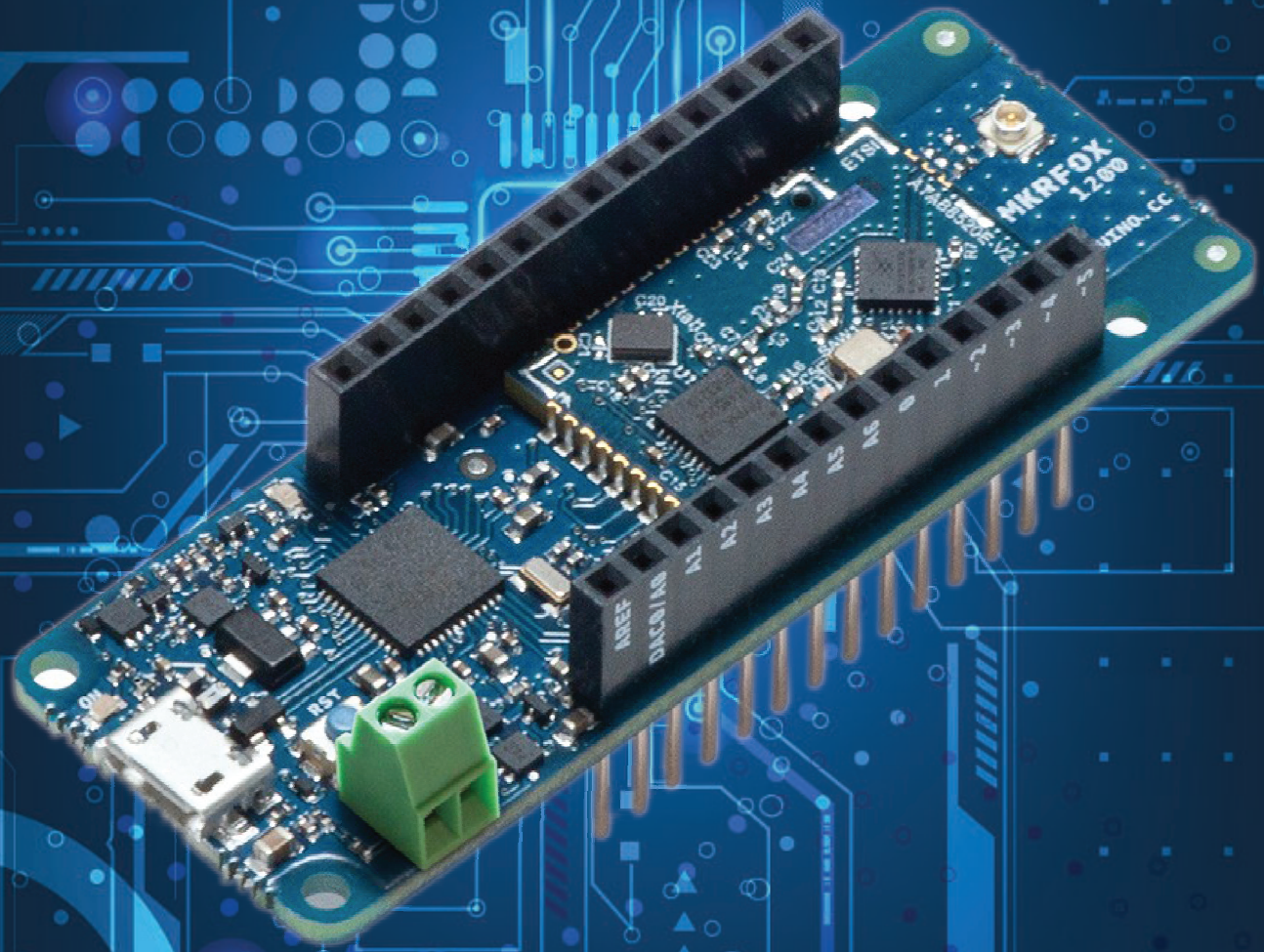
Светослав Енков

Програмиране в среда ARDUINO

ПРАКТИЧЕСКО РЪКОВОДСТВО



УНИВЕРСИТЕТСКО ИЗДАТЕЛСТВО
„ПАИСИЙ ХИЛЕНДАРСКИ“

Светослав Енков

Програмиране в среда Arduino

Практическо ръководство

УНИВЕРСИТЕТСКО ИЗДАТЕЛСТВО
„ПАИСИЙ ХИЛЕНДАРСКИ“

Рецензент: проф. дмн. Георги Атанасов Тотков

ISBN 978-619-202-261-7

За автора

Светослав Енков е доцент във ФМИ на ПУ „Паисий Хилендарски“. Разработил е избираемите дисциплини „Програмиране в среда Arduino“, „Програмиране в среда Arduino за напреднали“ и „Програмиране в среда Arduino с 32-битови контролери“.

Електротехниката и електрониката са негово хоби от малък. Завършил е курс по „Микропроцесорна техника“ през 1982 г. към Окръжния радиоклуб в Пловдив. Участник и многократен победител в студентски олимпиади по време на следването си във ФМИ, завършил е специалност Информатика с отличен успех. Има докторска степен по Информатика с тезис „Методика и средства за осигуряване на интернет достъпност за лица със специални образователни потребности“. Има научни и приложни интереси в областта на програмирането, е-обучението, електрониката, микроконтролерите, достъпността и мн.др. Работи като преподавател във ФМИ на ПУ от 1991 г.

Можете да се свържете с него чрез e-mail ***enkov@uni-plovdiv.bg***, както и във Facebook – Svetoslav Enkov.

Увод

В близкото минало, вградените системи и микроконтролерите се разработваха предимно от инженери или хоби-маниаци. Те бяха скъпи, трудно достъпни устройства и изготвянето на функциониращ прототип изискваше значителен опит и познания в областта на софтуерното инженерство и електрониката. С появата на Arduino – платформата за електронно прототипиране, базирана на отворен код, нещата са вече значително по-лесни и достъпни. Хардуерът е относително евтин, софтуерът е безплатен и самата платформа Arduino е проектирана за използване повече от артисти, дизайнери и любители, отколкото от професионалисти и инженери. [Arduino Bots'11]

Основната цел на това ръководство е да демонстрира как се разработват прототипи с Arduino. Изложена е достатъчно теория, за да се помогне да приложите новите познания за вашите нови проекти. Ще се запознаете с логиката, стояща зад кода и компонентите. Ще бъдат показани завършени проекти, с подробно обяснение на кода и компонентите, използвани в тях. Познанията, които ще получите, да послужат в бъдеще да реализирате собствени оригинални проекти и идеи. Книгата е предназначена за студенти и любители, желаещи да се научат да разработват интересни устройства, не за професионалисти, разработващи професионални устройства, но въпреки това, техниките, показани тук, може да се използват за реализиране на прототипи на комерсиални устройства. Акцентирано е на проекти с Arduino Uno, Mega и Due, като най-достъпни платки от семейството Arduino.

Arduino е сплав от три критични компонента: хардуер, софтуер и общност. За да извлечете максимума му, трябва да имате основни понятия от всичките три компонента, за повечето хора най-предизвикателното ще е хардуера. Преди да се опитате да се гмурнете в някой от проектите, моля, отделете време да прочетете цялата глава. Не само ще получите допълнителна информация, но и ще спестите много време, проблеми и нерви. [Arduino Practical'09]

Както беше споменато по-горе, един от ключовите аспекти за успеха на Arduino е общността, която се изгради около него, благодарение на отворената същност на хардуера и софтуера на Arduino. Софтуерът, използван в Arduino, е изцяло open source и хардуера (схемите, чертежите на платките и др.) са направени достъпни под Creative Common лицензи. На практика, това означава, че е лесно да се адаптират и софтуера и хардуера към Вашите нужди и после да се сподели обратно направеното към Arduino общността. Повечето автори на проекти подкрепят този подход и ви окуражават да направите и своите разработки достъпни за Arduino общността по подобен начин. За изходния код е достатъчно да се сложи ясна информация за лиценза в текста и по този начин ще е възможно за другите да пре-използват вашия код в техните собствени разработки с ваше разрешение. Затова, винаги, където е възможно, кода в това ръководство е сложен под GNU GPL лиценз. Също, за хардуерната част, е добре да слагате ясни указания дали и как може да се пре-използват схемите и чертежите от проектите.

Arduino официално се поддържа на два сайта – ***http://arduino.cc*** и ***http://arduino.org***, поради проблеми с разделянето на първоначалния екип на две групи (впоследствие противоречията са изгладени и двата сайта/екипа са вече равностойни, макар и не еднакви).

Вградените системи са навсякъде – това са устройства, базирани на микроконтролери, проектирани за специфична употреба. Някои примери са перални, мобилни телефони, асансьори, спирачни системи в автомобили, микровълнови печки, климатици, ръчни часовници и др. Вградените системи, за разлика от традиционните компютърни системи, обикновено не включват екран, мишка и клавиатура, вместо това, те се управляват с ключове, бутони и крачни педали, например. Повечето такива устройства са реактивни системи, действащи в непрекъснато взаимодействие с тяхното обкръжение и отговарящи с темпо, зададено от него. Това ги прави логичен избор за задачи, където трябва да се реагира мигновено, като спирачните системи на автомобилите. Повечето Arduino проекти са на практика вградени системи – работят непрекъснато, реагират веднага на околната среда и изпълняват точно и конкретно само функциите, заложени от разработчика. За разлика от комерсиалните вградени системи, които нямат възможност за лесна промяна на кода или схемата, тук имате възможност за бърза и ефективна подмяна, можете да превърнете откритията и идеите си в не-скъпи прототипи и да експериментирате лесно с различни подходи, компоненти и алгоритми.

Може да се създаде устройство за хранене на аквариумни рибки, да се управлява осветлението вкъщи или да се направи охранителна камера за гаража си, достъпна от всеки компютър в Интернет. Хората на изкуството могат да създават интерактивни инсталации или да интегрират сензори в неща, които се контролират без достъп до компютър. Възможните приложения са неограничено много и зависят само от Вашата фантазия.

Вградените системи съдържат сензори, микроконтролери и изходни устройства. Сензорите измерват условията вътре във физическото обкръжение, например разстояние, ускорение, движение, осветеност, налягане, отражение от повърхност и др. Микроконтролерът е мозъкът на вградената система. Той е малък компютър, с процесор и памет, което означава, че може да изпълнявате собствени програми на него. Микроконтролерът в Arduino се програмира с използване на истински компютър чрез USB кабел, а сензорите и изходните устройства са закачени към изводите му. Изходните устройства въздействат на физическото обкръжение, например, в тази книга ще срещнете светодиоди, дисплеи, релета и серво двигатели.

В това ръководство са изложени основите на вградените системи и са дадени съвети и примери как да конструирате и изпробвате своите първи „джаджи“. След това трябва да умеете да правите самостоятелно по-сложни проекти и прототипи, базирани на ваши собствени идеи. С помощта на съвременните средства за разработка и прототипиране, неща, преди изглеждащи невъзможно сложни, сега се оказват прости и разбираеми.

В продължение на повече от 6 години, по време на провеждането на избираемите дисциплини „Програмиране в среда Arduino“, голям брой студенти от ФМИ получиха основни познания за вградените системи, конструираха своите първи устройства, някои от тях избраха теми с Arduino за своите дипломни проекти (всички успешно защитени), доста се запалиха по това направление и го направиха свое хоби, а за някои това стана професия (работят по направления, свързани с контролери, управление на производствени процеси или разработка на хардуерни устройства).

От проведените курсове стана ясно, че студентите желаят да имат достъп до повече информация. Въпреки че вече всичко е достъпно в Интернет, то е разпиляно, не е подредено и не е на български език (това не е проблем за информатиците, но все пак е по-приятно да четеш на родния си език), затова беше поставено за цел да се обобщи събраната информация и на базата на постигнатите резултати, се издава тази книга.

Читателите ще усетят жажда за повече информация и ще започнат сами да проучват и разучават за теорията на електрониката, основите на управлението на процеси, програмирането на IoT устройства и много други неща, след като успешно построят и съживят своите първи „по-простички“ реални и работещи устройства, идеите за които са намерили в това ръководство. За тях са добавени в Приложение 3 връзки, указания и малко теория, за да се подпомогнат в по-нататъшното им пътешествие в пространството на вградените системи и микроконтролерите.

Какво е нужно да знаете, за да използвате това ръководство?

Да умеете да използвате компютър, да можете да инсталирате програми, да разрешавате проблеми по време на инсталацията на програмите и драйверите. Едно от предимствата на Arduino като платформа е неговата наличност за Windows, Linux и MacOS X. Вие ще можете да го използвате на практика с почти всеки компютър и операционна система. Умението да се програмира е от полза при изучаването на вградените системи, но на практика е достатъчно познаването на основните принципи от програмирането на C++, като функции, if-then условни оператори, цикли, сравнения и типове данни. За получаване на по-задълбочени познания в програмирането с Arduino, все пак трябва да се използват и учебници и книги по програмиране на C++, освен това ръководство.

Познанията от курса по физика, електротехника и електроника от средното училище ще са полезни и нужни за разбирането и усъвършенстването на устройствата от тази книга. Ако сте забравили вече тези неща, в началото на всеки проект ще има кратка теория на използваните устройства, обобщена информация ще намерите и в Приложение 3.

В началото на всеки примерен проект са изложени целите му и е даден списък с необходимите електронни детайли. Преди да сглобите устройството, може да разучите и изпробвате всеки компонент поотделно – свързването им заедно ще е по-лесно, когато разбирате техните основни функции и поведение. Не е необходимо да четете тези обяснения в началото, можете да изпробвате кода веднага, като го наберете в редактора или свалите от Интернет сайта. След като успешно сте го изпробвали, ще сте мотивирани да разберете как точно работи или да го промените за вашите собствени нужди и идеи. Когато по-нататък започнете да конструирате собствени устройства, тези обяснения ще са позволили по-лесно да разберете съответните секции от кода в своите нови проекти.

Не е необходимо да имате под ръка точно същите компоненти, като описаните тук. Може да имате достъп до различни компоненти или да откриете по-нови, по-добре изглеждащи или по-функционални компоненти, които би трябвало да работят и да се програмират по същия или подобен начин – това е вследствие на отворената същност на Arduino и на съвместимостта на компонентите, заложена от производителите им.

За Arduino

Arduino е термин, означаващ компания за компютърен софтуер и хардуер, проект и потребителско общество, които разработват и поддържат устройства с микроконтролери и комплекти за построяване на цифрови устройства с лесен за ползване свободен хардуер и софтуер, позволяващи постигането на интерактивност от неспециалисти, които могат да взаимодействат и управляват неща от реалния свят [Wikipedia – Arduino].

Продуктите Arduino се разпространяват като хардуер и софтуер с отворен код, лицензиран под LGPL/GPL лиценз, разрешаващ производството и разпространението от всеки. Платките и наборите се продават в сглобен или в „направи си сам“ вариант, достъпни са и като чертежи, схеми и изходен код за изцяло самостоятелно разработване.

Платките Arduino използват 8-битови (AVR) микроконтролери или 32-битови (ARM) процесори Atmel (най-новите модели Arduino включват и платки с други процесори, но в това ръководство ще се наблегне на платките Uno, Mega 2560 и Due, като най-разпространени и достъпни на пазара). Платките са снабдени с множество цифрови и аналогови входно-изходни изводи (изходи или pins), които могат да бъдат свързвани с разнообразни платки за разширение (shields) и други схеми. Те включват и сериен или USB интерфейс, с чиято помощ може да се зареждат програми от компютър. Микроконтролерите обикновено се програмират с помощта на езиците за програмиране C и C++, като специално за платките Arduino се препоръчва да се използва Arduino IDE средата, базирана на Processing.

Първата платка Arduino е представена през 2005 г. Екипът на проекта е целял да предостави на любители, ученици и професионалисти евтин и лесен начин да създават устройства, способни да взаимодействат с околната среда чрез сензори и изпълнителни устройства. Обичайни примери за използване са създаването на прости роботи, термостати и датчици за движение.

За това учебно помагало

С цел икономия на място, в това учебно помагало са поставени указания от кой уеб-базиран източник може да се открие допълнителна информация за по-подробно запознаване с дадена функция или възможност, с помощта на указанието:

„за повече информация, вижте <линк>“ .

За съжаление, често информацията ще е предимно на английски език. Ако в момента четете електронния вариант (PDF), това ще Ви улесни – трябва само да „кликнете“ на посочения линк.

Може да намерите материалите към това учебно помагало на адрес ***http://enkov.com/arduino***

Глава 1. Основни понятия за микроконтролерите и средите за разработка

Описание на микроконтролерите

Микроконтролерът е едночипова компютърна система, съчетаваща в себе си микропроцесор, тактов генератор, оперативна памет и програмируеми входно-изходни устройства. Често на същия чип има и различни видове компютърна памет. За разлика от микропроцесорите, които се използват в персоналните и други компютри, микроконтролерите са незаменими във вградените системи и са особено полезни, когато трябва да се реализира компютърно устройство, изпълняващо голям брой или сравнително сложни функции, например – комуникация с други устройства, управление на буквено-цифрови или графични дисплеи, измерване на различни величини, управление на технологични процеси и др.

Микроконтролерите се използват в продукти и устройства с автоматичен контрол, като контролни системи в автомобилите, дистанционни управления, офис машини, домашни уреди, електро-инструменти, играчки и други вградени системи. С намалените си размери и цена в сравнение с отделен микропроцесор, памет и периферни устройства, микроконтролерите са икономично решение за управление. [Wkipedia – Микроконтролер]

Микроконтролерът представлява система от затворен тип, която съдържа в себе си процесор и памет, и която може да бъде вградена в други устройства. По-голямата част от микроконтролерите, употребявани днес, са вградени в други машини като автомобили, телефони и периферни устройства за компютърни системи.

Докато една част от вградените системи са много сложни, много от тях имат минимални изисквания за памет и за продължителност на програмата, не изискват наличие на операционна система и не се нуждаят от програмиране на високо ниво. Типични устройства за вход и изход включват превключватели, релета, соленоиди, светодиоди, малки LCD дисплеи, радиочестотни устройства и сензори за данни, като например температура, влажност, светлинно ниво и т.н. Вградените системи обикновено не разполагат с клавиатура, екран, дискове, принтери или други разпознаваеми устройства за вход и изход, защото не е нужно да има директно взаимодействие с потребителя.

Обикновено програмите на микроконтролера трябва да се поберат в свободната памет на чипа, тъй като би било доста скъпо системата да бъде снабдена с външна памет. Използват се компилатори и асемблери за преобразуване на езици от високо ниво и програми на асемблер в компактен машинен код за съхранение в паметта на микроконтролера. В зависимост от устройството, паметта, предвидена за програмиране, може да бъде постоянна, памет само за четене, памет, програмирана още при самото производство или памет, която да се програмира на място при въвеждането в експлоатация.

Програмиране на микроконтролери

Микроконтролерите първоначално са се програмирали само на асемблер, но в наши дни за тази цел се използват езици за програмиране от по-високо ниво. Тези езици са или създадени специално за целта, или представляват версии на езици с общо предназначение, като например езика за програмиране C. Компилаторите на тези ези-

ци, обикновено имат заложени някои ограничения, а също така и подобрения, за да се постигне по-добра съвместимост с уникалните характеристики на микроконтролерите. Някои микроконтролери дори имат собствени среди за разработка на софтуер, чрез които се създават подходящи за тях приложения. Търговците на микроконтролери често разпространяват свободно такива инструменти, с цел по-лесното позициониране и налагане на пазара на техните продукти.

За някои видове микроконтролери има на разположение и симулатори. Те позволяват на разработчиците да анализират какво би било поведението на микроконтролера и на неговата програма, ако съответната част се използва в действителност. Симулаторът показва състоянието на вътрешния процесор, а също така позволява следене на изходите сигнали и генериране на входни такива. Въпреки че, повечето симулатори са ограничени от невъзможността си да симулират всичкия наличен хардуер в една система, те могат да симулират условия, които в повечето случаи са трудни за пресъздаване в реална среда, което ги прави най-бързият начин за изучаване и анализиране на даден проблем. За средата Arduino са налични няколко симулатора (вж. Приложение 3 – Fritzing).

Последните модели микроконтролери често са снабдени с вградена верига за дебъгване, която, достъпвана вътрешно чрез JTAG емулатор, позволява дебъгването на фърмуера с помощта на дебъгер.

Среди за разработка на софтуер за микроконтролери

Средата за програмиране се състои от инструментални средства, предназначени за създаване на програми. Създаването на дадена програма обикновено преминава през няколко стъпки, като за всяка от тях са необходими съответните инструментални средства. Към тези средства като минимум могат да се включат: езици за програмиране, текстови редактори, графични редактори, транслатори. програми за настройка и тестване на приложенията и др.

Едновременно с езиците за програмиране се развиват и транслаторите. Това са програми, превеждащи изходната програма в машинен код. Според начина, по който се осъществява това превеждане, транслаторите биват: компилатори и интерпретатори.

За микроконтролерите се използват предимно компилатори.

Компилаторът чрез непрекъснат процес прекодира целия текст на една програма от високо ниво, като създава нова програма на машинен език. Същата може да се съхрани и след това многократно да се изпълнява, без да се използва компилатора. Работата с език, реализиран чрез компилатор, обикновено минава през три етапа: текстът на програмата се създава с помощта на текстов редактор или програма за обработка на текст; вторият етап (след като програмата е тествана и са отстранени грешките) се състои в компилиране на програмата и третият етап е самото изпълнение на готовата програма.

Компилаторите избират за всяка от конструкциите на езика подходящ, предварително подготвен, фрагмент на машинен език и от всички фрагменти съставят (компилират) цялостната програма.

Текстови редактори

Текстовите редактори се използват за създаване и изменение на текстови файлове, представляващи програмния код. Текстът може да бъде последователност от опе-

ратори на някакъв език за програмиране, оформен като програмен модул. Някои текстови редактори поддържат функции, улесняващи програмирането на съответния език: цветово различаване на служебни думи, функции, класове, автоматична табулация, предложения за операнд.

Свързващи редактори

Следващата стъпка на разработване на потребителски програми е свързването. За изпълнение на тази стъпка се използват свързващи редактори. Програмите често се състоят от отделни обособени части. След компилиране на отделните части, те трябва да се свържат в едно цяло, при това включвайки доста сложни връзки между тях. В една обща изпълнима програма се обединяват няколко обектни модула и библиотеки. Тази дейност се извършва от специализирана програма на системата за програмиране, наречена свързващ редактор.

Дебъгери (анализатори на програмата)

Настройката е важна стъпка от процеса на разработването на потребителски програми. Могат да се използват различни средства за настройка. Дебъгерът е специална програма, която подпомага настройването и тестването на потребителските програми. Тя предоставя средства за трасиране (стъпково изпълнение на програмата и подробен запис на последователността от действия по време на изпълнение), за наблюдение как се променят стойностите на променливите и дава възможност за установяване на точки на прекъсване в настройваната програма. Тя може да се включва в или като част от компилатора или интерпретатора и помага на програмистите да откриват и коригират грешките.

При създаване на програмите могат да възникнат следните видове грешки:

- „Грешка“ при писане на програмата – отстраняват се с текстов редактор;
- Грешки по време на компилация (синтактични грешки) – откриват се от транслатора;
- Грешки по време на изпълнение (Run-time error) – откриват се с дебъгер;
- Грешки в алгоритъма (логически грешка) – най-трудни за откриване, откриват се по време на тестване на програмата и в най-лошия случай – при експлоатация.

Библиотеки

Към инструменталните средства най-често се включват библиотеките от програми, в които се съхраняват обектни модули за някои често използвани процедури, които автоматично се извличат оттам чрез свързващи редактори. Библиотеките съдържат подпрограми, които се използват най-често, например, подпрограми за вход/изход, тригонометрични функции, подпрограми за изчертаване на графични примитиви – линия, дъга, правоъгълник и др.

Интегрирани среди за разработване (IDE)

В повечето традиционни инструменти за разработване всяка от функциите по създаване на едно приложение (проектиране, редактиране, компилиране и настройване) работи като отделна програма, всяка със собствен интерфейс. Напоследък отделни-

те програми, включени в групата инструментални средства, се обединяват в т.нар. интегрирани среди (IDE).

Това понятие добива популярност при създаване на програмно осигуряване, работещо под Windows и MacOS. Интегрираната среда за програмиране най-често осигурява средства за подготовка, транслиране, тестване и коригиране на грешки, и изпълнение на програмата, както и за нейното документиране. Тя предоставя разширен многопрозоречен редактор, възможност за работа с мишка и обслужва пълния цикъл за създаване и поддържане на изпълняваната програма.

Интегрираната среда обикновено включва: език за програмиране; транслатор (компилатор или интерпретатор); свързващ редактор; изпълнителна система; система за проверка на програми; система за поддържане на библиотеки; текстови редактори и др.

Интегрираните среди се използват широко в практиката, тъй като при работа с тях компилирането, свързването и тестването се осъществява без да се напуска средата на редактора. При създаване на програми често се налага програмните файлове да се коригират и компилират многократно, като последователно се използват редактора и компилатора. Интегрираната среда повишава производителността на труда на разработчиците.

Много от съвременните интегрирани среди са ориентирани към създаване на приложения за операционни системи с графичен потребителски интерфейс (GUI), каквито са, например, операционните системи от фамилията Windows.

При такива системи вместо да се пишат много редове програмен код, за да се опише разположението на интерфейсните елементи, като бутони, текстови кутии, отметки и др., програмистът просто добавя предварително създадени обекти на някакво място върху екрана и задава началните им параметри, ако е нужно и после добавя кода за логиката – обикновено това е достатъчно за създаване на ефективен потребителски интерфейс.

Освен интегрирани среди, като допълнителни средства могат да се използват мениджъри на проекта, автоматически генератори на приложения, системи за автоматизирано тестване на софтуера и др.

Глава 2. Въведение в средата за разработка Arduino

Интегрираната среда за разработка на Arduino е мултиплатформено приложение, написано на програмния език Java и базирано на езика за програмиране „Processing“ и проекта „Wiring“. Конструирана е така, че да улесни програмирането от хора, които не са запознати с писането на софтуер. Съдържа редактор на код с функции syntax highlighting (показване на текст в различен цвят в зависимост от принадлежността на термините); brace matching (функция, която следи отварящите и затварящите скоби с цел по–лесна навигация през програмния код); автоматично подравняване; и също така е способно да компилира и да качва програми към устройството с един клик. [Arduino Wiki]

Arduino IDE е специална среда за програмиране, която върви на вашия компютър и ви позволява да пишете програми за Arduino на лесен език, базиран на езика Processing. Магията се получава, когато натиснете бутона, който upload-ва Вашата програма на платката. Вашият код се превежда на езика 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 машинен език. Тази последна стъпка е доста важна, защото с нея Arduino ни улеснява живота и елиминира възможно най-сложната част от програмирането на микроконтролера.

Може да се запознаете с Processing и Wiring от техните страници в Уикипедия:

https://en.wikipedia.org/wiki/Wiring_(development_platform)

https://en.wikipedia.org/wiki/Processing_(programming_language)

Всъщност, Вие може да използвате за програмиране на Arduino и средата Eclipse IDE, която е доста по-удобна и по-мощна, но е и по-трудна за начинаещите програмисти. За повече информация относно използването на Eclipse с Arduino, вижте

http://playground.arduino.cc/Code/Eclipse

Сваляне и инсталация на софтуера за Arduino IDE

За да програмирате на Arduino трябва да свалите средата за програмиране от ***http://www.arduino.cc/en/Main/Software***

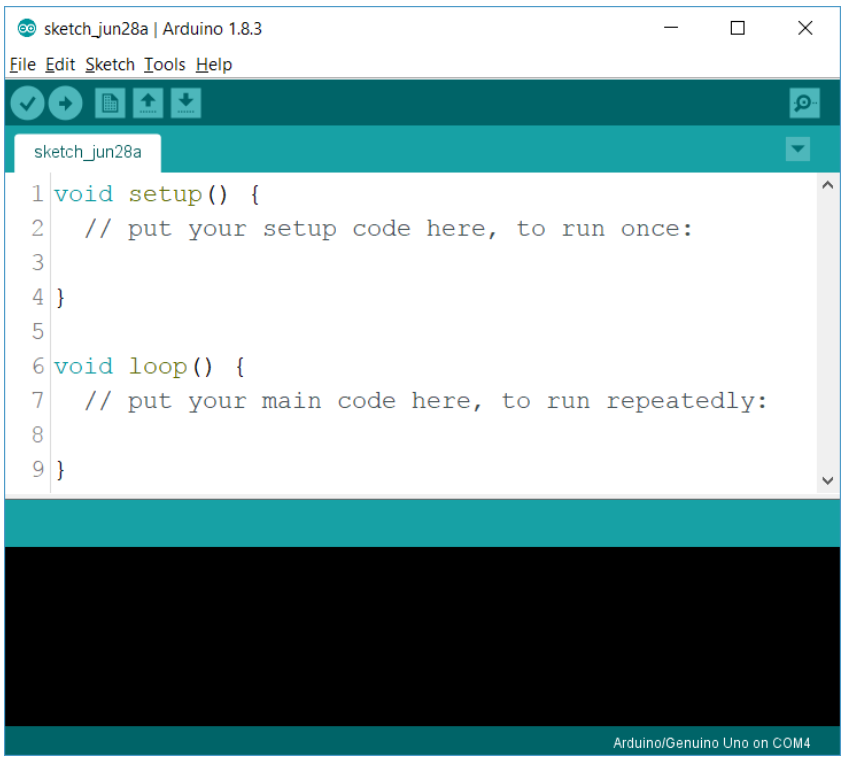
Изберете версията за вашата операционна система. Изтеглете файла и го разархивирайте. Първото, което трябва да направите е да инсталирате драйверите, които позволяват на компютъра да си „говори“ с платката Arduino през USB порта. Новите версии на средата Arduino правят това автоматично. Изключение се налага, ако ползвате нестандартна платка-клонинг, която използва друг чип за реализация на връзката с USB, тогава трябва да намерите драйверите от сайта на разработчика на тази платка. В някои случаи, операционната система ги намира и инсталира сама, когато включите платката за първи път – опитайте първо това.

Може да експериментирате и с онлайн версията на средата Arduino на адрес ***https://create.arduino.cc*** – базирана на браузър, също изисква плъгин за драйверите. Позволява разработка чрез браузъра, удобството е, че библиотеките и изходния код на проектите се пазят онлайн.

След като драйверите и средата са инсталирани, можете да стартирате средата за програмиране и да започнете да се забавлявате с Arduino.

Работа със средата за програмиране Arduino

Когато апликацията на средата Arduino се зареди, ще видите прозорец, като този по-долу:



Фиг. 1. Средата Arduino IDE

Ако в долния десен ъгъл не са показани правилно модела на вашата платка Arduino и порта, към който е свързана, трябва от менюто Tools-Board и Tools-Port да изберете точния модел и порт. Сега вече средата за програмиране на Arduino ще може да си „говори“ с платката и да я програмира.

Програма, написана за Arduino, се нарича „скеч“ (sketch) и е написана на програмния език C++, използвайки и синтаксиса на Processing. В това ръководство приемаме, че читателят е вече запознат с езика C++ и ще посочим само различията между C++ и Arduino.

Arduino програмите може да се разделят на 3 главни части: структура, стойности (променливи и константи) и функции. Структурата се състои от две функции, за разлика от C++, където обикновено е една функция (main): setup и loop:

setup() е функция, която е предназначена да се изпълни само веднъж и служи за инициализиране на променливи, избор на режим на използваните в скеча входове и изходи, начално инициализиране на библиотеките и др. Setup функцията се изпълнява еднократно, след всяко пускане или reset на Arduino платката.

loop() функцията се извиква след изпълнението на setup() функцията и се изпълнява циклично (безкрайно), позволявайки на програмата да реагира на промените. Използва се за (ре)активното управление на платката.

На практика, може да си представим, че на C++, освен функциите setup() и loop(), описани някъде в скеча, имаме и класическата функция main():

```
void main() {
  setup();
  while (1)
    loop();
}
```

Стандартният Arduino код не може да бъде видян от стандартен C++ компилатор като валидна програма, затова, когато потребителят кликне на бутона „Upload“ в средата за разработка, копие на кода се записва във временен файл, в който се добавя функцията *main()*, посочена по-горе, за да се валидира кода (нанасят се и други промени, разбира се, това се нарича *preprocessing*). Средата за разработка на Arduino използва GNU toolchain и AVR Libc, за да компилира програмите и използва Avrdude, за да качва програмите на устройството. Вследствие на това, че Arduino използва Atmel микроконтролери, средата за разработка на Atmel AVR Studio може също да се използва за писане на софтуер за Arduino, но тогава няма да може да се използва готов код от Arduino общността, тъй като той е написан на Arduino модификацията на C++. Всъщност, за Arduino платките с ARM и Intel процесори е по-различно, но в случая се разглеждат предимно стандартните платки с AtMega AVR процесори. [AVRStudio]

За да може средата Arduino да зареди програмата в микроконтролера, той трябва да има зареден т.нар. bootloader (зареждащ код). Всички платки Arduino са с предварително качен такъв (това е и причината да „липсват“ 0.5 до 2.5kB от флаш паметта на контролера – това е кода на зареждащия модул). Ако сте напреднали в Arduino, можете да си купите и „празни“ чипове (микроконтролер без записан зареждащ модул) и да си запишете модула с Arduino средата и ISP програматор на основата на AVRDude или с помощта на друга, работеща Arduino платка.

Детайли на варианта на езика C++, използван в Arduino

C++ (произнася се „си-плюс-плюс“) е неспециализиран език за програмиране от високо ниво. Той е обектно-ориентиран език със статични типове. От 1990-те, C++ е един от най-популярните комерсиални езици за програмиране. Bjarne Stroustrup разработва C++ през 1983 г. в Bell Laboratories като разширение на езика C – базиран на него, но с добавени редица допълнителни възможности и направени редица промени. Основната разлика между C и C++ е, че C++ съдържа вградена в езика поддръжка на обектно-ориентирано програмиране (ООП). В C++ са добавени класове, множествено наследяване, виртуални функции, overloading, шаблони (templates), обработка на изключения (exceptions) и вградени оператори за работа с динамична памет [Wikipedia – C++].

От познатите ни от C++ управляващи конструкции, ще посочим if, if…else, for, switch…case, while, do…while, break, continue и goto. Те нямат различия спрямо C++.

Ще споменем, че се препоръчва вместо #define да се ползва const. Това дава предимствата на проверките за тип и видимост на променливата.

Основните типове данни в Arduino са същите, каквито са в езика C++, на практика в Arduino са добавени някои нови дефиниции, константи и прототипи на функции, библиотеки и типове данни, специфични за AVR микроконтролерите, основно поради факта, че архитектурно те се различават от стандартните компютри. Нямаме файлова система, нямаме операционна система, затова почти всички стандартни функции на C++ за работа с файлове и с ОС са неприложими в Arduino.

Модификатор *volatile* – директивата указва на компилатора да зарежда променливата в RAM паметта, а не в регистър на процесора (което е памет за временно съхранение). Това е наложително, когато стойността на променливата може да се променя от нещо извън управлението на текущата секция код, например конкурентна нишка. В Arduino, единственото място, където това обичайно се случва, е в секция с код, асоцииран с прекъсвания (ISR, процедура за обработка на прекъсване).

Модификатор *PROGMEM* – указва променливата (често това са масиви с константи или низове) да се съхранява в програмната памет, а не в RAM. Това се налага, поради факта, че RAM паметта е много ограничен ресурс (2KB в Uno, 8KB в Mega). Наличен е само за AVR платките. Нужно е да имате *#include <avr/pgmspace.h>* в декларациите, както и самият достъп до данните от променливата става с помощта на функции, а не директно.

За повече информация относно PROGMEM, вижте

https://www.arduino.cc/en/Reference/PROGMEM

Константи, специфични за Arduino

HIGH, LOW – използват се за задаване на нивата на цифровите изводи, т.е. логическото ниво, съобразено със захранването на контролера (5V или 3.3V). За INPUT – при 5V – HIGH е при напрежение по-високо или равно на 3.0V, при 3.3V това е при 2.0V, а LOW е при напрежение по-ниско или равно на 1.5V за 5V платка и 1.0V при 3.3V платка. При OUTPUT – нивата са горната граница на захранващото напрежение за HIGH и 0V за LOW.

INPUT, OUTPUT, INPUT_PULLUP – вижте по-долу при описанието на функцията pinMode.

Предимството на повечето Arduino платки е наличието на вграден светлинен индикатор и специфичен резистор, свързани между някой извод и земя, което е доста удобно за прости тестове, без да се налага свързването на допълнителни светодиод и резистор. Изводът, на който е този светодиод, е зададен с константата *LED_BUILTIN*. За повечето платки това е цифров извод 13. Ще го използваме и обясним в **Проект 1**.

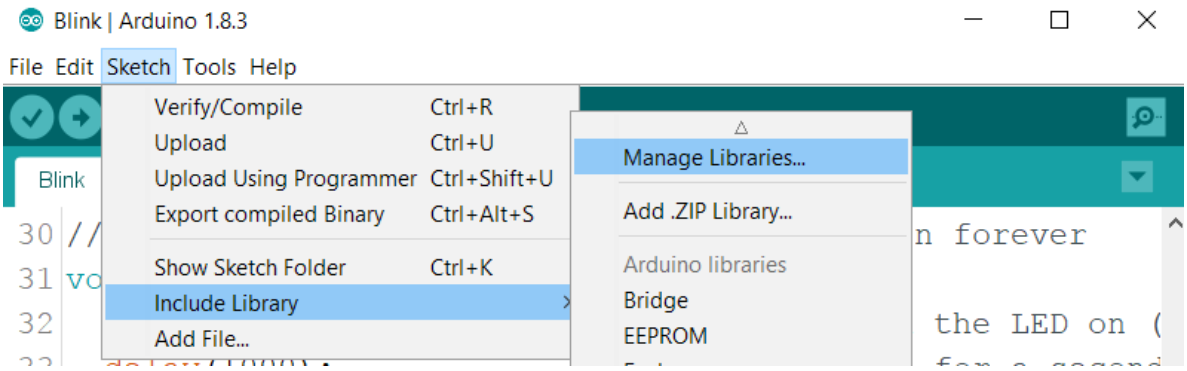
false и *true* са константи от C++, препоръчваме използването им, вместо 0 и 1 (всъщност true е еквивалент на всяка стойност „различна от 0“).

За повече информация относно константите в Arduino, вижте

https://www.arduino.cc/en/Reference/Constants

Библиотеки в Arduino

За да добавите библиотека в кода, от менюто Sketch изберете Include Library. Ако виждате нужната Ви библиотека в списъка, просто я изберете – тя ще се добави към проекта. От точката Add .ZIP Library можете да добавите библиотека, която е предоставена в zip файл, така Arduino IDE ще я сложи на правилното място и включи във Вашия проект.



Фиг. 2. Добавяне на библиотека с Library Manager

От точката Manage Libraries се извиква Library Manager на Arduino, с чиято помощ може да търсите библиотеки по ключови думи (в полето „filter your search“) и да ги свалите и инсталирате веднага автоматично (изберете Install now, след като намерите нужната библиотека).

Ако библиотеката е вече изтеглена под формата на папка, просто я преместете при другите библиотеки на Arduino, след разпакетирането ѝ:

- за Windows – My Documents\Arduino\libraries
- за Mac/Linux – Documents/Arduino/libraries

Ако търсите дадена библиотека и не я откриете с Library Manager по ключови думи, потърсете в GitHub repository и свалете библиотеката във вид на ZIP файл с помощта на „Clone or Download“ – „Download ZIP“ и после я разпакетирайте и преместете папката, както е показано по-горе в нужната директория.

Когато направите преизползваем код или напишете код за някой нов компонент, може да направите и собствена библиотека и да я публикувате в GitHub, като сложите подходящи таг-ове в library.properties файла и после тя ще се покаже в Library Manager на цялата Arduino общност. За повече информация, вижте:

https://github.com/arduino/Arduino/wiki/Library-Manager-FAQ

Видове платки Arduino

Arduino, освен система с отворен код, е и отличителна търговска марка и за да гарантира качеството и надеждността на продуктите, пълен екип професионалисти от компанията стриктно проверява новите платки, преди да бъдат пуснати в търговските мрежи. Официалните платки могат да се разпознаят винаги първо по името (започват с Arduino или Genuino) – Arduino Pro, Arduino Nano или Arduino Lilypad, например. Други, неофициални платки, често включват в името си „Arduino compatible“ или „for Arduino“. Друг начин за разпознаване на оригиналните платки е по брандирането – те са тюркоазени на цвят и някъде по тях имат изобразен символа за безкрайност, последван от официалния сайт на производителя: ***http://arduino.cc/***. [Wikipedia – Arduino]

Други компании също произвеждат платки, които се приемат като официални, например следните брандове: Adafruit Industries и SparkFun Electronics.

Поради това, че Arduino е с отворен код и схеми, има много неофициални варианти на платките. Обикновено те са базирани на същите микроконтролерни чипове и са съвместими със софтуера на Arduino, но изискват повече грижи, за да работят коректно. Нап-

ример Seeeduino (от Seeed studio) е базирана на Arduino Duemilanove и е 100% съвместима, но добавя допълнителни методи за свързване, допълнителни чипове и куплунги.

Основен „проблем“ на „официалните“ платки Arduino е тяхната цена, те са винаги значително по-скъпи от „неофициалните“ платки. Когато купувате „неофициални“ платки, проверявайте за потенциални проблеми със съвместимостта, най-вече откъм интерфейсния чип за връзка с USB и драйверите му, както и с какъв bootloader е презареден микроконтролера.

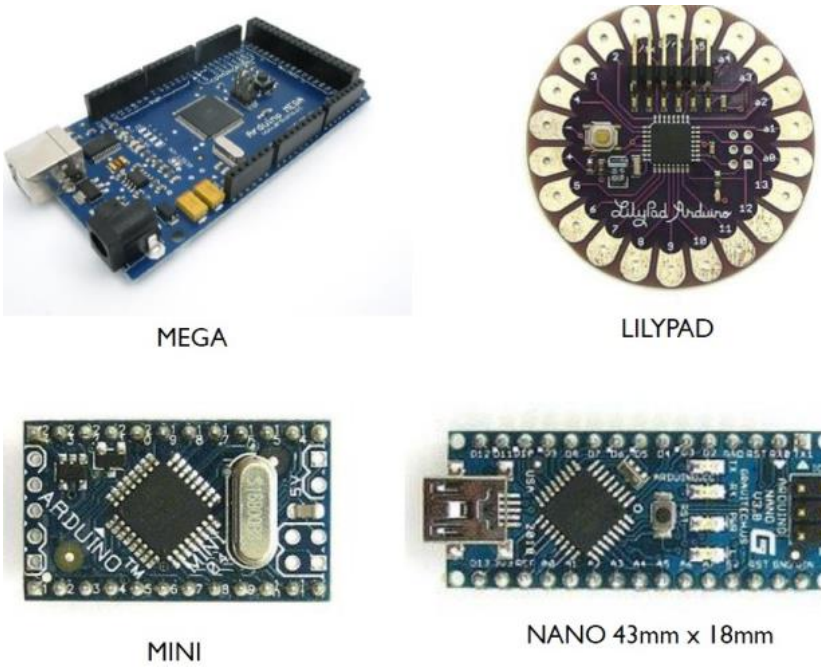
Някои официални Arduino платки са:

- Arduino Uno – микроконтролерна развойна платка с AVR микроконтролер ATmega328P. Свързването с компютър се осъществява чрез USB кабел USB A – USB B. Това е най-масовата платка и ще имаме нея впредвид в нашите проекти, ако не сме указали изрично друг вид платка.
- Arduino Leonardo – модификация на Arduino Uno, микроконтролерния чип е ATmega32u4. Той дава предимството платката да бъде разпозната като клавиатура или мишка от компютъра.
- Arduino Mega 2560 – както подсказва името ѝ, тази платка е по-голяма от Uno варианта. Създадена е за хора, които изискват повече – повече входове, повече изходи и повече процесорна мощ.
- Arduino Nano – Arduino Nano копира Uno, но е с размери само 1.8смх4.3см. Този размер е идеален за направата на по-малки проекти. Nano има цялата спецификация на Arduino Uno, използва същия ATMega328 микроконтролер, но е по-оптимизирана като размер и това я прави идеална за реални прототипи.

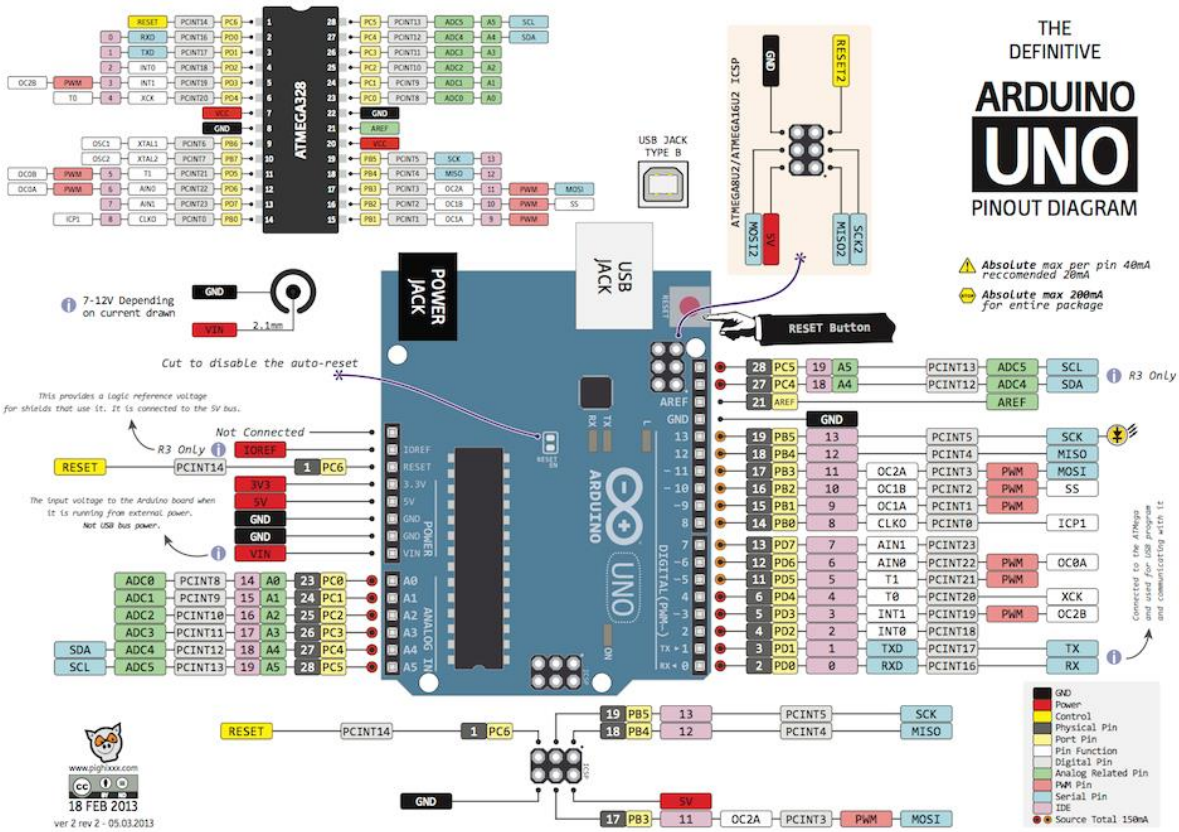
За повече информация относно точните параметри на всички модели Arduino платки, вижте

https://www.arduino.cc/en/Products/Compare и

https://en.wikipedia.org/wiki/List_of_Arduino_boards_and_compatible_systems.



Фиг. 3. Модификации на 8-битовите Arduino платки



Фиг. 4. Карта на изводите на Arduino Uno

Полезна информация дават т.нар. карти на изводите – те показват кой извод на платката за какво може да се използва. Поради липса на място, водещо до ограничен брой изводи, в микроконтролерите се практикува съвместяване на няколко извода в един, като текущата функция се задава от специални регистри на контролера, за удобство променяни чрез специални функции в библиотеките. В Arduino това става с функцията

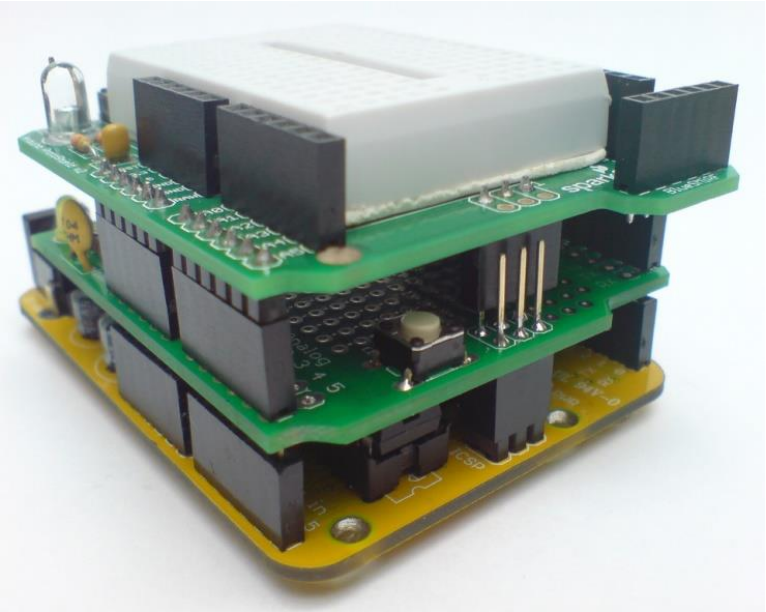
pinMode(pin, mode)

константата *mode* може да е INPUT, OUTPUT или INPUT_PULLUP. INPUT изключва задължително PullUp резисторите. Аналоговите входове A0…A5 (за Uno), могат да се използват и като цифрови изводи. От картата на изводите може да се съобрази кой извод на платката за какво може да се използва. За повече информация относно цифровите изводи, вижте

https://www.arduino.cc/en/Tutorial/DigitalPins

Платки за разширения (shields)

Предлага се голямо разнообразие от платки за разширение на възможностите, които се включват към нормалните куплунги (изводи) на платките Arduino. Те могат да осигурят контрол на мотори, връзка с GPS, Ethernet, LCD и др. Тези платки за разширения могат да бъдат изработени и по метода „направи си сам“. Разширителните платки могат да се стиковат и една над друга, ако имат и горен ред изводи. Всъщност, това е една от основните характеристики и предимство на Arduino платките, защото с помощта на тези платки за разширения се постигат бързи и успешни прототипи с минимално използване на кабели, междинни платки и поялник. Има и „празни“ разширителни платки за прототипи (Prototype shield, с места за запояване на елементи по тях), те също са много удобни, защото се стиковат лесно с Arduino платката. [Wikipedia – Arduino]



Фиг. 5. Платки за разширения (shields)

Глава 3. Проект 1 – Мигащи светодиоди. PWM регулиране. Захранване.

За запознаване с даден език за програмиране или среда за разработка обикновено се използват програми от вида „Hello, world!“. В Arduino по подразбиране няма буквено-цифрово изходно устройство и затова е прието да се използва код, който управлява някакъв светодиод или друго вградено устройство. Всяка Arduino платка притежава светодиод и резистор, свързани към определен извод на микроконтролера, зададен с константата *LED_BUILTIN* (по традиция, това е цифров изход D13, но на някои платки, като Gemma и MKR1000 е на D1 и D6, респективно). В повечето платки Arduino тази програма за мигане на вградения светодиод е фабрично програмирана, за да може да служи като „тест“ и проверка за работоспособността на платката веднага след покупката – нужно е само да се подаде захранване по някой от възможните начини, без необходимост от компютър или допълнителни компоненти. Поради този факт, този код се използва често и за проверка дали драйверите и средата са инсталирани правилно – зарежда се в контролера и се проверява работоспособността на връзката PC-USB, bootloader-а и настройките на компилатора (при грешен драйвер, неправилно избрана платка и проблем с кабелите няма да може да се зареди и стартира „мигането“). Кодът е наличен в средата Arduino в менюто File-Examples-01.Basics-Blink, но тук е даден съкратен вариант. Можете да го заредите от папката Project01/Project_01.1 – файла Project_01.1.ino.

```
// Проект 1.1 – Мигащ светодиод (вграден)
// Мигане на вградения светодиод с честота 1 с.
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Стартирайте средата Arduino IDE и наберете горния код (или го заредете от указаната папка). Натиснете Verify бутона, за да сте сигурни че няма грешки в кода. Трябва да получите статус „Done compiling“ и подобно на по-долното съобщение в секцията за съобщения на средата:

```
Sketch uses 928 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.
```

Ако това е успешно, можете да натиснете Upload бутона, за да заредите кода в Arduino платката. Би следвало скоро да наблюдавате как малкото вградено светодиодче на платката с надпис L в близост до извод 13 (на Arduino Uno) светва и изгасва на всяка секунда.

По-важните части от кода:

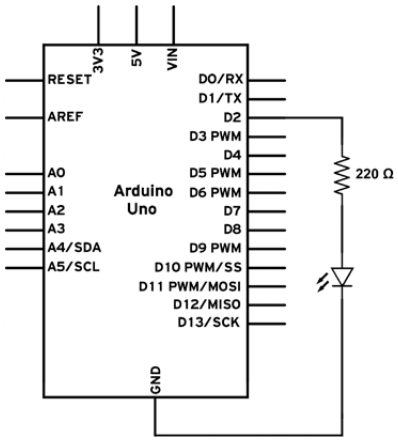
- във функцията setup() се указва с помощта на pinMode функцията, че извод с номер LED_BUILTIN трябва да е изходен;
- във функцията loop() е използвана функцията digitalWrite, която има два параметъра – номер на цифров изход и състояние на изхода, за да управляваме какво напрежение да има на този изход. В случая, HIGH предизвиква „светене“, LOW – „загасване“ на светодиода. Това е така, защото светодиода е свързан чрез резистор 1kΩ към „маса“ на платката (всъщност, вграденият светодиод е свързан към чип-компаратор LM358, свързан като повторител на сигнал, за да не пречи на нормалната работа на извода с номер LED_BUILTIN като вход, по-долу ще бъде показано как може да се преработи кода и свързването, за да работи с външен светодиод на друг цифров изход на Arduino платката);
- функцията delay(1000) предизвиква изчакване от 1000 милисекунди (1 секунда). Ако се изисква мигане на всеки две секунди, трябва да се сложи параметър 2000 на тази функция (и на двете извиквания);
- след това изчакване от 1 секунда, се изпълнява digitalWrite с параметър LOW, което предизвиква загасване на светодиода;
- отново имаме изчакване от 1 секунда – второто извикване на delay(1000);
- тъй като кода е част от функцията loop(), той се зацикля (повтаря) и така светодиода ще мига постоянно с честота 1с., докато платката получава захранване или не се зареди различен код в микроконтролера.

Този проект няма нужда от електронна (принципна) схема или Fritzing скеч, защото се използва вградения светодиод и не са необходими никакви външни елементи и свързването им.

Ако трябва да се управлява отделен (външен) светодиод, трябва непременно да се използва ограничителен резистор (ограничава максимално допустимия ток през светодиода, защитава и изхода на микроконтролера от превишаване на тока през него). Точната стойност на този резистор би трябвало да се пресметне по закона на Ом (вж. Приложение 3.b) по следните изисквания: при известен номинален ток през светодиода и известно изходно напрежение от изхода на контролера, спада на напрежение в ограничителния резистор да е равен на разликата м/у това изходно напрежение и работното напрежение на светодиода (forward voltage). При повечето светодиоди то е от 1.85V до 2.5V, може да се приеме средна стойност 2.2V (5-2.2 = 2.3V за „погасяване), може да се приеме и 20mA среден ток – дори това е допустимия максимално препоръчителен изходен ток за изход на Uno платките

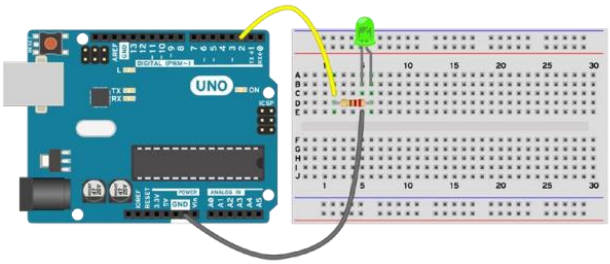
вж. ***https://playground.arduino.cc/Main/ArduinoPinCurrentLimitations***

От Закона на Ом следва R = U/I, тоест R = 2.3V/0.020A = 115Ω. На практика се избира съпротивление м/у 220 Ω и 1000Ω, защото има и ограничение за сумарния ток (често не е само един светодиод или управлявано устройство), както и често не е нужно светодиода да свети с максималната си яркост (скъсява неговия живот, както и дразни очите).



Фиг. 6. Управление на външен светодиод (схема)

Ако използвате монтажна платка (breadboard), се получава свързване, подобно на показаното:



Фиг. 7. Управление на външен светодиод (свързване)

Трябва да се промени и кода, да се замени LED_BUILTIN с номера на използвания цифров изход (в този случай – D2, изход 2) – може да се постави в началото const int LED_OUTPUT = 2; и да се замени LED_BUILTIN с LED_OUTPUT (*Project01/Project_01.2/Project_01.2.ino*).

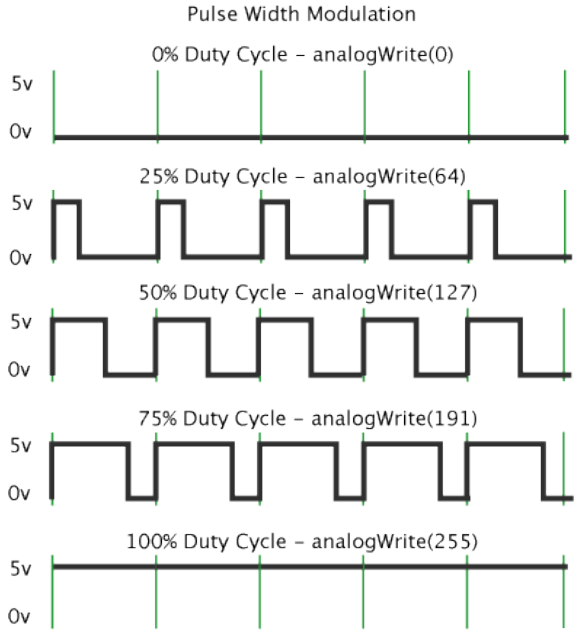
```
// Проект 1.2 – Мигащ светодиод (външен)
// Мигане на външен светодиод на изход 2 с честота 1 с.
const int LED_OUTPUT = 2;
void setup() {
    pinMode(LED_OUTPUT, OUTPUT);
}
void loop() {
    digitalWrite(LED_OUTPUT, HIGH);
    delay(1000);
    digitalWrite(LED_OUTPUT, LOW);
    delay(1000);
}
```

В Arduino Uno, 6 от 14-те цифрови извода с номера 3, 5, 6, 9, 10 и 11 могат да управляват PWM (pulse width modulation, широчинно импулсна модулация – ШИМ). Основното предназначение на PWM е за регулиране на мощността, подавана на изхода, без да има реално регулиране на изходното напрежение, казано иначе – да се получат

„аналогови“ резултати по цифров начин. Използва се цифрово управление на изхода, създавайки правоъгълни вълни – сигнал, превключван м/у „включено“ и „изключено“ състояние с шаблон, симулиращ напрежение между HIGH (5V) и LOW (0V), променяйки порцията от време, в което сигнала се намира във включено и изключено състояние. Продължителността на включено състояние се нарича „широчина на импулса“. Визуално тази широчина може да се определи в проценти (%), но в Arduino е избран подход, при който тя се задава като число 0...255, напр. 0% е стойност 0, 25% е 64, 50% е 127, 75% е 191 и 100% е 255. Използва се функцията

analogWrite(pin, value)

където pin е номер на цифров извод, поддържащ PWM, а value е стойност 0...255, задаващ широчината на импулса. [Arduino Beginning'10]



Фиг. 8. Графика на PWM и analogWrite()

В практиката с Arduino, PWM регулирането се използва често за:

- затъмняване на светодиод (регулиране на яркостта на светене на лампи);
- симулация на аналогов изход – ако се филтрува изходния сигнал, се получава „аналогово“ регулиране на напрежението между 0 и 100% (LOW и HIGH);
- генериране на аудио сигнали;
- управление на скоростта на мотори (ще се демонстрира в проект 3);
- генериране на модулиран сигнал, например за управление на инфрачервен светодиод за дистанционно управление и др.

За да се демонстрира регулирането на яркостта на светодиод, се налага модифициране на предходния проект за външен светодиод. Ще се наложи да се премести светодиода на изход, поддържащ PWM (избран е цифров извод 3) и да се добави код, използващ функцията analogWrite, вместо функцията digitalWrite.

Можете да използвате файла *Project01/Project_01.3/Project_01.3.ino*.

```
// Проект 1.3 – Преливане на светодиод (външен)
// Преливане на външен светодиод на изход 3 от 0 до 100%.
const int LED_OUTPUT = 3;
void setup() {
  pinMode(LED_OUTPUT, OUTPUT);
}
void loop() {
  for (int i=0; i<255; i++){
   analogWrite(LED_OUTPUT, i);
   delay(10);
  }
}
```

Ако стартирате този проект (не забравяйте да преместите светодиода на извод 3, ако все още използвате предходния проект), ще забележите как светодиода плавно преминава от изключено състояние (0% яркост) до пълна яркост (100%) за около 2.5 секунди (2550мс) и после пак загасва изведнъж до 0% и отново започва плавно повишение на яркостта до 100%.

Използване на външно захранване

При първоначалното разработване, Arduino проекта може да се захранва от компютъра по USB порта, но почти винаги при реализация на самостоятелен, работещ постоянно проект, се налага да се осигури постоянно, отделно захранващо напрежение с необходимата мощност.

По принцип, повечето платки Arduino имат куплунг за външно захранване, който приема от 6 до 20 волта (препоръчва се 7 до 19V, стандартните захранващи адаптери са предимно 7.5, 9 и 12V), минимум 250mA (ако няма допълнителна консумация от сензори, дисплеи и платки-разширения), 2.1mm кръгъл жак, с плюс в центъра. По-високото захранващо напрежение дава по-голям спад на напрежение в чипа на стабилизатора на напрежение и може да прегрее (пада КПД на стабилизатора на напрежение, защото е от LDO тип, не е импулсен). Има и възможност да се използва и батерийно захранване 9V или 12V, чрез куплунга за външно захранване, с батериен контейнер с кабел и накрайник 2.1mm (показан по-долу).

Може да се използва и USB захранващ адаптер 5V, 1A в USB порта. За повече информация, вижте на следния адрес:

http://playground.arduino.cc/Learning/WhatAdapter

Фиг. 9. Адаптер за външно (батерийно) захранване 9V

Това, отнесено към *Проект 01.3,* означава, че след зареждане на кода в контролера с Upload от Arduino IDE, може да се откачи платката от компютъра, да се захрани по някой от посочените начини за външно или батерийно захранване и да се радвате на първата си самостоятелна джаджа – „Мигащ светодиод“ (или „Преливащ светодиод“).

Управление с PWM на произволен цифров изход

Може да се реализира „ръчно“ PWM на всеки изход чрез последователното включване и изключване на изхода за определено време. Можете да използвате файла *Project01/Project_01.4/Project_01.4.ino*.

```
// Проект 1.4 – PWM управление на изход без analogWrite
// „Ръчно“ управление на PWM (ШИМ) 10% при 1kHz
void setup()
{  pinMode(2, OUTPUT); }
void loop() {
  digitalWrite(2, HIGH);
  delayMicroseconds(100); // приблизително 10% широчина на импулса при честота 1KHz
  digitalWrite(2, LOW);
  delayMicroseconds(1000 – 100);
}
```

Този начин на управление има предимството, че може да се използва с всеки от 14-те цифрови изхода, като в допълнение, имате пълен контрол над честотата и широчината на импулса. Главен недостатък на метода е, че прекъсванията и другия код от програмата пречат на точността на формиране на импулсите, както и трудно се пресмятане и напасване на константите за точните широчини и честоти („истинските“ PWM изходи се управляват от вътрешните таймери, които работят постоянно и независимо от кода, изпълняван в момента, докато не се зададе друга стойност с analogWrite).

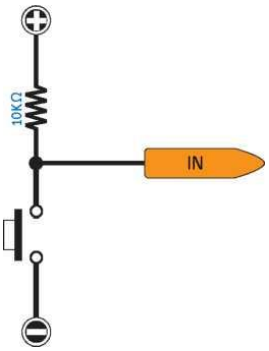
Един възможен начин на използване на този „ръчен“ подход за PWM е за моментно или периодично използване на някой цифров изход за управление на устройство, например управление на инфрачервен светодиод за дистанционно управление – тук се подават кратки, специално оформени импулси и не е необходима постоянна работа на изхода.

Може да ползвате библиотеката IRremote за IR дистанционно управление: ***https://github.com/z3t0/Arduino-IRremote***

Глава 4. Проект 2 – Бутони. Транзистори и релета за управление на мощен товар.

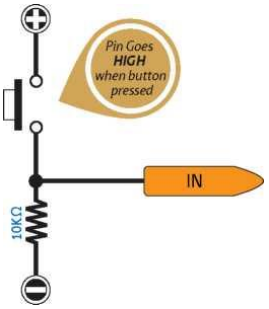
Едно от предимствата на Arduino проектите е, че може лесно да се реализира взаимодействие с потребителя. Тъй като това не е пълноценен компютър, най-удобно е реакцията на потребителя да става с помощта на бутони и превключватели (по-сложни бутони със задържане на състоянието или с превключване на вериги), така освен опростяването на кода и свързването, се опростяват и външния вид и реализацията на устройството. Най-простата реакция (взаимодействие с потребителя) е с помощта на обикновен бутон с две състояния – включено и изключено, свързан към цифров вход на Arduino платката. На входа ще има едното от двете състояния – HIGH или LOW. Много важен момент при работата с бутони и датчици, свързани към цифрови входове, е да се осигури подаване на постоянно напрежение (PULL_UP – към захранването или PULL_DOWN – към земята) , за да се избегне „плаването“ на входа – тъй като цифровите входове са високоомни и чувствителни, поради паразитния капацитет по кабелите (дори по самия извод на чипа като метал) се „натрупва“ напрежение и се превключва входа случайно и неконтролируемо, без да има реален сигнал. Затова, основно правило е неизползваните входове да се свързват към земята или захранващото напрежение (по-този начин, „свободният вход“ ще е винаги HIGH или LOW). Всички цифрови входове в Arduino имат възможност и за използване на вградения PULL_UP резистор (обикновено това е 20kΩ за Uno, 50kΩ за Due платките) при указване с pinMode(pin, INPUT_PULLUP).

Можете да използвате няколко възможни свързвания на бутон към Arduino – с или без вградения Pull-up резистор.



Фиг. 10. Бутон с външен Pull-up резистор

Когато се използва pull-up резистор, цифровия вход реагира по следния начин: не-натиснат бутон дава HIGH, натиснат бутон – LOW стойност.



Фиг. 11. Бутон с външен Pull-down резистор

При използване на pull-down резистор, цифровия вход реагира по обратен начин: не-натиснат бутон дава LOW, натиснат бутон – HIGH стойност.

За демонстрация на вградения Pull-up резистор и бутон, управляващ вградения светодиод, може да изпробвате следния код (или да използвате файла *Project02/Project_02.1/Project_02.1.ino*)

```
// Проект 2.1 – Бутон с INPUT_PULLUP
// Бутон, управляващ вградения светодиод
// светодиода ще загасне при натиснат бутон
const int buttonPin = 2; // свържете бутона към вход 2 и земя
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
}
void loop() {
  int buttonState = digitalRead(buttonPin);
  digitalWrite(LED_BUILTIN, buttonState);
}
```

Този код работи по следния начин: при натиснат бутон (свързан вход към земя) digitalRead функцията дава стойност LOW, при не-натиснат бутон (или несвързан вход) – HIGH. По този начин, натискането на бутона кара светодиода да загасне. Ако се изисква обратното действие – натиснатия бутон да кара светодиода да светне, трябва да се използва свързване към захранващото напрежение и външен pull-down резистор или да „обърнем логиката“ с промяна в кода, показана по-долу (може да използвате файла *Project02/Project_02.2/Project_02.2.ino*):

```
// Проект 2.2 – Бутон с INPUT_PULLUP инверсно
// Бутон, управляващ вградения светодиод
// светодиода ще СВЕТИ при натиснат бутон
const int buttonPin = 2; // свържете бутона към вход 2 и земя
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
}
void loop() {
  int buttonState = !digitalRead(buttonPin); // инвертираме с !
  digitalWrite(LED_BUILTIN, buttonState);
}
```

Операцията ! е побитово Не (Not) и прави „инвертиране“ на 0 в 1 и обратно (респ. HIGH в LOW и обратно).

При свързване на бутони, особено с по-дълъг кабел, е много важно да се избегне шума или трептенето на контактите на бутона. Най-лесно това може да се направи с помощта на повторно прочитане на състоянието на бутона в кратък период след първото,

за да се избегне случайно включване (вижте ***http://arduino.cc/en/Tutorial/Debounce***), може да използвате файла *Project02/Project_02.3/Project_02.3.ino*:

```
// Проект 2.3 – Бутон с INPUT_PULLUP инверсно с DeBounce
// Бутон, управляващ вградения светодиод
// светодиода ще СВЕТИ при натиснат бутон
// използва се софтуерен debounce
const int buttonPin = 2; // свържете бутона към вход 2 и земя
int buttonState;
int lastButtonState = LOW;
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50; // 50ms
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(buttonPin, INPUT);
}
void loop() {
  int reading = digitalRead(buttonPin);
  if (reading != lastButtonState) {
        lastDebounceTime = millis();
  }
  if ((unsigned long)(millis() – lastDebounceTime) > debounceDelay) {
        if (reading != buttonState) {
              buttonState = reading;
        }
  }
  lastButtonState = reading;
  digitalWrite(LED_BUILTIN, !buttonState); // инвертираме логиката
}
```

Трябва да се обърне внимание, че debounce не избягва шумовете от свободен вход (без pull-up), а само шумовете от трептене на контактите на бутона – понеже е механичен, в момента на натискане или отпускане, се генерират понякога „паразитни“ трептения, които са в границите на 30-50mS и точно те се филтрират с тази проверка. В Проект 2.3 е използвана нова функция – millis(). Тя връща като резултат изминалото след стартирането на платката време в милисекунди в стойност от тип unsigned long. Внимавайте с преобразуването и сравняването на този резултат с променливи от тип int – възможни са големи проблеми, поради преобразуването на типовете. Така също, стойността се нулира (превърта) приблизително на всеки 49 дни, затова, когато сравнявате разлики във времена, измерени с millis, се препоръчва използването на абсолютна стойност или други мерки – например, разликата да се прави с typecast през unsigned long (в горния пример, това е направено с typecast в (unsigned long)).

За повече информация за millis(), вижте

https://www.arduino.cc/en/Reference/Millis

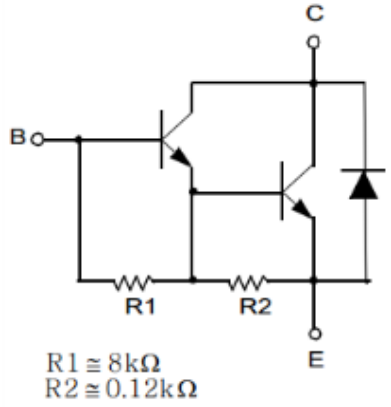
Транзистори и релета за управление на мощни товари

В проектите дотук, като единствено изходно устройство са използвани светодиоди (вграденият LED_BUILTIN или външни, свързани с ограничителен резистор). При управлението на реални устройства, често се налага микроконтролера да може да контролира ток, по-голям от 20mA или напрежения, по-високи от захранващото (5V или 3.3V, според модела на платката). За тази цел най-удобно е използването на „силов“ превключвател или „усилвател“ на ток или напрежение – мощен транзистор или реле (съчетанието *мощен транзистор* е употребено в значението на външен, отделен и по-мощен или високо-волтов транзистор като електронен компонент – самият микроконтролер също използва транзистори за изходите си, но с ток до 20-40mA и напрежение до 3.3/5V, като има ограничение и за сумарния ток от всички изводи).

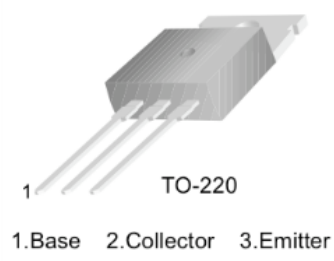
Транзисторите са усилватели на мощност, които могат да бъдат използвани и като цифров превключвател. Те са основно два типа – биполярни и полеви (MOSFET). Ще бъде показано използването на биполярни NPN транзистори за превключване на сигнали (ел. крушка) и за управление на постояннотокови двигатели (в следващия Проект 3).

Биполярните транзистори са най-широко разпространените дискретни полупроводникови елементи. Използват се за усилване, преобразуване и генериране на електрически сигнали. Най-често се използва схема с общ емитер (COE), тя дефазира входния сигнал на 180 градуса. [Wikipedia – Биполярен транзистор]

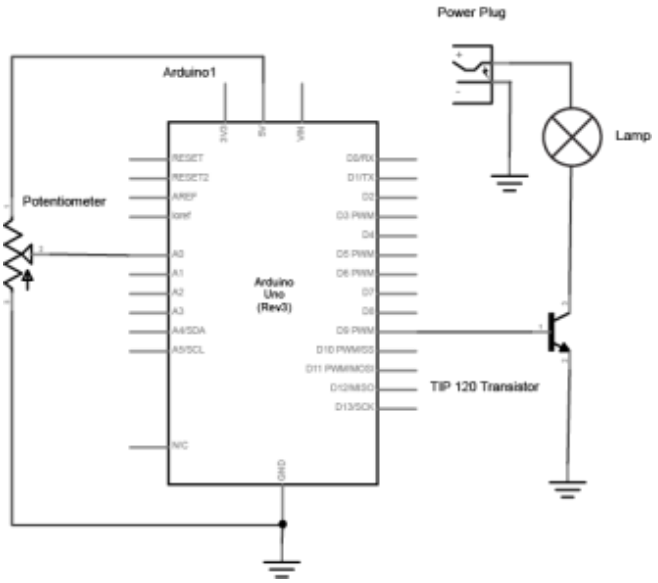
В проектите най-удобен е транзистор NPN модел TIP120, евтин, масов и с добри превключващи характеристики – 60V максимално напрежение колектор-емитер, 5A максимален постоянен ток (8A импулсна стойност) и висок коефициент на усилване (h_{FE} над 1000), позволяващ управление от изхода на микроконтролера, без да се претоварва. Друго предимство на TIP120 транзистора е вградения обратен диод, предпазващ от импулса при превключване на индуктивни товари (релета и двигатели), което спестява пари и улеснява свързването (опростява се схемата).



Фиг. 12. Транзистор TIP120 – вътрешна схема (означение)

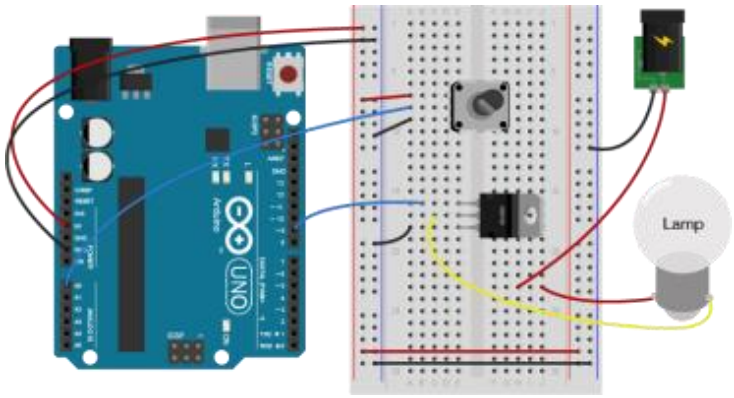


Фиг. 13. Транзистор TIP120 – корпус и изводи



Фиг. 14. Управление на ел. крушка с транзистор TIP120 (схема)

Посочената схема на свързване е опростена (но работеща) – в реалните проекти би било добре да има ограничаващ резистор (в случая с примерна стойност 330Ω) между изхода на микроконтролера и базата на транзистора с цел да предпази изхода при повреда в транзистора (окъсяване на базата към маса при прегряване или токов импулс), понякога се слагат и ограничаващи резистори с ниска стойност на съпротивлението за ограничаване на максималния ток през транзистора.



Фиг. 15. Управление на ел. крушка с транзистор TIP120 (свързване)

Кодът, отговарящ за управлението на ел. крушка с транзистор е същият, като кода за управление на мигащ светодиод, може да използвате файла *Project02/Project_02.4/Project_02.4.ino*:

```
// Проект 2.4 – Мигаща ел. крушка с транзистор
// Транзистор TIP120, управляващ външна ел. крушка с отделно захранване
const int transistorPin = 9; // свързан към базата на транзистора
void setup() {
  pinMode(transistorPin, OUTPUT);
```

```
}
void loop() {
  digitalWrite(transistorPin, HIGH);
  delay(1000);
  digitalWrite(transistorPin, LOW);
  delay(1000);
}
```

Понякога вместо транзистор, е удобно използването на релета. Те са устройства с 2 положения – включено и изключено. При тях електромагнитна бобина превключва механичен контакт и се създава импулс при изключването на тока през бобината, затова се налага да се комутират през маломощен или средномощен транзистор с вграден или добавен диод в обратна посока. Например, в горната схема, може да се замени електрическата крушка с бобината на реле, а контактите на релето да управляват мощен и високо-волтов товар – ел. печка, бойлер, пералня и др. мощни уреди. Релето осигурява и галваничното развързване на управлявания товар от микроконтролера. За повече информация, вижте

https://en.wikipedia.org/wiki/Relay

Глава 5. Проект 3 – Управление на постоянно-токови, стъпкови и серво мотори

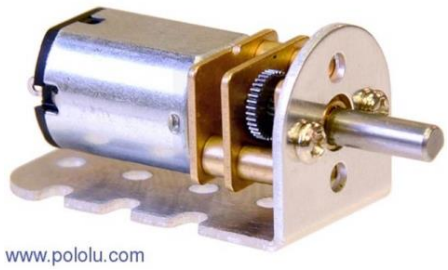
Електрическият двигател е електрическа машина, която преобразува електрическа енергия в кинетична. По вид на използваната електроенергия биват: постояннотокови двигатели и променливотокови двигатели, а по конструктивно изпълнение биват: колекторни двигатели и безколекторни двигатели. В Arduino проектите двигателите се използват за извършване на някакви механични действия – задвижване (роботи, дронове), преместване (в принтери или стругове), извършване на полезна работа (двигател на вентилатор или перална машина) и др. [Wikipedia – Електрически двигател]

Управление на постоянно-токови двигатели

Постояннотоковите електрически двигатели са двигатели, захранвани с постоянен ток. Това са първите електромотори, намерили практическо приложение. Тъй като четките се износват и се нуждаят от подмяна, откритите впоследствие асинхронни двигатели на променлив ток са заели местата на колекторните постояннотокови мотори в много сфери. Въпреки това, колекторните постояннотокови двигатели продължават и до днес да бъдат използвани без алтернатива на много места – автомобилни стартери, машини за хартия, роботи и др. [Pololu]

Фиг. 16. Постоянно-токов двигател – външен вид

Постояннотоковите двигатели имат най-голям въртящ момент от всички електродвигатели, при ниски скорости на въртене и еднакви масообемни показатели. За регулиране на въртящия момент и намаляване на оборотите на двигателя се използват редуктори. Редуктор е съвкупност от зъбни предавки, предназначена да намалява (редуцира – откъдето идва и наименованието редуктор) оборотите на въртене на двигател, повишавайки въртящия момент без значителни загуби на мощност. Почти всички проекти за Arduino роботи използват и редуктор, има двигатели с вграден редуктор. [Wikipedia – Редуктор, Pololu]

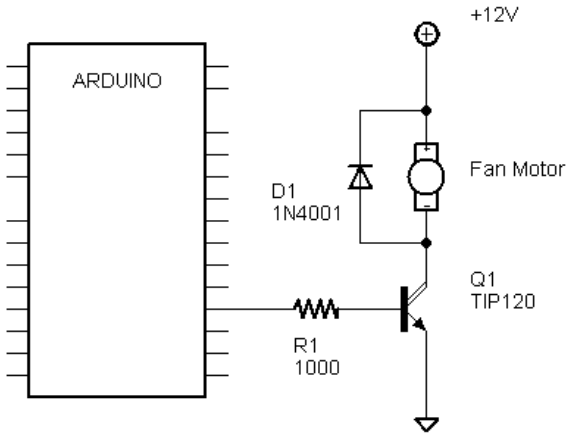


Фиг. 17. Постоянно-токов двигател с редуктор

За повече информация за управление на постоянно-токови двигатели с транзистори с помощта на Arduino, вижте

https://www.arduino.cc/en/Tutorial/TransistorMotorControl

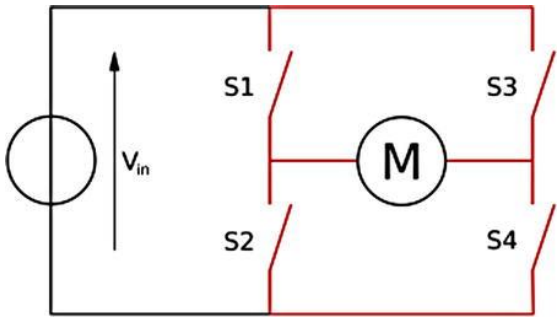
Постоянно-токов двигател може да се управлява по няколко начина. Ако се изисква опростено управление само в едната посока (например за двигател на вентилатор), може да се използва мощен транзистор (напр. TIP120) – схемата е подобна на използваната в предходния проект за управление на електрическа крушка.



Фиг. 18. Опростено управление на постояннотоков двигател с TIP120

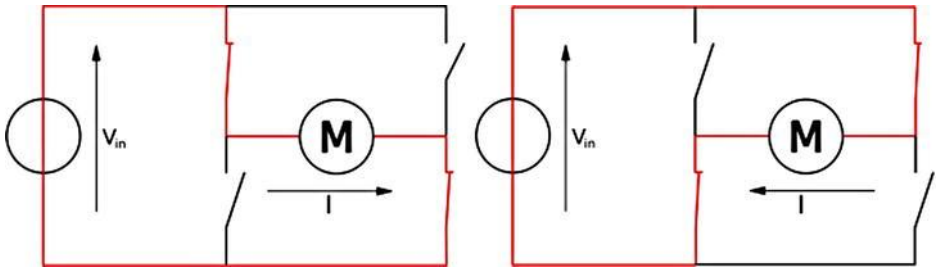
С помощта на горната схема, двигателят може да се пуска и спира (като се върти винаги само в една посока, в зависимост от свързването му), както и да се управляват оборотите му с PWM, ако е свързан към изход с PWM на Arduino платката. Поради приликата с проект 2.4, няма нужда да бъде показан изходния код.

Постоянно-токов двигател се управлява в двете посоки с помощта на т.нар. Н-мостов ключ (H-Bridge). Ще бъде показан пример с използване на ИС L293D, която е налична и монтирана на модули и платки за разширение (Arduino Motor Shield). L293D е сдвоен Н-мост, може да управлява 2 постояннотокови мотора. Идеализирана схема на Н-мост е дадена по-долу.

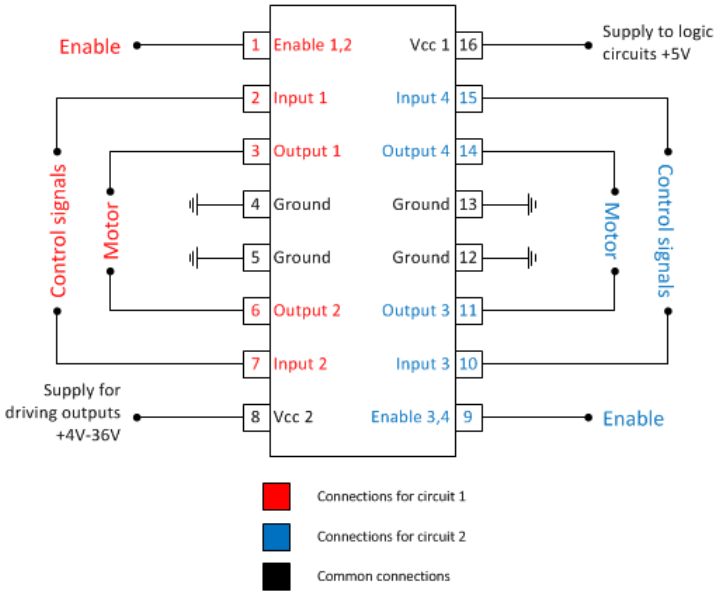


Фиг. 19. Схема на мостов ключ за упр. на постоянно-токов двигател

Когато ключовете S1 и S4 са включени – двигателят се върти в едната посока, а когато са включени S2 и S3 – в обратната посока. Фигурата онагледява принципа на действие на Н-моста. В ИС L293D ключовете са с мощни транзистори (и допълнителна логика) и позволяват използването на PWM за регулиране на оборотите (чрез входа Enable на съответната „половинка“ от ИС).



Фиг. 20. Действие на Н-мост за управление на двигател



Фиг. 21. Схема на изводите на ИС L293D (двоен Н-мост)

Примерен код за управление на постоянно-токов двигател с ИС L293D е даден по-долу (свързването на чипа към Arduino платката е описано в кода), може да използвате файла *Project03/Project_03.1/Project_03.1.ino*:

```
// Проект 3.1 – Управление на постояннотоков двигател с L293D чип
// Двигателят тръгва плавно за 10с. работи на макс. 30с. и спира плавно за 10с.
// после повтаря това в обратната посока и отново отначало.
// Свързване: D3 и D3 към Input 1 и Input 2, D9 към Enable 1,2
// двигателят към Output 1 и Output 2, отделно захранване за двигателя на Vcc2, обща земя
#define motorPin1 3 // L293D Input 1
#define motorPin2 4 // L293D Input 2
#define speedPin 9 // L293D enable 1,2
int Mspeed;
void setup() {
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(speedPin, OUTPUT);
```

```
}
void loop() {
   // настройване на моста в права посока
   digitalWrite(motorPin1, LOW);
   digitalWrite(motorPin2, HIGH);
  // плавно увеличение от 0 до 255
  for( Mspeed = 0; Mspeed <= 255; Mspeed++ ) {
     analogWrite(speedPin, Mspeed); // задаване на PWM обороти
     delay(40); // 10 секунди / 255 – прибл.
  }
   delay(30000); // 30 секунди на макс обороти (PWM е 255)
   // настройване на моста в обратна посока
   digitalWrite(motorPin1, HIGH);
   digitalWrite(motorPin2, LOW);
  // плавно намаляване от 255 до 0
  for( Mspeed = 255; Mspeed >= 0; Mspeed-- ) {
     analogWrite(speedPin, Mspeed); // задаване на PWM обороти
     delay(40); // 10 секунди / 255
  }
}
```

Управление на стъпкови двигатели

Стъпковите електродвигатели (Stepper Motor, Step Motor) намират приложение в голяма част от индустриалните и потребителските механични и електронни устройства, в системите за автоматизация и автомобилостроенето. Нещо повече, чрез използването на свойствата на тези двигатели се създават устройства с нови характеристики и принципи на действие. Управлението на стъпковите електродвигатели се извършва само чрез средствата на електрониката.

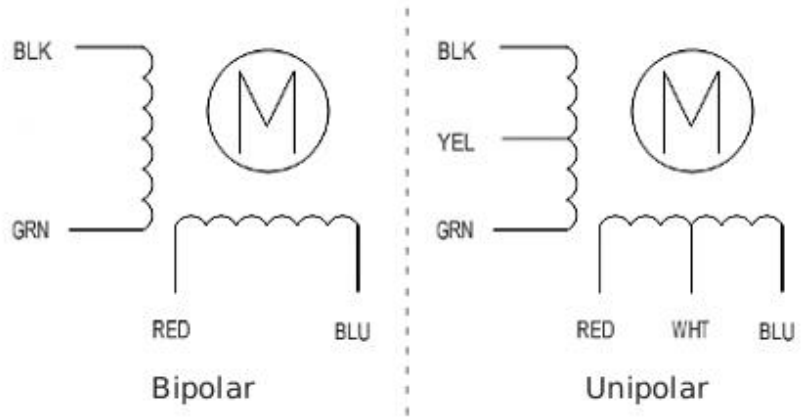
Две са основните различия на стъпковите електродвигатели от останалите постояннотокови електродвигатели, които дават отражение върху тяхното управление. Едното е сравнително малката им мощност, което определя по-прости, евтини, с малка постояннотокова консумация и размери схеми за управление. Второто различие е, че техният ротор не се върти плавно в класическия смисъл, а се завърта на определен ъгъл (прави една стъпка), като може да остане неподвижен за известно време и след това отново да се завърти в същата или обратна посока. Това движение на стъпки определя наименованието на електродвигателите. Както бе споменато, на всяка стъпка роторът може да остане неподвижен неограничено дълго време, което е удобно за позициониране на изпълнителния механизъм. Описаното действие се осигурява от импулси (а не с аналогово напрежение, както при другите електродвигатели) от управляващата електроника. [Wikipedia – Стъпкови мотори]

Фиг. 22. Стъпков двигател – външен вид

За повече информация за управление на стъпкови двигатели с помощта на Arduino, вижте

https://www.arduino.cc/en/Reference/Stepper

Стъпковите двигатели биват два вида – униполярен и биполярен. Униполярни (еднополярни електродвигатели) – те са с постоянен магнит или хибридни и наименованието им (Unipolar Stepper Motor) се дължи на протичането през намотките им на ток само в една посока. При биполярните стъпкови двигатели, посоката на тока през техните намотки се сменя в процеса на работата им (Bipolar Stepper Motor). Предимството е в по-простите намотки (нямат среден извод), което е за сметка на усложняване на управлението и на първо място в необходимостта от 2 пъти повече ключове.



Фиг. 23. Схема на униполярен и биполярен стъпков двигател

Ще бъде демонстрирано използването на вградената в Arduino библиотека Stepper. За управление на униполярен мотор се препоръчва ИС U2004 (UL2004N) масив от Дарлингтон транзистори или SN754410 четворен H-полумост (подобен на L293D, може да се използва и L293D) за биполярен мотор. Независимо, че схемата на свързване се различава, от гледна точка на използването на библиотеката, управлението е еднакво. Може да видите схемите на свързване от посочения по-горе адрес, те няма да бъдат показани тук, поради ограничения обем на това учебно помагало.

Проект 3.2 завърта стъпковия мотор със скорост 60 оборота в минута 1 оборот надясно, прави пауза от 1 секунда, връща назад 1 оборот наляво, пак прави пауза от 1 секунда и повтаря действието. Трябва да смените константата stepsPerRevolution с броя стъпки на оборот на използвания мотор. Може да използвате файла *Project03/Project_03.2/ Project_03.2.ino*:

```
// Проект 3.2 – Управление на стъпков мотор с библиотеката Stepper
// завърта 1 оборот и връща 1 оборот обратно с една секунда интервал
// Кода работи с униполярен и с биполярен мотор. Схема на свързване
// на https://www.arduino.cc/en/Tutorial/StepperOneRevolution
#include <Stepper.h>
const int stepsPerRevolution = 200; // това трябва да е спрямо използвания мотор – броя стъпки за цял оборот
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11); // свържете мотора на изходи 8,9.10,11 чрез ИС U2004/SN754410
void setup() {
  myStepper.setSpeed(60); // 60 оборота в минута скорост
}
void loop() {
  myStepper.step(stepsPerRevolution); // един оборот в посока на час.стрелка
  delay(1000);
  myStepper.step(-stepsPerRevolution);  // един оборот на обратната посока
  delay(1000);
}
```

Управление на серво мотори

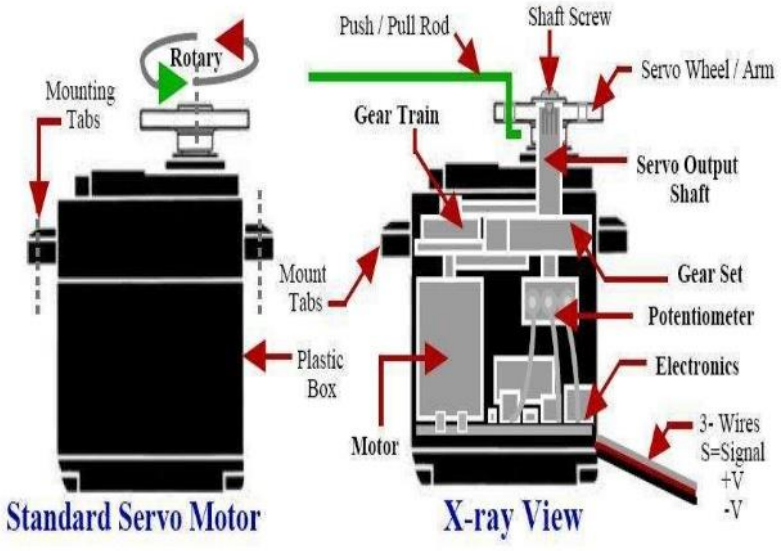
Сервомоторът е вид електродвигател, предназначен за привеждане в движение на устройства за управление. Обикновено сервомоторите са с малки габарити и мощност. Важни характеристики на сервомотора са също масата, динамиката на двигателя, равномерността на движение и ефективността. Сервомоторите се използват широко в промишлеността, например, в металургията, в автомобилостроенето, робототехниката, металообработващите машини, космическата и авиационна промишленост и т.н. За серводвигатели се използват както променливотокови, така и постояннотокови електродвигатели, но задължително с датчици за скорост и положение на ротора спрямо статора. Това е главната отличителна черта, спрямо другите двигатели. Освен контрола за положение, характерното за серводвигателите е минималното време за стартиране до достигане на зададени обороти, както и минималното време за спиране. Това означава мигновено достигане на електрически зададените от електронното управление обороти, както и прекратяване на действието. [Wikipedia – Серво-мотор]

Фиг. 24. Сервомотор – външен вид

За повече информация за директното управление на малки серво-двигатели с помощта на Arduino, вижте

https://www.arduino.cc/en/Reference/Servo

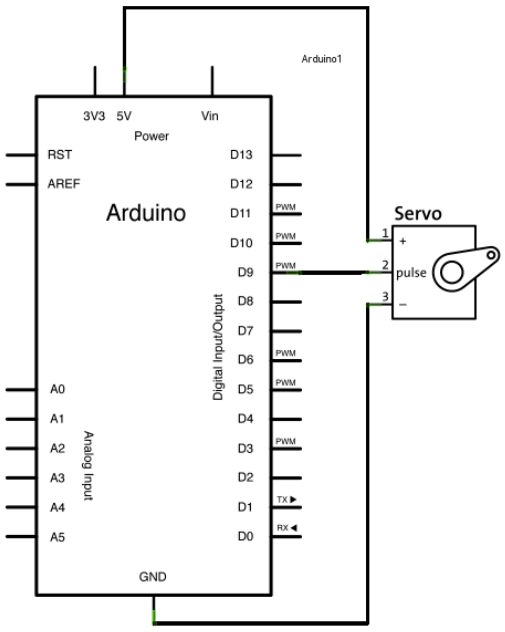
При управление на серво мотори с Arduino с помощта на вградената Servo библиотека, трябва да се отчита типа на сервомотора – дали е само до 180˚ (половин оборот) или 360˚ (непрекъснато въртене – този вид позволяват и управление на скоростта на въртене). Сервомоторите имат вградени предавки и ос, които могат да се управляват прецизно. Библиотеката Servo поддържа до 12 мотора на по-малките и 48 на Arduino Mega платките. На платките, различни от Mega, използването на тази библиотека забранява PWM функционалността на изходи 9 и 10, независимо дали на тези изходи има закачен сервомотор. На платките Arduino Mega, до 12 сервомотора не спират PWM, от 12 до 23 мотора спират PWM на изходи 11 и 12.



Фиг. 25. Вътрешна структура на сервомотор

Сервомоторите имат три извода: захранване, земя и сигнал. Захранването е обикновено червена жичка, земята е черна или кафява, а сигналната е жълта, оранжева или бяла и се свързва към цифров изход на Arduino платката. Обърнете внимание, че сервомоторите консумират значителен ток и ако се наложи да използвате повече от 2, трябва

да ги захраните от външен източник на захранване (не от +5V изхода на Arduino платката) – трябва да осигурите свързване на земята на външния токоизточник с тази на платката (това е основен принцип в електрониката, просто се припомня тук). На показаната по-долу схема, сигналният (управляващия) извод на сервомотора е означен *pulse*.



Фиг. 26 Схема на свързване на сервомотор към Arduino

Проект 3.3 завърта сервомотора плавно със стъпка от 1 ˚ – от 0 ˚ до 180˚ и после го връща обратно до 0˚ (като размахване на ветрило).

Може да използвате файла *Project03/Project_03.3/Project_03.3.ino*:

```
// Проект 3.3 – Управление на сервомотор на изход 9 със Servo библиотеката
// развърта мотора 0-180 и обратно
#include <Servo.h>
Servo myservo; // обект myservo за управление на сервомотора
int pos = 0;  // променлива за позицията на остта на сервомотора
void setup() {
 myservo.attach(9); // attaches the servo on pin 9 to the servo object
}
void loop() {
 for (pos = 0; pos <= 180; pos++) { // завъртане от 0 до 180 градуса
  myservo.write(pos);
  delay(15); // трябват около 15мс, за да достигне позицията (да реагира)
 }
 for (pos = 180; pos >= 0; pos--) { // завъртане от 0 до 180 градуса
  myservo.write(pos);
  delay(15);
 }
}
```

Глава 6. Проект 4 – Музика и звуци – пиезо-елементи. MIDI устройства.

Важен аспект от взаимодействието с потребителя при вградените системи, отчитайки липсата на дисплей, както е при компютрите, се оказва генерирането на звук – за предупреждение при достигане на някакъв резултат, за потвърждение на натискането на бутон, при грешка и др. (например повечето микровълнови фурни и перални с електронно управление имат звукова сигнализация при натискане на бутон и при приключване на текущата операция).

Вградени функции tone() и noTone()

В Arduino IDE са имплементирани (вградени) две функции за работа със звук – tone() и noTone():

- tone(pin, frequency)
- tone(pin, frequency, duration)
- noTone(pin)

Tone генерира правоъгълен сигнал със зададената честота (frequency е unsigned int) и 50% запълване на указания изход. Ако е използван варианта със зададена продължителност (duration е unsigned long в mS – милисекунди), след изтичане на времето, сигналът спира да се извежда, иначе се генерира непрекъснато, докато не се извика функцията noTone за същия изход. Изходът може да е свързан с активен пиезо-елемент или с високоговорител или пасивен пиезо-елемент (с транзистор за усилване и формиране на сигнала или поне през ограничителен резистор).

Може да бъде генериран само един звук с функцията tone() в даден момент, ако вече се изпълнява на друг изход, tone() няма да има ефект. Ако се изпълни tone() на изход, на който вече има звук, то ще се смени честотата му. Затова е важно използването на noTone(), преди да се подаде звуков сигнал на друг изход.

Не свързвайте директно Arduino изход към линеен или микрофонен вход на аудио техника – изходните нива (3.3V/5V) са много високи за този вид входове – използвайте непременно делител на напрежение и ограничителен резистор.

Използването на функцията tone() се влияе и оказва влияние на PWM изходите 3 и 11 на платките, различни от Mega (на практика, tone() използва таймерите, които се използват и за PWM, на тези изходи). Честотата, задавана на функцията, може да е от 31Hz до 65535Hz.

Tone Library на Brett Hagman

Използвайте Library Manager (описан в Глава 2), за да инсталирате библиотеката Tone Library от Brett Hagman, ако е необходима подобрена функционалност спрямо вградените в Arduino IDE tone()/noTone() функции. Тя предоставя клас Tone с повече възможности. Методите на класа са:

- begin(pin) – подготвя даден изход (pin) за работа със звук;
- isPlaying() – връща true, ако се просвирва звук;

- play(frequency[, duration]) – просвирва звук с дадена честота и опционално – продължителност в mS. Не-блокираща, връща управлението веднага;
- stop() – спира просвирването – аналог на noTone.

Библиотеката Tone предоставя за по-голямо удобство във вид на константи всички музикални ноти и октави NOTE_xxx (например NOTE_A4 е нота А от 4-та октава – честота 440Hz).

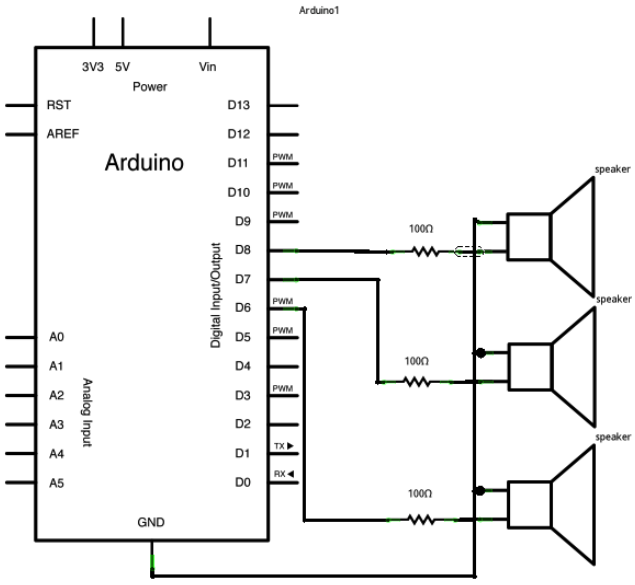
За разлика от стандартната функция note(), класът Tone може да извежда звук на произволен изход, при това едновременно няколко (стартират се няколко инстанции на класа). Трябва да се внимава, защото броят на едновременно просвирваните звуци зависи от модела на платката (броя на хардуерните таймери със CTC характеристика в микроконтролера). За съответния микроконтролер, това са:

- ATmega8: 2 (таймери 2 и 1);
- ATmega168/328: 3 (таймери 2, 1 и 0) – Uno са с 328;
- ATmega1280/2560: 6 (таймери 2, 3, 4, 5, 1, 0).

Редът на таймерите в горното изброяване е редът, в който те се заделят и използват от множеството инстанции на Tone. Пускането с play() заделя таймер, спирането със stop() или изтичането на зададената продължителност, освобождават таймера. Таймер 0 е най-добре да остане свободен, ако не е необходимо, защото той се използва от функциите millis(), delay() и при PWM. Вижте ***https://github.com/bhagman/Tone***

Управление на високоговорители и пиезо-елементи

Ще бъде показано как се прави свързване на три маломощни високоговорителя към Arduino Uno и просвирването на три тона (последователно) – защото първо ще се демонстрира с вградените функции tone()/noTone() и после ще се покаже с Tone библиотеката как може да се просвирят 3 звука на трите високоговорителя едновременно.



Фиг. 27. Свързване на 3 високоговорителя към Arduino

Не трябва да се забравя за ограничителните резистори (100Ω), предпазващи изходите на микроконтролера от претоварване. Използвани са цифровите изходи D6, D7 и D8.

Проект 4.1 – три високоговорителя с вградената функция tone() – може да използвате файла *Project04/Project_04.1/Project_04.1.ino*:

```
// Проект 4.1 – Просвирване на 3 звука последователно
// Използват се вградените функции tone()/noTone()
// Не забравяйте 100 ома резистори към всеки от 3-те високоговорителя по 8 ома
// свързани към цифрови изходи 6, 7 и 8
void setup() { // нямаме нужда от setup
}
void loop() {
  tone(6, 440, 200);  // просвирване на нота A4 на изход 6 200mS
  delay(200);
  noTone(6); // спиране на звука на изход 6
  tone(7, 494, 500); // просвирване на нота B4 на изход 7 500mS
  delay(500);
  noTone(7); // спиране на звука на изход 7
  tone(8, 523, 500); // просвирване на нота C5 на изход 8 500mS
  delay(800);
  noTone(8); // спиране на звука на изход 8
}
```

Същата схема на свързване, но с използване на библиотеката Tone (не забравяйте да я инсталирате), позволява просвирване на 3 звука едновременно – на всеки високоговорител отделен звук.

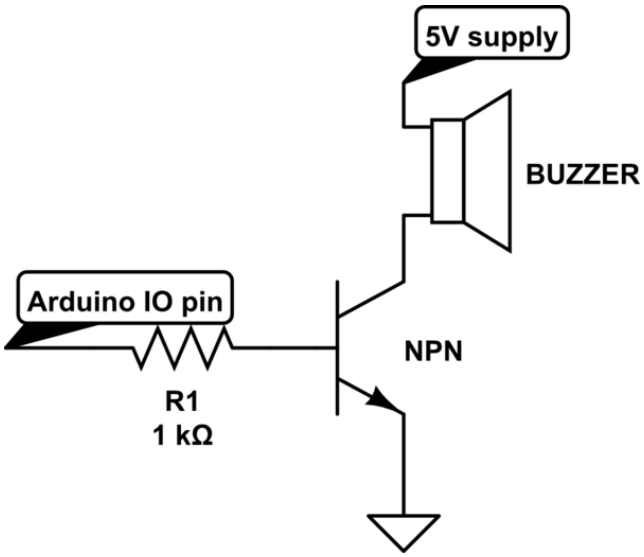
Може да използвате файла *Project04/Project_04.2/Project_04.2.ino*:

```
// Проект 4.2 – Просвирване на 3 звука едновременно
// Използва се библиотеката Tone на Brett Hagmann
// Не забравяйте 100 ома резистори към всеки от 3-те високоговорителя по 8 ома
// свързани към цифрови изходи 6, 7 и 8
#include <Tone.h>
Tone tone1;  // инстанция на Tone tone1
Tone tone2;  // инстанция на Tone tone2
Tone tone3;  // инстанция на Tone tone3
void setup() {
  tone1.begin(6); // задаване на изход 6 за tone1
  tone2.begin(7); // задаване на изход 7 за tone2
  tone3.begin(8); // задаване на изход 8 за tone3
}
void loop() {
  tone1.play(NOTE_A4, 200);  // просвирване на нота A4 на изход 6 200mS
  tone2.play(NOTE_B4, 500); // просвирване на нота B4 на изход 7 500mS
  tone2.play(NOTE_C5, 500); // просвирване на нота C5 на изход 8 500mS
  delay(1000);
}
```

Може да се обърне внимание на леснотата, с която се просвирват звуци с библиотеката Tone, с изключение на по-сложното начално задаване на инстанциите и изходите.

Вместо маломощен високоговорител, може да се използва модул с активен пиезо-елемент. Активен в случая означава, че на малката платка, освен пиезо-елемента, има и транзистор и ограничителен резистор. Трябва да се подадат и +5V и земя към модула, освен сигналния (I/O) от изхода на микроконтролера, за да може да работи правилно транзистора, включен като усилвател. Може да се използват и отделни транзистор, резистор и пиезо-елемент, като се спазва схемата на модула по-долу.

Фиг. 28. Модул с активен пиезо-елемент



Фиг. 29. Схема на модул с активен пиезо-елемент

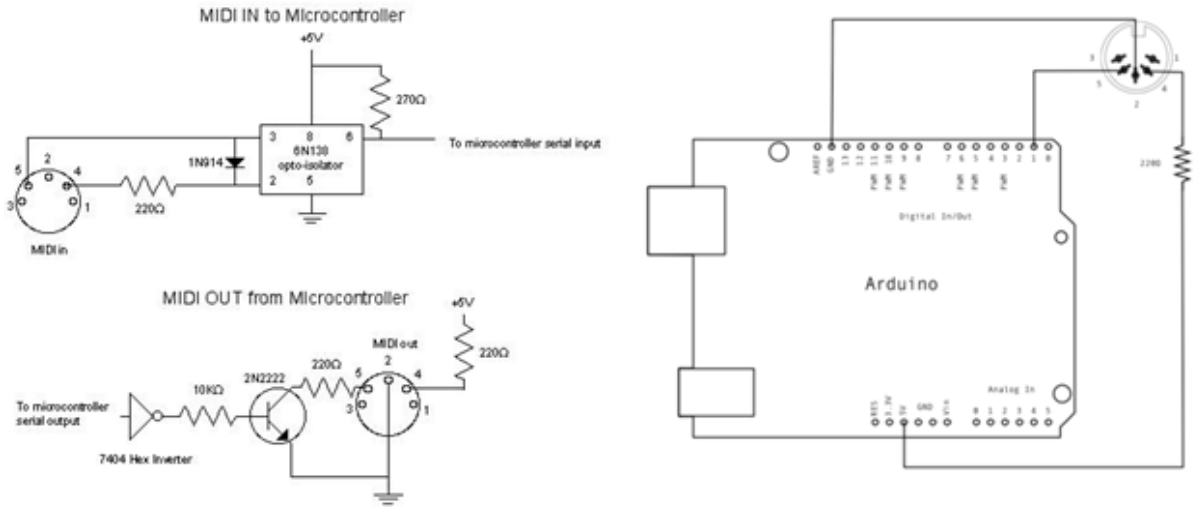
Управление на MIDI устройства с Arduino

MIDI (Musical Instrument Digital Interface) е широко използван интерфейс за връзка между музикални инструменти, предимно синтезатори, позволяващ реализиране на достъпно домашно студио. Развитието на MIDI оказва голямо влияние върху развитието на електронната музика. Модерните денс, техно, хаус, дръм енд бейс и др. жанрове не биха могли да достигнат нивото си без тази технология. MIDI позволява създаването на „домашно студио“ бързо и относително евтино, което облекчава развитието на експерименталната музика. [Wikipedia – MIDI]

MIDI-интерфейсът има три съединителни куплунга: IN, OUT и THRU. IN е вход, OUT е изход, а THRU повтаря входния сигнал и се използва за свързването на повече от едно MIDI-устройство каскадно. Интерфейсът на MIDI е тип „токов кръг“. Всеки байт започва със „стартов бит“, когато токът достига 5 mA, следван от 8 бита данни (най-младшият бит (Least Significant Bit, LSB) се предава първи) и завършва със „стоп бит“, когато токът се завръща към 0 mA – по подобие на RS-232 серийния интерфейс.

Има два основни вида съобщения: статус-байт, при който най-старшият бит винаги има стойност 1, и байт с данни, в който най-старшият бит винаги има стойност 0. Статус-байтът е разделен на две части. Старшите 4 бита задават типа събитие (има 8 типа, от 8 до 15), а младшите 4 – номера на канала (0–15). Например 1001 0001 (в двоична система) означава „Note On“ (включена нота) на канал 1 (забележка: тук стартовият и стоп битът не са указани). Всеки статус-байт бива последван от байтове данни. Статус-байтът „Note On“ бива последван от два байта, задаващи височината на нотата и силата на звука, с която е изсвирена. Например 0011 1100 означава нотата със стойност 60 („до“ от първа октава), а 0111 1111 означава сила на звука 127 (максимална). Ако статус-байтът съвпада с този на предходното съобщение, той може да се изпусне, а да се предават само байтовете данни. Това състояние се нарича „текущ статус“.

За повече информация за използване на MIDI библиотеката за Arduino, вижте ***https://playground.arduino.cc/Main/MIDILibrary***



Фиг. 30. Свързване на MIDI устройства

Любознателните музиканти-любители могат да експериментират сами с MIDI и Arduino, ако имат под ръка някакви MIDI устройства (синтезатори, клавиатури или секуенсъри).

Ако разполагате с платка Arduino Zero, Due или 101, може да използвате „Native USB port“-а като MIDI устройство, с помощта на библиотеката Arduino MIDI USB Library (може да я инсталирате с помощта на Library Manager). Примерен вариант е с помощта на свързани към Arduino Due бутони или клавиатура, да подавате MIDI команди към вградения в MS Windows софтуерен MIDI синтезатор. За повече информация, вижте: ***https://www.arduino.cc/en/Tutorial/MidiDevice***

Глава 7. Проект 5 – Аналогови сигнали. Аналогови датчици за температура и осветеност.

Потенциометър, терморезистор и фоторезистор

Преди да се премине към обработката на аналогови сигнали в Arduino, трябва да бъдат разгледани няколко възможни „източника“ на аналогов сигнал – това са датчици или сензори, генериращи промяна на изходното си напрежение при промяна на следената величина или някое действие на потребителя. Освен електронни схеми и сензори, това са и прости устройства като потенциометри, терморезистори и фоторезистори.

Потенциометърът е резистор с 3 извода, с който е възможно при промяната на съпротивлението чрез плъзгащ контакт в електрическата верига, да се променя изходното електрическото напрежение в предварително конструктивно зададени граници. Плъзгащия контакт на този пасивен компонент е един от изходните електроди и работи като делител на напрежение. Ако се използват само двата му края, действа като регулируем резистор (реостат). Използва се често за плавно регулиране на настройки, например за регулиране на силата или усилването на звука в аудио и Hi-Fi техниката. [Wikipedia – Потенциометър]

Терморезисторът е резистор, чието съпротивление се влияе силно от температурата на околната среда, т.е. има голям температурен коефициент на съпротивлението. В зависимост от това, дали температурния коефициент е положителен или отрицателен, терморезисторите се делят съответно на позистори (PTC) и термистори (NTC). Термо–резисторите се използват основно за измерване на температури, ограничаване и стабилизация на тока в електрическите схеми (защита от прегряване при претоварване) и за температурна компенсация на електронни схеми и на други видове сензори, чиито показания се променят при изменение на околната температура. [Wikipedia – Терморезистор]

Фоторезисторът или още LDR (Light Dependent Resistor) представлява полупроводников електронен елемент – резистор, чието съпротивление е в обратнопропорционална зависимост от падащия върху него светлинен поток. Използва се за реакция спрямо околната осветеност – например, включване на осветлението при свечеряване, автоматично регулиране на силата на яркост на дисплей и др. [Wikipedia – Фоторезистор]

Измерване на напрежение (аналогов входен сигнал)

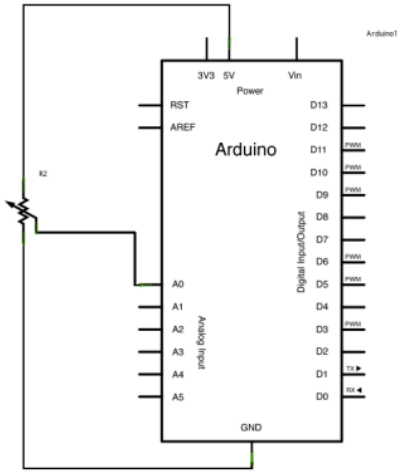
Общото между гореизброените три вида електронни елемента е, че промяната в тяхното съпротивление предизвиква промяна в напрежението (при съответното свързване), което подадено на някой от аналоговите входове на Arduino платката, може да бъде прочетено с помощта на функцията analogRead(pin). Тя връща целочислена int стойност 0..1023 (10-бита). Стойността на pin трябва да е 0..5 при Arduino Uno, Micro и Zero, 0...7 при Mini и Nano, 0..11 при Due и 0..15 при Mega платките.

Трябва да се отбележи, че отнема около 100 µs (0.0001 s) за прочитане на сигнала с функцията analogRead, тоест максималната скорост на четене е около 10 000 пъти в секунда. При нужда от по-висока скорост на четене или от по-голяма точност (по-голям брой битове), трябва да се използват външни АЦП (аналогово-цифров преобразувател, ADC), свързани по I^2C или друга шина (разполагаща с необходимата скорост на трансфер на данни) към Arduino платката.

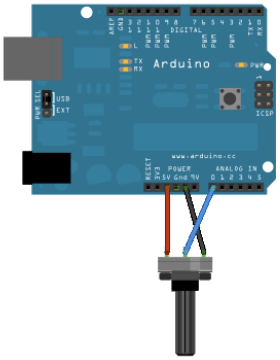
При използване на платки Arduino Due или Arduino Zero, поради факта, че те имат 12-битов АЦП (използват ARM Cortex процесор), може да се укаже на функцията analogRead да връща 12-бита (0..4096), макар по подразбиране (с цел съвместимост с AVR) тя е зададена да бъде 10-бита – това става с помощта на функцията analogReadResolution(bits) – bits може да е всяко цяло число от 1 до 32, но реално допустимите стойности са 10 или 12 (ако се укаже брой битове, различен от 10 или 12, се извършва автоматично „закръгляне“ или „отрязване“ до зададения брой битове от функцията analogRead, според броя зададени битове точност (над или под реалната стойност) – трябва да се внимава.

PWM с потенциометър

Ще бъде демонстрирана работата на аналоговия вход и потенциометър, като се използва въртенето на потенциометър за регулиране на яркостта на светодиода (по подобие на Проект 1.3, но вместо плавно в цикъл, яркостта ще зависи от позицията на потенциометъра). Стойността на потенциометъра не е от голямо значение, но при 5V захранване (Arduino Uno) ще е добре да е над 1kΩ и по-малка от 50kΩ. Средният извод на потенциометъра се свързва към аналоговия вход A0, а другите 2 края към земя и +5V (в този случай без значение, но иначе зависи от това, в коя посока на въртене ще искаме намаляване или увеличаване на стойността, отчетена с analogRead()).



Фиг. 31. Схема на свързване на потенциометър към аналогов вход A0



Фиг. 32. Свързване на потенциометър към аналогов вход A0

Проект 5.1 показва примерния код, реализиращ четене на стойността на потенциометър, свързан по показания по-горе начин. Може да използвате файла *Project05/Project_05.1/Project_05.1.ino*:

```
// Проект 5.1 – Преливане на светодиод  с потенциометър
// Преливане на светодиод на PWM изход 5 от 0 до 100% с въртене
// Потенциометър 10 kOhm на аналогов вход A0, LED на пин 5 през резистор 220 ома към GND
const int POT_INPUT = A0; const int LED = 5;
void setup() {
  pinMode(LED, OUTPUT);
}
void loop() {
  int value = analogRead(POT_INPUT);
  value = map(value, 0, 1023, 0, 255);
  analogWrite(LED, value);
}
```

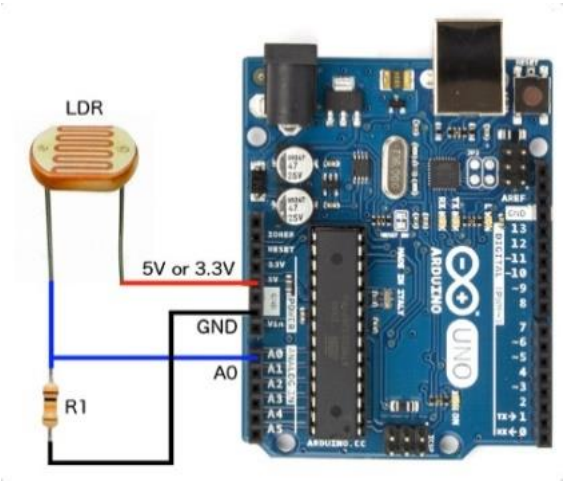
В Проект 5.1 е използвана функцията map(). Тя има следния формат:

map(value, fromLow, fromHigh, toLow, toHigh)

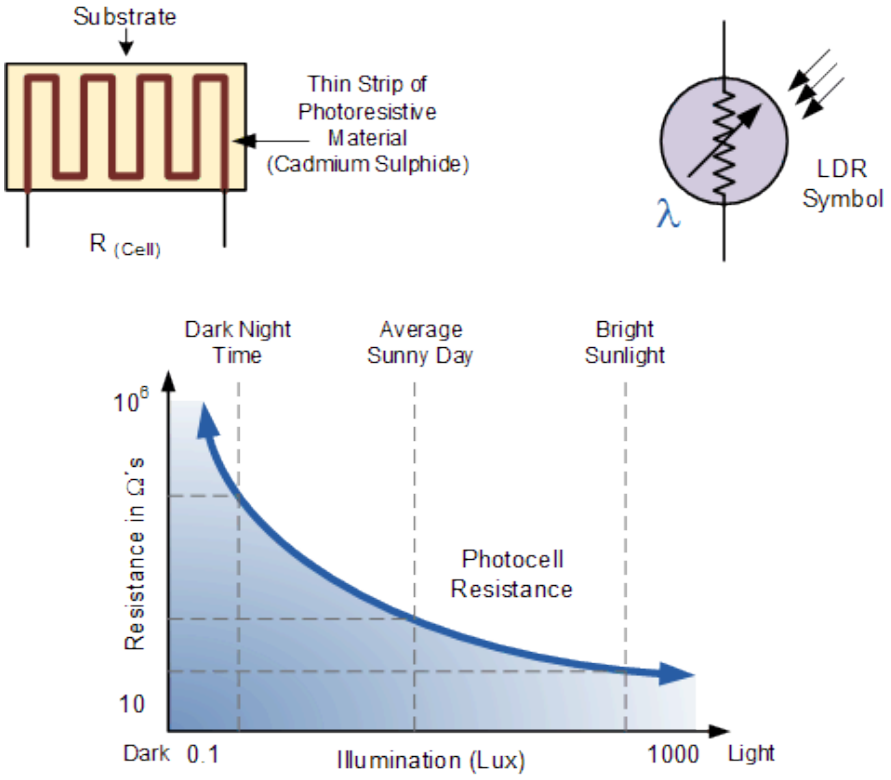
и се използва за преизчисляване на дадена стойност от един обхват числа към друг обхват числа. По този начин, получената входна стойност в аналоговия вход A0 от функцията analogRead 0..1023 се преизчислява в обхвата 0...255, който се изисква за функцията analogWrite за PWM изходите.

Използване на фоторезистор или терморезистор

Използването на фоторезистор или терморезистор с Arduino е подобно на показаното по-горе свързване на потенциометър, само че едната „половина“ на потенциометъра се заменя с фоторезистор или терморезистор. По този начин, в получения делител на напрежение, подаваното към аналоговия вход напрежение ще зависи от съпротивлението на сензора. При високо съпротивление ще има по-ниско напрежение, при по-ниско съпротивление ще има по-високо напрежение. Ако се цели обратната логика, може да се сложи сензора към земя, а ограничителния резистор от делителя да се свърже към захранващото напрежение. Трябва да се отчита, че за разлика от потенциометъра, повечето сензори не променят своето съпротивление от 0 до безкрайно голяма стойност, а в някакъв обхват, също така, както се вижда от показаната по-долу графика на характеристиката на фоторезистор от тип CdS (кадмиев сулфид), тя не е строго линейна. За по-непретенциозни проекти, където е важно да се „усети“ промяната в осветеността или температурата, а не точната ѝ стойност, може да се приеме, че графиката на характеристиката на сензора е линейна (да се апроксимира или приближи до линейна). При нужда от точно измерване, се използват цифрови датчици с вградена компенсация или се извършва прецизна многоточкова линеаризация с емпирични или опитно измерени стойности.



Фиг. 33. Свързване на фоторезистор (LDR)



Фиг. 34. Характеристика на фоторезистор (LDR)

Проект 5.2 показва примерния код, реализиращ четене на стойността на фоторезистор, свързан по показания по-горе начин. Може да използвате файла *Project05/Project_05.2/Project_05.2.ino*:

```
// Проект 5.2 – Фоторезистор
// Включване на вградения светодиод при стъмняване
// CdS LDR фоторезистор на аналогов вход A0
const int LDR_INPUT = A0;
const int NIGHT_VALUE = 800; // опитно се измерва или пресмята
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
```

```
}
void loop() {
  int value = analogRead(LDR_INPUT);
  if ( value < NIGHT_VALUE ) {
        digitalWrite(LED_BUILTIN, HIGH);
  }
  else {
        digitalWrite(LED_BUILTIN, LOW);
  }
  delay(1000);
}
```

Стойността NIGHT_VALUE се пресмята емпирично или по характеристиката от описанието на съответния аналогов сензор (фоторезистор, терморезистор или друг вид сензор). Закъснението се прави с цел да се избегне „честото включване и изключване“ при гранични стойности, макар че в реални проекти е по-добре да се реализира хистерезис или защита, подобна на показаната в Проект 2.3 за debounce на бутон (на практика стойността на сензора играе по подобен начин, но около някаква стойност, не вкл./изкл., както при бутоните). Хистерезис е също и факта, че сензора не се „връща“ точно линейно при обратното въздействие. [Wikipedia – Hysteresis]

Глава 8. Проект 6 – I²C, SPI и 1-Wire шина. Цифрови датчици за температура. CAN шина.

I²C (TWI) устройства

I²C (Inter-Integrated Circuit) е серийна компютърна шина, разработена от Philips (сега NXP), използвана за комуникация с бавни периферни схеми към процесори и микроконтролери на къси разстояния (вътре в платките или с къси кабели). Тя позволява наличието на множество главни и множество подчинени устройства (multi master, multi slave). За връзка между устройствата по I²C се използват само 2 линии – SDA (Serial DAta line) и SCL (Serial CLock line), като дължината на шината е ограничена само от сумарния паразитен капацитет (до 400pF), като има и вариант на I²C с източник на ток, вместо с pull-up за по-бърз трансфер и по-голям капацитет (дължина). Всяко устройство, свързано по шината, трябва да притежава уникален (за шината) адрес, с който останалите да се обръщат към него. При предаване на пакет данни между 2 устройства, винаги едното е главно (master), а другото – подчинено (slave). TWI (Two Wire Interface) е 100% идентичен с I²C, означението е въведено от Atmel, за да се избегнат патентни проблеми с Philips – трябва да се отбележи, че след октомври 2006 г. I²C е вече свободен от лицензионни такси и се използва безплатно.

Може да се използват 7, 8 и 10 бита за адресите в I²C. Библиотеката Wire в Arduino използва само 7 бита, така че, например, при използване на устройства с 8-битови адреси, се налага да се „изпусне“ 8-мия бит (да се измести адреса надясно с 1 бит) – ще се загубят адресите от 0 до 7 и ще се използват само адреси 0..127. Трябва да се запомни също, че SDA/SCL линиите винаги изискват pull-up резистори, като Arduino Mega2560 платките имат вградени такива резистори на изводи 20 и 21).

Изводите на I²C са на различно място, според модела на Arduino:

- Uno и 2009 A4 (SDA), A5 (SCL)
- Uno rev.3 SDA, SCL до AREF (след извод 13)
- Mega2560 20 (SDA), 21 (SCL)
- Leonardo 2 (SDA), 3 (SCL)
- Due 20 (SDA), 21 (SCL), втори на SDA1 , SCL1

Използване на Wire библиотеката за TWI/I²C

begin() или begin(address) – инициализира Wire библиотеката и стартира комуникацията по шината като master, ако не е указан адрес (празен параметър) или като slave с указания 7-битов адрес.

requestFrom(slaveaddress, quantity, stop=true) – кара master-а да заяви получаването на quantity байтове от slave, те се получават чрез available() и read(). Ако има параметър stop true (или няма такъв), освобождава шината, ако е false, изчаква следваща комуникация, без да освобождава шината, за да не пречат другите устройства. Връща броя на наистина получените байтове от това устройство.

beginTransmission(address) – започва комуникация с дадения адрес и после може с write() да се изпращат байтове към това устройство.

endTransmission() – приключва комуникацията, започната с beginTrasmission, като първо изпраща неизпратените байтове.

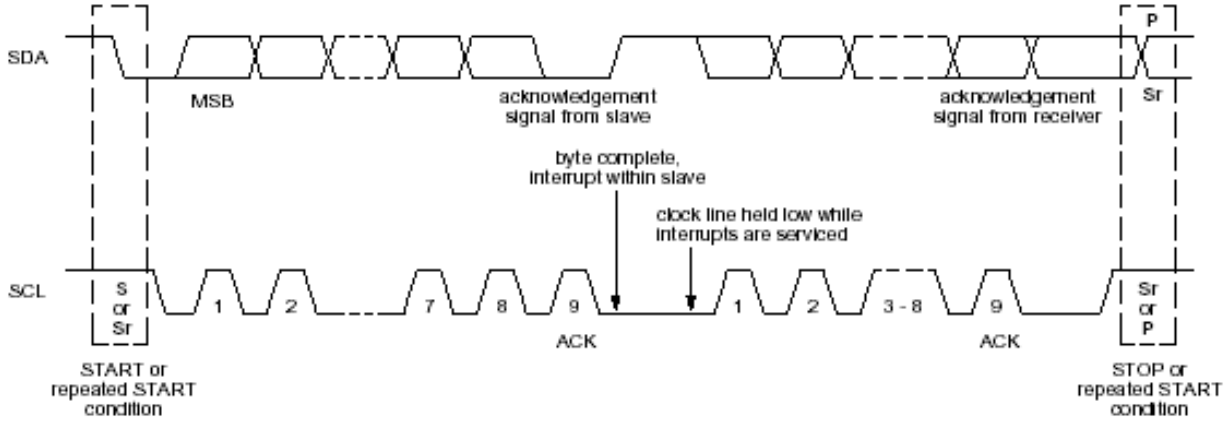
write() – изпраща дадените байтове. Синтаксис: write(value) или write(string) или write(data, length). Връща броя действително изпратени байтове, макар че това не винаги е нужно.

available() – проверява дали има данни (връща логическа ст-ст).

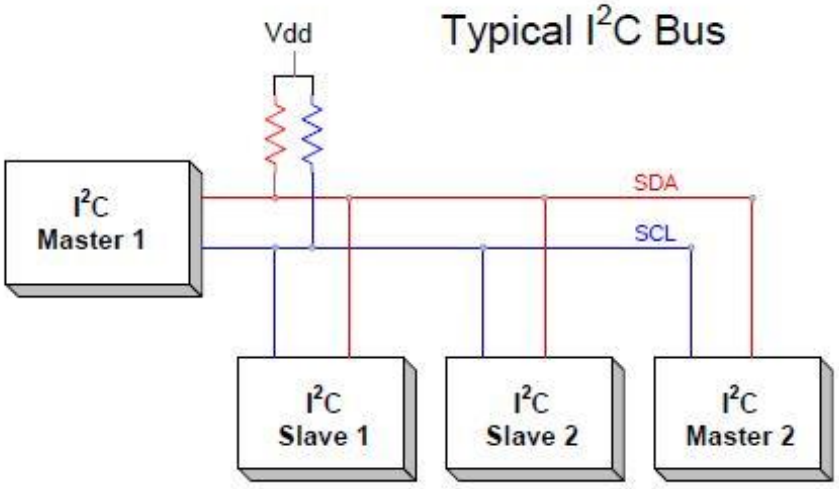
read() – прочита наличен байт, използва се с available().

onReceive() – задава функция от вида void myHandler(int numBytes), която да обработи заявката от master-а.

onRequest() – задава функция от вида void myHandler(), която да обработи заявката от slave устройството.



Фиг. 35. Време-диаграма на I²C протокола



Фиг. 36. I²C шина – схема на свързване

В Arduino IDE версиите след 1.0 функциите send()/receive() са заменени с read()/write(), това трябва да се отчита при използване на стар код. Вижте повече за Wire библиотеката на следния адрес:

https://www.arduino.cc/en/Reference/Wire

Библиотеки за серийни протоколи SPI и Serial

Освен I^2C, при работата с микроконтролери често се използват SPI и обикновена серийна комуникация. Разликата между I^2C и SPI, освен в самия протокол на обмен, и в реализацията – има само един master, SPI е и пълен дуплекс, докато I^2C е полу-дуплексен протокол, SPI не гарантира изрично получаването на пакета с данни и др. особености, като най-важната е, че се изискват 3 или 4 линии за връзка, спрямо 2 при I^2C.

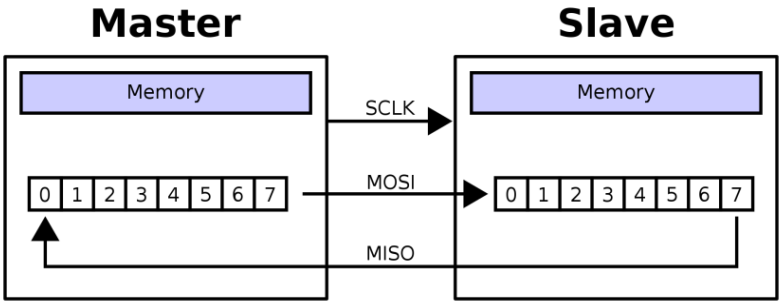
Шината SPI задава пет логически сигнала:

- SCLK: Serial Clock (изход от master устройството);
- MOSI: Master Output Slave Input (изход на данни от master у-во);
- MISO: Master Input Slave Output (изход на данни от slave у-во);
- SDIO: Serial Data I/O (двупосочен вход-изход по 1 линия) – опционален сигнал;
- SS: Slave Select (често е активно LOW ниво, изход от master у-во).

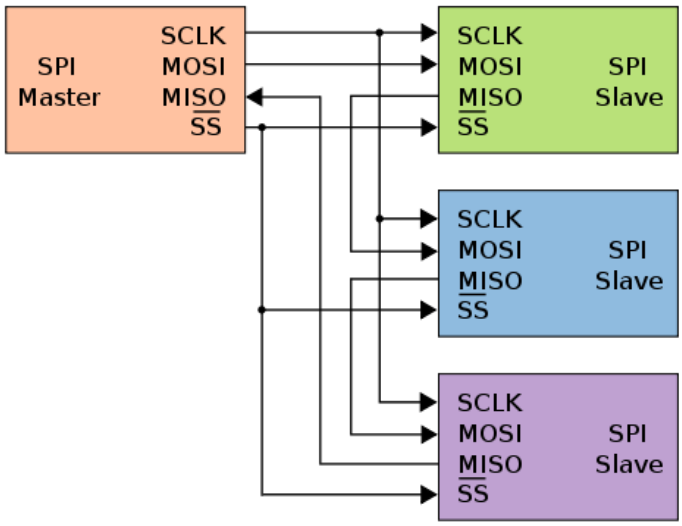
Разположение на изводите за SPI в Arduino – в Arduino Due SPI е изведен само на ICSP цокъла, за разлика от Arduino Uno, където е дублиран на изводи 11 MOSI, 12 MISO и 13 SCK (+ избран от потребителя изход за SS, ако микроконтролера е master или 10 за SS, ако е slave). В Arduino Due SPI се реализира по различен начин – трябва да се внимава. За повече информация, вижте:

http://arduino.cc/en/Reference/SPI и

http://arduino.cc/en/Reference/DueExtendedSPI

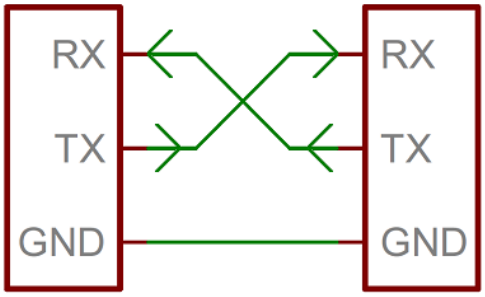


Фиг. 37. Свързване на две SPI устройства



Фиг. 38. Свързване на няколко SPI устройства

Серийната комуникация с две линии Rx/Tx е класическа, изисква само 3 линии за свързване – Receive Rx, Transmit Tx и земя. Недостатък е, че позволява директното свързване само между две устройства, а предимството е в простотата на реализация и възможността за дълги кабели при използване на повторители или усилватели на сигнала.



Фиг. 39. Свързване на две серийни устройства

Serial – в платките Arduino има поне 1 стандартен (хардуерен) сериен порт, като в платките Arduino Mega2560 и Arduino Due има по 3 допълнителни серийни порта (общо 4 порта). Трябва да се отчитат нивата на сигналите – дали са 3.3V или 5V на съответната платка и устройство! Често се използва и за диагностика при връзка по USB при свързан компютър с помощта на инструмента Serial Monitor на Arduino IDE в меню Tools – Serial Monitor. За повече информация, вижте

http://arduino.cc/en/Reference/Serial

SoftwareSerial – позволява всеки I/O изход да е Rx или Tx, но не позволява застъпване на сигналите, само 1 порт може да приема/предава в даден момент! Вижте ***http://arduino.cc/en/Reference/SoftwareSerial***

След като бяха посочени възможните начини за комуникация по сериен протокол, ще бъдат представени и няколко сензора, които ги използват. Основното предимство на серийните устройства в проектите с микроконтролери е малкият брой свързващи линии и унифицирания протокол. Едно устройство, използващо I^2C, например, е готово веднага за използване, само трябва да се подадат необходимите данни на неговия адрес (това се взима от спецификацията на устройството), като разработчиците няма нужда да се занимават с допълнителни хардуерни въпроси относно свързването му с микроконтролера – проблемите стават само „софтуерни“.

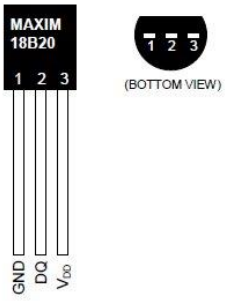
Сензор за температура с 1-Wire – DS18B20

DS18B20 е лесен за използване сензор за температура, удобен поради малките си размери и възможността за свързване на множество сензори по една обща линия за комуникация. Използва се протокол с една линия за връзка, но различен от този при DHT22 (той ще бъде показан по-долу) – Dallas 1-Wire (OneWire), това е двупосочен, полу-дуплексен протокол до 15.4 kbps, подобен като концепция на I^2C протокола, но с по-ниска скорост на предаване на данните на по-дълго разстояние, което го прави подходящ за датчици за температура и други метереологични данни. Позволява и паразитно захранване по линията за данни, по този начин може няколко 1-Wire сензора да се захранват и четат/управляват само по 2 линии при достатъчно дълъг кабел за връзка с тях. Всяко устройство има уникален 64-битов сериен номер – най-младшия байт е 8-битов код на типа на устройството, най-старшия байт е 8-битова контролна сума, това позволява устройствата с 1-Wire да се използват и за идентификация или управляващия кон-

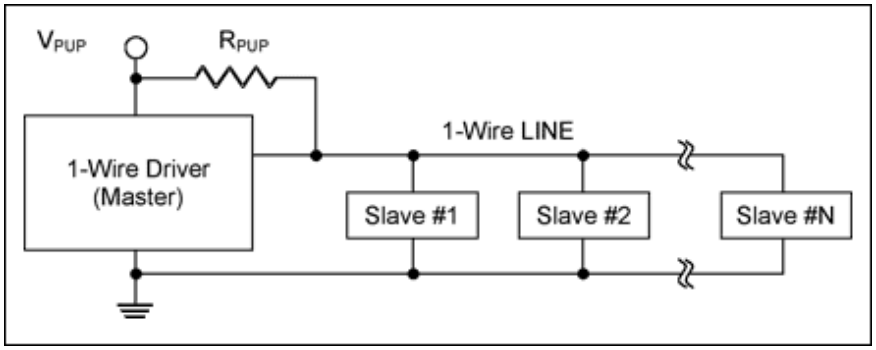
тролер да разпознава отделните сензори или да реагира при подмяна (например да изиска начална настройка – калибриране на показанията на „новия“ сензор). За повече информация, вижте:

http://www.scienceprog.com/understanding-1-wire-interface и

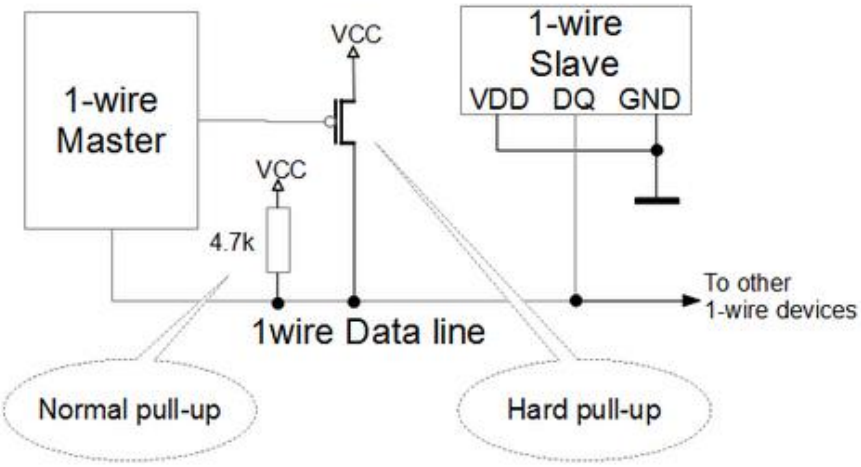
http://playground.arduino.cc/Learning/OneWire



Фиг. 40. Сензор за температура DS18B20 (1-Wire)



Фиг. 41. 1-Wire шина – схема на свързване

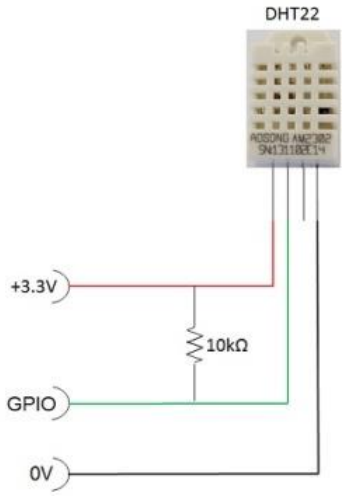


Фиг. 42. Паразитно захранване на 1-Wire сензори

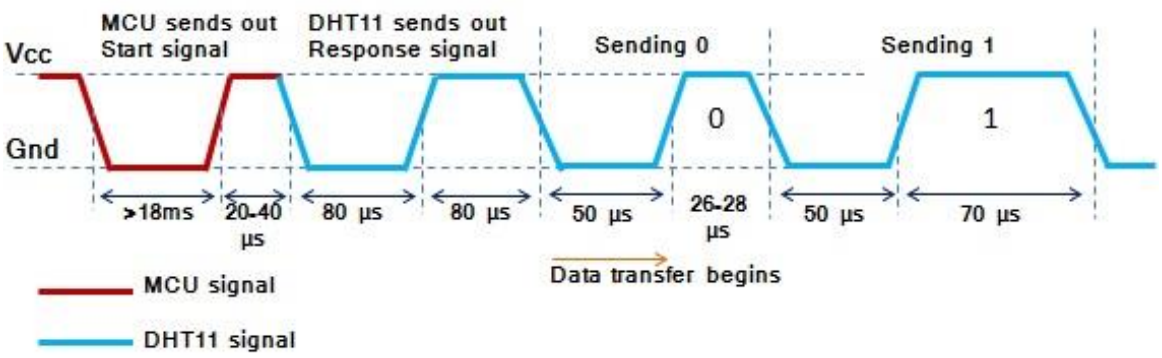
Сензор за температура и влажност DHT22

DHT22 е лесен за използване сензор за влажност и температура. Много е удобен за метеостанции, измерването става само след иницииране на сигнал от контролера, през останалото време е в режим на ниска консумация. Използва се еднопосочен протокол с една линия за връзка (не е 1-Wire). Има и версия само за положителни темпе-

ратури (DHT11), която е по-евтина, но не става за метеостанции (зимния период). За повече информация, вижте ***http://playground.arduino.cc/Main/DHTLib***



Фиг. 43. DHT22 (AM2302) сензор за температура и влажност



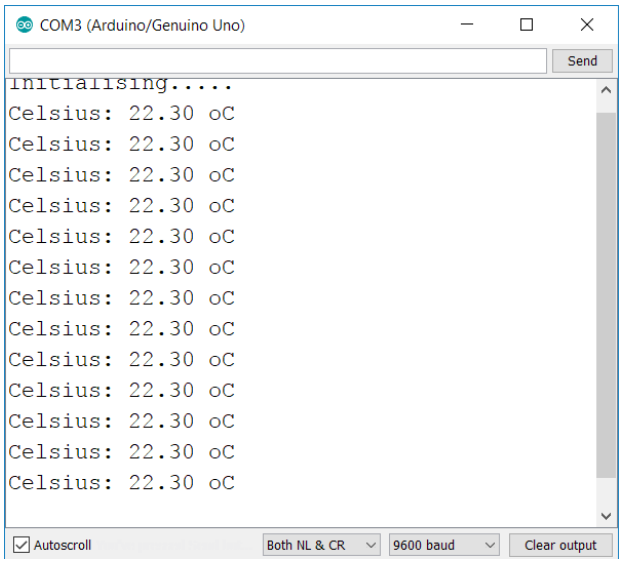
Фиг. 44. Протокол за връзка на DHT11(DHT22) сензорите

По долу ще бъдат показани примерни проекти с тези сензори за температура, а използването на I^2C и SPI ще бъде онагледено в следващата глава с помощта на I^2C и SPI дисплеи и индикатори.

Използване на сензорите DHT22/18B20 в проекти

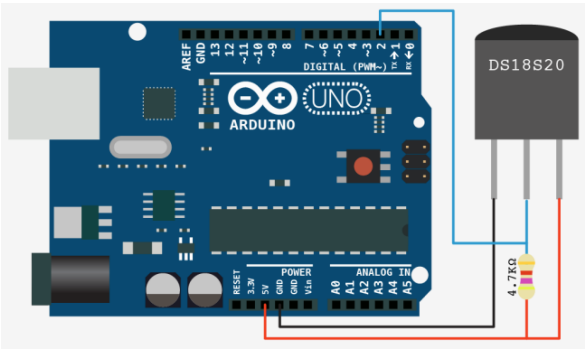
Сензорите за температура DS18B20 се предлагат в няколко варианта – като чип в корпуси TO-92, SO-150mils и µSOP, както и във вид на модули (запоени на малки платки) или монтирани във водоустойчиви неръждаеми тръбички за използване във външна среда. Независимо кой вариант е избран, винаги има нужда от свързването на 3 активни извода – DQ (Data Input/Output) сигнален извод, GND (земя) и V_{DD} (захранване – при паразитно захранване, трябва да се заземи). За демонстрация на използването на този сензор ще бъде показано нормалното свързване – с подаване на захранване по V_{DD} извода. Показанията се извеждат на компютър, свързан с Arduino платката по USB кабел (на практика, след програмирането на кода, с използване на Serial Monitor за показване на прочетените от сензора показания за околната температура. Serial Monitor е удобен инструмент за настройване на Arduino проектите, защото при липса на свързан компютър (или не-включен Serial Monitor) не пречи на изпълнението на програмата в микроконтролера. Обекта Serial комуникира едновременно с Tx/Rx изво-

дите на платката и по USB. За инициализация се използва begin(baud) – параметъра указва скоростта на комуникация по серийния порт. Може да се извежда с print() и println().



Фиг. 45. Serial Monitor с проекта за DS18B20

Примерният проект 6.1 изисква един сензор DS18B20, един pull-up резистор 4.7kΩ и монтажна платка (кабелите и самата платка Arduino се подразбират). С цел икономия на място тук е показано свързването без монтажната платка. Използвайте Library Manager (вижте Глава 2), за да инсталирате библиотеките OneWire (изберете тази на Jim Studt) и DallasTemperature, ако ги нямате инсталирани (това се прави еднократно). Може да използвате файла *Project06/Project_06.1/Project_06.1.ino*.



Фиг. 46. Свързване на DS18B20 с Arduino

```
// Проект 6.1 – Сензор за температура DS18B20
// Извежда на всеки 2 секунди температурата в oC на серийния монитор
#include <OneWire.h> // библиотека за шината OneWire
#include <DallasTemperature.h> // библиотека за сензорите DS18B20
const int ONE_WIRE_BUS = 2; // извод 2 за връзка със сензора
// дефиниране на променливите за OneWire класа и за сензорите
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
```

```
void setup()
{ Serial.begin(9600);
  Serial.println("Initialisng.....");
  sensors.begin();
}
void loop()
{ sensors.requestTemperatures();
  Serial.print("Celsius: ");
  Serial.print(sensors.getTempCByIndex(0)); // първият сензор, ако са повече от 1
  Serial.println(“ oC");
  delay(2000);
}
```

Кратки пояснения към горния код: функцията requestTemperatures() подава команда към всички сензори по шината да стартират измерване на текущата температура, а функцията getTempCByIndex(id) прочита температурата от сензора с индекс id (в случая = 0, първият и единствен по шината). Важно е да се отбележи, че DS18B20 не измерва постоянно, а само при команда от микроконтролера, през останалото време е в режим на изчакване с цел икономия на енергия. Ако не се изпълни функцията requestTemperatures(), ще се прочете „предходно измерената“ температура, която може вече да не е актуална.

В Проект 6.2 се демонстрира използването на сензора за влажност и температура DHT22 (известен и като AM2302). Изисква се само сензора и монтажна платка (кабелите и самата платка Arduino се подразбират). С цел икономия на място, не е показано свързването (свържете извод 2 на Arduino към извод 2 GPIO на DHT22). Използвайте Library Manager (вижте Глава 2), за да инсталирате библиотеката SimpleDHT, ако я нямате инсталирана. Може да използвате *Project06/Project_06.2/Project_06.2.ino*.

```
// Проект 6.2 – Сензор за температура DHT22 (AM2302)
// Извежда на всеки 2 секунди температурата в oC на серийния монитор
#include <SimpleDHT.h>; // библиотека за сензорите DHT
const int pinDHT22 = 2;  // извод 2 за връзка със сензора
SimpleDHT22 dht22; // инициализиране инстанция на променливата dht22 за връзка със сензора
float hum;  // влага
float temp; // температура
void setup() {
  Serial.begin(9600); // стартиране Serial Monitor
}
void loop() {
  if (dht22.read2(pinDHT22, &temp, &hum, NULL) == SimpleDHTErrSuccess) {
    Serial.print("Humidity: ");
    Serial.print(hum);
```

```
    Serial.print(“ %, Temp: ");
    Serial.print(temp);
    Serial.println(“ oC");
    delay(2000); //Delay 2 sec.
  }
}
```

Проект 6.2 е подобен на Проект 6.1, разликата е само в библиотеката и в начина на извикване на функцията за четене на сензора – read2(). Обърнете внимание, че изисква подаване на двете променливи за температура и влажност по адрес с &temp, &hum. Тази библиотека е максимално опростена и заема много малко място в паметта на контролера. Показана е за пример, препоръчва се използването на Adafruit DHT Sensor Library – за сметка на по-големия обем, тя предоставя по-голяма гъвкавост и използва унифицирани обекти за сензорите.

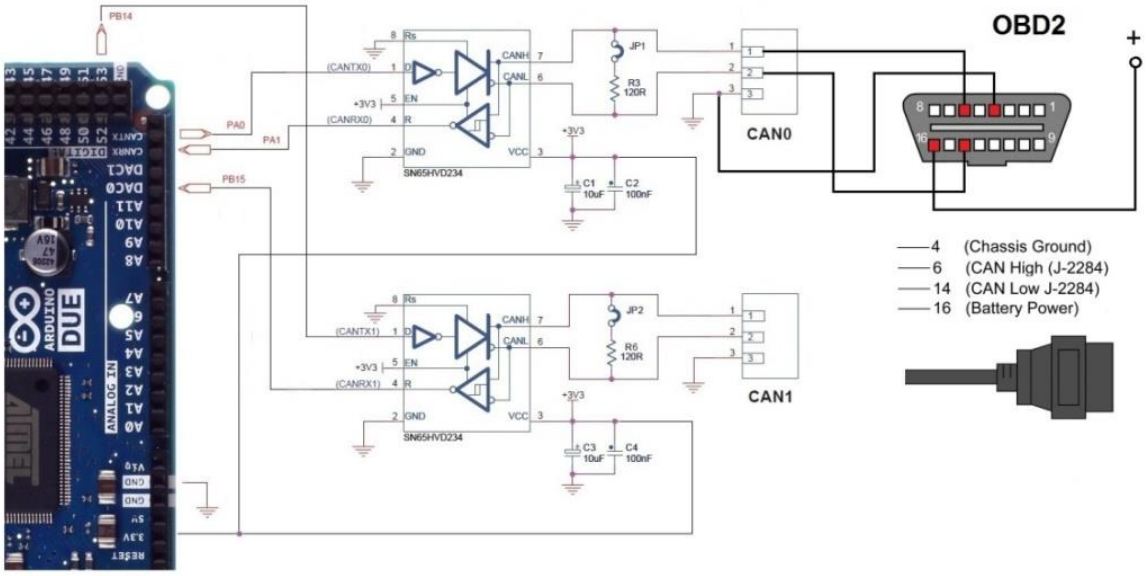
Използване на CAN (за автомобили)

CAN (Controller Area Network) е двупроводна, последователна, асинхронна шина с равноправни възли и подтискане на синфазни смущения, използвана в съвременните автомобили. Вградена поддръжка на CAN има в платките Arduino Due (разгледани в глава 13). Към останалите платки CAN може да се свърже чрез преобразуватели на нива към серийните портове.

Включването на електронните модули в CAN шината става посредством специално устройство – трансивър, основния елемент на който представлява диференциален усилвател. След обработка на сигналите от входовете CAN-High и CAN-Low, те се предават на електронния модул.

Вижте: ***http://media3.ev-tv.me/CANDueUserManual.pdf***

Има доста библиотеки за CAN за Arduino Due. Нужен е трансивър за връзката на CAN шината на автомобила с Arduino Due – свързва се с OBD2 цокъл и кабел. Схема на примерен трансивър е показана по-долу:



Фиг. 47. Схема на трансивър за CAN шина с Arduino Due

Има и готови модули с трансивър за CAN шина на ниска цена (под 5$) – като показания тук (с чип SN65HVD230 на TI).

Фиг. 48. Модул за CAN трансивър с SN65HVD230

Експериментаторите – собственици на автомобили с OBD2 връзка към системата за управление може да експериментират със собствени CAN устройства с Arduino – да следят разхода на гориво и на масло, да управляват някои компоненти на автомобила и др.

Глава 9. Проект 7 – Извеждане на буквено-цифрова информация – LED и LCD дисплеи

За взаимодействие с потребителите във вградените системи се използват LED и LCD дисплеи. Има няколко вида – отделни 7-сегментни дисплеи, свързани директно (изискват се много изходи на контролера и голям брой кабели за връзка с дисплея); блокове от няколко 7-сегментни дисплеи, свързани директно (с мултиплексиране), недостатък е отново големия брой необходими изходи и кабели; 7-сегментни дисплеи и модули с I^2C или SPI управляващ чип (изискват се само 2 или 3 управляващи изхода и 4-5 свързващи кабела, вкл. захранване и земя); LED матрици с точки, обикновено управлявани с чип за икономия на изходите; LCD дисплеи с буквено-цифрова информация с 1, 2 или 4 реда по 8, 16 или 20 знака с или без вградено осветление на екрана – могат да се управляват серийно и паралелно, според управляващия чип, препоръчва се I^2C или друг сериен интерфейс и накрая, големи, с висока резолюция LCD или OLED дисплеи с или без touchscreen управление (с относително висока цена и сложно управление, основен недостатък е, че библиотеките за тяхното управление заемат доста памет и ресурс в микроконтролера и се препоръчват за Arduino Mega2560 или Due платките или при по-големи проекти с необходимост от по-интерактивно взаимодействие с потребителя).

Ще бъде разгледано свързването и управлението на матрици 8x8 LED с MAX7219, модул 8x7-сегментен LED дисплей с TM1638, модули 4 и 8 7-сегментен LED с I^2C управляващ чип и модул LCD1602 с I^2C управление, като най-често използвани в практиката, поради относително ниската си цена и лесното свързване към Arduino платката.

Управление на LED матрица 8x8 с MAX7219

MAX7219 е подходящ за управление на матрици 8x8, 8 бр. 7-сегментни дисплея или до 64 отделни LED (с общ катод) със сериен вход-изход, позволяващ каскадно свързване на няколко модула. Свързването с Arduino е с 3-проводен интерфейс. Има налични много библиотеки за управление на тези чипове и разнообразни модули. Може да видите повече на ***http://playground.arduino.cc/Main/LEDMatrix***

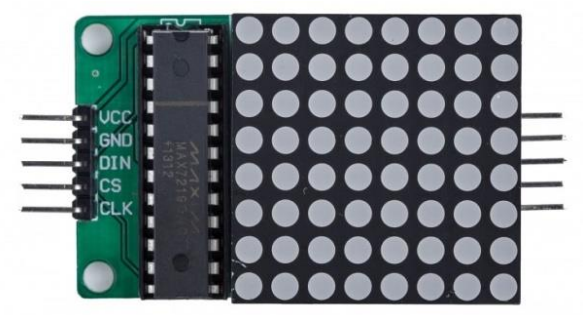
В Проект 7.1 ще бъде демонстрирана работата на модул с MAX7219 и матрица 8x8 LED точки с помощта на библиотеката LEDControl. Използвайте Library Manager (вижте Глава 2), за да инсталирате библиотеката LEDControl, ако я нямате инсталирана. Може да използвате *Project07/Project_07.1/Project_07.1.ino*.

```
// Проект 7.1 – Модул с LED матрица с MAX7219 (само 1 модул да е свързан)
#include "LedControl.h"
LedControl lc = LedControl(12,11,10,1); // DataIn – 12, CLK – 11, LOAD – 10, само 1 модул
unsigned long delaytime=100;
void setup() {
  lc.shutdown(0,false); // забрана на спящия режим
  lc.setIntensity(0,8); // яркост 50%
  lc.clearDisplay(0); // изчистване
```

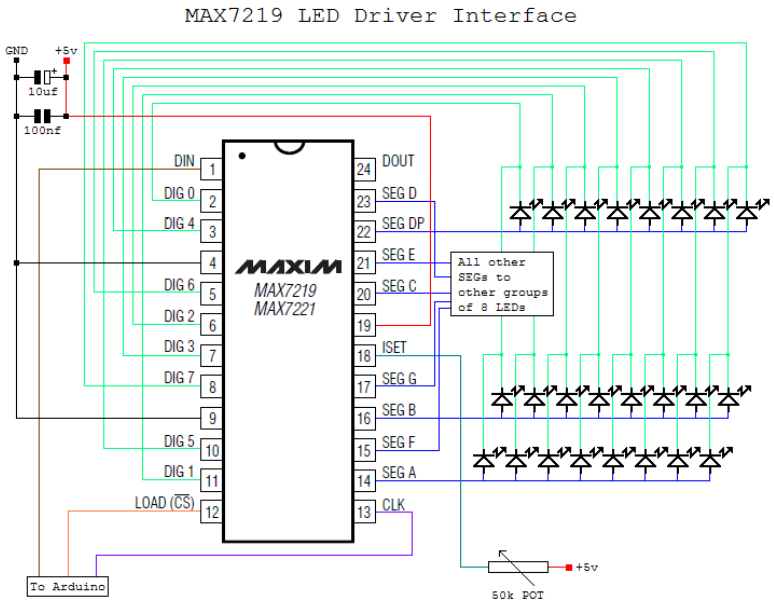
```
}
void writeArduinoOnMatrix() { // данни за думата Arduino
  byte a[5]={B01111110,B10001000,B10001000,B10001000,B01111110};
  byte r[5]={B00111110,B00010000,B00100000,B00100000,B00010000};
  byte d[5]={B00011100,B00100010,B00100010,B00010010,B11111110};
  byte u[5]={B00111100,B00000010,B00000010,B00000100,B00111110};
  byte i[5]={B00000000,B00100010,B10111110,B00000010,B00000000};
  byte n[5]={B00111110,B00010000,B00100000,B00100000,B00011110};
  byte o[5]={B00011100,B00100010,B00100010,B00100010,B00011100};
  // извеждане 1 по 1 на отделните букви от "Arduino"
  lc.setRow(0,0,a[0]);
  lc.setRow(0,1,a[1]);
  lc.setRow(0,2,a[2]);
  lc.setRow(0,3,a[3]);
  lc.setRow(0,4,a[4]);
  delay(delaytime);
  lc.setRow(0,0,r[0]);
  lc.setRow(0,1,r[1]);
  lc.setRow(0,2,r[2]);
  lc.setRow(0,3,r[3]);
  lc.setRow(0,4,r[4]);
  delay(delaytime);
  lc.setRow(0,0,d[0]);
  lc.setRow(0,1,d[1]);
  lc.setRow(0,2,d[2]);
  lc.setRow(0,3,d[3]);
  lc.setRow(0,4,d[4]);
  delay(delaytime);
  lc.setRow(0,0,u[0]);
  lc.setRow(0,1,u[1]);
  lc.setRow(0,2,u[2]);
  lc.setRow(0,3,u[3]);
  lc.setRow(0,4,u[4]);
  delay(delaytime);
  lc.setRow(0,0,i[0]);
  lc.setRow(0,1,i[1]);
  lc.setRow(0,2,i[2]);
  lc.setRow(0,3,i[3]);
  lc.setRow(0,4,i[4]);
```

```
  delay(delaytime);
  lc.setRow(0,0,n[0]);
  lc.setRow(0,1,n[1]);
  lc.setRow(0,2,n[2]);
  lc.setRow(0,3,n[3]);
  lc.setRow(0,4,n[4]);
  delay(delaytime);
  lc.setRow(0,0,o[0]);
  lc.setRow(0,1,o[1]);
  lc.setRow(0,2,o[2]);
  lc.setRow(0,3,o[3]);
  lc.setRow(0,4,o[4]);
  delay(delaytime);
  lc.setRow(0,0,0);
  lc.setRow(0,1,0);
  lc.setRow(0,2,0);
  lc.setRow(0,3,0);
  lc.setRow(0,4,0);
  delay(delaytime);
 }
 void rows() { // примигване на редове
  for(int row=0;row<8;row++) {
    delay(delaytime);
    lc.setRow(0,row,B10100000);
    delay(delaytime);
    lc.setRow(0,row,(byte)0);
    for(int i=0;i<row;i++) {
      delay(delaytime);
      lc.setRow(0,row,B10100000);
      delay(delaytime);
      lc.setRow(0,row,(byte)0);
    }
  }
 }
 void columns() { // примигване на колони
  for(int col=0;col<8;col++) {
    delay(delaytime);
    lc.setColumn(0,col,B10100000);
    delay(delaytime);
```

```
      lc.setColumn(0,col,(byte)0);
      for(int i=0;i<col;i++) {
        delay(delaytime);
        lc.setColumn(0,col,B10100000);
        delay(delaytime);
        lc.setColumn(0,col,(byte)0);
      }
    }
  }
  void single() { // обхождане на всички точки
    for(int row=0;row<8;row++) {
      for(int col=0;col<8;col++) {
        delay(delaytime);
        lc.setLed(0,row,col,true);
        delay(delaytime);
        for(int i=0;i<col;i++) {
          lc.setLed(0,row,col,false);
          delay(delaytime);
          lc.setLed(0,row,col,true);
          delay(delaytime);
        }
      }
    }
  }
  void loop() {
    writeArduinoOnMatrix();
    rows();
    columns();
    single();
  }
```



Фиг. 49. Модул с MAX7219 и 8x8 LED матрица



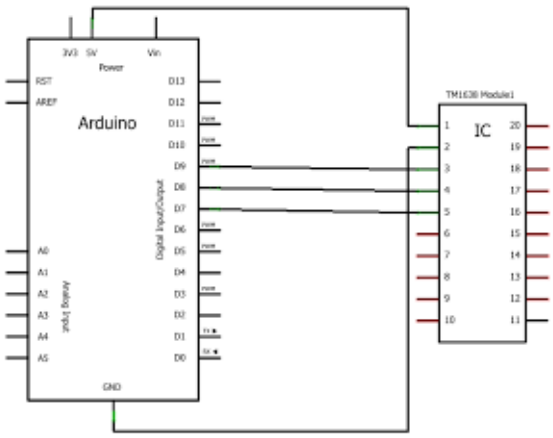
Фиг. 50. Схема на свързване на MAX7219 за 8x8 LED матрица

Управление на LED дисплей с TM1638

TM1638 е чип, подходящ за управление на 8 броя 7-сегментни дисплеи със сериен вход-изход (позволяващ каскадно свързване на няколко модула), както и до 8 бутона и до 8 отделни LED. Свързването с Arduino е с 3-проводен интерфейс SPI. Има удобна за използване библиотека и е евтин като цяло (под 9$) във вид на готов модул, заедно с бутони и отделни светодиоди. Инсталирайте библиотеката TM1638 със ZIP файл от

https://github.com/rjbatista/tm1638-library

Фиг. 51. Модул 8 x 7-сегментен LED с TM1638



Фиг. 52. Схема на свързване на TM1638 за 8x7-сегментен LED

Демонстрация на използването на модул с TM1638 е дадена в Проект 7.2. Може да използвате *Project07/Project_07.2/Project_07.2.ino*.

```
// Проект 7.2 – TM1638 модул с 8x7, 8 LED и 8 бутона
#include <TM1638.h> // библиотека от Ricardo Batista
TM1638 module(8, 9, 7); // свързване data 8, clock 9 strobe 7
void setup() {
  module.setDisplayToHexNumber(0x1234ABCD, 0xF0); // показване на 1234ABCD и 4 точки
}
void loop() {
  byte keys = module.getButtons();
  // светва първите 4 червени и последните 4 зелени LED при натискане на бутон
  module.setLEDs(((keys & 0xF0) << 8) | (keys & 0xF));
}
```

7-сегментен LED модул с I²C управление

По-долу е показано използването на 4x7-сегментен LED дисплей с размери на знаците 1.2” и I²C управление от Sparkfun Electronics, но на практика има голям избор в eBay, трябва да се внимава за различия в управлението им (обикновено търговеца предоставя и примерен код при поискване). Може да вземете указания и код за посочения модул на Sparkfun от ***https://github.com/sparkfun/Serial7SegmentDisplay***

Фиг. 53. 4x7-сегментен дисплей с I²C управление

На практика това е универсален модул, който може да се използва и по I²C и по SPI. Ще бъде показано използването по I²C. Използва се вградената Wire библиотека за управление на I²C шината и се подават отделни команди към чипа, може да се разработи или открие готова библиотека. Ако е свързан само 1 модул, може и без добавяне на pull-up резистори по I²C шината, иначе се препоръчва използването на 4.7kΩ pull-up резистори към SDA/SCL сигналните изходи на Arduino платката.

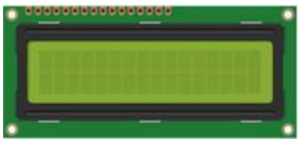
Може да използвате *Project07/Project_07.3/Project_07.3.ino*.

```
// Проект 7.3 – 4x7-сегментен LED модул с I2C на SparkFun Electronics
// Свързване към Arduino Uno: A5 to SCL, A4 to SDA, Vin to PWR, GND to GND
#include <Wire.h>
#define DISPLAY_ADDRESS1 0x71 // настройте адреса спрямо реално използвания I2C в модула
word counter = 0;
```

```
void setup() {
  Wire.begin();
  Wire.beginTransmission(DISPLAY_ADDRESS1);
  Wire.write('v'); // команда за reset – изчиства дисплея, зависи от I2C модула
  Wire.endTransmission();
}
void loop() {
  counter++; if ( counter > 9999 ) counter = 0;
  i2cSendValue(counter); // извежда брояча към дисплея
  delay(100);
}
void i2cSendValue(int number) { // извежда 4-цифрено число на дисплея
  Wire.beginTransmission(DISPLAY_ADDRESS1);
  Wire.write(number / 1000);
  number %= 1000;
  Wire.write(number / 100);
  number %= 100;
  Wire.write(number / 10);
  number %= 10;
  Wire.write(number);
  Wire.endTransmission();
}
```

LCD1602 LCD дисплей с I²C управление

LCD1602 е масово използван буквено-цифров LCD дисплей с LED подсветка, 2 реда по 16 знака с размери на знак 3.2x5.95mm (размери на прозореца 16 x 64.5mm). Използва се често в принтери и други устройства и поради ниската си цена и лесното управление се препоръчва за употреба в Arduino проекти. Има версии с паралелен и сериен интерфейс.



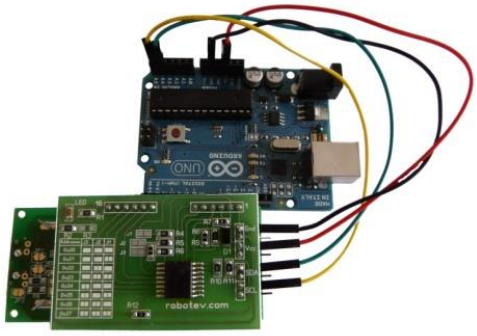
Фиг. 54. LCD1602 дисплей – без модул I²C

Използваният модул има монтирана конверторна платка, която преобразува паралелния интерфейс в I²C. Конверторната платка е изградена с чип MCP23008, има потенциометър за контраста, PCB джъмпери за I²C адрес и 4-пинов конектор за захранването и интерфейсните изводи.

По подразбиране адреса на модула е 0x27. С окъсяване на джъмперите за настройка на I²C адрес може да се зададе друг адрес и така към една I²C шина да се свърже

повече от един дисплей. Извод GND се свързва към маса, извод Vcc към захранване (между 2.7V и 5.5V), извод SDA към A4 на Uno или D20 на Mega2560 и SCL към A5 на Uno или D21 на Mega2560. За повече информация, вижте

https://playground.arduino.cc/Bulgarian/I2CLCDModul



Фиг. 55. Свързване на I²C LCD1602 модул към Arduino Uno

Използвайте Library Manager (вижте Глава 2), за да инсталирате библиотеката LiquidCrystal_I2C, ако я нямате инсталирана – потърсете библиотека, управляваща точния модел управляващ чип, който е на конкретния дисплеен модул LCD1602 (има няколко варианта). Може да използвате *Project07/Project_07.4/Project_07.4.ino*.

```
// Проект 7.4 – LCD1602 2 реда по 16 знака LCD модул
// Свързване към Arduino Uno: A5 to SCL, A4 to SDA, Vin to PWR, GND to GND
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);  // настройте адреса спрямо реално използвания I2C в модула, както и броя знаци и броя редове
void setup() {
  lcd.init();
  lcd.backlight();
  lcd.setCursor(1,0);
  lcd.print("Hello, world!");
  lcd.setCursor(1,1);
  lcd.print("From Arduino!");
}
void loop() {
}
```

Глава 10. Проект 8 – Четене и запис на SD карта и външен EEPROM. RTC.

Едно от основните ограничения при работа с микроконтролери, в частност 8-битовите AVR платки Arduino, е малкият обем на паметта за програмата и постоянните данни, както и липсата на външна памет като стандартна опция (нещо, което е задължително в обикновените компютърни системи), затова се налага използването на външна памет чрез I^2C или SPI интерфейс. Друг потенциален проблем на платките Arduino е липсата на часовник за реално време. Ще бъде демонстрирано как се използват SD карти и RTC модули за добавяне на тази липсваща функционалност.

Карти с памет – SD (microSD)

SD (microSD) картата е удобен начин за добавяне на външна памет към Arduino, като се работи с файлова система FAT16 или FAT32 и картата може да се чете и записва и на компютър или други устройства – използва се SPI за връзка с модула. Повечето библиотеки за Arduino имат ограничение от 2GB обем на картата и поддържат само 8.3 (кратки) имена на файловете. Има разширителни платки с вграден SD или microSD слот – например Ethernet Shield или DataLogger Shield, които спестяват място и улесняват използването на карти с памет. Ако имате модул за SD карти, а искате да работите с microSD карта, използвайте преходник microSD-SD.

Трябва да се внимава за някои различия при работата с файлове спрямо пълноценна OS – работната директория е винаги / и трябва винаги да се използват flush() и close(), защото няма таймаут или shutdown (няма операционна система). Трябва и да се внимава за SS извода по SPI интерфейса. Проблем е и липсата по подразбиране в Arduino на часовник за реално време (RTC), нужен за правилното отразяване на времето на създаване и промяна на файловете.

За повече информация, вижте ***http://arduino.cc/en/Reference/SD***

Фиг. 56. SD карта и модули за SD карти

За работа със SD карта може да използвате вградената библиотека SD, т.е. няма нужда да инсталирате външна библиотека с Library Manager. Трябва само да включите SD.h и SPI.h в кода за работа със SD карта. Важно е да се запомни, че картата трябва да е предварително форматирана на компютър и да бъде във FAT16/FAT32 формат (повечето карти с обем под 4GB се продават форматирани така, но ако е била използвана в телефон или в linux устройство, може да не е в този формат).

Проект 8.1 демонстрира как се работи с файлове – създава и изтрива примерния файл example.txt.

Можете да използвате кода от *Project08/Project_08.1/Project_08.1.ino*.

```
// Проект 8.1 – Създаване и изтриване на файл от SD карта
// SD модула трябва да е закачен на SPI шината MOSI – D11, MISO – D12, CLK – D13 и CS – D4
#include <SPI.h>
#include <SD.h>
File myFile;
void setup() {
 Serial.begin(9600);
 Serial.print("Initializing SD card...");
 if (!SD.begin(4)) {
  Serial.println("initialization failed!");
  return;
 }
 Serial.println("initialization done.");
 if (SD.exists("example.txt")) {
  Serial.println("example.txt exists.");
 } else {
  Serial.println("example.txt doesn't exist.");
 }
 // open a new file and immediately close it:
 Serial.println("Creating example.txt...");
 myFile = SD.open("example.txt“, FILE_WRITE);
 myFile.close();
 // Check to see if the file exists:
 if (SD.exists("example.txt")) {
  Serial.println("example.txt exists.");
 } else {
  Serial.println("example.txt doesn't exist.");
 }
 // delete the file:
 Serial.println("Removing example.txt...");
 SD.remove("example.txt");
 if (SD.exists("example.txt")) {
  Serial.println("example.txt exists.");
 } else {
  Serial.println("example.txt doesn't exist.");
 }
}
void loop() {}
```

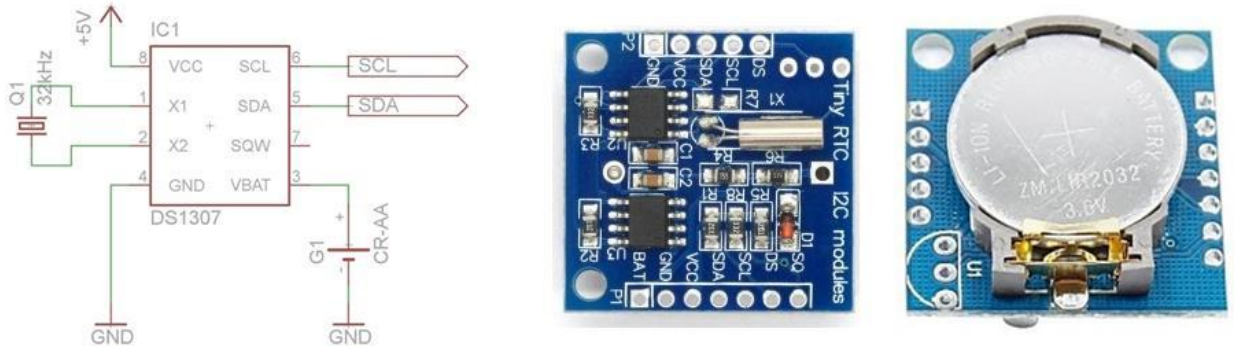
RTC DS1307 – часовник за реално време

DS1307 е евтин и удобен за употреба чип за часовник за реално време, използващ двупроводен I2C протокол за връзка с Arduino. Много често се комбинира с 4К EEPROM на същата платка, за да може да се пазят настройки и данни. 3 волтовата литиева батерия може да държи времето актуално до 5 години. Удобно се комбинира с модулите за SD карти, за да може да се поддържа коректно времето на файловете. Да се внимава при свързване с 3.3 волтови устройства, защото чиповете DS1307 са за 5V – налага се преобразуване на нивата с 1.5kΩ pull-up резистори при свързване с 3.3 волтови платки като Arduino Due и Leonardo. Ако намерите модули с DS1337 или DS3231 – те могат да работят и с 3.3 и с 5 волта нива на сигналите. За повече информация, вижте:

http://www.pjrc.com/teensy/td_libs_DS1307RTC.html и

https://learn.adafruit.com/ds1307-real-time-clock-breakout-board-kit/overview

Показаният по-долу модул (платка) е с добавен EEPROM чип AT24C32 с 4kB памет. Ще бъде демонстрирана работата само с DS1307 RTC чипа. На платката има изводи Vcc, GND, SDA и SCL – те са достатъчни, за да се свърже DS1307 с Arduino платката по I^2C шината (при Arduino Uno – както в предходните проекти, SDA е на A4, SCL е на A5.



Фиг. 57. RTC модул и схема на свързване на DS1307

Използвайте Library Manager, за да инсталирате библиотеката RTClib на Adafruit, ако я нямате инсталирана – тя поддържа и RTC модули, базирани на чиповете PCF8523 и DS323, освен DS1307. Ако желаете да използвате EEPROM чипа AT24Cxxx от комбинираните модули, трябва да инсталирате библиотеката uRTCLib на Naguissa (тя е с друг синтаксис, но включва поддръжка на този EEPROM чип).

Може да използвате *Project08/Project_08.2/Project_08.2.ino*.

```
// Проект 8.2 – DS1307 RTC модул с I2C
// Свързване към Arduino Uno: A5 to SCL, A4 to SDA, Vin to PWR, GND to GND
#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 rtc;
char daysOfTheWeek[7][12] = {"Sunday“, "Monday“, "Tuesday“, "Wednesday“, "Thursday“, "Friday“, "Saturday"};
void setup () {
  rtc.begin();
  Serial.begin(9600);
```

```
}
void loop () {
   DateTime now = rtc.now();
   Serial.print(now.year(), DEC);
   Serial.print('/');
   Serial.print(now.month(), DEC);
   Serial.print('/');
   Serial.print(now.day(), DEC);
   Serial.print(“ (");
   Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
   Serial.print(") ");
   Serial.print(now.hour(), DEC);
   Serial.print(':');
   Serial.print(now.minute(), DEC);
   Serial.print(':');
   Serial.print(now.second(), DEC);
   Serial.println();
   delay(3000);
}
```

Интерес предизвиква въпроса как се сверява RTC в микроконтролер без клавиатура и дисплей? Един възможен подход е, да се направи проверка дали е сверен и ако не е, да се вземе информация за времето на последната компилация от самата програма. Това може да стане със следният програмен фрагмент, поставен на подходящо място в кода (най-добре е да се зарежда малка програма, еднократно сверяваща RTC и впоследствие да се разчита на батерията и на точността на кварцовия генератор): rtc.adjust(DateTime(F(__DATE__), F(__TIME__))); – този код взима датата и времето на създаване на програмата и ги записва в RTC. При смяна на батерията, обаче, ще се върне това време – трябва да се отчете това.

Използване на вградения EEPROM

Ще бъде демонстрирана работата с вградения в Arduino AVR микроконтролерите EEPROM (няма EEPROM в Arduino Due). В различните микроконтролери има различен обем EEPROM: 1024 байта в ATmega328, 512 байта в ATmega186, 4096 байта (4KB) в ATmega1280/ATmega2560 и 1024 байта емулиран EEPROM в Genuino 101. EEPROM паметта издържа до 100 000 презаписа и може да се използва за запаметяване на настройки, променливи и данни, които да остават без промяна при рестартиране или загуба на захранване. За повече информация, вижте

https://www.arduino.cc/en/Reference/EEPROM

Има клас EEPROM с функции за достъп до EEPROM паметта. Най-лесно се използва с емулиращия масив от байтове оператор EEPROM. Трябва да се внимава да не се чете извън достъпния обем. Понеже броят презаписвания, макар и голям, не е безкраен, се препоръчва, когато се прави чест презапис, да се проверява дали стойността е различна

и само тогава да се записва. С тази цел има и функция update(), тя извършва запис, само ако има разлика в записваната и вече записаната стойност. Удобни са и функциите put() и get() – те четат и записват цял обект (структура) наведнъж, спестявайки работата с отделни байтове един по един. При работа с EEPROM паметта трябва да се има впредвид и бавния достъп при апис – записа на един байт отнема 3.3 ms (максимална скорост на запис 300 байта в секунда). Все пак, въпреки малкия си обем и ниската си скорост, вградения EEPROM е добър избор за записване на важни настройки в по-големите проекти, без използване на външни чипове или използване на модул за SD карта.

Може да използвате *Project08/Project_08.3/Project_08.3.ino*.

```
// Проект 8.3 – Проверка за работоспособност на вградения EEPROM
#include <EEPROM.h>
void setup(){
  Serial.begin(9600);
  byte val;
  val = EEPROM[ 0 ]; // Прочитане на клетка 0
  EEPROM[ 0 ] = val; // Запис в клетка 0
  if( val != EEPROM[ 0 ] ){ // Сравняване отново с клетка 0
    // клетка 0 е повредена – взимат се някакви мерки...
    Serial.println("EEPROM is defective!");
  }
}
void loop(){ }
```

Глава 11. Проект 9 – Ethernet модул с W5100 – връзка с Интернет и LAN

Ethernet е технология за свързване и мрежов протокол за LAN и MAN, създадена през 1980 г. и превърнала се в стандарта IEEE 802.3 през 1983 г. Оттогава насам е усъвършенствана на няколко пъти с цел поддръжка на по-високи скорости (bit rate) и по-дълги разстояния. С времето Етернет постепенно измества конкурентните жични технологии за LAN като Token Ring, Token Bus, FDDI и ARCNET. При съвременните LAN основната алтернатива вече не е по LAN кабели, а безжичния стандарт IEEE 802.11, известен също като Wi-Fi. [Wikipedia – Ethernet]

Протоколът за връзка, който се използва, е предимно TCP/IP (Transmission Control Protocol / Internet Protocol), това е концептуален модел на семейство от протоколи за комуникация между компютрите, който се използва в Internet и в почти всички други съвременни компютърни мрежи. Този модел се състои от много протоколи, но тъй като ключова роля имат протоколите TCP и IP, името се определя от тях. Моделът TCP/IP е създаден през 1980 г. заради необходимостта от единен начин за комуникация между компютрите, като по този начин предоставя възможност мрежите да бъдат свързвани помежду си. В модела TCP/IP информацията се пренася под формата на пакети. Всеки пакет се състои от 2 части – заглавна част (от английски header) и данни. Често е наричан Интернет модел. [Wikipedia – TCP/IP]

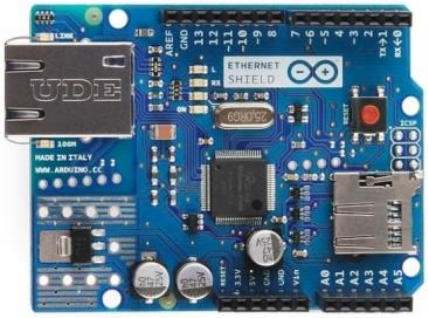
Ще бъде демонстрирано как от Arduino платка с помощта на Ethernet модул или разширителна платка с такъв чип, може да се направи комуникация по TCP/IP между Arduino платката и Internet или друга машина или друга Arduino платка в локална мрежа.

Wiznet W5100 модул за Ethernet

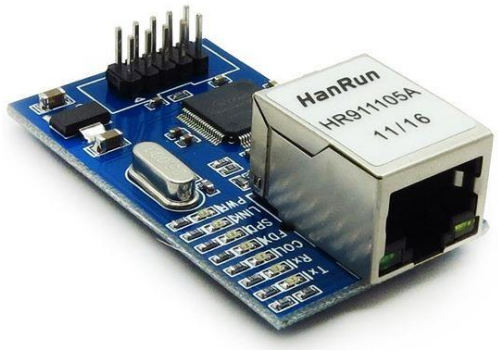
Wiznet W5100 е евтин и удобен за използване чип за управление на Ethernet връзка в Arduino – използва SPI. Наличен е, както на отделни модули, така и комбиниран с микроконтролер (Arduino Leonardo ETH – рядък модел платка с Arduino и Ethernet) или на разширителни платки. Често е комбиниран с (micro)SD слот в модулите на негова основа. За него има много библиотеки, освен стандартната, което го прави много подходящ за използване в проекти, свързващи се с Internet – сървери, логери, клиенти, сигнализации и аларми, както и модерното напоследък направление IoT (Internet Of Things). В новите версии се използва W5500.

За повече информация за библиотеката за работа с W5100, вижте:

http://arduino.cc/en/Reference/Ethernet



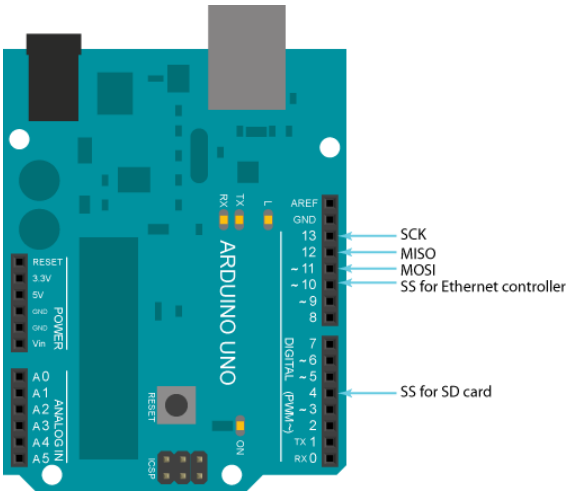
Фиг. 58. Arduino Ethernet Shield – разширителен модул



Фиг. 59. Ethernet модул с Wiznet W5100

Библиотеката Ethernet управлява W5100, а Ethernet2 – W5500 чипове, всички функции са еднакви. Смяната на библиотеката позволява портиране на код, написан за модул с W5100 към използване на модул с W5500.

Arduino комуникира с модулите W5100/W550 по SPI шината. При използване на разширителен модул (като Arduino Ethernet Shield или съвместим), това са изводи 11, 12 и 13 на Arduino Uno и 50, 51 и 52 на Mega. И на двете платки извод 10 е свързан към SS на модула (т.е. на Mega не е използван хардуерния SS изход на D53, но не трябва да се използва D53, защото няма да работи SPI интерфейса). Ако се използват отделни модули с W5100/W5500, трябва да се спази това свързване за съвместимост с вградените Ethernet библиотеки.



Фиг. 60. Свързване на Ethernet Shield към Arduino Uno

Ethernet библиотеката предоставя няколко класа за управление на модула и връзка с мрежата: Ethernet, IPAddress, Server, Client и EthernetUDP. В проект 9.1 е показан малък примерен web сървър, четящ показанията на сензор, свързан към аналогов вход.

Уточнение: в проект 9.1 не се използва SD картата. Ако модулът, който използвате има слот за (micro)SD карта и тя не се използва в кода, може да се получи блокиране на кода, защото изход D4 се използва като SS (с активно ниско ниво от четеца на SD карти и когато не се използва, е настроен като INPUT по подразбиране). Има 2 възможни решения – извадете SD картата от слота или добавете в setup() функцията на проекта pinMode(4, OUTPUT); digitalWrite(4, HIGH); с цел избягване на този проблем.

Може да използвате *Project09/Project_09.1/Project_09.1.ino*.

```
// Проект 9.1 – Web Server с Ethernet Shield (или W5100 модул)
// Показват се стойностите на аналоговите входове A0-A5
// Изисква се W5100 модул на изходи 10, 11, 12 и 13
#include <SPI.h>
#include <Ethernet.h>
byte MACAddress[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; // Изберете си уникален MAC адрес
IPAddress ServerIP(192, 168, 1, 167); // и уникален за вашата LAN IP адрес
EthernetServer WebServer(80); // Инстанция на сървера на порт 80 (HTTP)
void setup() {
  Ethernet.begin(MACAddress, ServerIP); // стартиране на връзката с LAN
  WebServer.begin();  // стартиране на web сървера
}
void loop() {
  EthernetClient client = WebServer.available(); // изчакване за клиенти
  if (client) { // свързан клиент
    boolean LineIsBlank = true;
    while (client.connected()) { // получаване на заявката
      if (client.available()) {
        char c = client.read();
        if (c == '\n' && LineIsBlank) { // празна линия е край на HTTP Req
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");
          client.println("Refresh: 5");  // refresh 5 sec
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          for (int AI = 0; AI < 6; AI++) { // чете всички аналогови входове
            int sensorValue = analogRead(AI);
            client.print("Analog input ");
            client.print(AI);
            client.print(“ is ");
            client.print(sensorValue);
            client.println("<br />");
          }
          client.println("</html>");
          break;
```

```
        }
        if (c == '\n') {
          LineIsBlank = true;
        } else if (c != '\r') {
          LineIsBlank = false;
        }
       }
      }
      delay(1); // изчакване за web браузъра да реагира
      client.stop(); // затваряне на HTTP връзката
     }
    }
```

Неудобство на този примерен проект е, че трябва да задавате „ръчно“ MAC адрес и IP адрес. Внимавайте да нямате няколко платки с еднакъв MAC адрес в една и съща LAN мрежа.

Media Access Control адрес или накратко MAC адрес е уникален идентификатор на производителя на мрежовите адаптери по технологията Ethernet. Използва се в едноименния протокол Media Access Control protocol, който е част от каналния слой на OSI модела. MAC адресите се използват и при други мрежови технологии, базиращи се на IEEE 802 стандарта. При изписване представлява дванадесет символа в шестнадесетична бройна система. Пример: 00-17-31-9B-00-7E. Първите шест символа дават информация за производителя на хардуера, а останалите шест са уникални, и се предполага че няма два хардуера с еднакъв MAC адрес в една локална мрежа. Много често изглежда така — 6 символа име на производител, следвано от останалите 6 символа. MAC адреса се използва от комутаторите (суичовете) за определяне на пътя, по който да минават пакетите информация, за да достигнат крайната точка. [Wikipedia – MAC адрес]

Библиотеката Ethernet поддържа и DHCP и DNS, може да видите как се използват в описанието на метода begin() на класа Ethernet:

https://www.arduino.cc/en/Reference/EthernetBegin

Използване на NTPClient библиотека за точно време

Може да видите и реално използване на DHCP, DNS и UDP в примерния проект 9.2, илюстриращ тяхната употреба за имплементиране на NTP клиент за точно време. Използвайте Library Manager (вж. Глава 2), за да инсталирате библиотеката NTPClient от Fabrice Weinberg. Може да използвате файла *Project09/Project_09.2/Project_09.2.ino*.

```
// Проект 9.2 – NTP клиент с W5100 Ethernet Shield
// Изисква се W5100 модул на изходи 10, 11, 12 и 13
// Използва time.nist.gov и DHCP, UDP, DNS
// Базиран на примера от https://www.arduino.cc/en/Tutorial/UdpNtpClient
// и използване на NTPClient от Fabrice Weinberg https://github.com/arduino-libraries/NTPClient
```

```
// За информация за NTP протокола вижте
http://en.wikipedia.org/wiki/Network_Time_Protocol
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>
#include <NTPClient.h>
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // задайте уникален MAC за ва-
шата LAN Мрежа
unsigned int localPort = 8888;
EthernetUDP Udp; // Udp инстанция

NTPClient timeClient(Udp); // инстанция на клиента за NTP
// Може да смените time server и часовия пояс (отместване в секунди) и интервал
за ъпдейт (в мс).
// NTPClient timeClient(Udp, "europe.pool.ntp.org“, 3600, 60000);

void setup() {
  Serial.begin(9600);
  while (!Serial) ; // само за Leonardo и Due

  // стартиране на Ethernet, UDP и timeClient
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    for (;;) ; // безкраен цикъл – reset бутона за нов опит
  }
  Udp.begin(localPort);
  timeClient.begin();
}

void loop() {
  timeClient.update(); // сверява от NTP сървъра

  Serial.println(timeClient.getFormattedTime());

  delay(10000); // 10 секунди изчакване
  Ethernet.maintain(); // опресняване на DHCP ако има нужда
}
```

В заключение, може да се спомене наличието на разширителни платки WiFi Shield и WiFi Shield 101 за достъп до WiFi мрежи, както и GSM Shield и 3G Shield, позволяващи

мобилен достъп до Интернет по GPRS/3G – изисква се карта от мобилен оператор с активирани мобилни данни.

Повечето от тези модули са с относително висока цена, затова се препоръчва използването на модули с ESP8266 за WiFi проекти. ESP8266 са със собствен процесор, но може да се програмират и чрез Arduino IDE.

Описанието на тези модули би могло да бъде разгледано в следващо издание на това ръководство.

Глава 12. Проект 10 – Метеостанция

Общи понятия за метеостанция

Като пример за завършен и реално използваем уред може да бъде избрано устройството за измерване на околната среда, наречено домашна метеостанция. На пазара има голямо разнообразие от много фирми-производители, но повечето модели са с фиксирани функции, няма голяма яснота по какви алгоритми се правят предсказанията на времето (ако имат тази функция) и често не е указано на какъв клас точност отговарят използваните в тях сензори. [Wikipedia – Метеорологична станция]

Предизвикателство за любознателния експериментатор е създаването на собствена метеостанция. Благодарение на гъвкавостта на платформата Arduino и използвайки показаните в предходните проекти сензори, протоколи за връзка и управление на дисплеи, лесно може да се проектира и разработи устройство, събиращо и обработващо данни за околната среда, предоставящо информацията на буквено-цифров дисплей или по Интернет.

Една класическа домашна метеостанция обикновено включва събиране и обработване на следната информация:

- Външна и вътрешна температура (вътрешна – в стаята);
- Външна и вътрешна относителна влажност на въздуха;
- Барометрично налягане;
- Качество на въздуха – измерват се нивата на въглероден окис, алкохол, бензин, дим, бутан, пропан и др. газове;
- Точно (актуално) текущо време и дата – от тях, ако е зададена информация за географското положение, може да се пресметнат времената на изгрев и залез, както и фазите на луната. Така също, предизвикателство е да има автоматично сверяване на времето по Интернет (NTP сървър) или от атомен часовник DCF77 по радиовълни (изисква се специален AM приемник на дълги вълни) [Wikipedia – DCF77];
- Опционално – осветеност в Lux и ниво на UV лъчите;
- Опционално може да се добавят сензори за нивото на падналите валежи, за скорост и посока на вятъра (анемометър), за далечина на видимост, степен на облачност – поради значителната им цена и трудностите по монтажа им, в този проект няма да бъдат включени такива сензори, но ще бъде показано как биха могли да се използват.

Основна трудност предизвиква факта, че някои от сензорите трябва задължително да бъдат монтирани извън помещението на подходящо място – това налага да се използва безжична връзка с тях, оттук идва необходимостта сензорите да се реализират с помощта на допълнителен микроконтролер и батерийно захранване. За нуждите на това учебно помагало, с цел опростяване на проекта, ще бъде използвана кабелна връзка с тези сензори, но ще бъде показано как може да се реализира безжичната връзка с тях и външното им захранване. [Arduino Atmospheric'12]

Ако решите да реализирате автоматично сверяване на времето, може да използвате два подхода – по Интернет с NTP клиент (Проект 9.2) и по радиовълни от атомния

часовник на DCF77 станцията в Германия. Сверяването по Интернет е най-лесно, но не винаги има постоянна връзка с Интернет (може устройството да работи в отдалечен район или да е мобилно), а използването на GSM Shield + карта за достъп до мобилен Интернет е сравнително скъпо решение, свързано и с плащане на месечна такса.

Повече информация за DCF77 и Arduino може да намерите на следния адрес: ***https://playground.arduino.cc/Code/DCF77***

В тази версия на проекта за метеостанция, с цел опростяване на кода и схемата, няма да бъде реализирана опцията за автоматично сверяване на времето – ще се използва RTC DS1307 от Проект 8.2, но при желание, може да добавите кода от Проект 9.2 за автоматично сверяване с NTP сървър. Няма да се използва и DCF77 сверяване, пак с цел икономия на място и памет.

Избор на компоненти за метеостанцията

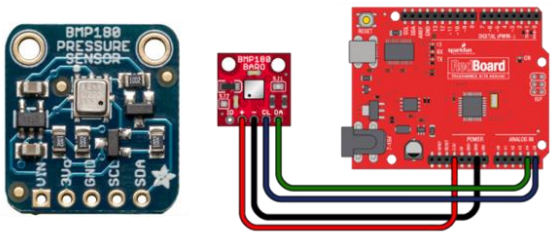
С цел опростяване на проекта и намаляване на разходите са избрани следните сензори, свързани към Arduino Uno с използване на дисплей LCD1602 с I^2C – изобразяващ 2 реда по 16 символа:

- Датчик за качеството на въздуха MQ-135;
- Барометричен датчик BMP180;
- Датчик за влажност и температура DHT22;
- LDR фоторезистор за околна светлина.

Употребата на датчика DHT22 беше показана в Проект 6.2, на LDR фоторезистор в Проект 5.2 и на дисплей LCD1602 с I^2C в Проект 7.4, затова тук ще бъде обяснено само използването на датчиците BMP180 и MQ-135.

BMP180 е подобрен и по-евтин вариант на популярния сензор за барометрично налягане BMP085. Много е удобен за показване на атмосферното налягане, а след калибриране на морското равнище и за определяне на надморската величина, което го прави подходящо за мобилни метеостанции или уреди за навигация в самолети и вертолети (по-точно в модерните напоследък дронове). Използва I^2C. Може да използвате код и примери с BMP085 от Интернет за работа с BMP180 (съвместими са). Инсталирайте библиотеката Adafruit BMP085 library (версия 1.0.0 без unified sensor) – използвайте Library Manager (Глава 2).

За повече информация, вижте ***https://learn.adafruit.com/bmp085*** и ***http://playground.arduino.cc/Bulgarian/BMP085***

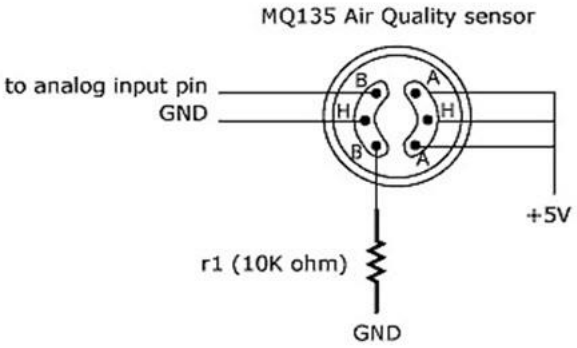


Фиг. 61. BMP180 сензор за барометрично налягане

Сензорът MQ-135 измерва качеството на въздуха, по-конкретно нивата на въглероден окис, алкохол, бензин, дим, бутан, пропан и др. газове. Ако в околния въздух

има по-висока концентрация от тези вещества, то може да се каже, че въздуха е замърсен. Сензорът измерва промяната в концентрацията и реагира с промяна на изходното напрежение – то е право пропорционално на концентрацията на тези химически вещества във въздуха (не е строго линейна зависимост, както и за различните газове е различно нивото, но за нашите цели може да се приеме, че то е линейно и еднакво за всички замърсители).

Фиг. 62. Сензор за качество на въздуха MQ-135



Фиг. 63. Свързване на сензор за качество на въздуха MQ-135

Датчикът MQ-135 използва вътрешно подгряване (има нагревателен елемент, затова и показанията са валидни 2 минути след първоначалното включване, за да се достигне нужната температура) и консумира около 1W мощност при 5V захранване, затова е важно да се подават 5V от външен източник, а не от +5V изход на Arduino Uno. Може да се използва, например, външен 7805 стабилизатор на напрежение или отделен 2A/5V захранващ модул, от който да се подава напрежение и на Arduino платката и на сензора MQ-135.

Изменението на напрежението на изхода на сензора се подава на зададен аналогов вход на Arduino платката. Предварително трябва да се настроят прагови нива в кода на програмата за метеостанцията и когато те се надвишат, да се реагира със съобщение на дисплея, че въздуха е замърсен (прави се проверка в три точки – 4 състояния).

Проект 10.1 използва наученото дотук, за да реализира проста метеостанция, показваща температура и влага, налягане и надморска височина, степен на замърсеност на въздуха и ниво на околната светлина през 4 секунди на екрана на LCD1602.Може да използвате файла

Project10/Project_10.1/Project_10.1.ino.

```
// Проект 10.1 – Метеостанция
// LCD1602 и BMP180 към I2C SDA A5, SCL A4
// LDR към A3
// MQ-135 към A2
// DHT22 към A1 (аналоговите входове са и цифрови)
// за по-лесно ползване на рейка или лентов кабел към A1-A5 изводите
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SimpleDHT.h>; // библиотека за сензорите DHT
#include <Adafruit_BMP085.h>
// свързване
const int MQ135pin = A1;
const int DHT22pin = A2;
const int LDRpin = A3;
// обекти
LiquidCrystal_I2C lcd(0x27,16,2); // настройте адреса спрямо реално използвания I2C в модула, както и броя знаци и броя редове
SimpleDHT22 dht22; // инициализираме инстанция на променливата dht22 за връзка със сензора
Adafruit_BMP085 bmp180; // за BMP180
// променливи
float hum; // влага
float temp; // температура
int AirQuality; // MQ-135 ст-ст
int Lighting; // LDR ст-ст
// константи за качеството на въздуха с MQ-135
const int low = 300;
const int med = 500;
const int high = 700;

int bmpready = 1;
int aDelay = 4000; // закъснение при смяна на показанията

void setup() {
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Sensors are");
  lcd.setCursor(0,1);
  lcd.print("warming-up!");
```

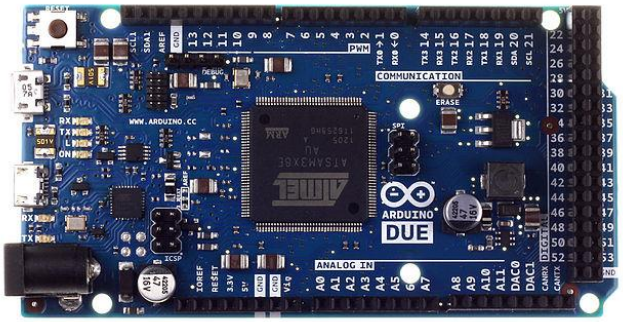
```
  delay(2*60*1000); // 2 минути подгряване на сензора MQ-135
   lcd.clear();
 }

 void loop() {
  // четене на DHT22
  int dhtready=0;
  if (dht22.read2(DHT22pin, &temp, &hum, NULL) == SimpleDHTErrSuccess) { dhtready
= 1; }
  if( dhtready ){
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Temp(*C)= ");
        lcd.print(temp);
        lcd.setCursor(0,1);
        lcd.print("Humidity(%) = ");
        lcd.print(hum);
        delay(aDelay); // показване 4с
  }
  else { // DHT not ready
        lcd.setCursor(0,0);
        lcd.print("NO DATA!");
        lcd.setCursor(0,1);
        lcd.print("Check DHT22!");
        delay(2000); // изчакване 2с при грешка в DHT22
  }
  // четене BMP180
  if (!bmp180.begin()){
        bmpready = 0;
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("BMP180 sensor");
        lcd.setCursor(0,1);
        lcd.print("not found!");
        delay(1000);
  }
  if ( bmpready ) {
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("----Pressure----  ");
        lcd.setCursor(0,1);
        lcd.print(bmp180.readPressure());
```

```
            lcd.print("  Pascal");
            delay(aDelay);
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("----Altitude----");
            lcd.setCursor(0,1);
            lcd.print(bmp180.readAltitude(101500));
            lcd.print("  meter");
            delay(aDelay);
    }
    AirQuality = analogRead(MQ135pin); // четене на сензора MQ-135
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("  Air Quality:");
    lcd.setCursor(0,1);
    if(AirQuality==0){
            lcd.print("  Sensor Error!");
    }
    if(AirQuality<=low && AirQuality>0){
            lcd.print("      GOOD");
    }
    if(AirQuality>low && AirQuality<med){
            lcd.print("  GETTING BAD");
    }
    if(AirQuality>=med && AirQuality<high){
            lcd.print("   VERY POOR");
    }
    if(AirQuality>=high){
            lcd.print("     WORST");
    }
    delay(aDelay);
    Lighting = analogRead(LDRpin);
    lcd.setCursor(0,0);
    lcd.print("  Lighting:");
    lcd.setCursor(0,1);
    lcd.print(Lighting);
    delay(aDelay);
  }
```

Глава 13. Arduino платки с 32-битови контролери

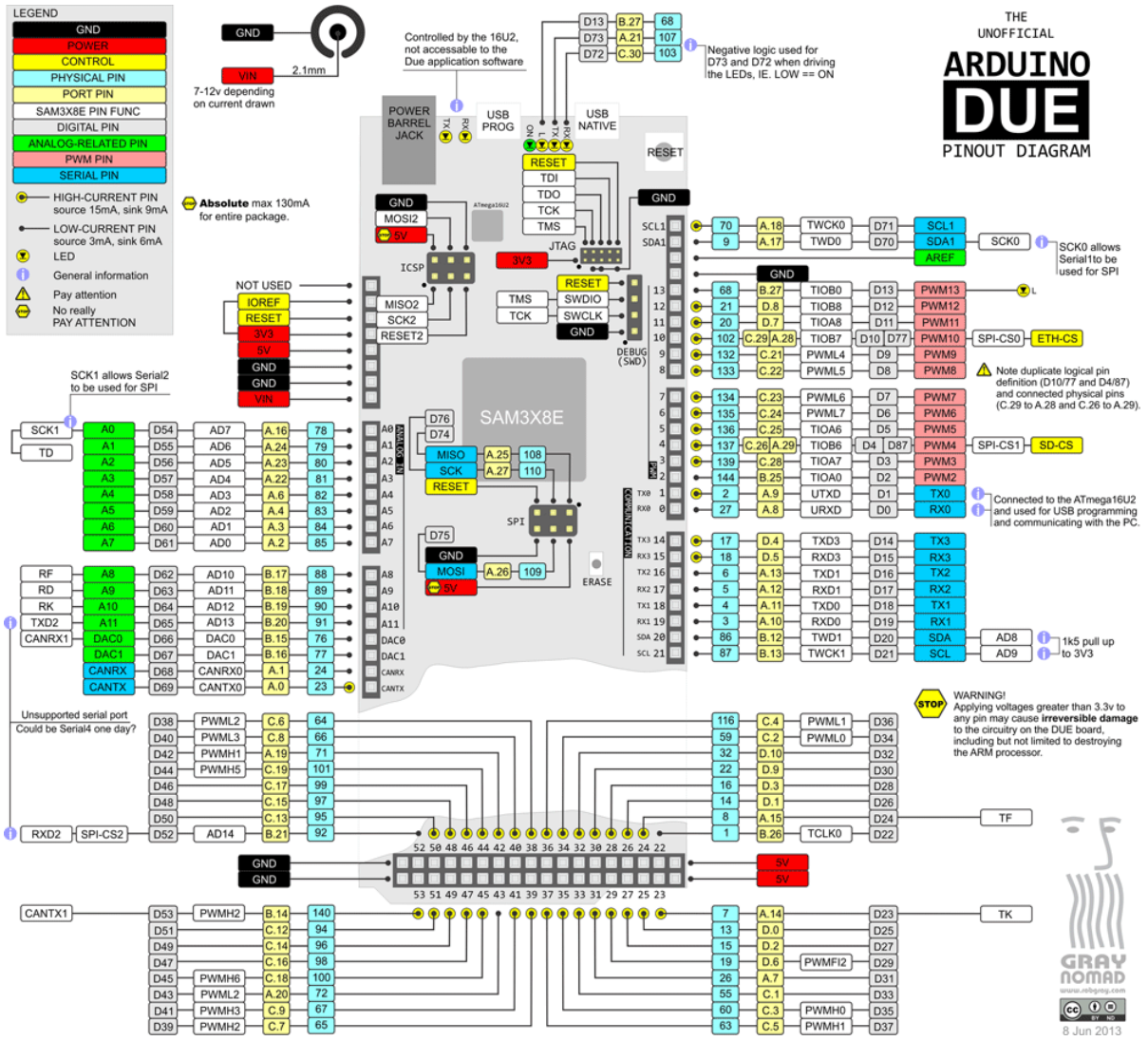
В последно време цената на устройствата с ARM микроконтролери спадна значително, това не можа да остане настрана от Arduino общността и през 2012 година беше представена платката Arduino Due, базирана на 32-битовия ARM Cortex-M3 Atmel SAM3X8E микроконтролер. Платката е подобна на Arduino Mega, но с някои съществени различия – работното напрежение е 3.3V, срещу 5V в Mega; има 2 USB порта (единият е класическия сериен порт, а вторият е USB OTG, директно свързан с микроконтролера); DAC изходи; CAN входове; много нови функции и възможности, както и по-голям обем памет и по-високо бързодействие. [Wikipedia – Arduino]



Фиг. 64. Платка Arduino Due

Основни характеристики на Arduino Due

- Микроконтролер: AT91SAM3X8E – ARM Cortex M3 процесор
- Работна честота: 84 MHz
- Работно напрежение: 3.3V
- Захранващо напрежение (препоръчително): 7-12V
- Захранващо напрежение (абсолютни граници): 6-20V
- Цифрови I/O пинове: 54 от тях 12 PWM изхода с 8 или 12 бита
- Аналогови входни пинове (ADC): 12 x 12-битови
- Аналогови изходни пинове (DAC): 2 x 12-битови
- Общ изходен ток за всички I/O: 130mA
- Максимален ток на 3.3V 800mA, на 5V 800mA
- Flash памет: 512KB (2 блока по 256KB)
- SRAM: 96KB (64 + 32 KB), 9 x 32 битови таймери
- Reset и Erase бутони, светодиод на изход 13
- 6 x DMA,1 x CAN, 2x TWI, 1 x SPI, 4 x UART, 3 x USART



Фиг. 65. Карта на изводите на Arduino Due

Сравнение с Arduino Uno платките

- Микроконтролерът AT91SAM3X8E – ARM Cortex M3 е 32 битов, обработва 4 байта, вместо 1 на такт, както AtMega328P
- Работна честота: 84 MHz, вместо 8 до 16 MHz
- Работно напрежение: 3.3V, вместо 5V (по-икономично)
- Цифрови I/O пинове:54 (12 PWM), вместо 14 (6 PWM)
- 12 бита вместо 8/10 бита за PWM и аналоговите вх./изх.
- Аналогови входове (ADC): 12 12-битови – Uno има 6 10 битови
- Аналогови изходи (DAC): 2 12-битови – Uno няма
- Flash памет: 512KB (2 блока по 256KB) – Uno 32KB
- SRAM: 96KB (64 + 32 KB) – Uno 2KB
- DMA 6 броя – в Uno няма DMA
- няма EEPROM, докато в UNO има 1KB EEPROM

Нови библиотеки, специфични за Arduino Due

- USBHost – позволява Due да функционира като USB Host. Не поддържа устройства зад хъбове – а някои устройства имат вграден USB Hub.
- SerialUSB – използва native USB Serial порта на SAM3XE в Due.
- Keyboard – изпраща клавишни комбинации вместо USB клавиатура към прикачен PC.
- Mouse – изпраща позициите на курсора и бутоните вместо USB мишка към прикачен PC.
- Scheduler – кооперативен неблокиращ планировчик за паралелни процеси (цикли), изпълняващи се заедно с главния loop. Методи: startloop за деклариране на паралелен цикъл и yield за предаване на управлението към другите процеси.
- Audio – използва DAC0/DAC1 изходите за просвирване на .wav файлове от SD карта.

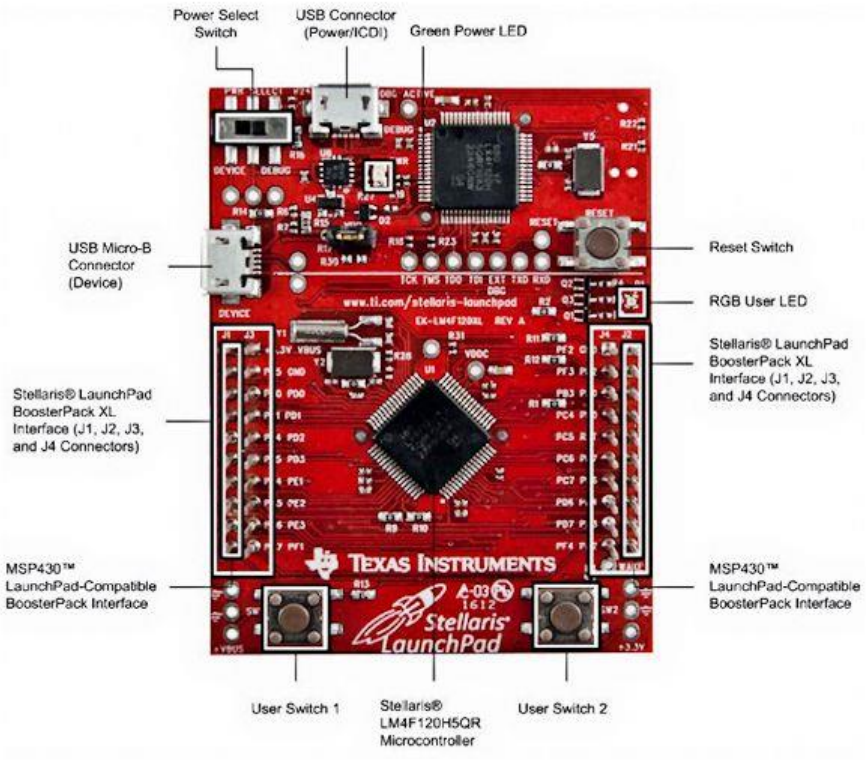
Има налични и много други библиотеки от независими разработчици, както и доста недокументирани функции от самите glibc библиотеки, когато изберем тип на платката да е Arduino Due в Arduino IDE средата.

Като цяло, голяма част от кода, наличен за Arduino платки с AVR микроконтролери (каквито са Uno и Mega), се компилира и изпълнява безпроблемно без промени на Arduino Due платките. Средата Arduino IDE при избиране на платка Arduino Due, сменя библиотеките с такива, пригодени за ARM процесора на Due. Все пак, по-специфичен код, взаимодействащ с някои липсващи в ARM хардуерни елементи от AVR чиповете, както и код, който е време-зависим и е написан с използване на точната честота на изпълнение на инструкциите в AVR, ще има проблеми при изпълнението на Due. Други потенциални проблеми са с липсващите PROGMEM модификатори (няма avr\pgmspace.h), поради различната организация на флаш паметта.

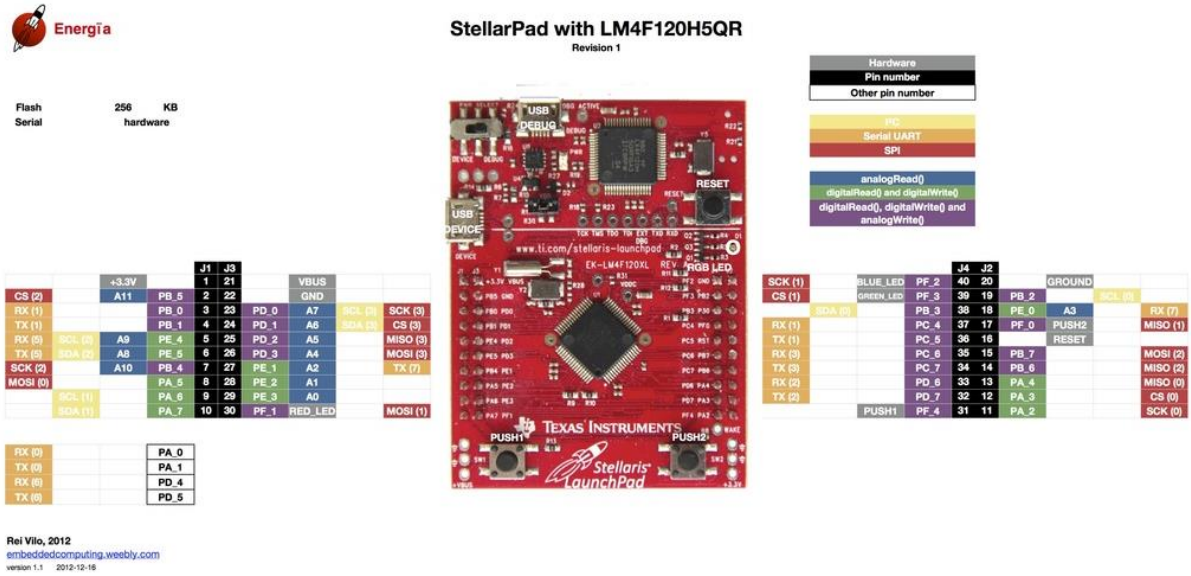
Други платки с 32-битови ARM или Intel микроконтролери от фамилията Arduino са Arduino 101, Arduino Zero, Arduino Yún, Arduino MKR1000 и др.

Може да се споменат и платките на Texas Instruments от серията Launchpad, които не са съвместими с Arduino по изводи, използват други микроконтролери, но за тях има разработен клонинг на Arduino IDE средата – Energia IDE (вижте ***http://energia.nu***), която осигурява интересна съвместимост с Arduino кода – това се постига с помощта и на преработени библиотеки от Arduino (предимно Wiring съвместимите). На избираемите дисциплини, на които е посветено това учебно помагало, са разглеждани и използвани, освен Arduino платки и платки TI Stellaris (в момента новото им име е TI Tiva Launchpad), както и платки TI MSP430 Launchpad.

Разликата с Arduino Due на хардуерно ниво е значителна. Използван е микроконтролер LM4F120H5QR (TM4C123GH6PM), който е от тип ARM Cortex-M4 с вградени FPU и DSP. Изводите на платките Stellaris LaunchPad не са съвместими с платките за разширения на Arduino. Имат RGB LED за тестване и 2 потребителски бутона, за разлика от Arduino. Имат съвместимост с другите платки на TI от серията LaunchPad – MSP430, които имат голяма популярност в САЩ (Arduino в началото е по-популярно в Европа, но напоследък чрез Genuino платките, става популярно и в САЩ).



Фиг. 66. Платка TI Stellaris LaunchPad



Фиг. 67. Карта на изводите на Stellaris Launchpad

Предимството на платките LaunchPad в контекста на това учебно помагало е само в използването на Energia IDE, с отчитане на големия брой различия. Можете да използвате голяма част от натрупания опит в програмирането на Arduino за работа с тези платки. Разбира се, това дава и по-голяма гъвкавост и диверсификация при реализацията на проектите.

Заключение

Авторът се надява настоящото учебно помагало да е предизвикало в читателите интерес към необхватния свят на Arduino. За съжаление, поради ограничения обем, не е възможно да бъдат изложени всички проекти и материали, които са разработени по време на провеждането на избираемите дисциплини „Програмиране в среда Arduino“. Отчитайки това, е направен опит навсякъде да бъдат поставени достатъчно упътвания за намиране на допълнителна информация от Интернет за конкретния проблем с препратки към Интернет сайтове. В Приложението е добавен списък с интересни адреси на сайтове и форуми за Arduino, както и множество допълнителни указания за покупка и избор на платки и компоненти, заедно с информация за основните понятия в електрониката.

Някои възможни насоки за развитие на курсовете, които са планирани да бъдат разработени в близко бъдеще, са в областта на:

- Програмиране на Raspberry Pi;
- Роботика и автоматика с микроконтролери;
- 3D-печат (разработка на 3D принтери с микроконтролери);
- IoT системи (Internet of Things) и „умен дом“;
- Изкуствен интелект с микроконтролери;
- Разпознаване на говор (реч) с микроконтролери;
- Дронове и други летателни апарати (отделно от роботиката);
- и много други…

Приложение 1. Анотации на избираемите дисциплини „Програмиране в среда Arduino“

Избираемите дисциплини (три части) са разработени и се провеждат от доц. д-р Светослав Енков след 2011 г. във ФМИ на ПУ и са предназначени за студентите бакалаври, редовно и задочно обучение на ФМИ на ПУ от специалностите „Информатика“, „Бизнес Информационни Технологии“, „Софтуерни Технологии“ и „Софтуерно Инженерство“. Засиленият интерес към тях доведе до написването на това ръководство.

По-долу са представен анотациите на тези избираеми дисциплини, ако читателят открие нещо интересно, което липсва (поради ограничения обем) в тази книга, може да се свърже с автора (доц. д-р Светослав Енков) за повече информация и съдействие.

Част 1 – Програмиране в среда Arduino

Анотация: Целите на курса са студентите да придобият основни познания за програмирането на 8-битови микроконтролери Atmel AVR в среда Arduino и да се запознаят с принципите на създаването на прости схеми и устройства с тези контролери. Arduino е платформа за електронно прототипиране с отворен код, базирана на гъвкави и лесни за използване хардуер и софтуер. Системата общува със средата, получавайки сигнали от множество сензори и датчици и може да взаимодейства с нея, контролирайки светлини, дисплеи, двигатели и други устройства. Микроконтролерът в платката се програмира с Arduino език за програмиране (базиран на Wiring и подобен на C++) в развойна среда Arduino (базирана на Processing). За нуждите на курса са осигурени платки Duemilanove и Mega2560, както и множество сензори, дисплеи, бутони и прототипни платки, с цел практическото усвояване и тестване на получените знания.

Съдържание на курса (по седмици)

1. Обзор на микроконтролерите и развойните платки. Представяне на хардуерната среда Arduino. Запознаване със софтуерната среда Arduino IDE.
2. Работа с цифровите входове и с аналоговите изходи. Принципи на PWM регулирането. Обща структура на софтуерен проект в Arduino средата. Секция Setup и секция Loop. Зареждане на проекта в паметта на контролера.
3. Първи реално работещ проект – управление на светодиод, зумер и 16x2 дисплей.
4. Работа с таймери, прекъсвания, управление на двигатели и релета. Роботи и други изпълнителни устройства.
5. Основни принципи на връзката с потребителя при микроконтролерите – бутони, клавиатури, IR дистанционно управление и 320x240 сензорен дисплей.
6. Управление на паметта, работа с външна памет – SD карта и външна RAM/EEPROM. Комуникация по I2C протокол. Безжични комуникации по 433MHz ASK протокол. Управление на LAN модул и връзка с Интернет.
7. Задаване на индивидуален проект за разработване. Обсъждане на основните трудности и проблеми. Начален sketch на проекта.
8. Работа по проекта. Отстраняване на възникналите проблеми. Изготвяне на хардуерен прототип с прототипна платка.

9. Довършване и тестване на проекта. Писане на техническа документация.
10. Представяне и защитаване на проекта. Крайна оценка.

Част 2 – Програмиране в среда Arduino за напреднали

Анотация: Целите на курса са студентите да придобият допълнителни познания за програмирането на 8-битови микроконтролери Atmel AVR в среда Arduino и да се запознаят с принципите на създаването на по-сложни схеми и устройства с тези контролери. Arduino е платформа за електронно прототипиране с отворен код, базирана на гъвкави и лесни за използване хардуер и софтуер. Системата общува със средата, получавайки сигнали от множество сензори и датчици и може да взаимодейства с нея, контролирайки светлини, дисплеи, двигатели и други устройства. Микроконтролерът в платката се програмира с Arduino език за програмиране (базиран на Wiring и подобен на C++) в развойна среда Arduino (базирана на Processing). За нуждите на курса са осигурени платки Duemilanove и Mega2560, както и множество сензори, дисплеи, бутони, шасита, серво механизми и прототипни платки, с цел практическото усвояване и прилагане на получените знания.

Съдържание на курса (по седмици)

1. Обзор на микроконтролерите и развойните платки. Представяне на хардуерната среда Arduino. Запознаване със софтуерната среда Arduino IDE. Преговор на наученото в част 1 на избираемата („Програмиране в среда Arduino”).
2. Работа с дисплеи, бутони и I2C/TWI устройства.
3. Контролен проект – управление на TM1638 дисплей с бутони и индикация.
4. Експерименти с таймери, прекъсвания, управление на двигатели, серво и релета. Роботи и други изпълнителни устройства – проектиране на механизмите и алгоритмите.
5. Тест на робот с 4WD и серво – реален експеримент.
6. Управление на робота с безжична връзка – Bluetooth модул HC-06 и модул 433MHz.
7. Задаване на индивидуален проект за разработване на робот, серво или друг механизъм. Обсъждане на основните трудности и проблеми. Начален sketch на проекта.
8. Работа по проекта. Отстраняване на възникналите проблеми. Изготвяне на хардуерен прототип с прототипна платка.
9. Довършване и тестване на проекта. Писане на техническа документация.
10. Представяне и защитаване на проекта. Крайна оценка.

Част 3 – Програмиране в среда Arduino с 32-битови контролери

Анотация: Целите на курса са студентите да придобият допълнителни познания за програмирането на 32-битови микроконтролери ARM Cortex-M3 и Cortex-M4F (Atmel SAM3X8E и TI LM4F120 процесори) в среди Arduino и Energia и да се запознаят с принципите на създаването на схеми и устройства с тези контролери. Arduino е платформа за електронно прототипиране с отворен код, базирана на гъвкави и лесни за използва-

не хардуер и софтуер. Системата общува със средата, получавайки сигнали от множество сензори и датчици и може да взаимодейства с нея, контролирайки светлини, дисплеи, двигатели и други устройства. Микроконтролерът в платката се програмира с Arduino език за програмиране (базиран на Wiring и подобен на C++) в развойна среда Arduino (базирана на Processing). За нуждите на курса са осигурени платки Arduino Due и TI Stellaris LM4F120 Launchpad, както и множество сензори, дисплеи, бутони, шасита, серво-механизми и прототипни платки, с цел практическото усвояване и прилагане на получените знания.

Съдържание на курса (по седмици)

1. Обзор на 32-битовите ARM Cortex микроконтролери и развойните платки Arduino Due и Stellaris Launchpad. Представяне на хардуерните среди Arduino и Launchpad. Запознаване със софтуерните среди Arduino IDE 1.5 и Energia. Преговор на наученото в предходните части на избираемата („Програмиране в среда Arduino“ и „Програмиране в среда Arduino за напреднали”).
2. Работа с датчици за напрежение, ток, влага, температура, барометрично налягане, осветеност, акселерометри, жироскопи, магнитометри, RTC и др.
3. Контролен проект – управление на TM1638, MAX7219, I2C 7-сегментни и TFT сензорни дисплеи с бутони и индикация, с получаване на информация от разнородни датчици.
4. Експерименти с таймери, прекъсвания, DMA, управление на двигатели, серво и релета. Управление на паметта при по-сложни проекти. Работа с EEPROM.
5. Разработка на метеорологична станция – реален експеримент.
6. Управление на устройства с безжична връзка – Bluetooth модули HC-05/06 и безжични модули nRF24L01 2.4GHz. Работа с Wiznet W5100 Ethernet модули за връзка с LAN и Интернет. Управление по Интернет – локален web сървер.
7. Задаване на индивидуален проект за разработване на метеорологична станция, управление на робот или система за домашна автоматизация „умен дом“. Обсъждане на основните трудности и проблеми. Начален sketch на проекта. Симулиране в симулатор.
8. Работа по проекта. Отстраняване на възникналите проблеми. Изготвяне на хардуерен прототип с прототипна платка.
9. Довършване и тестване на проекта. Писане на техническа документация.
10. Представяне и защитаване на проекта. Крайна оценка.

Приложение 2. Кратък тест по Arduino за проверка на знанията

1. В стандартния пример Blink, функцията delay(1000) кара светодиода да мига на всяка секунда. Как може да направите светодиода да мига на всеки 2 секунди?
   а) delay(2);
   б) delay(200);
   в) delay(2000);
   г) delay(2000000).

2. Как може най-лесно да направите мигане на светодиод, без да използвате delay функцията?
   а) не може;
   б) с функцията millis(), междинна променлива (предходна ст-ст) и сравняване;
   в) с функцията millis(), без междинна променлива (предх. ст-ст);
   г) с програмиране на таймерите и прекъсвания;
   д) с функцията micros(), междинна променлива (предходна ст-ст) и сравняване.

3. Как се разбира дали вашия скеч (програма) е ок?
   а) не може да се разбере;
   б) „Done compiling“ и няма червени съобщения;
   в) „Done compiling“ и има червени съобщения;
   г) червен статус в линията под редактора.

4. Ако на цифровия вход 1 на Arduino Uno е подадено напрежение 3.1 волта, каква стойност ще върне функцията digitalRead(1)?
   а) LOW;
   б) HIGH;
   в) зависи от настройките на цифровия вход;
   г) неизвестно дали HIGH или LOW (случайна стойност).

5. Как се коментират в Arduino програмите?
   а) не може;
   б) // до края на реда;
   в) Загражда се коментара в /* */;
   г) и двата горни отговора (б и в) са верни.

6. На входа за отделно захранване на Arduino платката може да се подаде напрежение:
   а) само 5V DC;
   б) от 7 до 19V DC;
   в) само 9V DC;
   г) няма значение какво е напрежението.

7. Кои са двете главни секции в една програма за Arduino?
   а) setup and loop;
   б) main and setup;
   в) loop and main;
   г) int and setup.

8. На Arduino Uno – колко са аналоговите входове?
    а) пет и са означени A1-A5;
    б) шест и са означени A0-A5;
    в) шест и са означени A1-A6;
    г) пет и са означени A0-A4.

9. С коя команда може да укажете, че извод 9 е ИЗХОД?
    а) int sensorPin = 9;
    б) int sensorValue = 9;
    в) digitalWrite(9, HIGH);
    г) pinMode(9, OUTPUT).

10. С коя команда може да светнете светодиод, свързан към извод 5?
    а) int sensorPin = A0;
    б) int sensorValue = 0;
    в) digitalWrite(5, HIGH);
    г) pinMode(5, OUTPUT).

11. Кое от изброените устройства е аналогово входно устройство?
    а) сензор за налягане BMP180;
    б) бутон;
    в) сервомеханизъм;
    г) потенциометър.

12. Кое от изброените устройства е цифрово входно устройство (един вход)?
    а) сензор за налягане BMP180;
    б) бутон;
    в) сервомеханизъм;
    г) потенциометър.

13. С коя вградена функция може да просвирите звук от Arduino (не са показани параметрите на функцията)?
    а) tone();
    б) sound();
    в) audio();
    г) music().

14. С помощта на кое от изброените можете да комуникирате между Arduino платката и компютъра при свързан USB кабел?
    а) обекта SerialMonitor в кода и Serial Monitor в Arduino IDE;
    б) обекта Wire и USB конзола в PC;
    в) обекта Serial в кода и Serial Monitor в Arduino IDE;
    г) не можем да комуникираме с PC.

15. В Arduino Uno стойността, зададена на PWM изход, варира между?
    а) 0..5;
    б) 0..255;
    в) 0..1023;
    г) 0..256.

16. Съкращението PWM (ШИМ) означава?
    а) Pin Width Modulation;
    б) Pulse With Modulation;
    в) Pulse Width Modulation;
    г) Pulse Width Mode.

17. Кой от следните типове не е валиден тип данни в Arduino?
    а) void;
    б) byte;
    в) real;
    г) string.

18. В Arduino UNO LED_BUILTIN е свързан към цифров изход?
    а) 1;
    б) 3;
    в) 13;
    г) 1 и 3.

19. Какво прави следният код int ledpin[7];?
    а) задава се тип int на 7-мия елемент от масива ledpin;
    б) задават се стойностите на масив с изводите за 7 светодиода;
    в) декларация на тип от 7 елемента от тип int;
    г) инстанция на променлива – масив от 7 елемента от тип int.

20. Какво представлява I^2C?
    а) серийна комуникация по USB кабел;
    б) серийна комуникация между главни и подчинени устройства;
    в) съкращение от Interprocess Interrupt Call;
    г) нито едно от посочените.

Указания: винаги е верен само 1 от 4-те отговора. Тестът по Arduino е съобразен с всички платки Arduino, освен където изрично е указан друг (конкретен) модел на платката във въпроса.

Приложение З.а. Сайтове и форуми за Arduino

Въпреки че вече всеки умее да търси в Google, в тази секция е представена подборка от проверени от автора най-често използвани ресурси в Интернет за Arduino и Raspberry Pi. Включени са и ресурси за Raspberry Pi, като набираща все по-голяма популярност платформа.

Arduino

- ✓ ***https://www.arduino.cc/*** – основният сайт на Arduino. Ползвайте секциите Learn, Forum и Blog.
- ✓ ***http://playground.arduino.cc/Bulgarian/Nachalo*** – българската секция в сайта на Arduino.
- ✓ ***https://learn.adafruit.com/*** – ресурси за Arduino модулите, продавани от Adafruit.
- ✓ ***https://www.sparkfun.com*** – секциите за Arduino, Blog, Tutorials и Learn.
- ✓ ***http://www.instructables.com/howto/arduino/*** – пълни указания за проекти с Arduino.
- ✓ ***http://duino4projects.com*** – проекти с Arduino.
- ✓ ***https://www.circuitsathome.com/category/mcu/arduino/*** – интересни проекти с Arduino.
- ✓ ***http://robotshop.com/letsmakerobots/*** – проекти с роботи.
- ✓ ***http://www.robotev.com/*** – сайта на фирма Роботев, ползвайте секциите Библиотека и Блог, имат ресурси и за Raspberry Pi.
- ✓ ***https://www.facebook.com/groups/arduinobulgaria/*** – може да се включите в групата Arduino Bulgaria във Facebook.
- ✓ ***http://www.robotics-bg.com/discussion/index.php?board=41.0*** – форум по роботика, секция за Arduino.
- ✓ ***https://www.olimex.com/forum/*** – форум на фирма Olimex, има секция за Arduino.
- ✓ ***http://arduino.start.bg/*** – каталог с линкове за Arduino.
- ✓ ***https://arduinofordisabilities.wordpress.com/*** – материали за хора с увреждания и Arduino.
- ✓ ***https://esp8266.ru/arduino-ide-esp8266/*** и ***http://samopal.pro/esp8266-4/*** – ресурси за програмиране на ESP8266 чрез Arduino IDE (на руски език).
- ✓ ***http://arduino-project.net/*** – ресурси за Arduino на руски език.

Raspberry Pi

- ✓ ***https://www.raspberrypi.org/*** – основния сайт на Raspberry Pi.
- ✓ ***https://www.adafruit.com/category/105*** – ресурси за Raspberry Pi модулите, продавани от Adafruit.
- ✓ ***https://github.com/raspberrypi*** – ресурси за Raspberry Pi в GitHub.
- ✓ ***https://www.hackster.io/raspberry-pi*** – общност за проекти с Raspberry Pi.
- ✓ ***http://hackaday.io/projects/tag/raspberry%20pi*** – Raspberry Pi проекти.
- ✓ ***http://www.instructables.com/howto/raspberrypi*** и

http://www.instructables.com/id/Raspberry-Pi-Projects/ – указания за проекти с Raspberry Pi.

- ✓ ***https://diyhacking.com/diy-projects/raspberry-pi-projects/*** – проекти с Raspberry Pi.
- ✓ ***https://softuni.bg/*** – може да търсите за ресурси и записани на видео безплатни семинари на български език за Arduino и Raspberry Pi.

Отговори на въпросите на теста от Приложение 2:

1в, 2б, 3б, 4б, 5г, 6б, 7а, 8б, 9г, 10в, 11г, 12б, 13а, 14а, 15б, 16в, 17в, 18в, 19г, 20б

Приложение З.в. Кратко въведение в електрониката и електротехниката

Едно от предимствата при проектирането на проекти с Arduino е, че за голяма част от електронните детайли на ниско ниво вече са се погрижили за Вас – платката е готова, схемата е нарисувана. Имате нужда само от някои основни умения и познания в областта на електрониката, за да разберете какво се случва зад кулисите. За тази цел са подбрани за разяснение основните понятия в електрониката.

Ток, напрежение и мощност(и съпротивление и капацитет)

Токът, напрежението и мощността са взаимосвързани понятия и трябва да ги схванете правилно, ако не искате да повредите Вашите електронни схеми и проекти.

Напрежението се измерва в единици Волт (V). Означава се със символа U, измерва потенциала в схемата.

Токът се измерва в единици Ампер (**A**). Означава се със символа **I**, измерва силата на потока на електрическия заряд в дадена точка.

Мощността се измерва в единици Ват (**W**). Означава се със символа **P**, измерва отношението на пренесената енергия за определен интервал от време към големината на този интервал. В повечето случаи е валидна формулата P = I x U (мощността е равна на произведението на тока по напрежението).

Съпротивлението се измерва в единици Ом (**Ω**). Означава се със символа **R**, измерва отношението на електрическата потенциална разлика към тока в проводник, в който няма електродвижещо напрежение – на практика 1 ом е съпротивлението на проводник, върху който ток със сила 1A предизвиква спад на напрежение от 1V.

Капацитета (capacitance) се измерва в единици Фарад (**F**). Означава се със символа **C**, измерва отношението на големината на заряда към потенциала на проводника. На практика се използват малки стойности (пико, нано и микро фаради).

Всички измервания и величини в електрониката използват системата SI и са валидни следните означения:

pico (p) 10^{-12}, nano (n) 10^{-9}, micro (μ) 10^{-6}, mili (m) 10^{-3},

kilo (k) 10^{3}, mega (M) 10^{6}, giga (G) 10^{9},

последвани от единицата на величината, например 20mA означават 20 x 10^{-3} A, което е 0.020 Ампера (изговаря се 20 милиампера).

Ако се отнесат тези понятия към Arduino платките – захранващото напрежение е 5V за Uno. Всеки отделен изходен извод на микроконтролера може да отдаде ток до 20mA (0.02A), но сумарния ток от всички изводи трябва да е по-малък от 100mA (0.1A), следователно максималната управлявана мощност директно от микроконтролера ще е 0.5W (5V x 0.1A = 0.5W, или още казано, половин ват).

Макар да не е част от системата SI, в електронните схеми често се използва следната конвенция – десетичната запетая се замества с единицата за измерване, по този начин 3.3V става 3V3, 1.5kΩ става 1k5 (тук дори се изпуска и знака за съпротивление Ω). Тази конвенция се прилага само в самите изчертавания на схемите, не в описанията им.

Закон на Ом – основен закон в електротехниката (и електрониката), това е правилото, определящо зависимостта между напрежението, тока и съпротивлението на проводника в една електрическа верига. Същността на закона е проста: създаваният от напрежението ток е обратно пропорционален на съпротивлението, което той трябва да преодолява, и е право пропорционален на пораждащото го напрежение.

Законът на Ом може да се запише по три еквивалентни начина:

I = U / R

U = R x I

R = U / I

Това позволява, когато се знаят две от величините (ток, напрежение и съпротивление) в една верига, да се пресметне третата. Често се използва, за да се изчисли правилно съпротивлението, да се прецени какво напрежение да се приложи или да се пресметне каква е максималната сила на тока в дадена верига, ако са известни другите две величини. Понякога при проектирането на електронни схеми се решават системи от линейни уравнения, защото за даден компонент са неизвестни две от трите съставки, но този компонент участва в по-сложен контур от схемата, за която има известни данни (ето, къде се използва тази „пуста и ненужна“ математика – без нея и без Закона на Ом сигурно нямаше да може да бъде четена тази книга на компютър и да се работи с Arduino).

Основни електронни елементи

Резистор (*съпротивление*) – ограничава силата на тока, протичащ във веригата. Обикновено са с цилиндрична форма, значението на номинала се указва обикновено с цветни ленти (може и с цифри). Брой изводи – 2. Лесно може да се объркa стойността, бъдете внимателни. Имат и максимална разсейвана мощност, която е важно да не се надвишава.

Вж. ***https://en.wikipedia.org/wiki/Resistor***

При работа с резистори, освен Закона на Ом, често се налагат и някои пресмятания – при последователно и паралелно свързване на резистори и изчисляване на делител на напрежение. Валидни са следните правила:

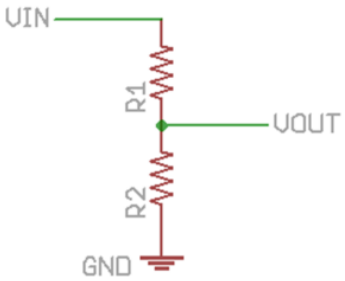
При последователното свързване на резистори, сумарното съпротивление R_t е равно на сумата от съпротивленията на отделните резистори.

$$R_t = R_1 + R_2 + \ldots + R_n$$

При паралелното свързване на резистори реципрочната стойност на полученото съпротивление R_t е равна на сумата от реципрочните стойности на отделните резистори.

$$1/R_t = 1/R_1 + 1/R_2 + \ldots + 1/R_n$$

За свързване и пресмятане на *делител на напрежение* се използват дадените подолу схема и формула.



Фиг. 68. Схема на делител на напрежение с 2 резистора

$$V_{out} = V_{in} \frac{R_2}{(R_1 + R_2)}$$

Фиг. 69. Формула на делител на напрежение с 2 резистора

Кондензатор (*capacitor*) – предназначен е за временно съхранение на електрически заряд в електрическо поле, често се използват за филтруване, натрупване на енергия или генериране на сигнали. Обикновено са с цилиндрична форма, значението на номинала се указва обикновено с цифри във Фаради (нано, пико или микро). Брой изводи – 2. Лесно може да се объркa стойността, бъдете внимателни. Имат и максимално работно напрежение, а електролитните кондензатори имат и полярност – тези параметри трябва да се отчитат при употребата им.

Вж. ***https://en.wikipedia.org/wiki/Capacitor***

При паралелното свързване на кондензатори, получения общ капацитет C_t е равен на сумата от капацитета на отделните кондензатори.

$$C_t = C_1 + C_2 + \ldots + C_n$$

При последователното свързване на кондензатори, реципрочната стойност на получения общ капацитет C_t е равна на сумата от реципрочните стойности на отделните кондензатори.

$$1/C_t = 1/C_1 + 1/C_2 + \ldots + 1/_n$$

Диод – електронен еквивалент на еднопосочен клапан. Тока протича само в едната посока. Обикновено са с цилиндрична форма, с лента на едната страна, определяща полярността (катода). Брой изводи – 2 (анод и катод). Тока тече от анода към катода, катода трябва да е свързан към ниския потенциал, а анода – към по-високия потенциал.

Вж. ***https://en.wikipedia.org/wiki/Diode***

С*ветодиод (LED*) – излъчва светлина, когато се пусне ток през него. Тока протича само в едната посока. Обикновено прилича на лампичка. Брой изводи – 2. По-дългия извод (анод) се включва към положителния потенциал. Работи само при правилно включване. Нуждаят се от резистор за ограничение на силата на тока. Вж. ***https://en.wikipedia.org/wiki/LED***

Транзистор – използва се за превключване или усилване на сигнали. Може да са в различни корпуси, наименованието се нанася на корпуса. Обикновено има три извода (база, колектор и емитер при биполярни транзистори), важно е да не се объркат. За ограничаване на тока често се използват резистори.

Вж. ***https://en.wikipedia.org/wiki/Transistor***

Интегрална схема (ИС, микрочип или чип) е електронна схема с миниатюрни размери, състояща се от полупроводникови устройства и пасивни компоненти. Реализира се обикновено върху тънък кристал от силиций или друг полупроводник (чип). В зависимост от функционалното си предназначение, ИС могат да се класифицират като аналогови, цифрови и комбинирани. Цифровите ИС се състоят главно от транзистори. Аналоговите схеми съдържат също така и резистори и кондензатори. Микроконтролерът в платките Arduino е пример за цифрова интегрална схема.

Вж. ***https://en.wikipedia.org/wiki/Integrated_circuit***

За по-любознателните експериментатори се препоръчва да разгледат *Портала за електроника* в Уикипедия:

https://en.wikipedia.org/wiki/Portal:Electronics

Приложение З.с. Кратък справочник на най-често използваните инструменти и компоненти

В концепцията на Arduino не влизат елементи за монтаж и корпус. Разработчиците са длъжни да си изберат и реализират самостоятелно начина на монтаж, сглобяване и механичната защита на платките (и целия проект). В повечето Arduino платки и платки за разширения са предвидени монтажни отвори, които може да се използват за прикрепването им заедно или към корпуса на цялостния прототип. В магазините за електронни компоненти, както и в онлайн магазините (описани в следващата глава) се намират и подходящи кутии, закрепвания и панели, въпрос на творчество и фантазия е да ги откриете и да направите финалната версия на Вашите Arduino проекти в използваем и приятен вид.

Не е необходимо да имате веднага всички изброени по-долу инструменти и компоненти подръка, с течение на времето ще добиете опит и усет и ще си направите колекция от необходимите за Вашите проекти.

Инструменти

- Защитни очила и очила-лупа (може и отделна лупа) – много електронни компоненти имат много дребни означения;
- Лампа-челник и/или настолна лампа;
- Клещи-резачки и резачка за кабели, както и плоски клещи;
- Отвертки – различни размери (кръстати и прави);
- Бормашина с набор от свредла;
- Мултицет (multimeter) – незаменим измервателен инструмент. Много помага за проверка на свръзките, за замерване на стойностите на компоненти с изтрит или неясен надпис, както и за проверка на работоспособността на голяма част от по-простите електронни елементи (диоди, резистори, кондензатори, бутони, релета и др.);
- Пистолет за топло лепене – за прикрепяне на механични компоненти, кабели и сглобяване на кутии;
- Поялник и тинол (сплав за запояване) – препоръчваме да се снабдите с поялник със стойка, както и поялника да е с регулируема температура и защита от прегряване (оскъпява се, но е полезна функция);
- Гайки, винтове, шайби и шпилки – различни размери и дължини, метални и пластмасови;
- Тиксо – едностранно и двустранно лепещо за прикрепяне на компонентите и кабелите;
- Изолационно тиксо за кабели и компоненти към 220V (или друго високо напрежение).

Компоненти

- Монтажна платка (breadboard) – удобна е за бърз и лесен монтаж без запояване на компоненти и детайли, трябва да се има впредвид, че финалната версия на проекта се препоръчва при възможност да е на прототипна или специално изработена печатна платка (със запояване), ако е нужна устойчивост на вибрации, външни въздействия и по-малък размер. Има различни размери монтажни платки, могат и да се стиковат една към друга с помощта на щифтовете по краищата им.
- Захранващ модул за монтажна платка MB102 – осигурява гъвкаво (може да се избира 3.3 и 5 V) и филтрувано напрежение на монтажната платка от няколко вида източници на напрежение. Много е удобно, когато на монтажната платка има компоненти с по-висока консумация, която не може да се осигури от Arduino платката.
- Рейки за монтажна платка – за свързване с кабели с друг конектор, както и за монтаж на чипове и по-големи елементи.
- Кабели с накрайници тип „крокодил“ – за подаване на напрежение, измерване и за лесно временно свързване на външни устройства.
- Кабели за монтажна платка (breadboard cables) – препоръчва се да си набавите трите вида (м-ж, м-м, ж-ж), често се наричат Dupont wire (Dupont cable) с указан накрайник (male/female). Може да ги купувате на лента и да си откъсвате нужния брой за връзката с конкретния компонент, така няма да се разпиляват и разбъркват. Вземете си различни дължини (10, 20, 40 и 80 см).
- Кондензатори за филтруване – препоръчва се на всяка монтажна платка да има по един общ електролитен филтруващ кондензатор 10µF до 100µF и до всяка отделна интегрална схема да има по един неелектролитен развързващ кондензатор 0.1µF (100nF).

Електронни компоненти (елементи)

Описаните по-долу електронни компоненти са препоръчителни. Можете да стартирате и с готови комплекти от компоненти за Arduino – в тях има нужния минимален брой от всички компоненти, с които се реализират проектите, за които е предназначен дадения комплект (с или без платката Arduino). Продават се и отделни комплекти от достатъчен асортимент от отделните елементи (напр. набор резистори по 10 броя от 10-12 стойности), както и универсални набори от разнородни компоненти за любители (несвързани с Arduino или конкретен проект). В повечето реални магазини (на място) продават и отделни бройки от елементите, докато в онлайн магазините често това са само комплекти или има указан минимален брой за покупка.

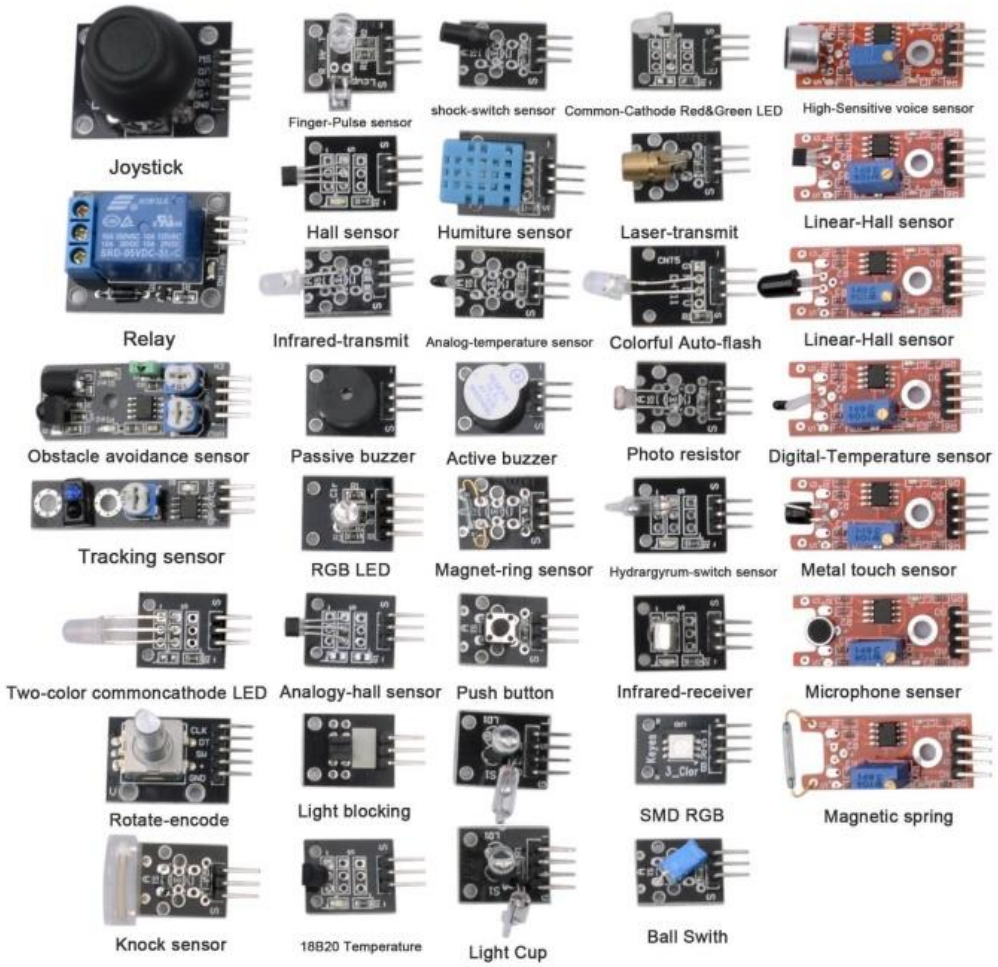
Все пак, всеки уважаващ себе си любител на електрониката, трябва да има под ръка следните елементи:

- набор резистори и потенциометри – различни стойности и мощност.
- набор кондензатори – нисковолтови електролитни, няколко стойности на капацитета, както и керамични за аналогови сигнали и генератори.

- набор диоди – нисковолтови (за сигнали и изправители) и високоволтови (за обратен ток при релета и двигатели).
- набор транзистори – маломощни PNP/NPN, средно и мощни и MOSFET.
- набор светодиоди – трите основни цвята, различни размери, както и RGB.
- набор бутони, DIP превключватели, релета и др. механични компоненти за избор и управление на електрически вериги.
- за по-напредналите – набор основни логически интегрални схеми (CMOS и TTL), стабилизатори на напрежение, изместващи регистри 74595 за по-сложните проекти и др.

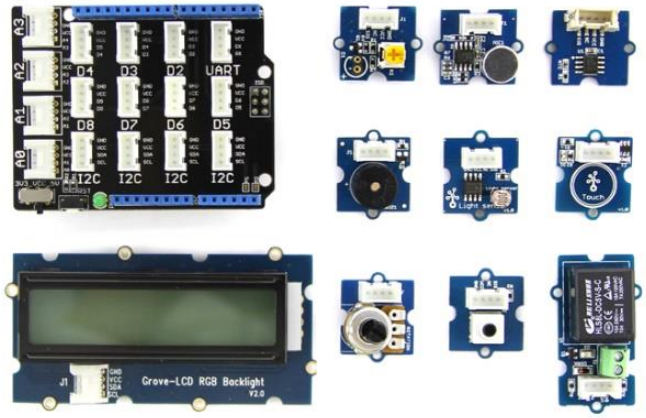
Комплекти с модули – сензори за Arduino

Може да се закупят и комплекти с готови модули – сензори за Arduino (не набор от компоненти). Всеки сензор е монтиран на малка печатна платка с накрайници, заедно с pull-up и ограничителни резистори, филтруващи кондензатори, транзистори, диоди и другите необходими компоненти, нужни за работата на сензора; някои имат светодиод за наличие на захранващо напрежение и монтажни отвори за закрепване към крайното устройство. Изискват експериментална платка (breadboard) – понякога не е включена в комплекта и комплект кабели със съответните накрайници.



Фиг. 70. Комплект модули – сензори за Arduino

Друг подход за комплект модули сензори и компоненти за Arduino е Grove – по-добен на показания по-горе, но с куплунги и с разширителна платка (Grove Base shield), в която се включват Grove-модулите. Предимство е сравнително лесното разучаване и използване, недостатък е по-високата цена и по-ограничения набор модули.



Фиг. 71. Комплект модули – сензори Grove с Base shield

Приложение 3.d. Съвети за покупка и избор на компоненти за проектите

Когато се разработват прототипи с Arduino, често се налага да се закупуват допълнителни компоненти и материали, най-вече електронни елементи, както и някои механични детайли – кутии, задвижвания, кабели, захранвания, батерии и др.

Може да се използват онлайн магазини от България и чужбина, изброени са някои от тях по-долу.

eBay.com е най-известният онлайн магазин, в него може да откриете почти всичко. Amazon.com също предлагат много електронни компоненти, вкл. и Arduino китове. Препоръчва се покупка от онлайн магазини и търговци от Обединеното Кралство и Германия (или друга страна от ЕС), поради липсата на митнически проблеми и ускорената доставка. При поръчка от САЩ или Азия на стоки с цена, по-висока от 15 евро или с доставка по куриер има голяма вероятност Вашата пратка да бъде задържана в митницата, да доплатите ДДС, понякога и мито (за потребителска електроника и стоки до 150 евро по принцип се дължи само ДДС без мито), но най-неприятно е чакането и голямата бюрокрация покрай самия процес на митническо освобождаване.

Проверявайте откъде се изпраща стоката, с какъв метод на доставка е и каква е крайната цена (повечето онлайн магазини имат сортиране по „цена+доставка“, не само по цена – понякога митницата гледа цената с доставка, не само цената на стоката). Например, използвайте ebay.co.uk, ebay.de, amazon.de или amazon.co.uk, но самият факт, че поръчвате от сайт .co.uk или .de не Ви гарантира, че стоката ще е изпратена от UK – следете описанието, непременно влезте във връзка с търговеца, ако имате и най-малките съмнения, преди да поръчате.

Aliexpress.com е също добър избор, но трябва да отчитате вероятността за забавяне на доставката (често се чака 30-45 дни и повече), както и вероятността пратката да бъде задържана на митницата.

Някои търговци от Китай (Азия) използват само за определени стоки т.нар. „склад в ЕС“, от който изпращат стоката без проблеми с митниците, но на по-висока цена, други търговци предлагат предварително да си удържат парите за митническото освобождаване и да го извършат чрез техни посредници (често на завишена цена, с комисиона за „услугата“), което все пак е удобство и спестява време.

При поръчка от онлайн магазини внимавайте и с начина на доставка, обикновено се предлагат няколко различни начина (поща, куриери, ускорена доставка), за някои от които може да заплатите цена, многократно превишаваща цената на стоката, която купувате.

Покупка от България

Може да използвате и български онлайн магазини, за да спестите значително времето за доставка. Някои от тях са:

- Роботев – ***http://robotev.com***
- Вактор – ***https://vactor.bg/***
- ERElement – ***http://erelement.com***

- Квазер – ***http://kvazer.com***
- Викиват – ***http://vikiwat.com***
- Олимекс – ***http://olimex.com***
- Комет Електроникс – ***http://store.comet.bg/Catalogue***
- Кузмов Електроникс – ***http://kuzmovelectronics.com***
- Пасат Електроникс (Елимекс) – ***http://elimex.bg***
- Електрон Пловдив – ***http://eshop-bg.com*** и др.

Електронни компоненти и Arduino платки може да намерите в България и в онлайн платформите за безплатни обяви OLX ***http://olx.bg***, Базар.БГ ***http://bazar.bg*** и др. В Базар.БГ има специализиран магазин за Arduino платки и компоненти – ***http://bgarduino.bazar.bg***

Посочени са някои магазини за електронни части и компоненти, които са с търговски салон или офис в град Пловдив, така ще може да се изберат на място компонентите (око да види, ръка да пипне) – *адресите по-долу са в град Пловдив*:

- Кузмов Електроникс – ул. „Митрополит Панарет“ №6, тел. 0878 438 241;
- Комет Електроникс – бул. "Свобода“ №69, секция 5, офис 3, тел. 032 634 186, 0885 222 550;
- Квазер – бул . „Хр. Ботев“ №71, тел. 032 675 134;
- Викиват – ул. „Мостова“ №3, тел. 0700 45 445;
- Елимекс – ул. „Цар Асен“ №7, тел. 032 549 017;

Съвети за избор и замяна на компоненти

Относно избора на компоненти, с течение на времето, ще се научите да разчитате техните т.нар. datasheet (технически характеристики) и да сравнявате и търсите компоненти по характеристики, а не само по техните наименования и модел, описани в избрания от Вас проект от това ръководство (или срещнати в описание на проект по Интернет). Най-важните детайли, на които трябва да обърнете внимание, са:

- Захранващото напрежение – освен опасност от директна повреда на компонента, това води и до смяна на логическите нива. Ако все пак се наложи смяна на компонент от 5 на 3.3 волта, при 5 волтова платка, трябва да сложите и преобразуване на логическите нива на цифровите изводи, както и да внимавате захранването на компонента да е от 3.3 волтовата захранваща линия в платката;
- Ако компонентът е с цифрово управление (I^2C, SPI, 1-wire и други подобни протоколи), трябва да внимавате за възможни различия в протоколите (версия и скорост), при нужда да смените библиотеките, както и да следите за адреса по подразбиране (може да е различен в „новия“ компонент);
- Не съвсем маловажен аспект е и корпуса на компонента, често се намират „същите“ компоненти в друг корпус, при което трябва да намерите начин да използвате преходна платка или да пренаредите изводите на свързване на

компонента, спрямо посочените в оригиналния проект, може да се наложи запояване;

- Понякога, има компоненти с различни параметри, но със същите изводи и управление – например DHT11 и DHT22 са „почти еднакви“, но DHT11 не измерва и не показва отрицателни температури. Може да има различия в работния температурен обхват на компонентите, това е важно, ако правите проект, при който сензорите са „навън“ и са подложени на въздействието на околната среда. Има компоненти, предназначени за военно и индустриално приложение, които са по-устойчиви на външни въздействия, спрямо тези за „комерсиално“ или гражданско приложение, но са със съответно завишена цена или друг корпус. Разликата понякога е означена само в една различна цифра или буква в означението на детайла.

Приложение З.е. Използване на емулатори за разработване на Arduino проекти

Разработени са множество системи за симулация и емулация на Arduino проекти и схеми. Те позволяват разработване и тестване на проектите без налична Arduino платка и компоненти. Имат много предимства, но и доста ограничения – повечето емулатори не поддържат доста от външните устройства (платки за разширения), както и ограничен набор компоненти. Удобни са за разработка и оценка на малки и средни проекти, преди да се инвестира в реален хардуер. Повечето симулатори все още нямат 32-битова поддръжка (за ARM и Intel базираните Arduino платки). Ето някои от по-известните симулатори:

- Fritzing (***http://fritzing.org***)– безплатен, поддържа Due, чертаене на схеми, удобен за работа, визуално проектиране;
- Wokwi Online Arduino and ESP32 Simulator (***https://wokwi.com***) – удобен за работа, добре емулира, има голяма общност и групи във FB – има и безплатна версия;
- Codeblocks for Arduino (***http://arduinodev.com/codeblocks***) – безплатен;
- Emulino (***http://github.com/ghewgill/emulino***) – безплатен;
- ArduinoSim (***http://sourceforge.net/projects/arduinosim***) – безплатен;
- UnoArduSim (***https://www.sites.google.com/site/unoardusim***) – безплатен;
- Simuino (***http://code.google.com/archive/p/simuino***) – безплатен;
- ArduinoDebugger (***http://github.com/Paulware/ArduinoDebugger***) – безплатен симулатор с дебъгер;
- Softpedia Arduino Simulator (***http://www.softpedia.com/get/Programming/Other-Programming-Files/Arduino-Simulator.shtml***) – безплатен;
- Arduino Simulator iPhone App (***https://itunes.apple.com/us/app/arduino-simulator-learn-diy/id438426863?mt=8***) – US$ 2.99 ;
- Simulator for Arduino (http://virtronics.com.au/Simulator-for-Arduino.html) – AU$ 19.99, пробен период.

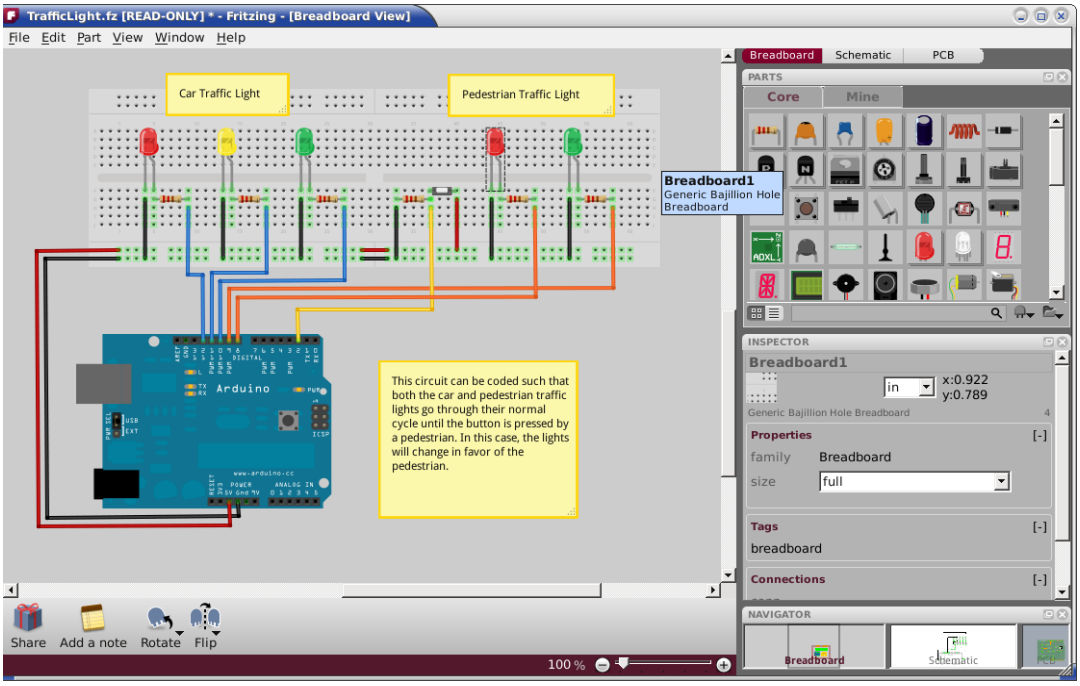
Най-удачен избор, според нас, е Fritzing, затова ще обърнем малко внимание на него по-долу (препоръчваме и Wokwi за онлайн използване).

Fritzing е проект с отворен код за разработване на CAD софтуер за дизайн на електронни схеми (хардуер) за любители, за да им помогне да преминат от експерименти с прототипи към построяване на краен вариант на техните устройства. Fritzing е създаден в духа на езика за програмиране Processing и Arduino контролерите и позволява на любителите да документират техните базирани на Arduino прототипи и да създават печатна платка (PCB) за производство. Уеб сайта на проекта помага на потребителите да споделят и обсъждат проектите си и да оптимизират разходите за пускане в производство [Wikipedia – Fritzing].

Основното предимство на Fritzing е, че може да се използва като инструмент за автоматизация на проектирането на електроника (EDA) за не-инженери: входната метафора е вдъхновена от обкръжението на дизайнерите (прототипна платка), докато крайният резултат е фокусиран над достъпното производство (печатни платки). След декември 2004 във Fritzing е наличен и изглед на сорс код, където може да се пише код и да се качва директно в Arduino платката от проекта.

Изображенията на компонентите се разпространяват под лиценз CC-BY-SA, под който се разпространяват и всички генерирани проекти.

Може да разгледате няколко урока от Instructables за използването на Fritzing ***http://www.instructables.com/id/Fritzing-an-Introduction*** (вижте и линковете вдясно – Related). В официалния сайт на Fritzing може да потърсите проекти с Arduino – ***http://fritzing.org/projects/by-tag/arduino***.



Фиг. 72. Изглед на Fritzing проект с Arduino

Приложение 3.ғ Списък със защитени дипломни работи (реални проекти)

Представен е списък на темите на няколко дипломни работи, свързани с избираемите дисциплини „Програмиране в среда Arduino“, с използване на микроконтролери, разработени и защитени успешно под ръководството на доц. д-р Светослав Енков. При желание за достъп до разработките – свържете се с доц. Енков.

1. Интегрирана безжична система за управление на дома, А. Христозов, спец. Софтуерни технологии, 2017 г.;
2. Прототип и реализация на система за управление на отдалечени микроконтролери „SmartHome“, Н. Недевски, спец. Информатика, 2016 г. (научен р-л докторант П. Кюркчиев, консултант и рецензент – гл.ас. д-р Св. Енков);
3. Система за "умно осветление“ с Ардуино, П. Петров, спец. Информатика, 2016 г.;
4. Създаване на система за управление, автоматизация и контрол над дома, А. Димитров, спец. Информатика, 2016 г.;
5. Създаване на Смарт Интернет Контролер (IoT) за управление на устройства от бита с IR-комуникация „Hydra remote control“, Г. Добришинов, спец. БИТ, 2016 г.;
6. Автомобилна система за сигурност с глобално позициониране, Т. Михайлов, спец. Софтуерни Технологии, 2016 г.;
7. Създаване на програмируем логически контролер (PLC) с графичен потребителски интерфейс (GUI), Т. Михайлов, спец. Информатика, 2015 г.;
8. Интелигентен дом, И. Симидчиев, спец. Информатика, 2014 г.;
9. Изграждане на електронна система за управление на домакинство, А. Христозов, спец. Информатика, 2014 г.;
10. Проектиране и изграждане на система за сигурност с Arduino, К. Петров, спец. Информатика, 2014 г.;
11. Създаване на Quadcopter с помощта на AeroQuad32 и Android, Т. Попчев, спец. Информатика, 2013 г.;
12. Система за контрол на достъп, М. Армянов, спец. Информатика, 2012 г.

Допълнение – нови дипломни работи (в прав хронологичен ред):

13. Изграждане на система за интелигентен дом, Й. Василев, Информатика, 2019 г.;
14. Автономно управление на радио управляем модел с Arduino Mega 2560, Ат. Николов, спец. Информатика, 2019 г.;
15. Система за оповестяване при бедствия и аварии, Б. Курдов, Информатика, 2019 г.;
16. Изграждане на система за „интелигентна стая“, Н. Атанасов, Информатика, 2019 г.;
17. Проектиране и реализация на умна домашна градина с Arduino, М. Кръстева, спец. Информатика, 2019 г.;
18. Система за наблюдение и защита на центрове за данни, С. Петков, спец. Информатика, 2019 г.;
19. Изграждане на интелигентен гараж с Arduino, К. Михайлов, спец. СТД, 2020 г.

Библиография

Книги

[Arduino Atmospheric'12] Patrick Di Justo, Emily Gertz, „Atmospheric Monitoring with Arduino", O'Reilly Media, 2012, ISBN: 978-1-449-33814-5.

[Arduino Beginning'10] Michael McRoberts, „Beginning Arduino" 2-nd edition, Apress , 2012, e-book ISBN: 978-1-4302-5017-3.

[Arduino Bots'11] Kimmo and Tero Karvinen, „Make: Arduino Bots and Gadgets (Learning by Discovery)", O'Reilly Media, 2011, ISBN: 978-1-449-38971-0.

[Arduino Cookbook'11] Michael Margolis, „ Arduino Cookbook", O'Reilly Media, 2011, ISBN: 978-0-596-80247-9.

[Arduino Notebook'08] Brian Evans, „ Arduino Programming Notebook", ***http://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf***, CC license 2008.

[Arduino Practical'09] Jonathan Oxer, Hugh Blemings, „Practical Arduino: Cool Projects for Open Source Hardware", Apress, 2009, ISBN 978-1-4302-2477-8.

[Arduino Sensors'11] Tom Igoe, „Making Things Talk: Using Sensors, Networks, and Arduino to See, Hear, and Feel Your World: Physical Methods for Connecting Physical Objects", O'Reilly Media, 2011, ISBN 978-1-449-39243-7.

[Отворен хардуер'15] Георги Петров, Филип Андонов, Тодор Дачев, „Разработка на приложения с отворени хардуерни платформи", учебник по договор BG051PO001-3.1.07-0062, София, 2015, ISBN: 978-619-160-506-4.

[Програмирование Arduino'12] Улли Соммер, „Программирование микроконтроллерных плат Arduino/Freeduino", ориг. издание „Mikrocontroller-Programmierung mit Arduino/Freeduino – Franzis", БХВ – Санкт Петербург, 2012, ISBN: 978-5-9775-0727-1 (на руски език).

Интернет сайтове (+ Приложение 3.а)

[Arduino] Arduino Home Page, ***http://www.arduino.cc*** и ***http://www.arduino.org***

[AVRStudio] Using AVR Studio for Arduino development, ***http://www.engblaze.com/tutorial-using-avr-studio-5-with-arduino-projects*** (последно посетен 23.07.2017)

[ARDX] ARDX Arduino Experimenter's Kit, ***https://solarbotics.com/product/ardx*** (последно посетен 01.08.2017)

[Pololu] Pololu Robotics and Electronics, ***https://www.pololu.com*** (последно посетен 02.08.2017)

[Wikipedia] – Уикипедия, свободната енциклопедия, ***https://bg.wikipedia.org***

Съдържание

Списък на фигурите

Светослав Енков

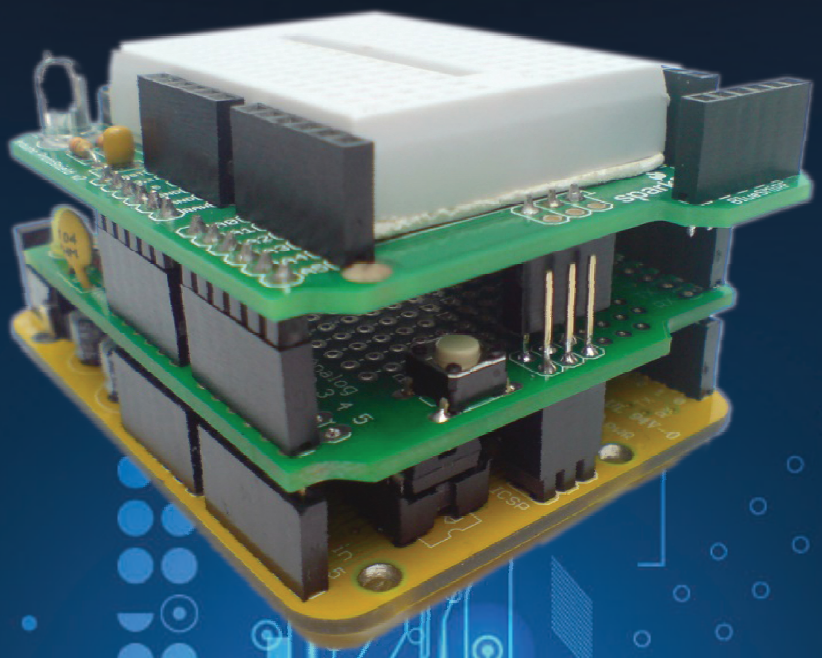
Програмиране в среда Arduino

Практическо ръководство

Българска, първо издание

Предпечатна подготовка: Георги Ташков
Пловдив, 2017 (последни корекции - август 2020)
ISBN 978-619-202-261-7

Пловдивски Университет „Паисий Хилендарски“
ул. „Цар Асен“ 24
Факултет по Математика и Информатика
бул. „България“ 236
Пловдив



9 786192 022617