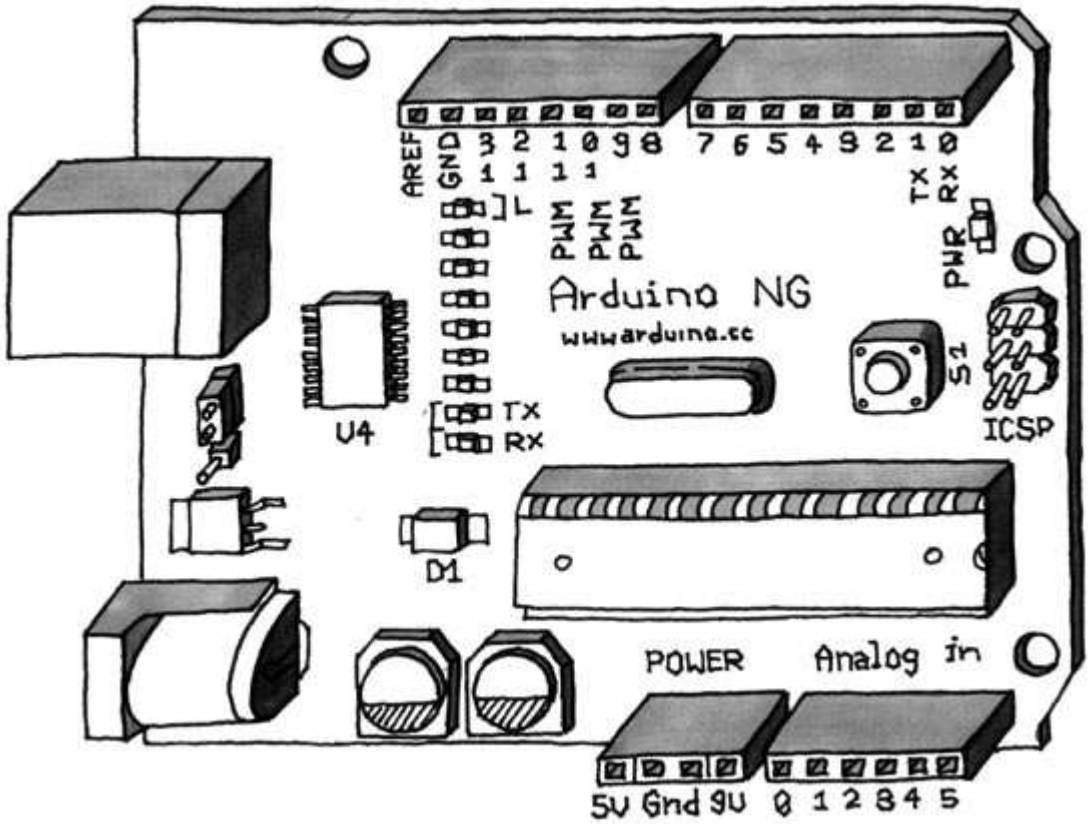
Запознанство с

ARDUINO



от Масимо Банци
с материали от Масимо Банци, Ерика Калогеро, Дейвид Куартиелес, Джеф Грей, Том Игое, Дейвид Мелис и Кристиан Нолд.
Илюстрации от Елиза Кандучи.

Благодарности

Екипът на Ардуино се състои от:
Масимо Банци, Дейвид Куартиелес, Том Игое, Дейвид Мелис и Джианлука Мартино.

Екипът иска да благодари на следните хора и институции за помощта при съставянето на тази книжка:

Алиади Кортелети за дизайна.

Барбара Гела, Стефано Митри, Клаудио Модерини.

Interaction Design Lab, Милано
Domus Academy, Милано
Interaction Design Institute, Милано
Университета в Малмьо, факултет по изкуства, култура и комуникации (КЗ)

ARDUINO

Съдържание

(въведение)

Ардуино е платформа за хоби роботика и физикъл компютинг проекти, базирана на входно-изходна платка и среда за програмиране близка до езика Processing/Wiring. Ардуино може да се използва за създаване на самостоятелни интерактивни предмети или да си взаимодейства с външни софтуерни програми като Flash, Processing, MaxMSP, PureData.

Средата за програмиране, която също е с отворен код, може да бъде свалена безплатно за Windows, Mac OS X и Linux.

Ардуино се различава от останалите подобни платформи на пазара по това че:

Платформата е разработена в образователна среда, което я прави идеална за бързо навлизане в материята от начинаещи.
Работи под различни операционни системи – Windows, Mac, и Linux
Средата за програмиране е базирана на езика Processing.
Програмира се през USB кабел, а не през сериен порт. Това е особено полезно тъй като повечето съвременни компютри нямат сериен порт.
Софтуерът и хардуерът са с отворен код – ако желаете може да свалите схемата от сайта на Ардуино, да си закупите частите и сами да си сглобите платката без да плащате нищо на създателите на Ардуино.
Платката е абсолютно достъпна и струва около 60 лева, а подмяната на изгорял чип около 15 лева така че да можете да си позволите да правите грешки.
Има изградено и силно общество от ентусиасти и съответно много хора, които биха могли да ви помогнат при нужда.

Чудите се какво ли значи всичко това? Отговорите ще получите от тази книжка, създадена да помогне на ентусиасти по роботика, дизайнери и хора на изкуството да разберат ползата, която могат да извлекат като се научат да използват платформата на Ардуино и приемат неговата философия.

/ какво е интеракшън дизайн?

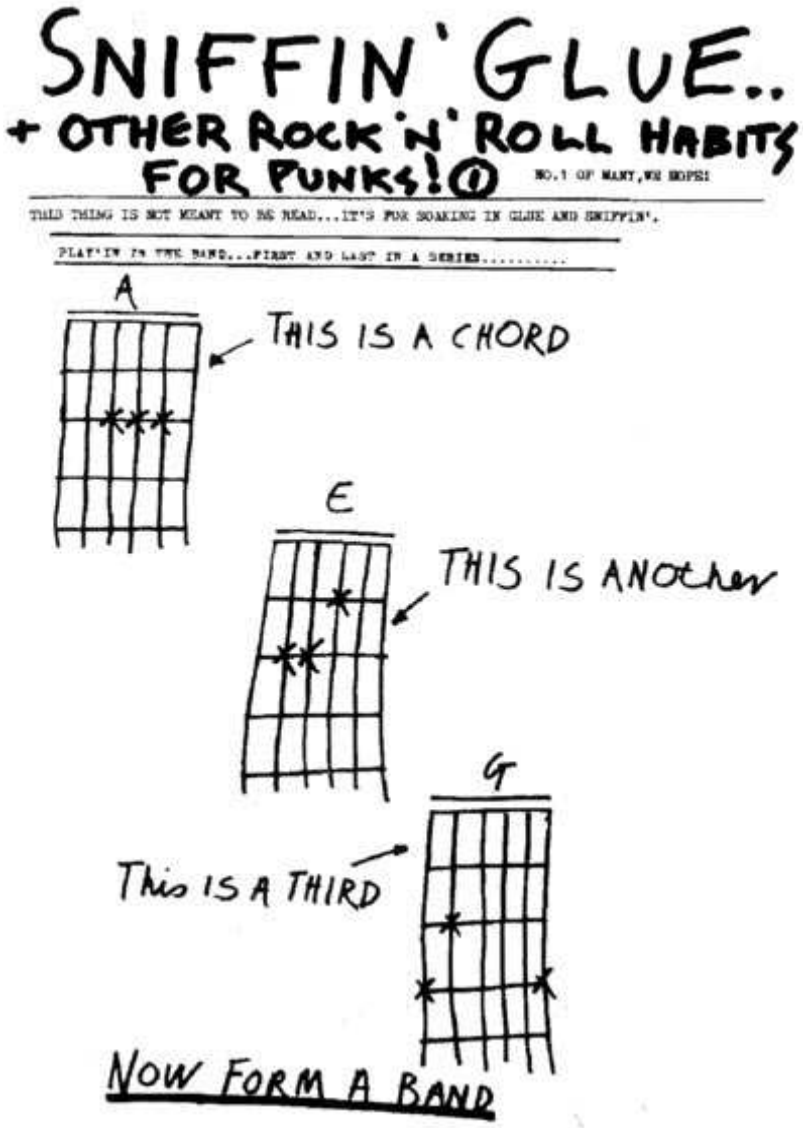
Има много определения за интеракшън дизайн, но моето любимо гласи просто: „Интеракшън дизайнът е създаването на каквото и да е интерактивно преживяване.” В днешни дни това обикновено се отнася за създаването на интерактивност между нас хората и различни предмети или продукти. Ние в Ардуино обичаме да търсим прекрасната, и може би дори нестандартна, интерактивност между хора и технология. Силно вярваме в проектирането чрез итеративен процес базиран на прототипи с постоянно растяща прецизност. Този подход, който е част от някои видове традиционно проектиране (или дизайн), може да бъде разширен за да включва прототипи от света на технологиите и електрониката.
Този род интеракшън дизайн се нарича физикъл компютинг. Настоящата книжка в никакъв случай не може да замени книга по физикъл компютинг и ние ви препоръчваме да си купите отличната книга на Том Игое „Physical Computing”.

/ какво е физикъл компютинг?

Физикъл компютингът се състои в създаването на електронни прототипи, превръщайки датчици, задвижващи механизми и микроконтролери в материали за дизайнери и хора на изкуството.
Това включва проектирането на интерактивни предмети, които могат да „общуват” с хората посредством датчици и задвижващи механизми, контролирани от софтуер вървящ на микроконтролер.

В близкото минало се налагаше намесата на инженери при използване на електронни компоненти в проектите, което не позволяваше на хората на изкуството да си играят директно с такъв вид медии. Повечето от електронните „инструменти” бяха предназначени за инженери и изискваха задълбочени познания. В последните няколко години микроконтролерите (малки компютри, съставени от един единствен чип) станаха евтини и лесни за използване, позволявайки да се създадат по-добри „инструменти” за проектиране и изработка на електронни проекти.

Постарахме се чрез Ардуино да направим тези инструменти още по-достъпни за начинаещите, позволявайки им да създават свои собствени прототипи само след 3-4 дни обучение. С Ардуино всеки ентусиаст може да се запознае с основите на електрониката и да проектира свои прототипи срещу скромна инвестиция.



От английското пънк списание „Sniffin' Glue" (Дишане на лепило), излязло около 1997.

(подходът „Ардуино”)

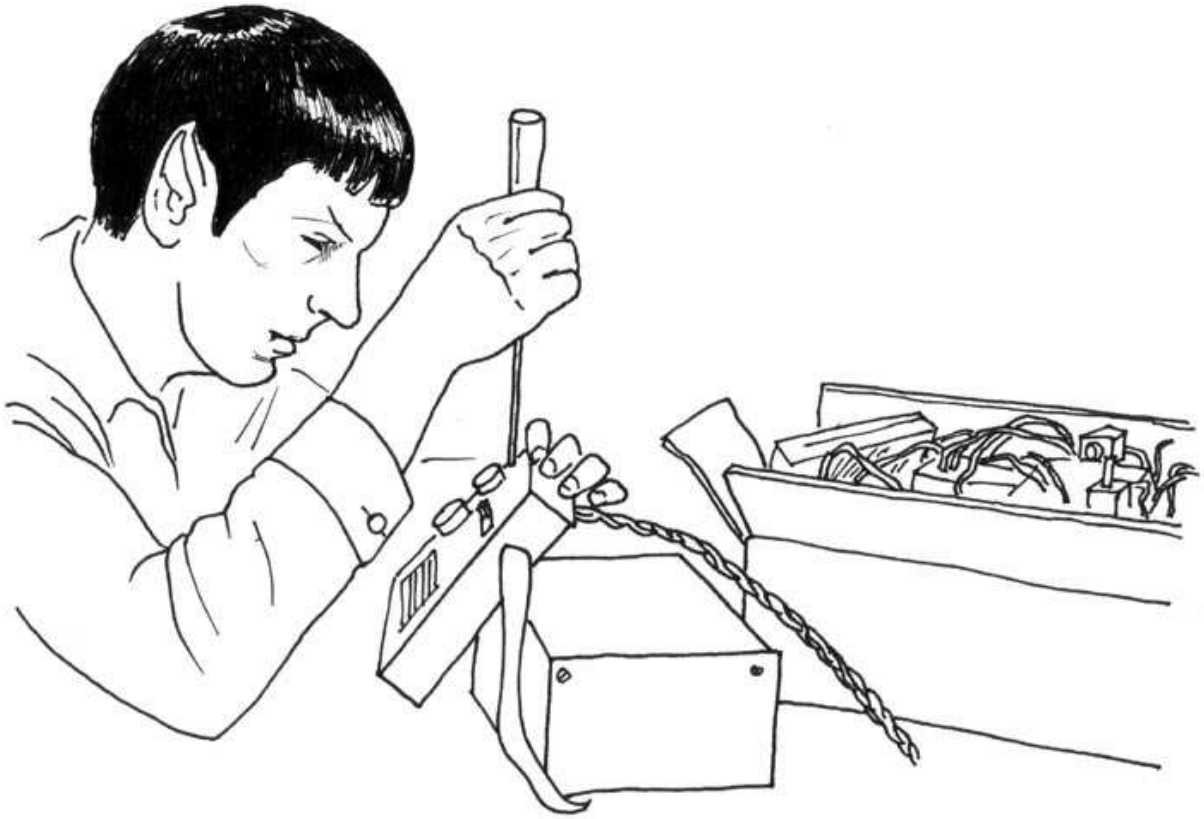
Философията на Ардуино се базира на правенето на проекти, а не на говоренето за проекти. Тя е постоянно търсене на по-бързи и точни начини за създаване на по-добри прототипи. Изпробвали сме много техники за правене на прототипи и сме се научили да мислим с ръцете си.
Докато инженерите се стремят да стигнат от точка А до точка В следвайки стриктни правила, подходът на Ардуино се основава на изгубването по пътя и откриване на точка С. Този процес се нарича тинкъринг и ние сме големи негови почитатели; да си играем без да следваме стереотипи и сами да се изненадваме на откритията си. В нашето търсене сме избрали няколко софтуерни продукта, които да ни помагат в постоянното манипулиране на хардуер и софтуер.

Друга от нашите концепции е наречена „възползнически прототипи”: защо да хабим време и енергия за създаване на прототипи от А и Б, процес изискващ време и дълбоки технически познания като можем да хакнем готови продукти и да се възползваме от тежкия труд вложен от големите компании и платените инженери. Например в Ивреа на няколко сметища е захвърлено наследството на Оливети. Там след фалита на италианската компания са изхвърлени компютърни части, електронно компоненти и уреди, които могат да бъдат купени за по няколко евро, хакнати и използвани в нови прототипи. Така значително се съкращава времето за разработка.

Последният елемент от философията е обществото – стремеж да запалим интереса на хората и да ги насърчаваме да споделят откритията си като споделяме първи.

В следващите няколко параграфа ще видите някои от нещата, които вдъхновиха „подходът Ардуино”.

/ **тинкъринг** (от анг. Tinkering)



Вярваме, че е абсолютно необходимо да се заиграваме с медията - да изпробваме всевъзможни хрумвания директно върху хардуера и софтуера, понякога дори и без да имаме ясна цел. Ние наричаме този процес „Тинкъринг". Едно изложение, което се състоя през 2004 в „Експлораториум" даде най-доброто определение за тинкъринг:

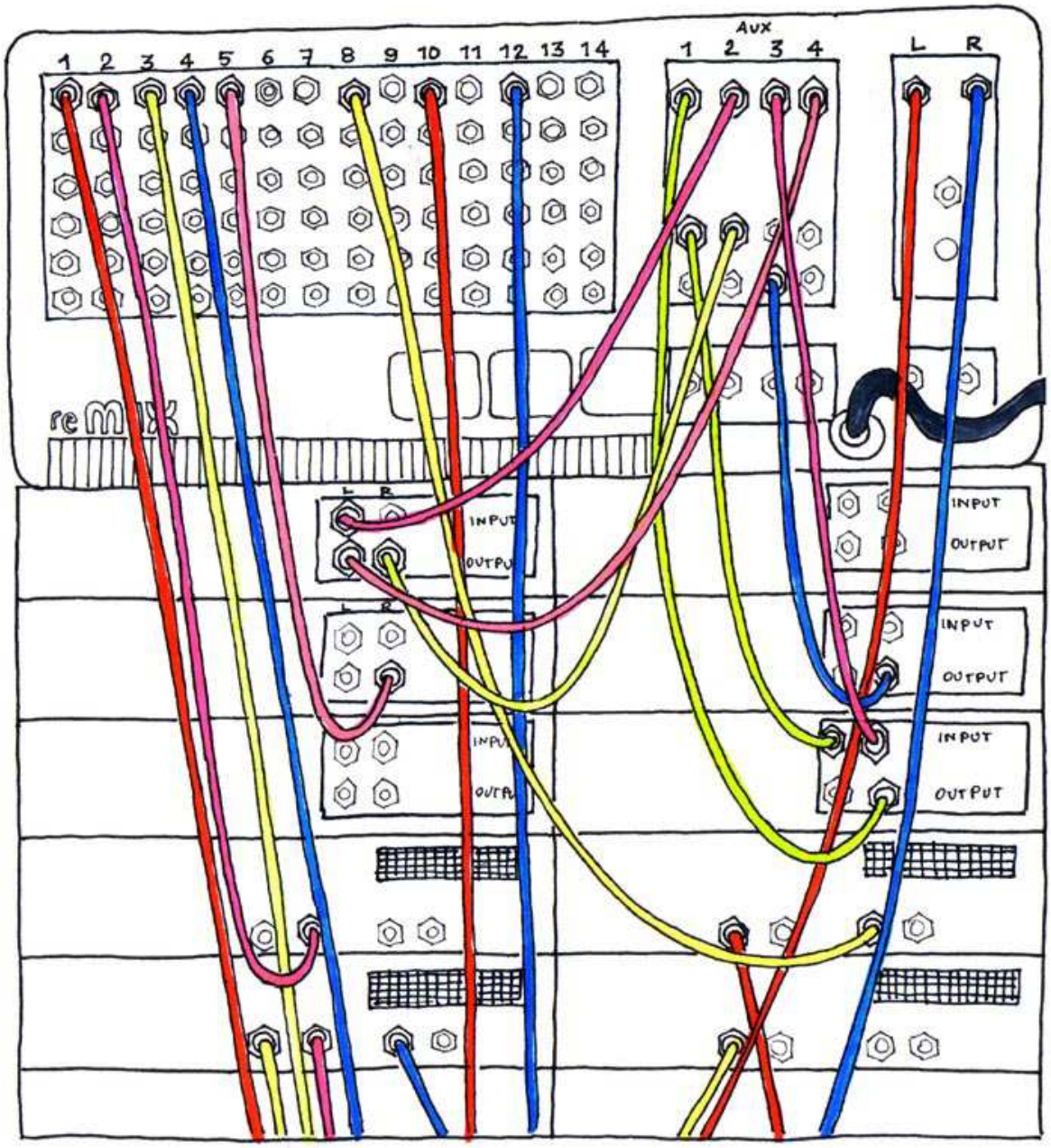
Тинкъринг е това, което се случва докато се опитваме да направим нещо без да знаем точно как да го направим, водени от импулс, въображение и любопитство.

При тинкъринга няма упътване, няма и провал, нито грешен или правилен подход. Същността е в това да разберем как нещата работят и да ги променим.

Джаджи, машинки и безумно разнородни компоненти събрани в един предмет и работещи в хармония – това са резултатите от тинкъринг.

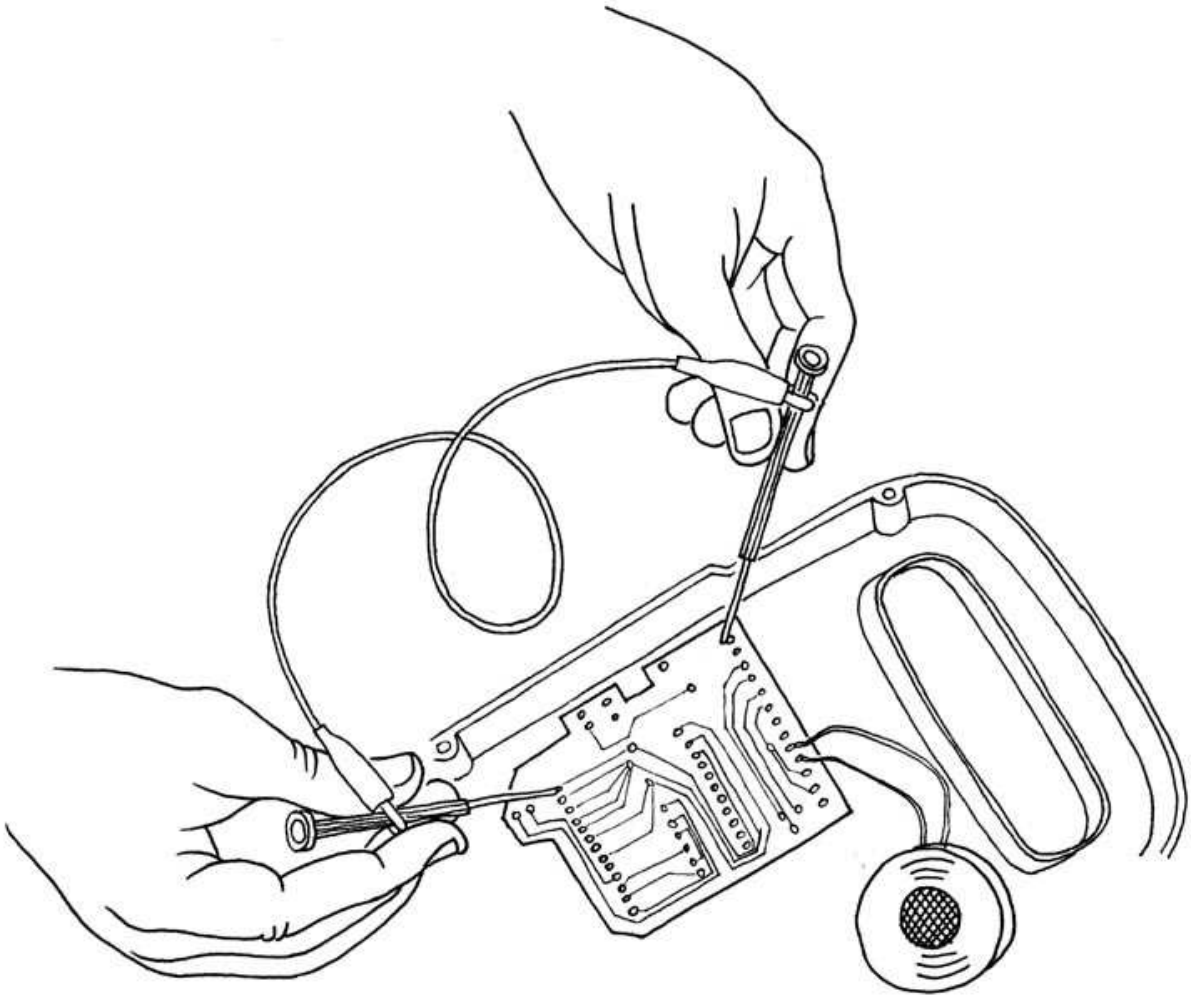
Преработката на вече открити технологии е един от най-добрите начини за тинкъринг. Хакването на евтини играчки или изхвърлени електронни компоненти и чаркове, така че да правят нещо ново, е сред най-добрите начини да се постигнат отлични резултати.

/ **пачване** (от анг. Patching)

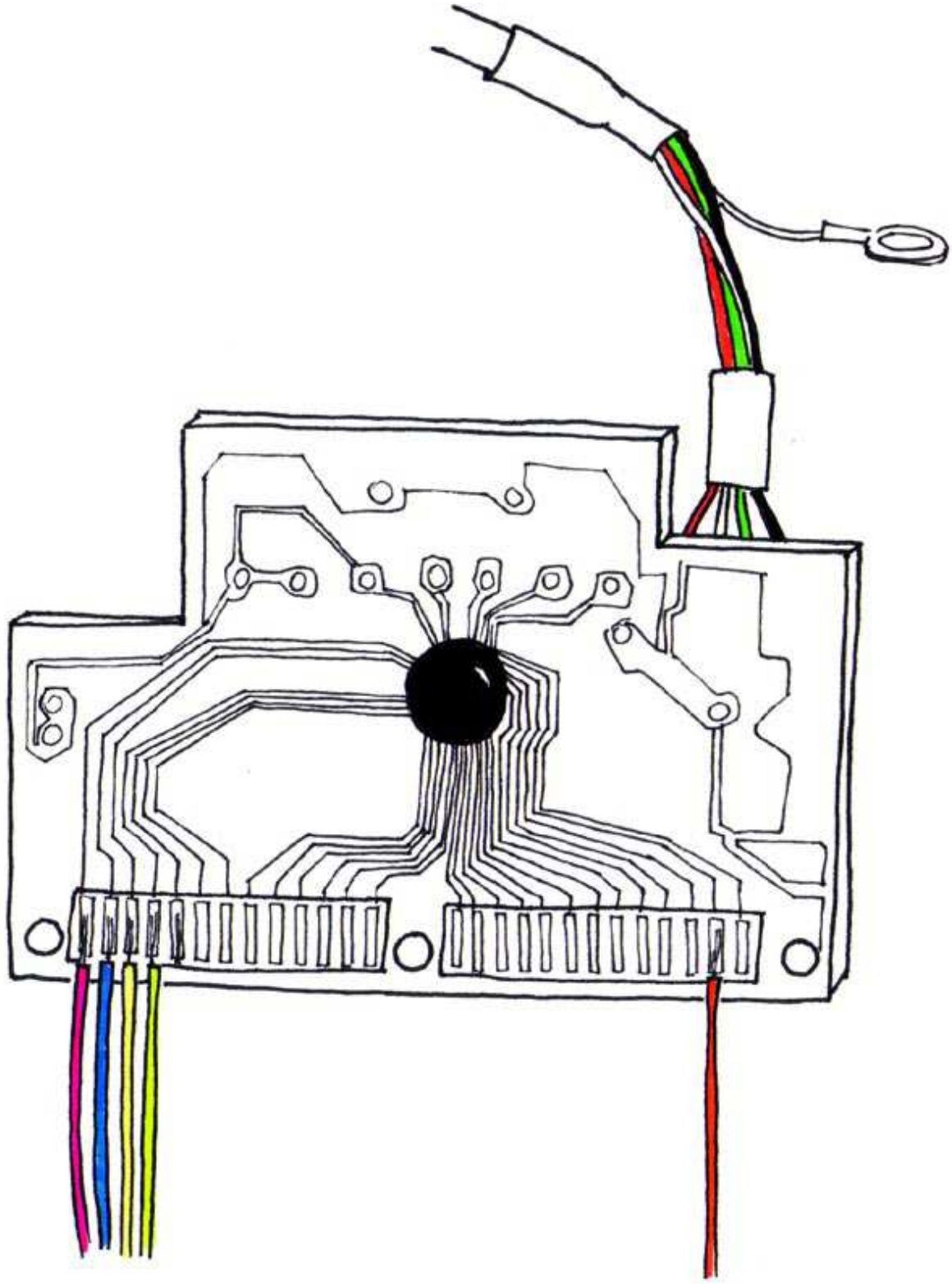


Робърт Муг прави първия аналогов синтезатор с модулен синтез, давайки свобода на музикантите да изпробват безбройни комбинации като „пачват" различните модули посредством кабели. Така синтезаторът прилича на едновремешна телефонна централа, но комбинацията с многобройните копчета е перфектна платформа за тинкъринг със звук и иновации в музиката. Този подход е пренесен в света на софтуерните програми от продукти като Max и PureData.

/ игра на „късо”

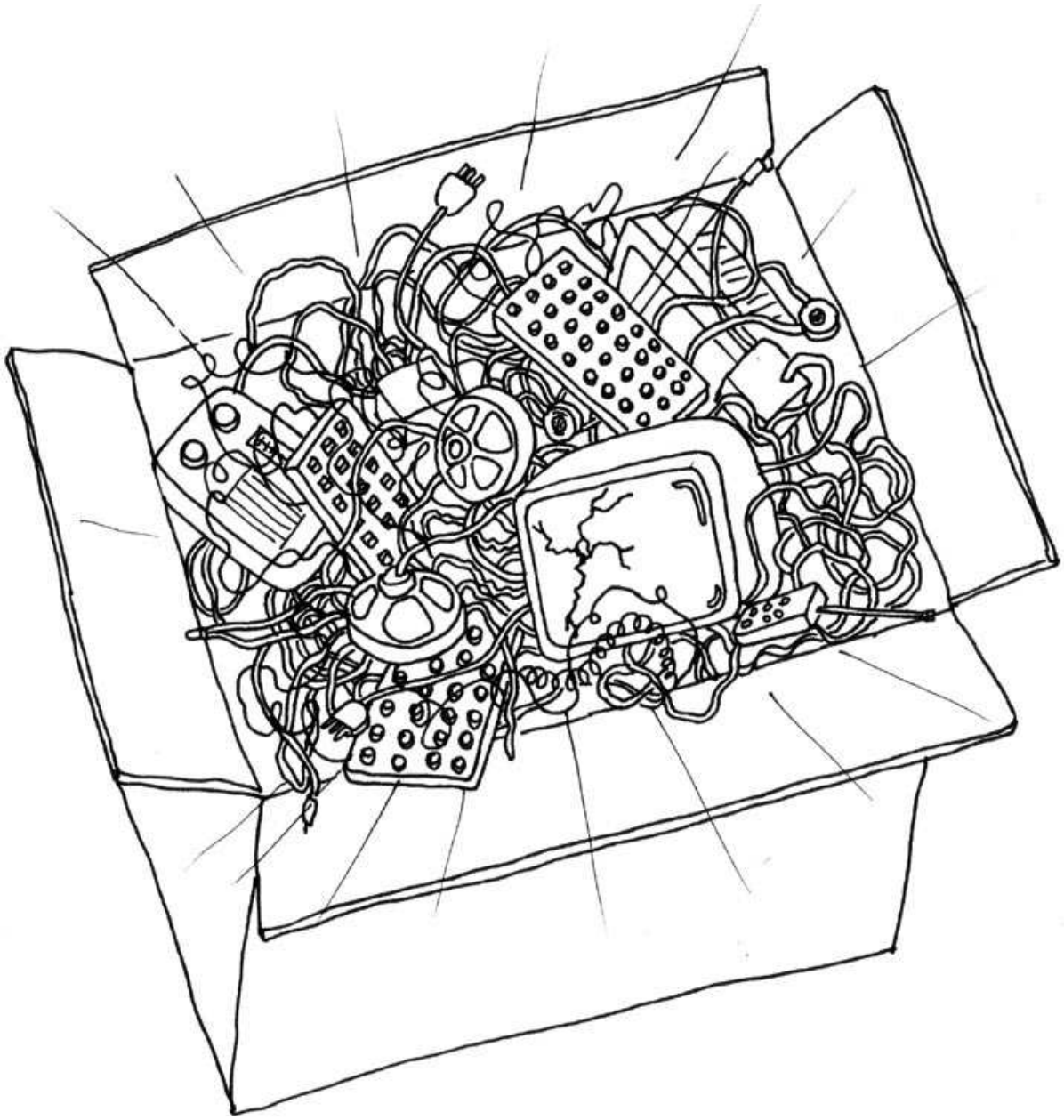


Играта на „късо” е една от най-интересните форми на тинкъринг. Състои се в креативното даване на късо на нисковолтови, захранвани от батерии, уреди като ефекти за китара, детски играчки, и малки синтезатори с идеята да се създаде нов музикален инструмент или генератор на звук. В основата на този процес е „изкуството на късмета”. Основоположник е Рийд Газала, който по случайност допира детски усилвател в метален предмет на своето бюро, като така го дава на късо и получава поредица от странни звуци.



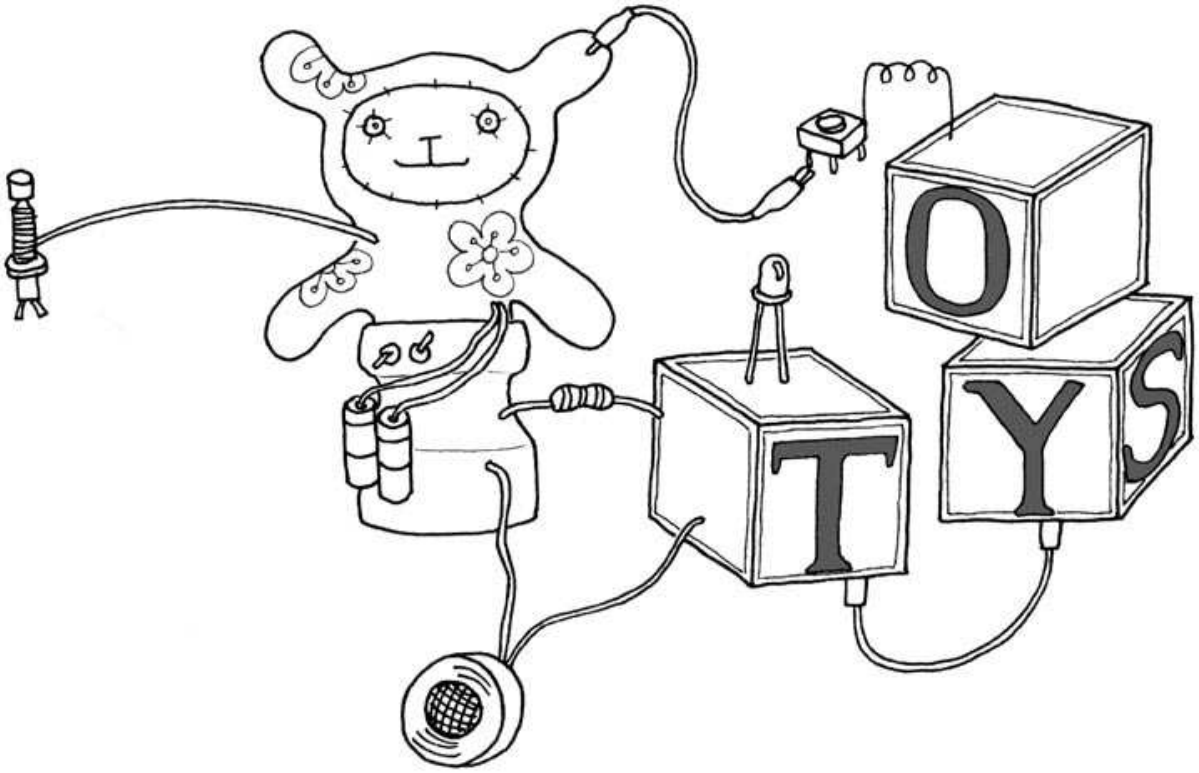
Клавиатурата е едно просто (и евтино) устройство, чието сърце е малка платка. Ако заменим матрицата на клавишите с датчици можем да имаме съвсем нов начин за взаимодействие със софтуера. Клавиатурата е ключов хардуерен компонент за начинаещи в областта на физикъл компютинга.

/ обичаме отпадъците



Един от най-добрите начини да се постигнат бързи резултати е ако намерим източник на електронни отпадъци, които да се използват за бързо (и евтино) създаване на нови прототипи. Понасъберете повечко отпадъци и ги преглеждайте преди да започнете създаването на нов прототип от А и Б.

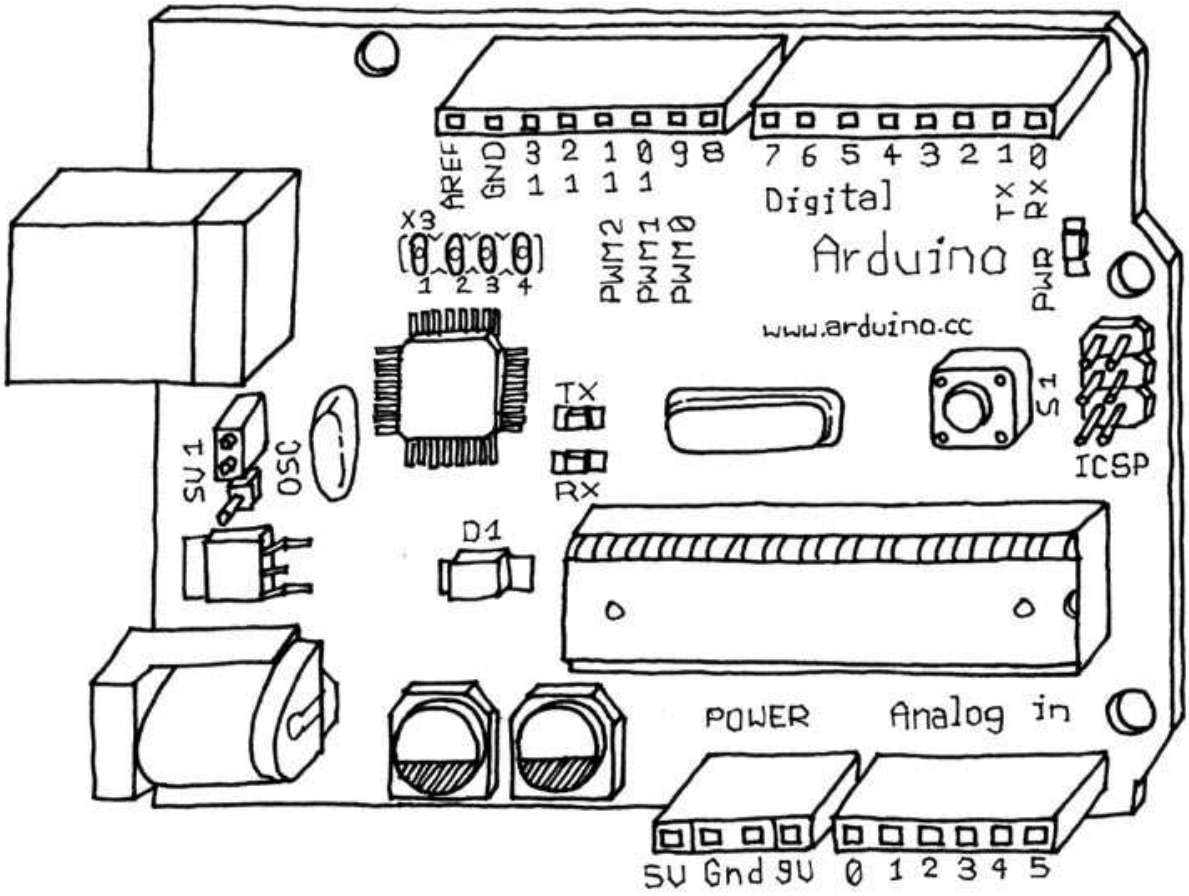
/ хакване на играчки



Играчките са чудесен източник на евтини технологии, които да се хакват и използват по ново предназначение. Най-добрата интерпретация на този начин на работа е от Хусман Хаке и Адам Сомлай-Фишер, които описват перфектно този подход в проекта „Нискотехнологични датчици и задвижващи механизми.”

/ взаимопомощ

/ хардуерът на Ардуино



Това е рисунка на хардуера на Ардуино. Възможно е на пръв поглед да се объркате от многото конектори, затова сега ще обясним какви са елементите на платката:

14 Цифрови входно-изходни пина (пинове 0-13), които могат да бъдат входове или изходи в зависимост от това как са зададени чрез софтуерната програма.

6 Аналогови входа (пинове 0-5), които приемат аналогови стойности (т.е. замервания за ел. напрежение) и ги превръщат в число от 0 до 1023.

3 Аналогови изхода (пинове 9, 10 и 11), те са 3 от цифровите пинове, на които може да бъде зададено да изпълняват ролята на аналогови изходи.

Платката може да бъде захранвана през USB порта или през захранващия жак. Предпочитанието се задава от скобата маркирана със SV1. Ако скобата е по-близо до USB порта, платката се захранва от него. Захранва се през жака ако скобата е по-близо до него.

/ софтуерът (средата за програмиране)

Софтуерът е последния от компонентите на Ардуино. Това е специална среда за програмиране, която върви на вашия компютър и ви позволява да пишете програми за Ардуино на лесен език, базиран на езика Processing. Магията се получава когато натиснете бутона, който ъплоудва вашата програма на платката. Вашият код се превежда на C, (който обикновено е доста труден за начинаещи) и се предава на avr-gcc компилатора, софтуер с отворен код който прави превода на разбираем за микроконтролера език. Тази последна стъпка е доста важна защото с нея Ардуино ни улеснява живота и елиминира възможно най-сложната част от програмирането на микроконтролера.

Даунлоуд и инсталация на софтуера

За да програмирате Ардуино трябва да свалите средата за програмиране от: http://www.robotev.com/biblioteka.php или от http://www.arduino.cc/en/Main/Software
Изберете версията за вашата операционна система.
Изтеглете файла и го разархивирайте.
Първото, което трябва да направите е да инсталирате драйвърите, които позволяват на компютъра ви да си говори с платката Ардуино през USB порта.

За Macintosh:

Намерете папката “Drivers” във фолдъра “arduino-0004” и кликнете два пъти на файла FTDIUSBSerialDriver_v2_0_1.dmg. След като този файл се отвори инсталирайте програмата от FTDIUSBSerialDrive.pkg. След като този файл се инсталира рестартирайте компютъра, за да е сигурно, че драйвърите са инсталирани както трябва. След като инсталирате успешно трябва да пуснете командата „macosx_setup.command” и следвайте инструкциите. Когато програмата попита за паролата ви, напишете тази с която се логвате на компютъра. След като тази програма мине успешно трябва да изключите компютъра. Недейте просто да рестартирате или да се разлогвате – буквално изключете компютъра и после го включете наново.
Сега вече може да свържете Ардуино към компютъра.

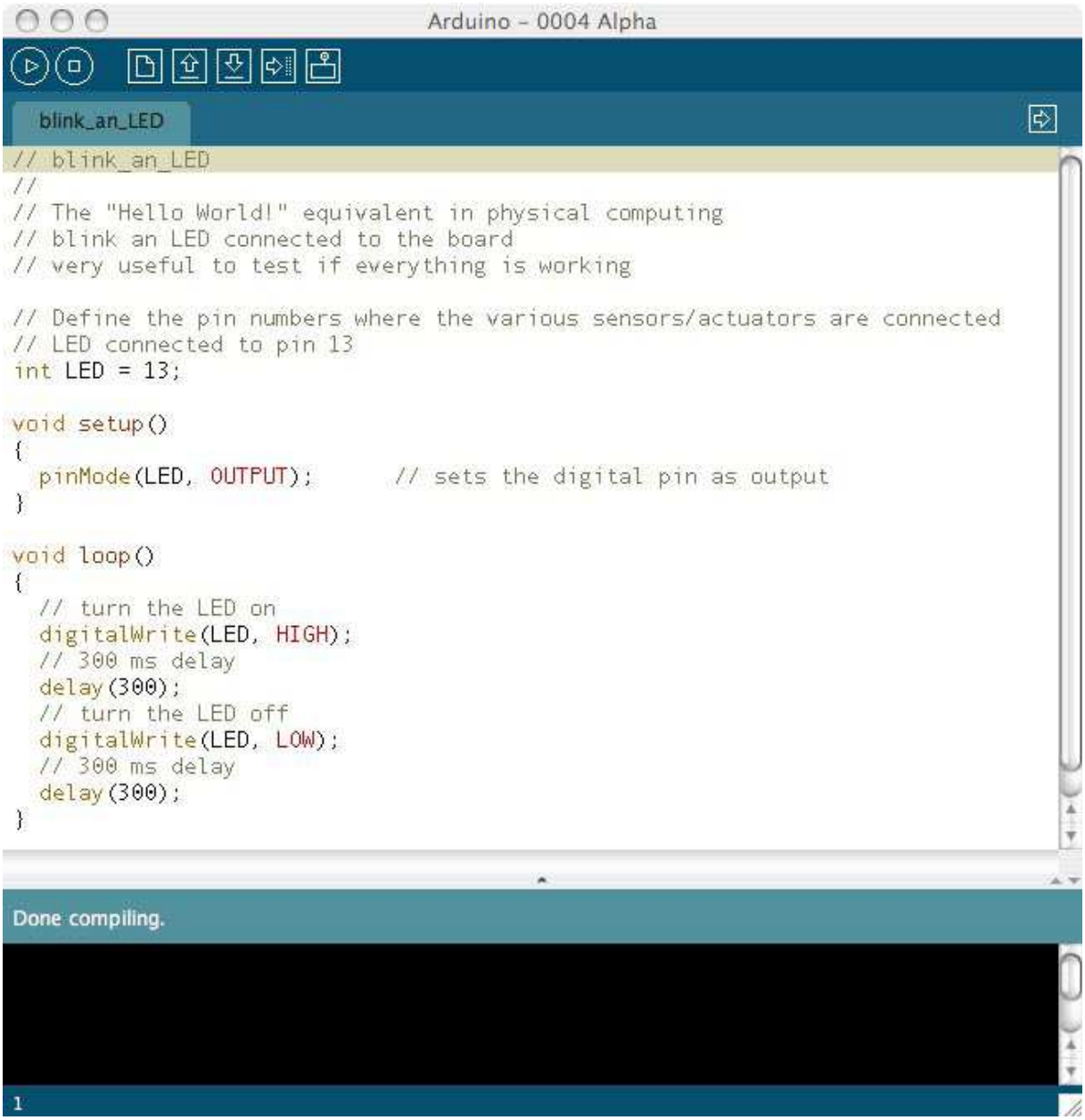
За Windows:

Разархивирайте файла [FIXME], намиращ се в папка “Drivers” в някоя директория, която лесно да откриете. Свържете платката Ардуино с компютъра и когато се появи прозорецът [FIXME] New Device Found [/FIXME], укажете на инсталационния уизард къде да намери драйвърите.
Тази операция ще се повтори два пъти защото първия път се инсталира драйвърът от ниско ниво и след това се инсталира код, който маскира платката като сериен порт.

След като драйвърите са инсталирани, можем да заредим и средата за програмиране и да започнем да се забавляваме с Ардуино.

Работа със средата за програмиране

Когато апликацията се зареди ще видите прозорец като този



Първо трябва да разберем към кой порт е свързана платката.

За Macintosh:

В меню “Tools” изберете “Serial Port” и изберете порта, който започва с “/dev/cu.usbserial-„. Последните три символа показват към кой USB порт е свързана платката Ардуино. Те ще се променят ако свържете платката към друг порт.

За Windows:

Тук процесът е малко по-сложен в началото.
Влезте в „device manager” от:

Start menu > Control Panel > Hardware > Device Manager

Потърсете устройството в списъка „Ports (COM & LPT)”:
Ардуино ще се появи като „USB Serial Port” и ще се казва нещо от рода на COM4.
Забележка: На някой Windows машини COM порта има число по-голямо от 9 и това създава проблеми когато Ардуино се опитва да си комуникира с него. В такива случаи потърсете помощ в troubleshooting секцията на www.arduino.cc или се свържете с администраторите на www.robotev.com.

Накрая в средата за програмиране изберете съответния порт от „Tools / Serial Port” менюто.

Сега вече средата за програмиране на Ардуино може да си говори с платката и да я програмира.

REALLY GETTING STARTED WITH ARDUINO

(същинското запознанство с Ардуино)

След като се запознахме с философията и съставните части на Ардуино сме готови да накараме платката да прави разни неща.
В следващите няколко страници ще се запознаем с няколко примера за основните неща, които можете да правите с платката Ардуино. В края на тези упражнения ще сте готови да експериментирате сами с Ардуино.

/ интерактивното устройство

Повечето от предметите, които ще създаваме с Ардуино следват проста схема, наричана интерактивно устройство.

То представлява електронна верига, която може да усеща заобикалящата среда посредством компоненти наричани **датчици**, да преработва информацията от тези датчици като следва **поведение** заложено в софтуера, и да отвърне на заобикалящия свят чрез **задвижващи механизми**.

/ датчици и задвижващи механизми

Датчиците и задвижващите механизми са компонентите, които позволяват на електронните устройства да взаимодействат със заобикалящата ги среда. Тъй като микроконтролерът е много опростен компютър, той може да обработва единствено електрически сигнали (подобно на токовите вълни изпращани между невроните в човешкия мозък) и за да усеща светлина, температура и други физични свойства се нуждае от нещо, което да ги конвертира в електричество. В човешкото тяло, например, окото конвертира светлината в сигнали, които изпраща чрез нерви до мозъка. В електрониката използваме просто устройство наречено фоторезистор, който измерва отразеното върху него количество светлина и изпраща сигнал, който може да бъде разбран от микроконтролера. След като получи данни от сензорите устройството има нужната информация за да „реши” как да реагира. Реакцията се извършва чрез задвижващи механизми. Те са електронни компоненти, които превръщат електрически сигнали във физически действия. Например в човешкото тяло мускулите получават сигнали от мозъка и ги превръщат в движене. В света на електрониката тази роля се изпълнява от задвижващи механизми като електромоторчетата.

В следващите глави ще научим как да получаваме данни от различни сензори и да управляваме различни задвижващи механизми.

/ въведение в основите на програмирането

Възможно е никога преди да не сте се занимавали с програмиране, ако имате опит може да прескочите тази глава.
Същността на програмирането се състои в това да преведем желаното „поведение" от нашето устройство в инструкции, които процесорът да може да разбере. Процесорите използват много детайлен език от ниско ниво, но с времето се появиха и езици от високо ниво. Те са по-близки до човешките езици от инструкции като

```
lda 0x0f
```

Сега да видим как можем да зададем просто „поведение" като го превърнем в програма. По-простите програми могат да се опишат и като изречение, например: „Когато е много тъмно искам лампите да светнат и моторчето да почне да се върти бавно. Това изречение може да бъде написано и като „псевдо-код" (нещо, което прилича на програма, но все пак да е човешки език.):

```
ако нивото на светлината е под 50
включи лампата
включи моторчето на „бавно"
повтори отначало
```

Бързо стигаме до заключението, че ни трябват два вида програмни структури – цикли и условни оператори. Цикълът е необходим за да може процесорът постоянно да получава информация от входните пинове и да изпраща инструкции до изходните. Условните оператори се използват за да проверим за наличието на определени условия и да променим реда на изпълнение на програмата в зависимост от това дали условията са налице или не.

Основната програма за Ардуино изглежда така:

```
// tova e komentar

// deklarirane na promenliva

int x;

void init () {
// sloji koda tuk
}

void loop () {
//sloji koda tuk
}
```

В началото на програмата декларираме променливите – части от паметта, в която се съхраняват данни. След това използваме void init () функцията за да подготвим програмата (да определим кои пинове ще използваме за входове и кои за изходи). Последната функция void loop () ще се изпълнява непрестанно докато не изключите платката Ардуино. Поради тази причина тук слагаме логиката на условията. Това е същинската програма и в тази част можем да контролираме „потока на програмата". Текст предхождан от // е коментар, който микроконтролерът не чете, но е много полезен за да ни подсказва каква задача изпълнява кода, ако отворим програмата след време или пък я отвори друг.

Променливи

Една от особеностите на програмирането е, че винаги когато искате да запазите дадена стойност трябва да използвате променлива, която на свой ред да е декларирана. Така компютърът знае каква стойност да очаква. Основните видове данни, които компютърът може да очаква са:

int (от integer) – цяло число – 1,2,3,5 и т.н.
byte – цяло число от 0 до 255, който е полезен когато трябва да пестите памет защото той използва само един байт от паметта. (Ардуино има само 1024 байта RAM памет).

За по-голямата част от вашите проекти тези два типа променливи ще са достатъчни. За повече информация вижте в сайта на Ардуино www.arduino.cc

Контролиране потока на програмата

If (ако) [условие] Then (тогава)

При Ардуино if операторът изглежда така:

```
if (израз)
{
оператор;
оператор;
}
```

Където изразът може да е едно от:

a = = b	a е равно на b
a ! = b	a не е равно на b
a > b	a е по-голямо от b
a < b	a е по-малко от b
a = < b	a е по-малко или равно на b
a > = b	a е по-голямо или равно на b

(Важно: Не бъркайте = = израза с = защото той сменя стойността на променливата от лявата страна на равенството. Това може да е много опасно.)

Операторите могат да са каквито пожелаете (включително и други условни оператори).

FOR цикли (FOR loops)

```
for (i = 1; i  <= 8; i++)
{
оператор
}
```

FOR цикълът е начин да се изпълни част от кода точно определен брой пъти. В примера по-горе, операторът се изпълнява точно осем пъти, като минава от 1 до 8.

Забележка: При Ардуино няма нужда да инициализирате локалната променлива i, но в java-базирани езици като Processing трябва да използвате int преди първото споменаване на i.

Забавяне (Delay)

```
delay (1000)  //
```

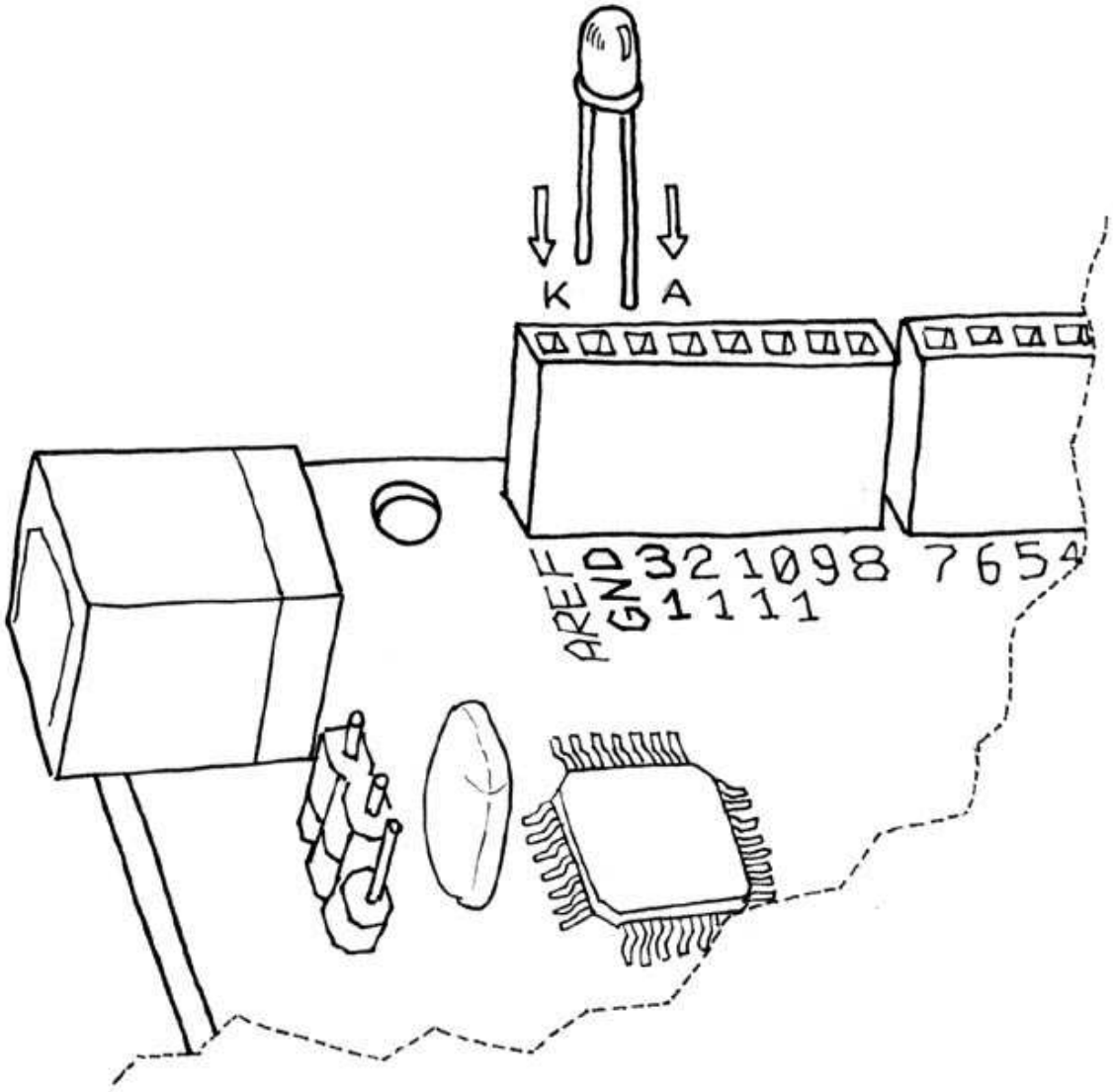
Функцията delay () служи за да забави темпото, с което процесорът се ъпдейтва с информация. Полезна е когато искаме нещо да се случва през определен интервал от време. Например ако искаме светодиод да мига на всяка секунда, след като подадем сигнал да светне можем да зададем delay (1000), което ще накара процесора да не прави нищо в продължение на една секунда (1000 милисекунди). Delay функцията е полезна и за откриване на бъгове и контрол над потока на програмата.

/ мигащ светодиод

Тази програма е първия код, който да стартирате и да се уверите, че платката работи и е конфигурирана правилно. Напишете кода по-долу в средата за програмиране на Ардуино. След като кодът е в средата за програмиране трябва да го проверим и да го „качим" на платката. Натиснете бутона „Verify" и ако всичко е наред най-отдолу на текста ще видите съобщение "Done compiling".
Вече сме готови да качим програмата на платката. Натиснете „reset" бутона на платката. Това ще накара Ардуино да преустанови това, което прави и да чака нови инструкции от USB порта. Сега имаме около 6 – 7 секунди, в които да натиснем „Upload to I/O board". Така ще изпратим програмата до Ардуино, ще я запазим в паметта на платката и ще я стартираме. Ще видите няколко съобщения в черната секция в дъното на прозореца – те ви помагат да разберете дали процесът е завършил успешно. На платката има два диода RX и TX, които примигват за всеки един байт получен или изпратен от платката. По време на качването на програма те трябва да примигват. Ако това не се случва значи има проблем в кабела или не сте избрали верния порт от менюто „Tools/Serial Port".

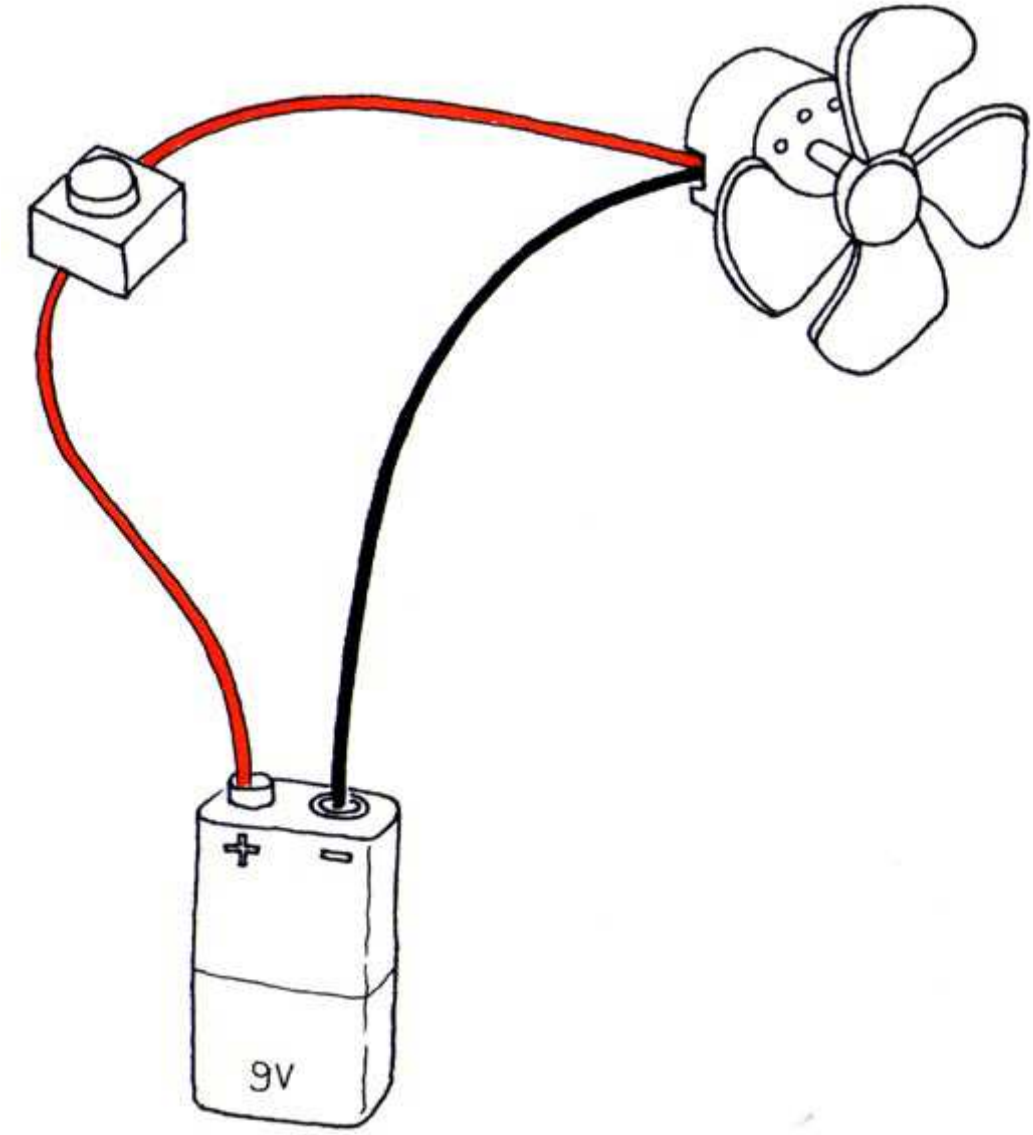
```
/* Migasht diod
* ------------
*
* pali i gasi diod svarzan kam pin 13
*
*/

int ledPin = 13;      // Diodat e svarzan kam
                      // cifrovia pin 13

void setup()
{
      pinMode(ledPin, OUTPUT);      // opredelya cifrovia
                                    // pin kato izhod
}

void loop()
{
      digitalWrite(ledPin, HIGH);   // zapalva dioda
      delay(1000);                  // izchakva sekunda
      digitalWrite(ledPin, LOW);    // izgasya dioda
      delay(1000);                  // izchakva sekunda
}
```

Ако програмата е качена успешно, свържете диода към пинове 13 и GND на платката, както е показано на илюстрацията.



/ какво е електричество

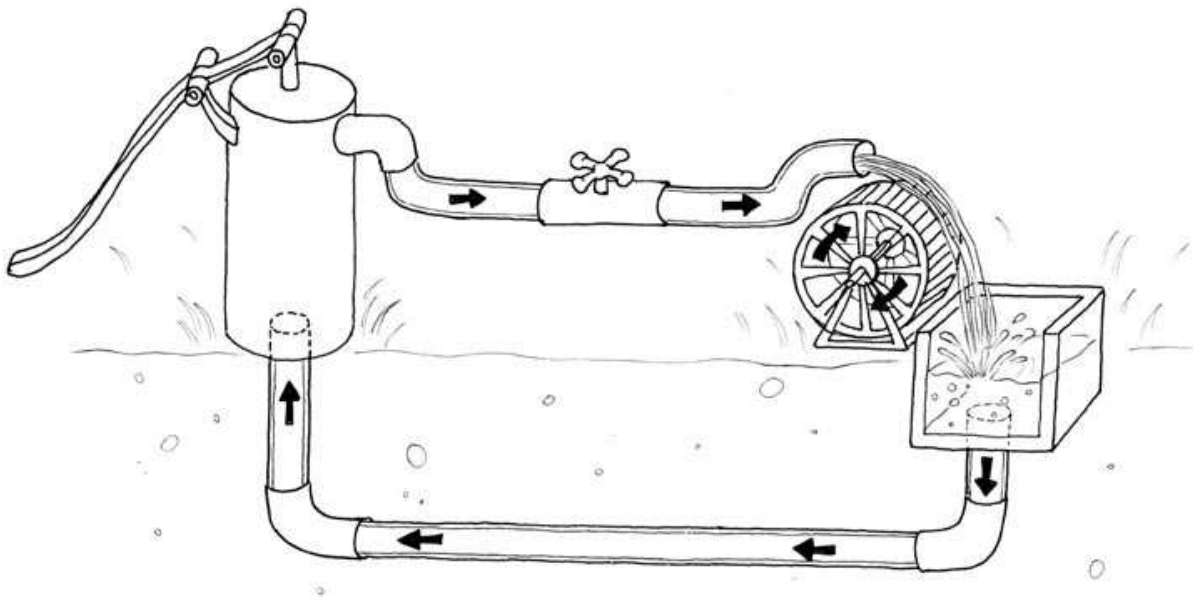
Ако някога сте се занимавали с водопроводната инсталация в дома, електрониката ще ви се стори лесна. Шегата настрана, най-добрият начин да разберете как електричеството и ел. мрежите работят е като си представите модел наречен „водна аналогия”. Да разгледаме един прост уред като преносимия вентилатор.



Ако го разглобим ще видим, че в него има малка батерия и две жици свързани към електромоторчето, като на едната от жиците има прекъсвач. Сложете заредена батерия и включете от прекъсвача – електромоторчето ще започне да се върти, доставяйки желаната свежест. Как работи този уред? Ами представете си, че батерията е водна помпа, прекъсвачът е кран, а пък електромоторчето е едно от онези колела, които сте виждали по едновремешните воденици. Когато отворите крана, водата от помпата стига до колелото и го завърта.

Тази проста хидравлична система има два важни параметъра: налягането на водата (зависи от това колко е мощна помпата) и количеството вода, което може да минава по тръбите (зависи от размера на тръбите и от съпротивлението, което колелото оказва на удрящата го струя вода).

Ако искате колелото да се върти по-бързо трябва да увеличите размера на тръбите (това действа до определен момент) и да увеличите налягането подавано от помпата. Като увеличаваме размера на тръбите ние позволяваме повече вода да преминава през тях. Всъщност с по-големите тръби ние намаляваме съпротивлението, което те оказват на течащата през тях вода. Колкото по-големи тръби слагаме, толкова по-бързо ще се върти колелото, но само до момента в който налягането вече няма да е достатъчно силно за да изпълва тръбите. След този момент ще ни е нужна по-мощна помпа.



Можем да увеличаваме размера на тръбите и мощността на помпата докато струята не стане прекалено силна и не счупи колелото. Друго, което се забелязва е, че когато колелото се върти, оста му се загрява. Без значение колко добре сме захванали колелото, триенето между оста и дупките, в които тя е захваната винаги ще отделя топлина. Важното в случая е, че в система като тази не всичката енергия, която излиза от помпата се превръща в движение. Част от нея се загубва в различни несъвършенства. Загубената енергия се познава по топлината отделяна от частите на системата.

Кои са важните части на гореописаната система? От една страна това е налягането произведено от помпата, а от друга съпротивлението, което тръбите и колелото оказват на течащата вода (да речем, че това може да бъде представено като брой литри за секунда).

Ако не обръщаме внимание на подробностите електричеството работи по подобен начин на водата – има помпа (всеки източник на електричество – батерия или ел. контакта на стената) която избутва електрически заряд (представете си го като „капки” електричество) по тръбите (жиците) до различни уреди. Някои от тях произвеждат топлина (като електрическата възглавничка на баба ви),светлина (лампата в стаята ви), звук (стерео уредбата ви), движение (вентилатора ви) и др.

Ако на една батерия пише 9V може да си представите волтажа на батерията като налягането, което тази „помпичка” би могла да произведе. Тази стойност се измерва във волтове, на името на Алесандро Волта – изобретателят на батерията. Течението на водата си има електрически еквивалент, наричан напрежение, който се измерва в ампери, от Андре Мари Ампер. Съпротивлението оказвано на потока по какъвто и да е начин се нарича, да, точно така, съпротивление. То се измерва в омове, на името на немския физик Ом.

Заслуга на г-н Ом е най-важният закон в електричеството и единствената формула, която трябва да запомните.

Той е успял да демонстрира, че в електрическата верига силата на тока (I), напрежението (U) и съпротивлението (R) са взаимносвързани и, че съпротивлението определя напрежението във веригата при постоянна сила на тока.

Като се замислим, това изглежда много логично: ако включим 9V батерия към проста електрическа верига и измерим напрежението ще видим, че колкото повече резистори слагаме (и така увеличаваме съпротивлението) толкова по-слабо става напрежението. Да се върнем на примера с водата течаща през тръбите. Ако имаме помпа с постоянна мощност и на една от тръбите поставим кран, който можем да оприличим на резисторите в електротехниката, то като затягаме крана (и така увеличаваме съпротивлението на водата) по тръбите ще тече по-малко вода. Ом изразява закона си със следната формула:

R (съпротивлението) = U (напрежението) / I (силата на тока)

$$U = R * I$$

$$I = U / R$$

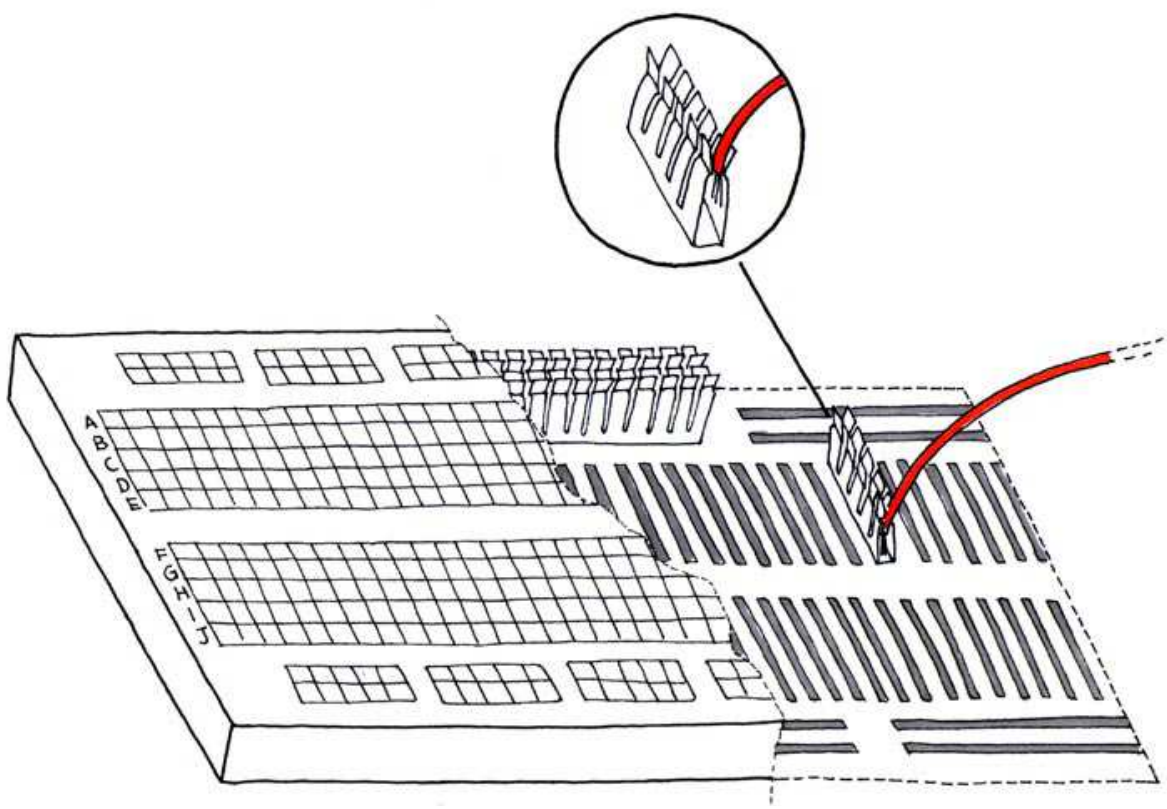
Това е единственото правило, което трябва да запомните и да се научите да прилагате, защото за повечето ви проекти само то ще ви е необходимо.

/ монтажната платка (бредборд)

В процеса на съставяне на електрическа верига се правят много промени докато тя се оптимизира и заработи по желания начин. Това е един много бърз и повтарящ се процес, който прилича на електронния вариант на скицирането. Всеки дизайн еволюира в ръцете ви докато пробвате различни комбинации. За най-добри резултати е препоръчително да ползвате нещо, с което да можете да пренареждате връзките между компонентите възможно най-бързо и практично, без да се налага да чупите каквото и да било.

Такова условие твърдо изключва запояването – времеемък процес, който подлага компонентите под напрежение всеки път когато се загряват и изстиват.

Отговорът на нашия проблем е един практичен уред, наречен монтажна платка – бредборд.



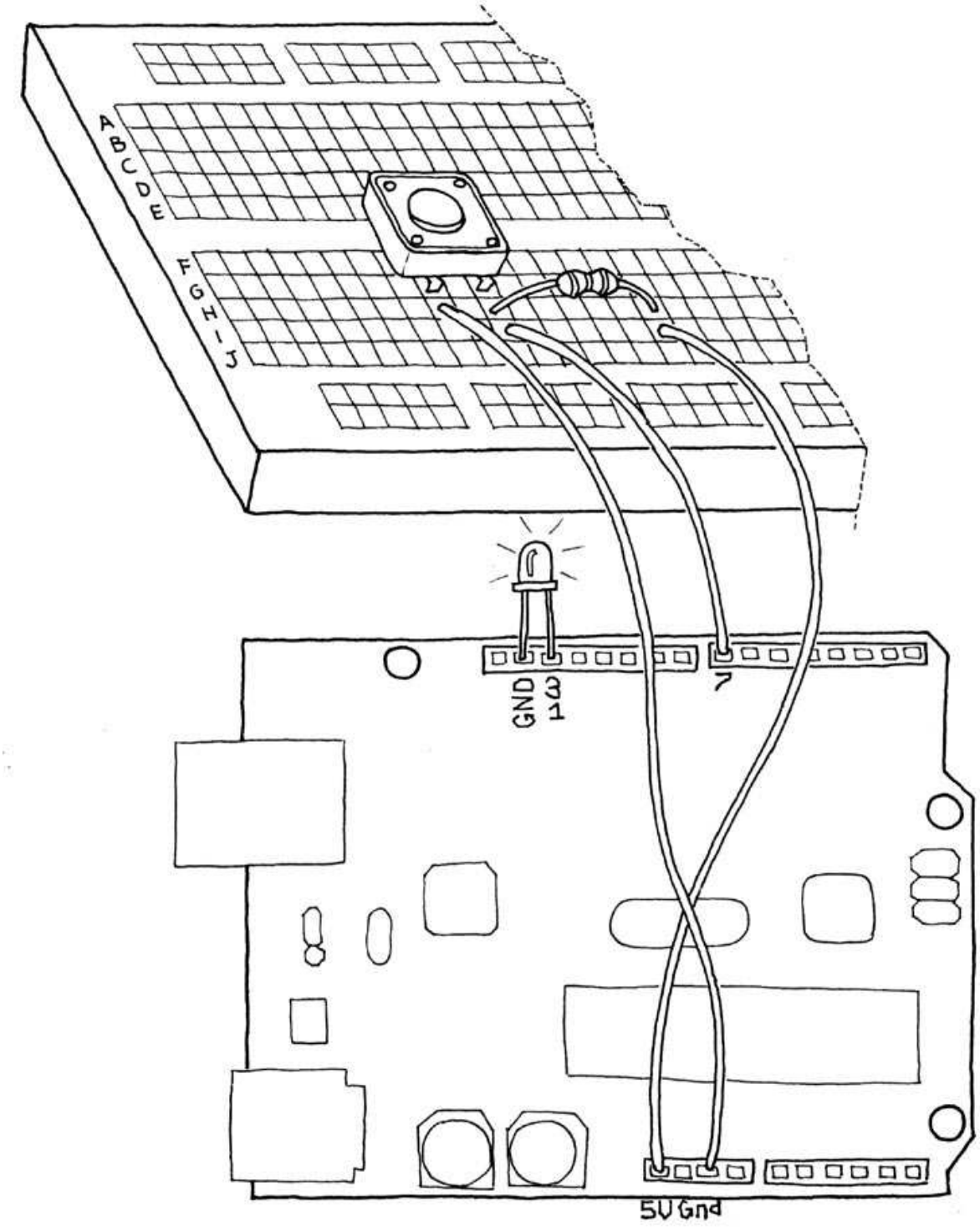
Както е показано на картинката, това е малка пластмасова дъсчица покрита с дупки. Във всяка дупка има пружиниращ контакт. Когато вкарате крачето на някой компонент в някоя от дупките, то се свързва с всички други дупки от вертикалната колона. Всяка дупка е на 2.54 mm от останалите, тъй като крачетата на повечето компоненти са на такова разстояние и чипове с повече крачета пасват идеално. Не всички контакти на монтажната платка са еднакви. Има известни разлики в най-горния и най-долния ред (оцветени с червено и синьо и уместно маркирани с + и -), които са свързани хоризонтално и служат да доставят електричество на цялата платка, така че като ни потрябва напрежение или заземяване да можем лесно да ги набавим с помощта на къса

жичка. Последното нещо, което трябва да знаете за монтажната платка бредборд е, че в средата си има по-голямо разстояние, широко колкото малък чип. Затова и всяка вертикална линия от дупки е прекъсната в средата – за да не може да дадете на късо крачетата от двете страни на чипа – хитро, нали?

/ отчитане на бутон

Сега ще покажем как Ардуино може да получава сигнали от външния свят. Това става с digitalRead () функцията, която казва дали има електрическо напрежение в даден пин. Ако силата на тока е 2.5 V или повече, digitalRead () ще покаже HIGH, а в противен случай ще покаже LOW.
Кодът по-долу проверява статуса на цифровия пин и включва диода ако бутонът е натиснат. Това е много прост пример, който ще ви въведе в материята много бързо, единствено ще трябва да построите веригата нарисувана на следващата страница.

```
/* migane na diod kogato butonat e natisnat
* -----------------------
*/

int ledPin = 13;      // izbirane na pin za dioda
int inPin = 7;        // izbirane na vhoden pin
                      // (za butona)
int val = 0;          // promenliva za otchitane statusa na pina

void setup() {
      pinMode(ledPin, OUTPUT);       // deklarirane na dioda kato izhod
      pinMode(inPin, INPUT);         // deklarirane na butona kato vhod
}

void loop(){
      val = digitalRead(inPin);   // otchitane stoinostta na vhodniya pin

      // proverka dali vhodniya pin e na HIGH (butonat ne e natisnat)
      if (val == HIGH) {
            digitalWrite(ledPin, LOW); // izgasya dioda
} else {
      // diodat miga i sled tova se izklyuchva
digitalWrite(ledPin, HIGH);
delay(200);
digitalWrite(ledPin, LOW);
delay(1000);
}
}
```


A B C D E
F G H I J
GND
13
5V Gnd

/ изпробване на различни вкл/изкл датчици

След като вече знаете как да използвате бутон, може да го замените с най-различни други сензори, които могат да подадат същата информация – имат контакти, които могат да са отворени или затворени.

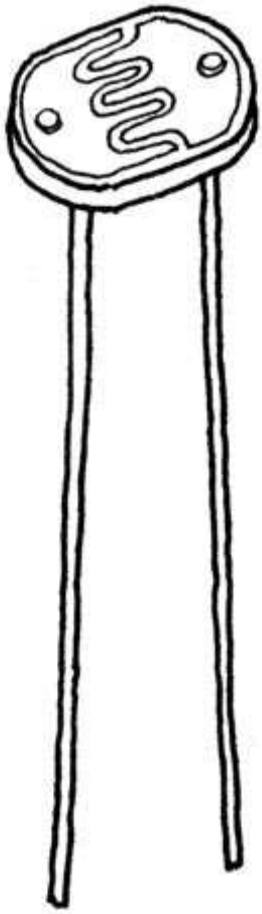
Подходящ пример е сензорът за наклон. Той е прост електронен компонент, съставен от две метални пластини и метално топче. Когато сензорът е в хоризонтално положение, топчето свързва двете пластини, давайки същия резултат както когато копчето на бутона е натиснато. Когато наклоните сензора, топчето се отмества и двете пластини вече не правят контакт, както когато отпуснете копчето на бутона. С такъв датчик може да направите, например, предмети които да реагират при преместване или разклащане.

Друг датчик, който бихте могли да изпробвате е инфрачервеният сензор от алармените системи. Този малък уред се задейства когато някой се движи в близост до него, осигурявайки лесен начин да засичате присъствието на хора.

Експериментирайте с всевъзможни уреди, чийто контакти могат да са свързани или не – като термостатите използвани да поддържат приятна температура в стаите. Или просто като сложите две жички една до друга и капнете вода между тях.

/ използване на фоторезистор вместо бутон

Сега ще опитаме един по-интересен експеримент. Вземете фоторезистор, като този нарисуван тук.

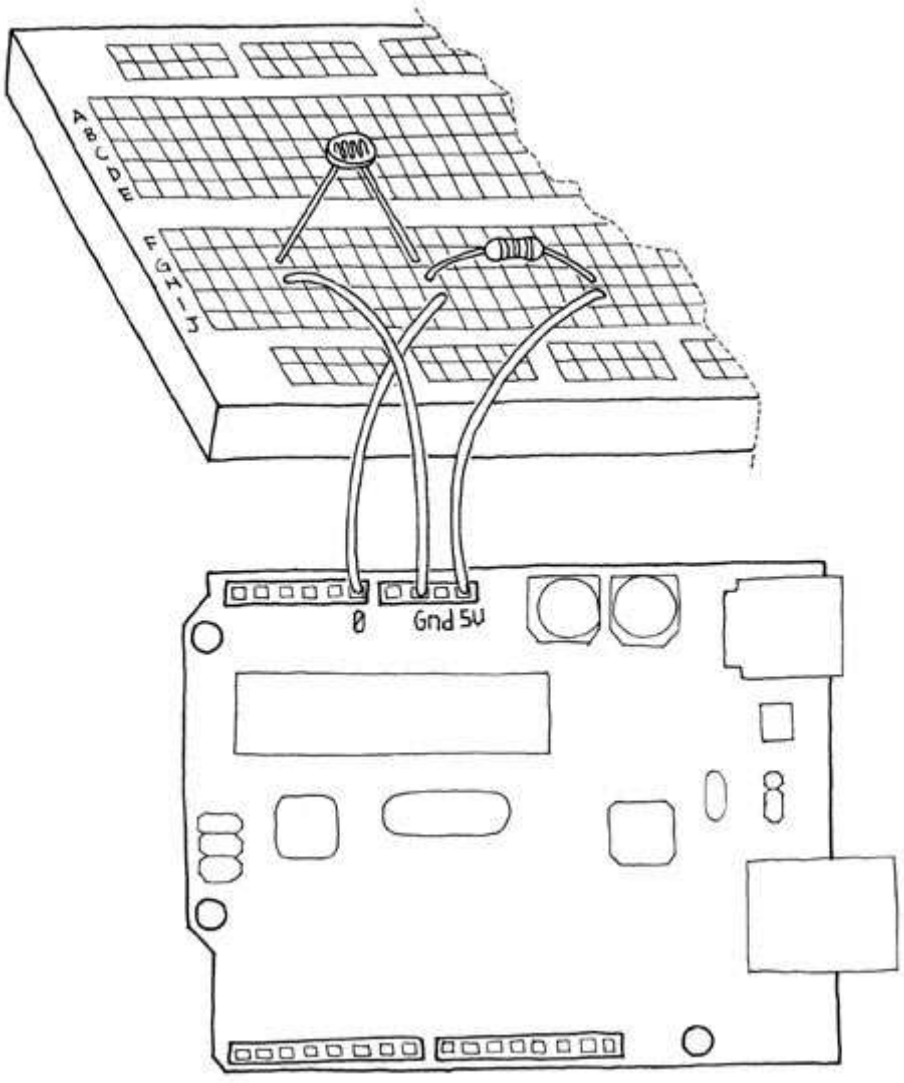


Това е фоторезистор или LDR. Когато е на тъмно съпротивлението му е много голямо, но когато го осветите то бързо намалява, правейки го относително добър проводник на електричество. Сложете фоторезистора на мястото на бутона. Когато го покриете с ръка диодът угасва, а когато я махнете диодът светва наново. Току що измайсторихте първия си диод контролиран от сензор.

/ аналогови входове

Както видяхме в предишната глава, Ардуино може да усети когато има електрически заряд в някой от неговите пинове и да обяви това чрез digitalRead () функцията. Тази функция ни върши работа за голяма част от приложенията на фоторезистора, който използвахме в предишния пример, и също така може да ни даде информация не само за това дали има светлина или не, но и за това колко е ярка. Това е разликата между вкл/изкл датчиците (който ни дават информация само за това дали дадено условие е изпълнено или не) и аналоговите сензори чиято стойност може да се променя. За да можем да получаваме такава информация от аналоговите сензори ни трябва различен вид пин. В долната част на Ардуино платката има шест пина маркирани „Analog In". Те са специални пинове, които не само ни казват дали в тях има електрически заряд или не, но също така и неговата сила (или стойност). С помощта на analogRead () функцията можем да отчетем силата на тока в тези пинове. Функцията връща числа от 0 до 1023 за стойности на силата на тока от 0 до 5 волта. Например ако в пин 0 има заряд със сила 2.5 волта analogRead () ще покаже стойност от 512 и т.н.

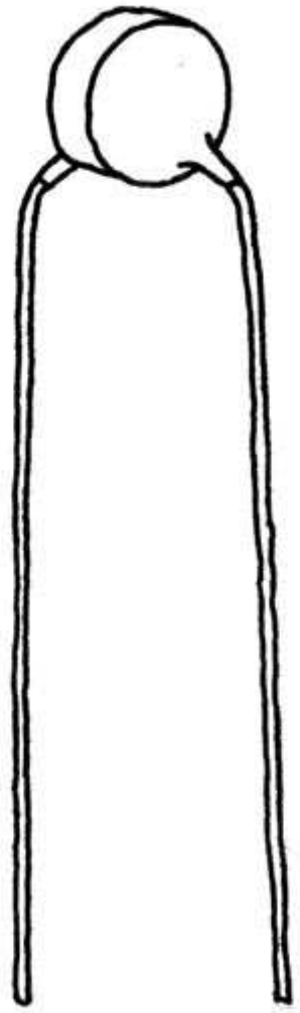
Ако построите веригата нарисувана тук с резистор от 4.7К ома и пуснете кода от следващата страница ще имате светодиод мигащ със скорост зависеща от количеството светлина върху фоторезистора.



```
/*  Funkciyata analogRead i diod
* -----------------
* copyleft 2005 David Cuartielles
*
*/

int potPin = 2;       // izbirane vhoda za
                      // potenciometara (ili fotorezistora)
int ledPin = 13;      // izbirane vhod za dioda
int val = 0;          // promenliva za zapazvane stoinostite
                      // idvashti ot fotorezistora

void setup() {
      pinMode(ledPin, OUTPUT);      // ledPin e izhod
}
void loop() {
      val = analogRead(potPin);     // prochitane stoinostta ot
                                    // datchika
      digitalWrite(ledPin, HIGH);   // zapalvane na dioda
      delay(val);                   // prekusvane na izpulnenieto na
                                    // programata za izvestno vreme
      digitalWrite(ledPin, LOW);    // izgasyane na dioda
      delay(val);                   // prekusvane  na izpulnenieto na
                                    // programata za izvestno vreme
}
```

/ изпробване на различни съпротивителни датчици

Използвайки веригата от предишната глава можете да свържете различни други съпротивителни датчици, които действат в голяма степен като фоторезистора. Например бихте могли да свържете терморезистор.



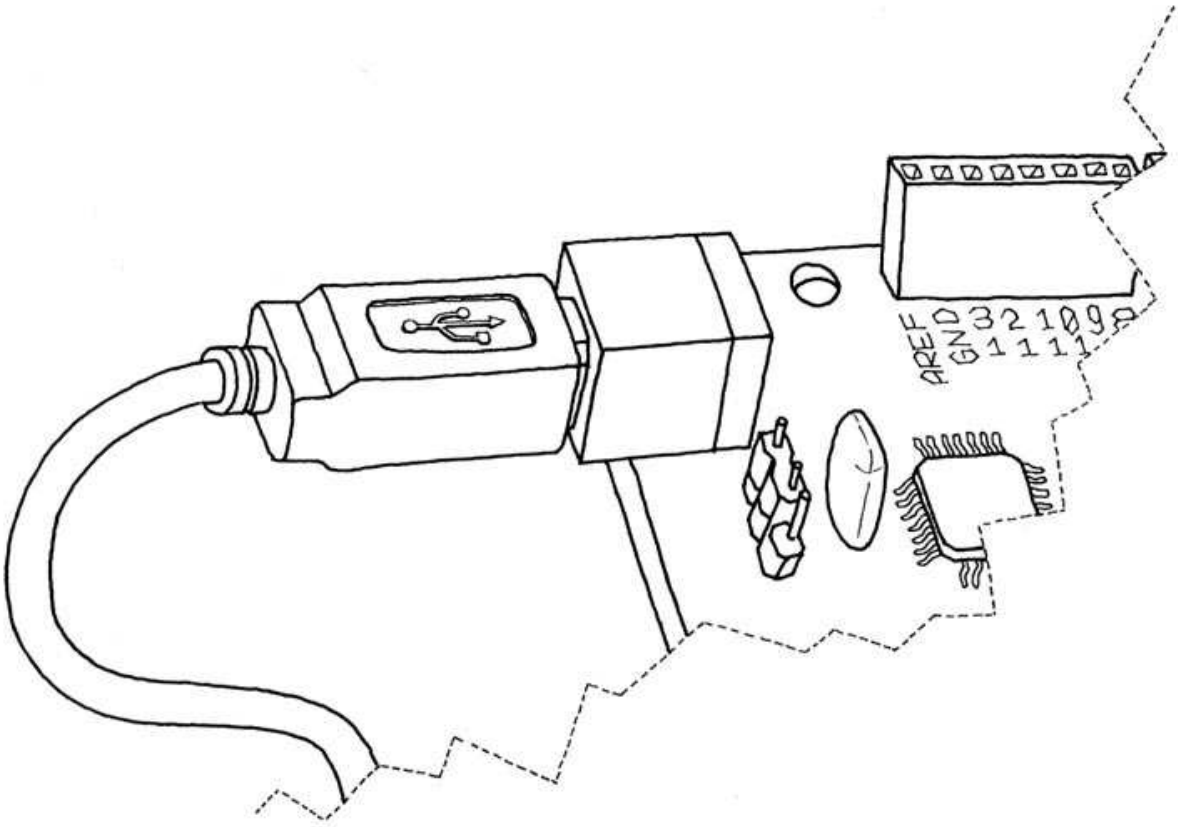
Това е просто устройство, чието съпротивление се влияе от температурата. В предишния пример видяхме, че промяната в съпротивлението променя и силата на тока, която на свой ред може да бъде измерена от Ардуино.

Запомнете, че показанията не показват температурата в градуси. Ако ви трябват точните стойности в градуси трябва да отчитате стойностите от analogRead () и същевременно да замервате температурата с термометър. След това трябва да вкарате резултатите в таблица за да откриете зависимостта помежду им.

Досега използвахме диода като четец, но как можем да видим стойностите, които Ардуино получава от сензорите? Идеята да накараме Ардуино да примигва стойностите използвайки морзов код не е добра. За подобни цели използваме така наречената серийна комуникация, описана в следващата глава.

/ серийна комуникация

Както видяхме в началото на тази книжка, Ардуино е свързан с компютъра чрез USB кабел, който се използва за качването на код в платката.



Добрата новина е, че тази връзка може да се използва от нашите програми за да изпращат данни обратно към компютъра или да получават команди от него.
За целта използваме така наречения „Сериен обект". Този обект съдържа всичкия код необходим за изпращане и приемане на данни.
Сега ще използваме описания по-рано пример с фоторезистора и ще изпращаме неговите показания обратно към компютъра.

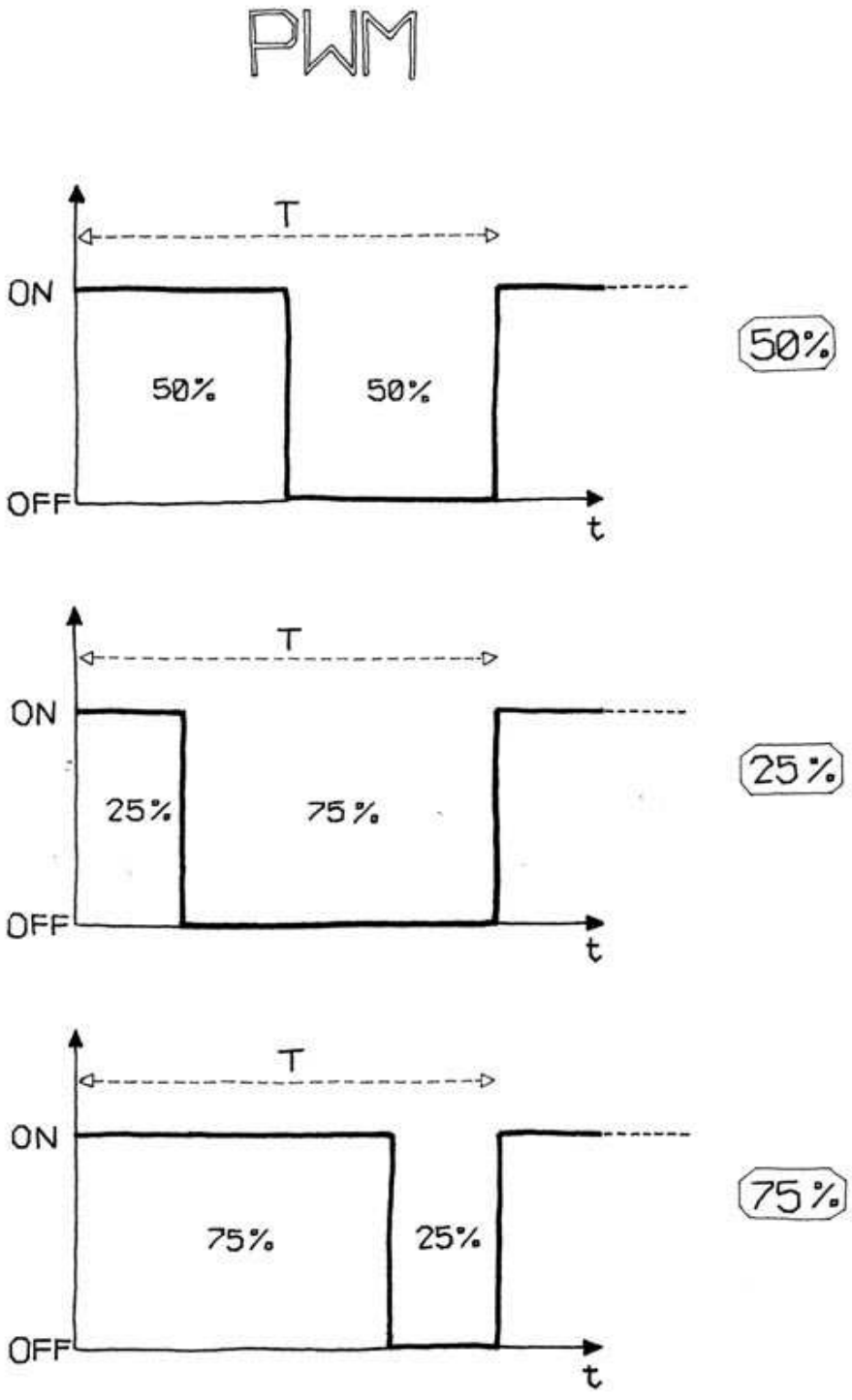
Наберете този код в нова форма в средата за програмиране:

```
int sensorPin = 2;    // izbirane vhoda za
                      // potenciometara (ili fotorezistora)

int val = 0;    // promenliva za zapazvane stoinostite
                // idvashti ot fotorezistora

void setup() {
        Serial.begin(9600); // otvaryane seriiniya port za izprashtane na
                            // danni obratno kam komputera
                            // sas skorost 9600 bita v sekunda
}

void loop() {
        val = analogRead(sensorPin);      // prochitane stoinostta ot
                                          // datchika
        Serial.println(val);              // otpechatvane na stoinostta
                                          // v seriiniya port
        delay(100);
}
```

Докато програмата върви натиснете бутона „Serial Monitor” в средата за програмиране на Ардуино и стойностите ще започнат да се подреждат в долната част на прозореца. Всяка програма, която може да чете данни от сериен порт може и да си „говори” с Ардуино.

/ аналогови изходи и ШИМ

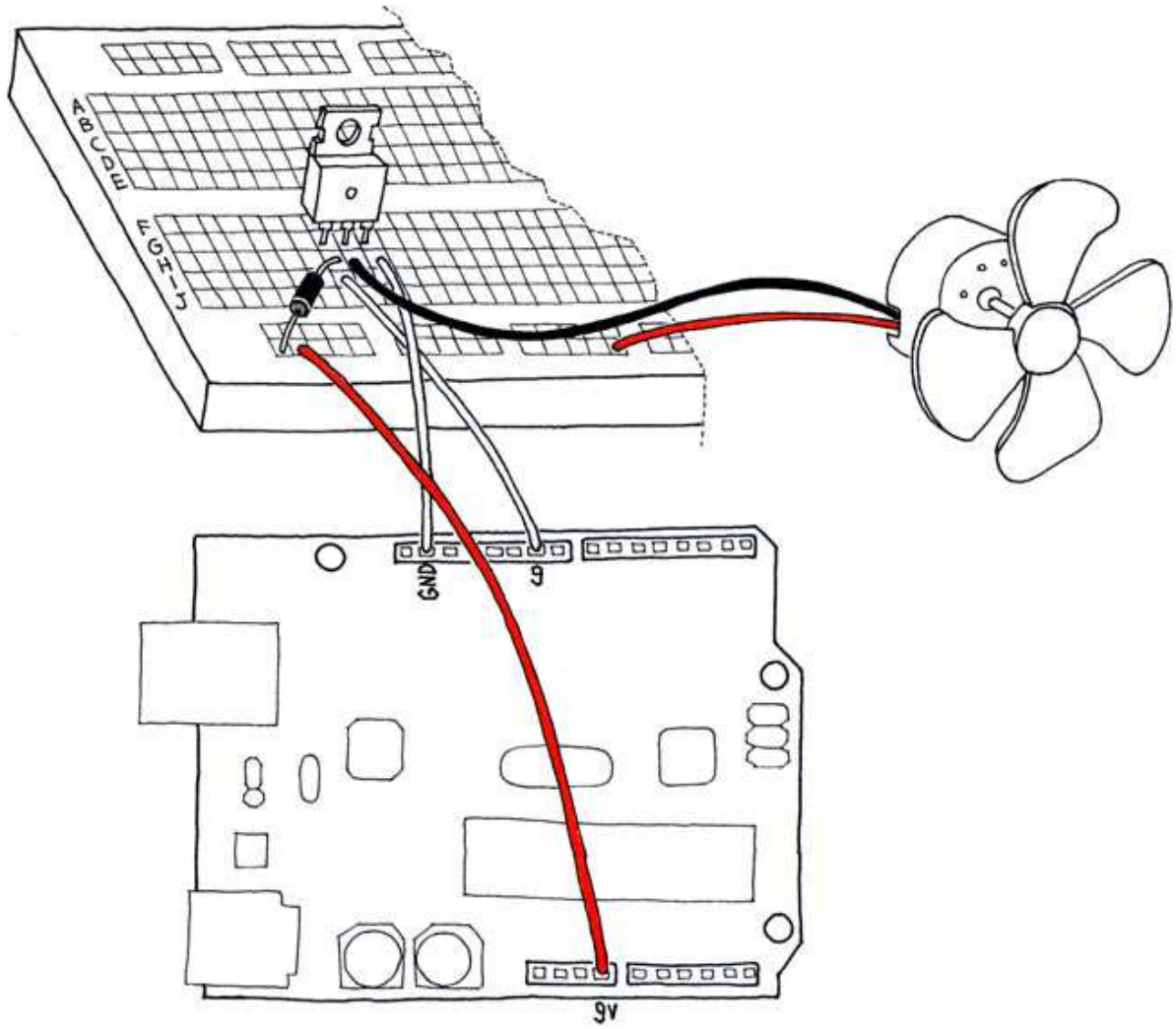
Със знанията натрупани дотук ще можете сами да си направите интерактивна лампа – такава която не се контролира само с бутона за включване и изключване, а по един по-поетичен начин. Един от недостатъците на досегашните примери с мигащия диод е, че само включвахме и изключвахме светлината, докато истинската тарикатска лампа трябва да може да изгасва постепенно. За да решим този проблем ще използваме малък трик, който прави възможни явления като киното и телевизията. Той се нарича широчинно импулсна модулация, или ШИМ.



Вземете първия пример с мигащ диод и променете числата в delay () командата докато мигането не стане прекалено бързо за окото ви и диода вече не се вижда като мигащ, а като наполовина по-блед. Сега нагласете числата така че съотношението включен – изключен да е 1 към 4. Сега яркостта му трябва да е около 25%.
Тази техника се нарича широчинно импулсна модулация – завъртян начин да се каже, че ако диодът мига достатъчно бързо не можете да видите мигането, и че можете да променяте яркостта му като променяте съотношението между времето когато е включен и изключен. Диаграмата от предишната страница показва как става това.
Тази техника може да се прилага и на други устройства. Например може с нея да контролирате скоростта на електромоторче.
Ако експериментирате ще видите, че контролирането на диода ръчно е неудобно защото всеки път когато искате да отчетете показанията на някой датчик или да изпратите данни по серийния порт диодът ще примигва. За наш късмет платката Ардуино има хардуерен компонент, който може много ефективно да контролира примигването на три различни диода докато програмата прави нещо друго. Това става през пинове 9,10 и 11, контролирани от analogWrite () командата.
Например, ако напишете analogWrite (9,128) ще зададете яркост от 50% на диода свързан към пин 9. Защото analogWrite () задава като параметър число от 0 до 255, където 255 означава 100%. Наличието на три такива канала е перфектно защото ако свържем към тях червен, зелен и син диод можем да използваме техните свойства и да си направим кой да е цвят.

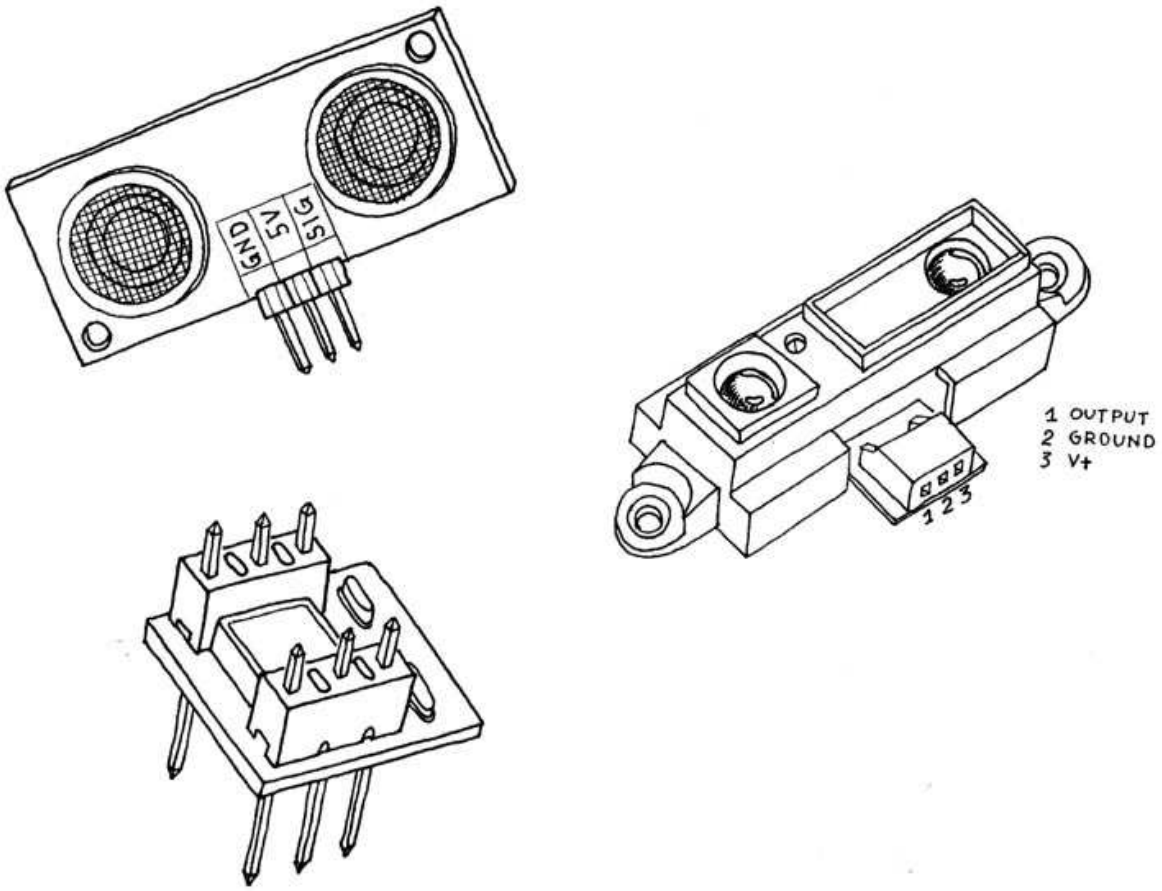
/ повдигане на по-тежки товари (електромоторчета, лампи и др.)

Всеки от пинове на Ардуино може да захранва уреди, които използват до 20 милиампера, което е доста слабо напрежение и достатъчно колкото да захрани диод. Ако се опитате да подкарате електромоторче, пинът моментално ще спре да работи и може дори да изгори процесора. За да задвижите по-тежки товари, като електромоторчета, е необходим външен компонент, който да може да включва и изключва такива устройства и да се контролира от Ардуино. Един такъв компонент е Мосфет транзисторът. Зад това странно име стои електронен превключвател, който може да се контролира като се подава ток към едно от трите му крачета, наричано „порта". Това работи по същия начин като ключа за лампата на стената ви, като функцията на пръста, с който включваме и изключваме лампата е иззета от пин на платката Ардуино, който изпраща токови сигнали до „портата" на Мосфета.
На картинката по-долу е изобразено как може да се използва Мосфет, като моделът IRF520, за да задвижи електромоторче с перка.
На картинката се вижда и, че електромоторчето се захранва от 9V конектора на платката Ардуино и това е друга от ползите на Мосфета – позволява ни да използваме захранване различно от това, което ползва самата платка. И тъй като Мосфета е свързан към пин 9, можем да контролираме скоростта на електромоторчето с analogWrite () командата.



/ сложни сензори

Като сложни определяме тези сензори, чийто данни не могат да бъдат прочетени само с digitalRead () и analogRead () команди. Обикновено тези сензори са малки вериги, съдържащи малък микроконтролер, който обработва данните преди да ги прати към Ардуино.
На картинката са изобразени ултразвуков датчик, инфрачервен датчик и акселерометър. Примери как да използвате такива датчици има на сайта на Ардуино http://www.arduino.cc .



/ общуване със софтуер

В тази глава ще ви покажем как можете да ползвате данните изпратени от Ардуино по USB кабела до външна програма, вървяща на вашия компютър.

Ардуино код за виртуален Etch-A-Sketch.

За този проект са необходими два променливи резистора, които ще контролират съответно X и Y осите. Уловката тук е да изпращаме данните предхождани от буквите A и B за отделните сензори. Този метод може да се използва и при изпращане на данни от повече от два сензора.

```
/* Virtualen Etch-A-Sketch s dva potenciometara
Tazi programa chete danni ot dva analogovi senzora prez seriiniya port i chertae
tehnite stojnosti po X i Y osite.

Arduino Code
Christian Nold, 22 Feb 06
*/
int potPin = 4;       // izbor na vhoden pin za potenciometara
int potPin2 = 5;      // izbor na vhoden pin za potenciometara
int ledPin = 13;      // izbor na pina za dioda
int val = 0;          // promenliva za zapazvane na dannite idvashti ot senzora
int val2 = 0;         // promenliva za zapazvane na dannite idvashti ot senzora

void setup() {
        beginSerial(9600);
        pinMode(ledPin, OUTPUT); // deklarirane na ledPin kato izhod
}

void loop() {

        val = analogRead(potPin);         // prochitane na stojnostite ot senzora
        val2 = analogRead(potPin2);       // prochitane na stojnostite ot senzora

        printString("A");
        printInteger(val);    // primeren identifikator za senzora
        serialWrite(10);
        printString("B");
        printInteger(val2);  // primeren identifikator za senzora
        serialWrite(10);
}
```

Виртуален Etch-A-Sketch код за Processing

Този код търси буквата A следвана от данни и ги използва за да чертае по X оста. След това чака за буквата B и чертае по Y оста. Светлосенките се постигат чрез неколкократно копиране на полупрозрачни (алфа) черни квадратчета.

```
 /* Virtualen Etch-A-Sketch s dva potenciometara
Tazi programa chete danni ot dva analogovi senzora prez seriiniya port i chertae
tehnite stojnosti po X i Y osite.

Processing Code
Christian Nold, 22 Feb 06
*/

import processing.serial.*;

String buff = “”;
String temp = “”;

float temporary = 0.0;
float screenwidth = 0.0;

float xCoordinate = 0;
float yCoordinate = 0;

int val = 0;
int NEWLINE = 10;

Serial port;

void setup()
{
        size(200, 200);
        strokeWeight(10); // shiroka
        stroke(255);
        smooth();

port = new Serial(this, “COM2”, 9600); // promenete COM2 s vernia port
}

void draw()
{
```

```
        fill(0,2); // izpolzvane na cherno s alfa 2
        rectMode(CORNER);
        rect(0,0,width,height);
        while (port.available() > 0) {
        serialEvent(port.read());
        }
point(xCoordinate, yCoordinate);
}

void serialEvent(int serial)
{

        if(serial != NEWLINE) {
                buff += char(serial);
        } else {

if (buff.length() > 1) {

        temp = buff.substring(0, buff.length()-(buff.length()-1));   // tova izolira
        samo purvonachalno identificiraniya senzor

if(temp.equals("A") == true) {    //stojnost ot senzor A
        temp = buff.substring(1, buff.length());
        temporary = float(temp);

xCoordinate = width/(1024/temporary);
println(xCoordinate);
}

if(temp.equals("B") == true) {    // stojnost ot senzor B
        temp = buff.substring(1, buff.length());
        temporary = float(temp);

yCoordinate = height/(1024/temporary);
println(yCoordinate);
}

// sazdavane na stojnost za "buff"
buff = "";
}
}
}
```