке SoftwareSerial те WiFi модул спојити према шеми споја као на сљедећој слици.

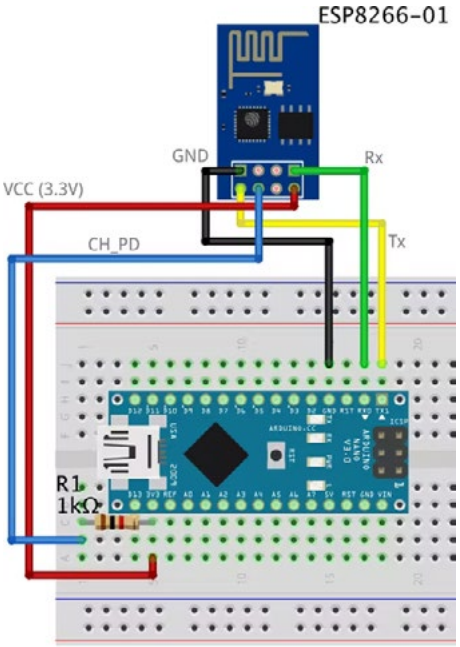


Слика 12.2. Шема споја ESP8266 и Arduino Uno

SoftwareSerial библиотека ради поуздано док год су брзине преноса података подешене на 19200 и мање. ESP8266 је фабрички подешен за рад на брзини преноса података од 115200, стога је потребно реконфигурисати модул користећи АТ команде.

Прије реконфигурације потребно је у Arduino учитати скицу BaseMinumum, с путање *Examples->Basic->BaseMiinumum.*

НАПОМЕНА:
За потребе реконфигурације ЕПС8266 модула потребно је спојити га на хардверски UART модул Arduin-a, дакле RX пин ESP модула спојити на TX пин Arduin-a, TX пин ESP-a спојити на PX пин Arduin-a. Примјер који је дат као водиља урађен је с другом развојном Arduino плочом, али принцип спајања остаје исти, било да се ради о ESP8266-01 модулу или ESP8266 SMD модулу.



Примјер спајања ESP модула за реконфигурацију помоћу АТ команди

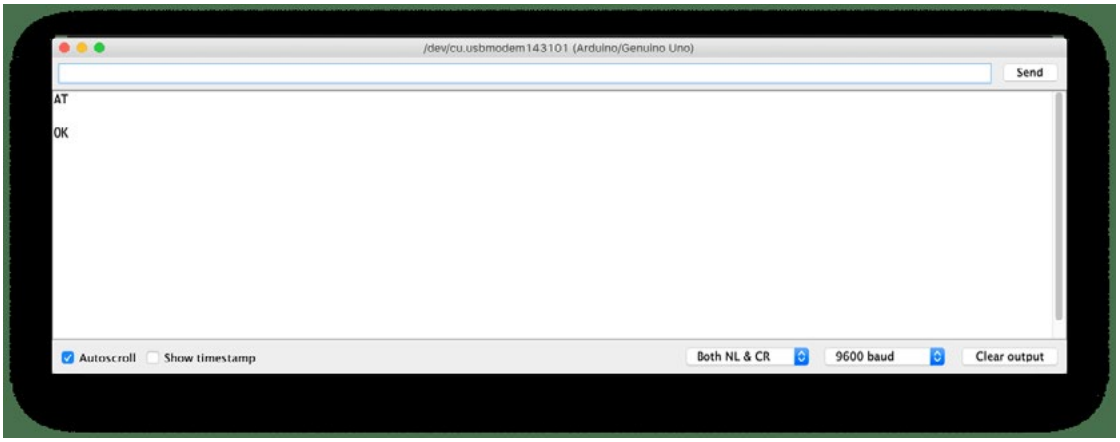
AT командни сет је стандардни начин комуникације с модемима, BT и GPS модулима. AT је скраћеница од ријечи *ATention* и представља префикс у свим наредбама, при чему модул зна да послије доласка ове кључне ријечи иде наредба. На сљедећој слици представљене су неке од основних AT наредби:

No.	Command	Example	Parameters	Description	returns
1	AT	AT	Nothing	Check if the module is working properly.	OK
2	AT+RST	AT+RST	Nothing	Resets the Module.	OK
3	AT+CWMODE=<MODE>	AT+CWMODE=1	1 = Station 2 = AP 3 = Both	sets the Wifi mode	OK
4	AT+CIOBAUD=<baud> OR AT+UART_DEF=<baud>,8,1,0,0	AT+CIOBAUD=9600 OR AT+UART_DEF=9600,8,1,0,0	baud rate	changes the module communication speed.	OK
5	AT+CWJAP=<SSID>,<PASS>	AT+CWJAP="meme", "69696969696"	SSID = Wifi Network name PASS = Wifi Password	Connects to a Wifi Network	WIFI CONNECTED WIFI GOT IP
6	AT+CIFSR	AT+CIFSR	Nothing	Tells you the module IP address and the MAC Address	IP Address Mac Address
7	AT+CIPSERVER=<MODE>,<PORT>	AT+CIPSERVER=1,80	MODE = 0 to close the server, 1 to open the server. PORT = port number	Sets the module as a server.	OK

Слика 12.3. AT команде

Потребно је отворити Serial Monitor апликацију и у њој у доњем десном углу подесити брзину комуникације на **115200** и селектовати опцију Both **NL & CP.**

Наредни корак је провјера комуникације с ESP модулом. Примјер успјешно остварене комуникације приказан је на сљедећој слици.



Слика 12.4. AT одзив с ESP8266 модула

У зависности од тога који је firmware дошао учитан на вашем ESP8266-01 или ESP8266 SMD модулу, промјену брзине протока података можете да пробате с једном од ове двије наредбе:

```
command AT+UART_DEF=<baud>,8,1,0,0

AT+CIOBAUD=<baud>.
```

Уколико је одзив на једну од ове двије наредбе **“OK“, ваш ESP модул даље комуницира при брзини 9600 bit/s. Не заборавите у свом Serial Monitor-у одабрати у доњем десном углу 9600**!

Потом је потребно да ESP8266 модул спојите на своју локалну WiFi мрежу, при чему је потребно уписати у AT команду назив локалне мреже те шифру за њу.

```
AT+CWJAP = "имемојеWiFiмреже","шифрамојеWiFiмреже"
```

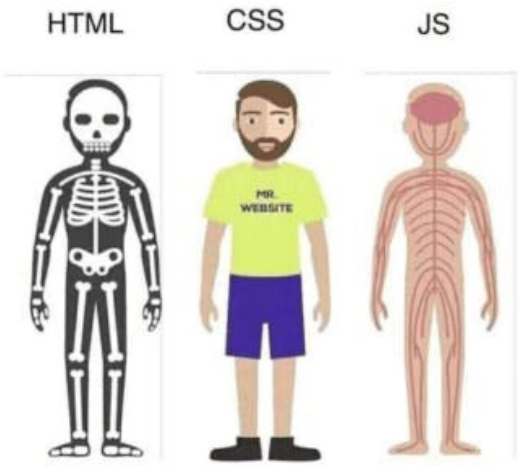
Уколико се оствари успјешна конекција на локалну WiFi мрежу, одзив ESP модула требало би да буде:

```
WIFI DISCONECTED
WIFI CONNECTED
WIFI GOT IP
```

Да бисте провјерили која је ИП адреса додијељена вашем ESP модулу у локалној WiFi мрежи, потребно је послати AT наредбу ***AT+CIFSR.***

За неке од вас овај дио биће први сусрет с веб-технологијама, стога ће овај сегмент бити мало детаљније описан.

Као што смо већ навели, за израду једноставне веб-апликације потребни су нам HTML, CSS и JavaScript. Најбољи опис ова три сегмента дат је у сљедећој слици.

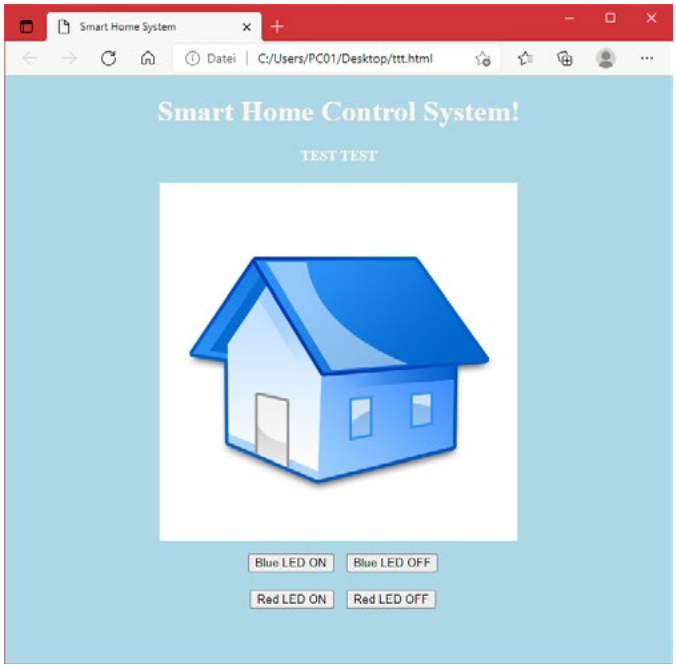


Слика 12.5. Структура веб-странице

HTML је скраћеница од HuperText markup language и користи се за израду главне структуре веб-странице те помоћу њега радимо додавање button-а, фотографија, заглавља, табела и осталих елемената. HTML као језик посједује низ елемената помоћу којих прегледнику шаље информацију како да прикаже кориснику веб-садржај. Ови елементи језика представљају тзв. тагове.

CSS је скраћеница од Cascading Style Sheet, те након израде главне структуре странице служи за фино подешавање њеног изгледа. Дакле, функција му је да дефинише изглед појединих HTML елемената.

Javascript је програмски језик помоћу којег веб-странице нуде интеракцију с њеним корисником. У овом конкретном примјеру користиће се за слање HTTP request-a од клијента (webbrowser-a) ка веб-серверу (ESP8266) за укључење или искључење потрошача.



Слика 12.6. Изглед тестне странице

За писање HTML кода можете да користите Notepad++ editor.

```
<!DOCTYPE html>
<html>
<head>
    <title>Smart Home System</title>
</head>
<body>
    <h1> Smart Home Control System!</h1>
    <h4>TEST TEST!</h4>
     <img src=" http://home.citycable.ch/fayachi/design/image/icone/
home.jpg" width="400px" height="400px">
    <br>
    <button id="111" class="button">LAMP ON</button>
    <br>
    <button id="110" class="button">LAMP OFF</button>
</body>
</html>
```

Опис синтаксе

- **<!DOCTYPE html >:** декларација да је ријеч о HTML5 документу
- **<html>:** root елемент HTML странице
- **<head>:** метаинформације о страници
- **<title>:** назив странице, видљив у browser табу
- **<body>:** видљиви дио садржаја веб-странице налази се унутар ових тагова
- **<h1>:** елемент дефинише велике елементе, рецимо, формат текста
- **<img>:** додаје слику на веб-страницу
- **
:** нови ред
- **<button>:** додаје button елеменат на веб-форму

Сваки button унутар веб-апликације има неколико атрибута који су веома важни за нашу конкретну страницу, а то су ***id*** и ***class.***

```
 <!DOCTYPE html>
<html>
<head>
    <title>Smart Home System</title>
</head>
<body style="background-color: seagreen; color: seashell; text-align:
center;">
    <h1> Smart Home Control System!</h1>
    <h4>TEST TEST!</h4>
      <img src=" http://home.citycable.ch/fayachi/design/image/icone/
home.jpg" width="400px" height="400px">
    <br>
     <button style="margin: 10px;" id="111" class="button">LAMP ON</
button>
    <br>
     <button style="margin:10px;" id="110" class="button"> LAMP OFF</
button>
</body>
</html>
```

У овој итеракцији на чисти HTML код додали смо мало стила да би он изгледао мало љепше. Унутар ***body*** тага додали смо style="background-color: seagreen; color: seashell; text-align: center;" да би страница била центрирана у прегледнику и промјенили смо јој позадинску боју, те боју фонта.

Такође, у button тагу је додано стуле="маргин: 10пх;" да бисмо направили маргину од 10 пиксела око button-а.

Након што смо завршили са структуром веб-странице, потребно јој је додати функционалност. Идеја је да browser, када корисник кликне на button „Lamp ON",

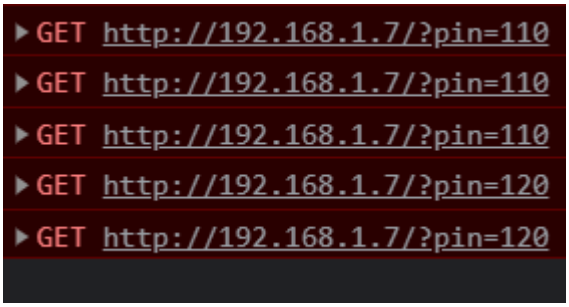
пошаље request у којем се налазе јединствени подаци за клик тог button-а. Arduino код треба организовати да препозна те податке те одради кореспондентну акцију на својим дигиталним излазима.

```
<!DOCTYPE html>
<html>
<head>
    <title>Smart Home System</title>
      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/
jquery.min.js"></script>
</head>
<body style="background-color: seagreen; color: seashell; text-align:
center;">
    <h1> Smart Home Control System</h1>
    <h4>TEST TEST!</h4>
    <img src="dogMeme.jpg" width="400px" height="400px">
    <br>
      <button style="margin: 10px;" id="111" class="button">LAMP ON</
button>
    <br>
<button style="margin:10px;" id="110" class="button"> LAMP OFF</button>
   // Start Javascript
        <script>
          // Ако је активиран button iz ".button" класе
        $(".button").click(function () {
         // покупи id и спреми га у "p" варијаблу
            var p = $(this).attr('id');
         // Button id који се шаље ка веб-серверу
            pin: p
        //Пошаљи request

            $.get("http://192.168.1.4:80/", {  pin: p
            });
        });
    </script>
</body>
</html>
```

Сва магија дешава се у сљедећем дијелу кода:

```
<script>
        $(".button").click(function () {
            var p = $(this).attr('id');
            pin: p
            $.get("http://192.168.1.4:80/", {
            pin: p
            });
        });
    </script>
```

Када корисник кликне на било који button на веб-страници, он у позадини покрене **.click** функцију. Та функција „покупи" атрибуте о којима смо раније писали, „id" и „p", који би требало да су јединствени у вашој апликацији и спреми их унутар „p" варијабле, те се потом пошаље **GET** request с припадајућом вриједношћу **id-a** кликнутог button-a. Уколико активирате опцију developer tools у свом browser-у, видјећете сљедеће request-е.



Слика 12.7. Изглед GET request-а

С овим смо завршили с веб дијелом пројекта. Ваш задатак би могао бити да умјесто једног button-a на веб-форму додате button-a онолико колико имате слободних дигиталних пинова на свом Arduin-у. Фајл с веб-апликацијом можете да отворите у неком од веб-прегледника. Обавезно укључити опцију developer tools. Важно је да оба уређаја, у овом случају PC и ESP8266 буду на истој WiFi мрежи.

Када сте креирали своју веб-страницу, било би је потребно „хостати" на неки remote веб-сервер или локално. Најједноставнија опција је да се искористи интегрисани IIS сервис унутар Windows оперативног система, који је потребно активирати пратећи сљедеће кораке:

- Отворите **Control panel и идите на Programs and Features > Turn Windows Features on or off**.
- Активирајте **Internet Information Services**.
- Проширите **Internet Information Services** ставку и порвјерите је ли веб-сервер компонента означена.
- Click **OK**.
- Копирајте своју веб-страницу на локацију C:/inetpub/wwwroot/imevasestranice.html
- Провјерите IP адресу свог PC -а
- Веб-страници можете да приступите с мобилног уређаја који је на истој WiFi мрежи на сљедећи начин:
- http://ipadresavasegPC:80/imevasestranice.html

```
// Позивањем SoftwareSerial библиотеке омогучиће ти се коришћење пинова
2 i 3 kao Rx, Tx
#include <SoftwareSerial.h>
// Подеси Rx ==> Pin 2; TX ==> Pin3.
SoftwareSerial esp8266(2,3);

// Креирај константу “serialCommunicationSpeed”

#define serialCommunicationSpeed 9600

// Креирај константу “debug” и придружи вриједност true

#define DEBUG true

 // Придружи варијабли “redLED” вриједност 12

int redLED =12;

 // Придружи варијабли “blueLED” вриједност 11

int blueLED =11;

void setup()
{
// Постави пин 12 као излазни пин

  pinMode(redLED,OUTPUT);

// Постави пин 11 као излазни пин

  pinMode(blueLED,OUTPUT);

// Деактивирај LED на почетку апликације

  digitalWrite(redLED,LOW);

// Активирај LED на почетку апликације

  digitalWrite(blueLED,HIGH);

// Иницијализуј хардверску серијску комуникацију на пиновима (0, 1) i
brzini 9600.

  Serial.begin(serialCommunicationSpeed);

// Иницијализуј софтверску серијску комуникацију на пиновима (2, 3) и
брзини 9600

  esp8266.begin(serialCommunicationSpeed);
// понови функцију “InitWifiModule()” за иницијализацију комуникације
између ESP8266 i WiFi рутера или мобилног хотспота.

 InitWifiModule();

// Послије успјешне иницијализације искључи LED
```

```
digitalWrite(blueLED,LOW);
}

void loop()
{
// Ако су запримљени подаци, парсирај их ако не улазе у петљу

  if(esp8266.available())
  {
// Тражи "+IPD," string у запримљеним подацима ако тражени string постоји ".find()" враћа true

if(esp8266.find("+IPD,"))
{

// Сачекај једну секунду да се buffer напуни подацима

      delay(1000);

// Одузимање 48 од запримљених података се ради јер функција read() враћа ASCII децималне вриједности

    int connectionId = esp8266.read()-48;

// Тражи "pin=" string унутар запримљених података

          esp8266.find("pin=");

// Прочитај први бајт из улазног buffer-а


          int pinNumber = (esp8266.read()-48)*10;

// Прочитај други бајт из улазног buffer-а

         pinNumber = pinNumber + (esp8266.read()-48);

// Прочитај трећи бајт из улазног buffer-а, сачувај га у "statusLed" варијабли

    int statusLed =(esp8266.read()-48);

// ON/OFF LED "pinNumber" у зависности од вриједности "statusLed" варијабле


      digitalWrite(pinNumber, statusLed);

// Принт "connectionId" вриједност у Serial Monitor-у за debugging

      Serial.println(connectionId);

 // Принт "pinNumber" вриједност у Serial Monitor-у за debugging
```

```
      Serial.print(pinNumber);
// Принт неколико празних мјеста ради лакшег праћења информација
      Serial.print("      ");
// Принт "statusLed" вриједност у Serial Monitoru za debugging
      Serial.println(statusLed);
// Затвори TCP/IP конекцију

String closeCommand ="AT+CIPCLOSE=";
// Додај "connectionId" вриједност у string
closeCommand+=connectionId;
// Додај "\r\n" string-у, симулирамо „нови ред као на тастатури"
closeCommand+="\r\n";
// Пошаљи команду ка ESP8266 модулу на извршавање
sendData(closeCommand,1000,DEBUG); /
    }
  }
}
/*********************************************************************
*********************************
* Name: sendData
* Description: this Function regulates how the AT Commands will ge sent
to the ESP8266.
*
* Params: command - the AT Command to send
*                 - timeout - the time to wait for a response
*                  - debug - print to Serial window?(true = yes, false
= no)
*
* Returns: The response from the esp8266 (if there is a reponse)
*/
String sendData(String command, const int timeout, boolean debug)
{
// Иницијализуј String варијаблу "response"
String response = "";
// Пошаљи AT команду ка ESP8266 (ARDUINO -> ESP8266).
esp8266.print(command);
// Покупи вријеме од када је апликација активна и сачувај га у варијабли
"time".
```

```
long int time = millis();

// Код унутар витичасте заграде извршава се сваке секунде

while( (time+timeout) > millis())

{
// Провјери постоји ли неки одзив са ESP8266 сачуван у улазном buffer-у
Arduin-а

     while(esp8266.available())
{
// Ако је ово тачно, узми сљедећи карактер из улазног buffer-а и сачувај
га у "response" String ватијабли

        char c = esp8266.read();

// Пуни елементе string-а са сваким новопристиглим карактером

    response+=c;
      }
}

// Ако је "debug" TRUE, принтај одзив у Serial Monitoru

     if(debug)
{
      Serial.print(response);
    }
return

 //vrati  String "response".

response;
}
/**************************************************************************
****
* Name: InitWifiModule
* Description: this Function gives the commands that we need to send to
the sendData() function to send it.
*
* Params: Nothing.
*
* Returns: Nothing (void).
*/

void InitWifiModule()
{

// Ресет ESP8266 модула

  sendData("AT+RST\r\n", 2000, DEBUG);

//delay(1000);
```

```
// Споји се на WiFi мрежу

  sendData("AT+CWJAP=\"PUT YOUR SSID\",\"PUT YOUR PASSWORD\"\r\n", 2000,
DEBUG);

  delay (3000);

// Подеси ESP8266 WiFi mode на station mode

  sendData("AT+CWMODE=1\r\n", 1500, DEBUG);

delay (1000);

// Прикажи IP адресу и MAC адресу

  sendData("AT+CIFSR\r\n", 1500, DEBUG);

delay (1000);

// Омогући више конекција

  sendData("AT+CIPMUX=1\r\n", 1500, DEBUG);

  delay (1000);

// Почетак комуникације на порту 80 с веб-сервером кроз http requests

  sendData("AT+CIPSERVER=1,80\r\n", 1500,DEBUG);
}
```

Сам код у овој фази требало би да буде у великој мјери јасан, имплементиране су двије функције InitWifiModule() і sendData(), при чему sendData() регулише начин на који се AT наредбе шаљу ка ESP8266 модулу. InitWifiModule() дефинише које ће AT наредбе sendData() функција слати ка ESP8266.
Унутар loop() функције чека се долазни HTTP request header и у њему се тражи "+IPD" што је знак да је захтјев успјешно дошао, потом се чита вриједност варијабле pin ->p, "111" се састоји из два броја, први дио 11 је број пина, а задња цифра је вриједност коју желимо да пин има. Рецимо, ако желимо да је пин укључен, то значи да ће задња цифра у троцифреном броју бити 1 ->HIGH. Односно, уколико корисник кликне на button „LAMP OFF", вриједност варијабле pin ->p биће "110".

Једна од важнијих ствари на коју морате да обратите пажњу јесте да упишете акредитиве своје WiFi мреже у линији

```
sendData("AT+CWJAP=\"PUT YOUR SSID\",\"PUT YOUR PASSWORD\"\r\n", 2000, DEBUG);
```

Вјероватно ћете на неки од претходних пројеката додати WiFi опцију. Срећно кодирање!!!

ИСХОД УЧЕЊА	РАЗРАДА ИСХОДА – ИНДИКАТОРИ
Ученик/ца:	**Ученик/ца:**
- описује улогу електронских компоненти које су коришћене у пројекту	- наводи улогу и основне карактеристике ESP8266 IoT модула - наводи pinout WiFi модула дефинишући улогу сваког појединачно - објашњава основе AT командног сета наводећи неке од команди
- користи електричну шему споја за повезивање Arduin-a и електронских компоненти	- повезује ESP8266 и Arduino Uno према шеми споја - спаја ESP8266 модул на локалну WiFi мрежу - повезује преостале компоненте у пројекту према шеми споја
- креира једноставан веб базирани control panel за управљање уређајима који су спојени на локалну WiFi мрежу	- креира веб-апликацију за реализацију пројектне идеје - креира embedded апликације за реализацију пројектне идеје - увезује embedded и веб-апликацију - проширује стечено знање из програмских језика HTML, CSS и Javascript - примјењује HTML тагове за креирање веб-апликације - креира button-е унутар веб-апликације користећи атрибуте id и class - форматира веб-апликацију користећи адекватне стилове - објашњава улогу функција InitWifiModule () и sendData() у коду
- користи одговарајуће библиотеке унутар програма за реализацију пројектног задатка	- користи библиотеку SoftwareSerial за debugging софтвера и troubleshooting
- верификује и извршава програм	- реконфигурише EPS8266 модул користећи AT команде - анализира програмски код - отклања грешке уколико су настале у коду према смјерницама наставника
- тестира пројектни задатак	- учитава код на Arduino платформу ради тестирања пројектног задатка - анализира пројектни задатак - користи ESP8266 IoT за развијање Smart Home пројектних идеја

Простор за вјежбу:

```
 void setup() {
}
void loop() {
}
```

Закључак

Прије свега, надамо се да вас овај приручник није уплашио. Јасно је да не постоји књига која ће вас научити све, али су оне које ће да вам отворе врата и прошире видике. Можда је и ова једна од њих. Као и у свим стварима које су повезане са животом, једноставно морамо да сакупимо храброст и да закорачимо даље. А када говоримо о програмирању, онда морамо да одемо и даље од апликације Hello World.

Овај приручник је ту да вам покуша приближити могућности Arduino развојне платформе уз комбиновање додатних сензора и актуатора који нису саставни дио Arduino Starter kit-a. Неки пројекти су једноставни, неки су мултидисциплинарни. Најважније је да не одустанете и када не иде како треба.

Приручник је ту да вас уведе у мало компликованије вјежбе, али његов прави циљ је да покушате самостално да комбинујете различите сензоре, дисплеје и актуаторе те да осмислите неки нови кул gadget који ће залудјети свијет, а вас учинити планетарно познатим. Како год ово тренутно звучало немогуће, никада не смијете да заборавите да је за успјех, осим труда и залагања, потребна још само идеја.

Наравно, треба да кажете себи да је само небо граница и да не заборавите **правило број 1** при реализацији било којег пројекта *keep it simple*. Не покушавајте да задивите техникама писања кода, већ његовом функционалношћу.

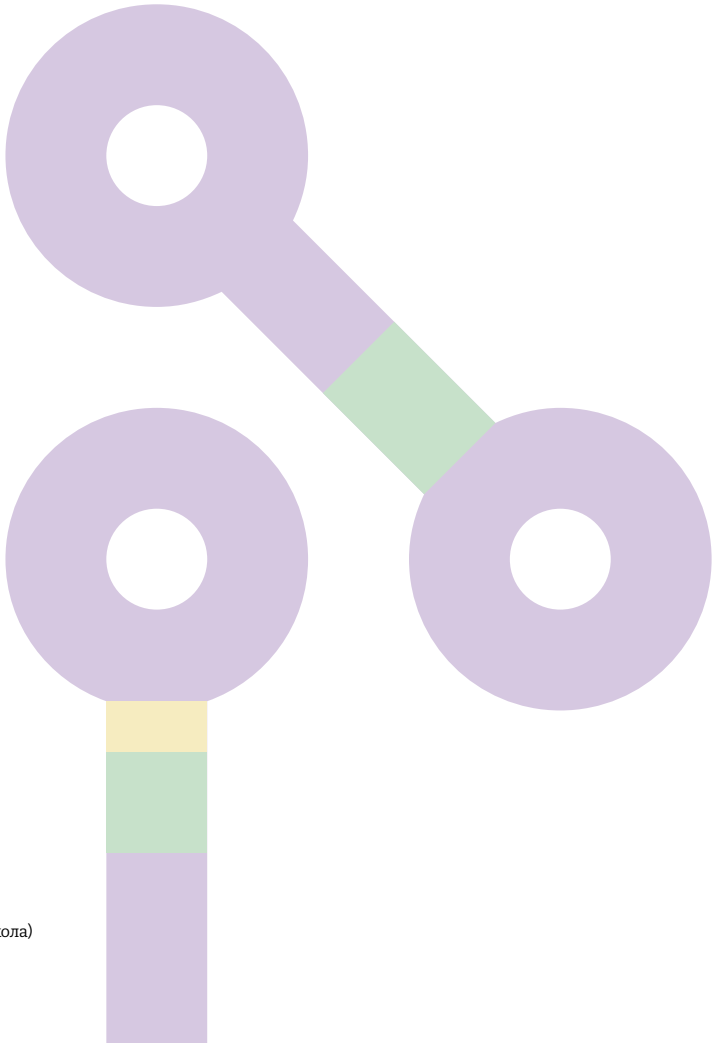
Клуб IT Girls жели вам успјешно кодирање и прегршт пројеката у којима ћете уживати.

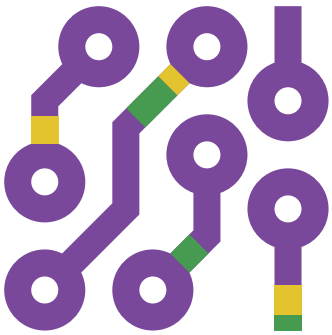
Литература

Halilović, M. (2019). *ARDUINO ZA SVE*. ITGirls inicijativa u okviru projekta "IT Girls dolaze u vaše škole".

Vuković, M. (2017). *Mikroupravljački sustav za emulaciju električnih*. Preuzeto sa https://repozitorij.etfos.hr/islandora/object/etfos:1559

1. https://docs.arduino.cc/
2. https://docs.arduino.cc/tutorials/
3. https://www.arduino.cc/reference/en/
4. https://learn.adafruit.com/category/learn-arduino
5. https://www.monolithicpower.com/
6. https://izradi.croatianmakers.hr/lessons/?technology=54





2021.