

Мазмұны

№1 сабақ: Arduinoмен таныстыру. Кіріспе.	2
№2 сабақ: Электроника және программалауға кіріспе	13
№3 сабақ: Математика операторлары, ауыспалы, шартты операторлармен таныстыру	25
№4 сабақ: Енгізу және шығару порттарымен жұмыс, константалар, ауыспалы булевалар	44
№5 сабақ: Ұқсас порттармен жұмыс, жарық датчигінен, есептуішінен мәліметтерді салыстырып оқу	54
№6 сабақ: Ұқсас порттармен жұмыс, үндесті динамикті қосу, температура датчигімен жұмыс	59
№7 сабақ: Мәліметтер массивімен таныстыру, циклдар, массивтерді сұрыптау, потенциометрмен жұмыс	63
№8 сабақ: Кітапханаларды қосу, ультрадыбысты дальномермен таныстыру	69
№9 сабақ: Arduino арқылы ДС двигателін басқару	75
№10 сабақ: Өз функцияларын жасау	82
№11 сабақ: Серводвигателмен таныстыру	89
№12 сабақ: Серводвигателмен жобалар	92
№13 сабақ: Бекітуге жобалар	93
№14 сабақ: Bluetooth модулімен таныстыру	95
№15 сабақ: Цифрлық тахометрмен таныстыру	99

№1 сабақ

Тақырыбы: Arduinoмен таныстыру. Кіріспе.

Электр түсінігі: Тоқ көздерінен тоқ тұтынушыға өткізгіш арқылы қозғалатын зарядталған бөлшектер (электрондар) ағымын электр тоғы деп атайды.

Аккумуляторлар, күн батареялары, сондай-ақ электростанциялар тоқтың қайнар көздері болады.

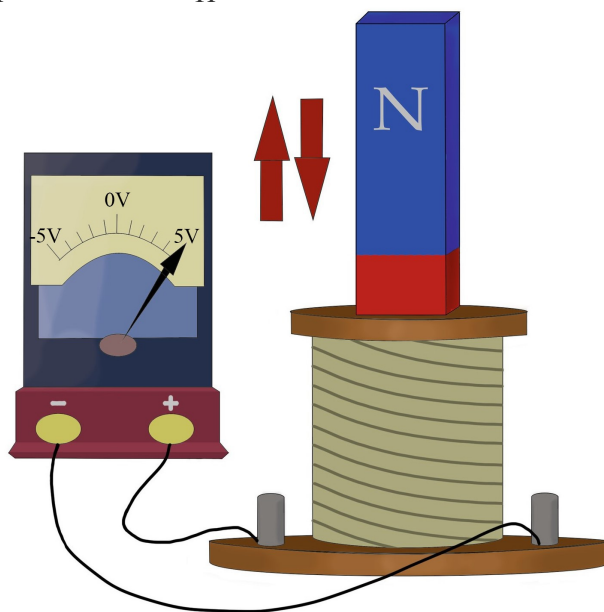
Тоқ өткізетін зат немесе кез келген нәрсе өткізгіш болуы мүмкін: металл сым, су мен тіпті дағды өткізгіш болуы мүмкін.

Өткізгіш арқылы қозғалатын бөлшектер электрондар деп аталады.

Тоқты тұтынушы – бұл электрде жұмыс істейтін кез келген құрал, мысалы жай шам.

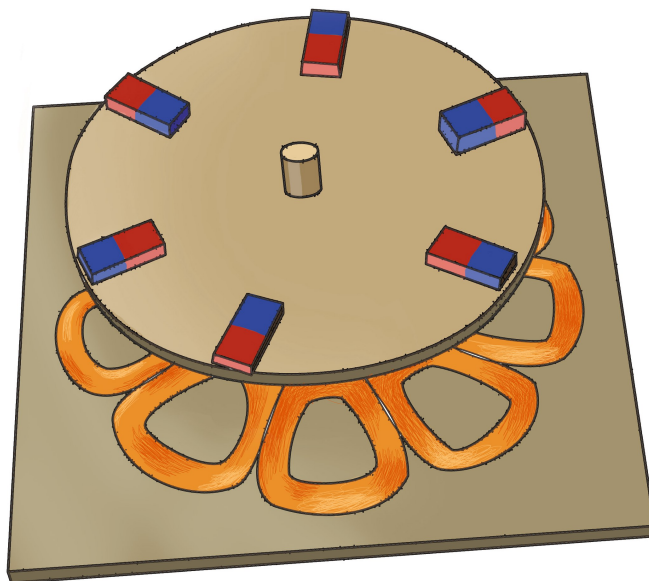
Электрондар сондай аз, оларды көрмейсің, ұстауға да болмайды. Барлық атомдар электроннан тұрады, электрондар әр затта кездеседі. Бірақ бір қатарға бірге қозғалатын электрондар электр тоғын жасап электр құралын жүргізеді не шамды жағады. Өткізгіште әрқашанда электрондар бар, бірақ олар қозғалмайды не астан-кестен қозғалса, жай шамды сымға қосып, ол жанбайды, ол үшін электрондарды бір бағытта қозғау қажет, сол үшін электростанциялар жұмыс істейді. Бірақ олар қалай жұмыс істейді?

Электрондарды өте жай қозғалуға мәжбүр етуге болады. Олар магнит өрісіне өте жақсы әсер етеді, яғни электрондар қозғалу мен сымда тоқ болу үшін сымның жанын магнитпен тез өту жеткілікті. Бұл магнит сымға жақындағанда болады, электрондар магниттің әсерінен өз орындарынан қозғалады. Бұның бәрі электрондардың өз өрісі, магниттің де өз өрісі бар, олар көрінбейді, бірақ олар бір-біріне өз әсерін тигізеді, мысалы, егер магнитті басқа магнитке жақындатса, олар бірін-бірі әрі итереді. Магнит электрондар жылжитын өріс жасайды, басқаша айтқанда – магнит күштірек әсер етсе, соншалықты кернеу көп. Осы қысқа мерзімде, сымға магнитті жақындатқанда, шамды жағуға болады. Электрондар орнынан қозғалғаннан кейін, олар тынышталады, олар қайтадан қозғалу үшін, магнитті алып тастау керек, сонда ғана олар кері жылжи бастайды. Егер магнитті тез жақындатып және алып отырса, тоқ өткізгіште тұрақты болып, шам кедергісіз жанып тұрады.

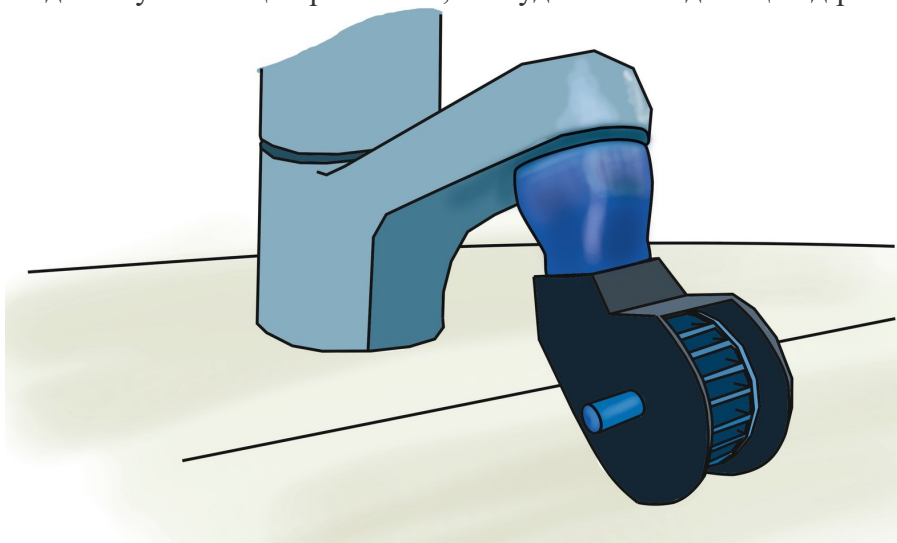


Барлық электростанциялар осындай құбылыстарға негізделеді. Әрбір электростанцияда генератор қондырылған, бұл генератор өте қарапайым, оның ішінде сымдардан көп катушкалар мен көп магниттер, магниттер тез де үзіліссіз катушкалармен қатар жылжиды да катушкалардың сымдарында тоқ пайда болады, сол тоқ сымдар арқылы біздің үйлерге беріледі. Магниттер катушкаларға жақындайды және алыстатылады, оны олар өте тез, секундына 50-60 рет жасайды, сондықтан бөлмеде шамдар үзіліссіз жанады, бірақ олар секундына 50-60 рет өтеді де жанады, алайда біздің көзіміз осындай жылдамдықты көрмейді, сондықтан біз бір қалыпты жануын көреміз. Магниттерді ақырын жылжытуға болады, онда біздің көзіміз

жарықтың жыпылықтауын байқайды да шаршайды (жыпылықтаған фонарьға қараумен тең). Магниттерді сондай тез не жылжытады? Мұнда бәрі қарапайым. Магниттер шеңберде орналасқан және катушкалармен қатар айналады. Мысалы, төмендегі суреттегідей.



Өте қарапайым магниттерді айналдыруға болады. Мысалы, гидроэлектростанциялар өзеннің ағымның арқасында жұмыс істейді, осындай магнитті дискіге турбина жай бекітіледі, судан магнитті диск айналады да, катушкада электр өндіреді. Төменде суретте 3D принтерде шығарылған турбина көрсетілген, ол үйдің дағдылы кранына қосылады, егер оған магнитті дискті қосып, сымды катушканы қатар жалғаса, біз суды ағызғанда ток өндіріле бастайды.



Су турбинының 3D моделін 3D принтерінде басу үшін мына сілтемеден:
http://3dtoday.ru/3d-models/games/toys/mini_vodyanaya_turbina/ аударуға болады

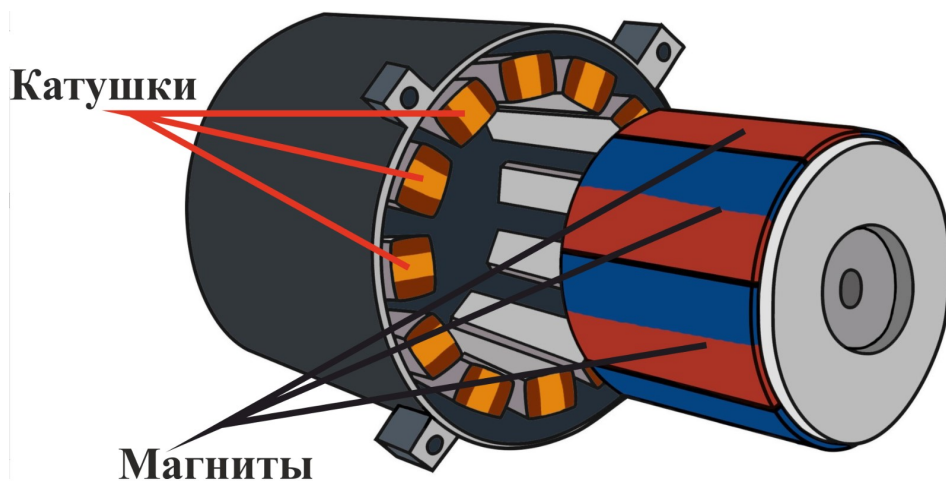
Жел электрогенераторлары да осындай принциппен жұмыс істейді, тек қана бұл жағдайда генераторға жел турбины бекітіледі.



Ондай генераторды өзіміз жасауға болады, оны катушкаға сымды орап не магнитті диск жасауға болады. Немесе қарапайым кішкене моторды пайдалануға болады.



Электрден жұмыс істейтін барлық двигательдер дәл осындай сымды катушкалар мен магниттерден тұрады. Төмендегі суретте двигательдің ішкі құрылысы көрсетілген. Шеттерінде катушкалар орналасқан, ортасында магниттер, егер ортаны айналдырсақ, онда электр тоғы өндіріледі.



Өзіңнің жел генераторыңды жасау үшін осындай двигателге тек вентилятордың қалақтарын бекіту жеткілікті, егер оларды айналдырса, двигателге қосылған шам жанады

Жел электростанциялары осылай жұмыс істейді. Қазақстанда жылу электростанциялары өте көп таратылған (қысқаша ЖЭС), олар көмірдің жануы арқылы өндірілетін жылудан жұмыс істейді. Олар жел электростанцияларына өте ұқсас, тек желдің орнына буды пайдаланады. Көмір жанғанда үлкен сыйымдылықта су қайнайды да бу шығады, бу соншалықты көп болғандықтан жел сияқты турбинаны айналдырады да, тоқ өндіреді.

Алайда барлық бұл электростанциялар айнымалы тоқ өндіреді, айнымалы тоқ дегеніміз не және ол неге осылай аталады? Жоғарыда айтқандай, катушкаға магнитті жақындатқанда, электрондар бір жаққа тез жылжиды да электр тоғы пайда болады, біз катушкадан магнитті алғанда, электрондар басқа жаққа жылжиды да, электр тоғы да пайда болады, егер үнемі магнитті жақындатып, алып отырсақ, онда тоқ алдымен бір жаққа, сосын өз бағытын үнемі өзгертіп басқа жаққа жылжиды, мұндай тоқ айнымалы деп аталады. Көптеген электроқұралдар осындай тоқпен жұмыс істейді, мысалы шәйнек, шам немесе желдеткіш.

Электроника сияқты күрделі құрылымдар оларда тоқ бағытының өзгеруін ұнатпайды, олар ондай тоқтан күйіп кетуі мүмкін, оларға тоқ әрқашанда бір бағытта болу керек. Мысалы, егер мобильді телефонды розеткаға тікелей қосса, ол бірден күйіп кетеді, сондықтан телефонды зарядтау үшін қоректену блогын пайдалану қажет (оны зарядтау, заряд т.б. атайды), қоректену заряды телефон үшін розеткадағы айнымалы тоқты тұрақты тоққа айналдырады, бұл тоқ әрқашанда бір бағытта қозғалады.

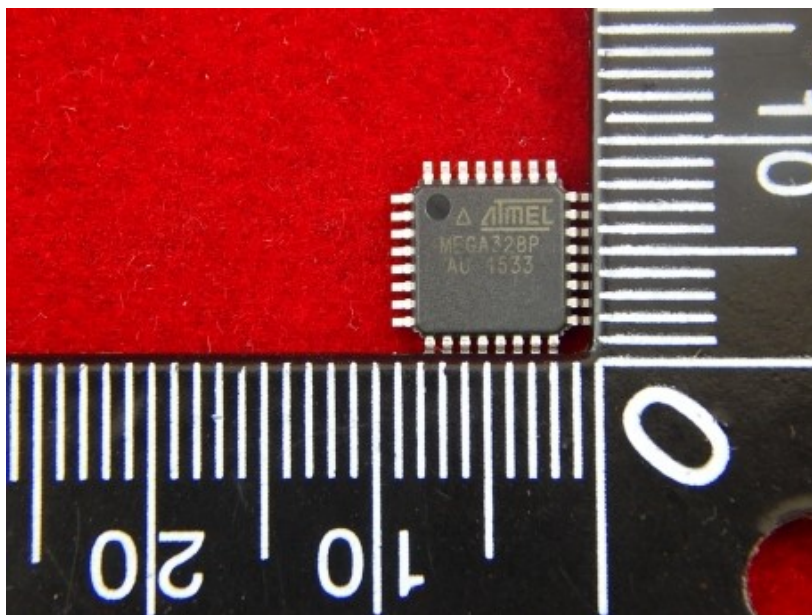
Барлық батареялар тұрақты тоқ өндіреді, оның электрондары оң зарядтан теріс зарядқа жылжиды (плюс (+) минус (-)). Күн батареялары тұрақты тоқ өндіреді.

Бірақ батареялар электростанция сияқты жұмыс істемейді, оларда магниттер де қолданылмайды. Батареялардың ішінде 2 химиялық зат бар, дәрік айтқанда осы заттардың бірі басқаға әсерін тигізеді де (жегідей жейді, ажыратады, ол ыдырап кетеді), электр тоғы өндіріледі, сондықтан батареялар ұзақ емес, олардың ішіндегі заттар бір-бірімен түгелдей әсерін тигізеді, сондықтан батареяны лақтырып тастау керек, өйткені одан пайда болмайды. Аккумуляторлардың зарядталу қасиеті бар, олар зарядталғанда зат бөлшектері (иондар) қайтадан қалпына келеді де, аккумуляторды тағы да пайдалануға болады, бірақ олар толық қалпына келмейді және әр кезде азая береді, сондықтан орташа 500-3000 рет зарядқадан кейін, аккумулятор жарамсыз болады. Күн батареялары ұқсас жұмыс істейді, олар арнайы материалдан жарық түскенде электрондарды беріп, электр өндіреді. Алайда аккумуляторға қарағанда, күн батареялары едәуір ұзақ қызмет етеді.

Біздің сабақтарымызда негізінен тұрақты тоқ пайдаланылады. Алда айтылғандай электрондар кернеуден жылжиды. Кернеу вольтпен (V) өлшенеді. Сендер батареядағы 1.5V жазуын бірнеше рет көрдіңдер, ол электрондар, бұл батареядан 1.5V кернеуімен жүретінін білдіреді. Тоқ нақты күшпен берілуі мүмкін, крандағы су сияқты қысым күшті болса, оған құбыр жуан қажет, тоқтың күші көп болса, сым да жуан керек. Егер сым тым жіңішке болса, ал тоқ күшті болса, онда сым қыза бастайды да, күйіп кетеді. Сол сияқты су үлкен қысыммен жіңішке құбырмен ақса, ол оны қысымнан жыртып жібереді. Электр тоғының күші Ампермен (A) өлшенеді.

Суға қойылған кран сияқты, оқты басқаруға сөндіргіштер көмектесетінін біз білеміз. Мысалы, кранды жартылай жабу сияқты, жылытқыштың температурасы не желдеткіштің жылдамдығын реттей отырып, оларға берілетін қысымды реттейміз.

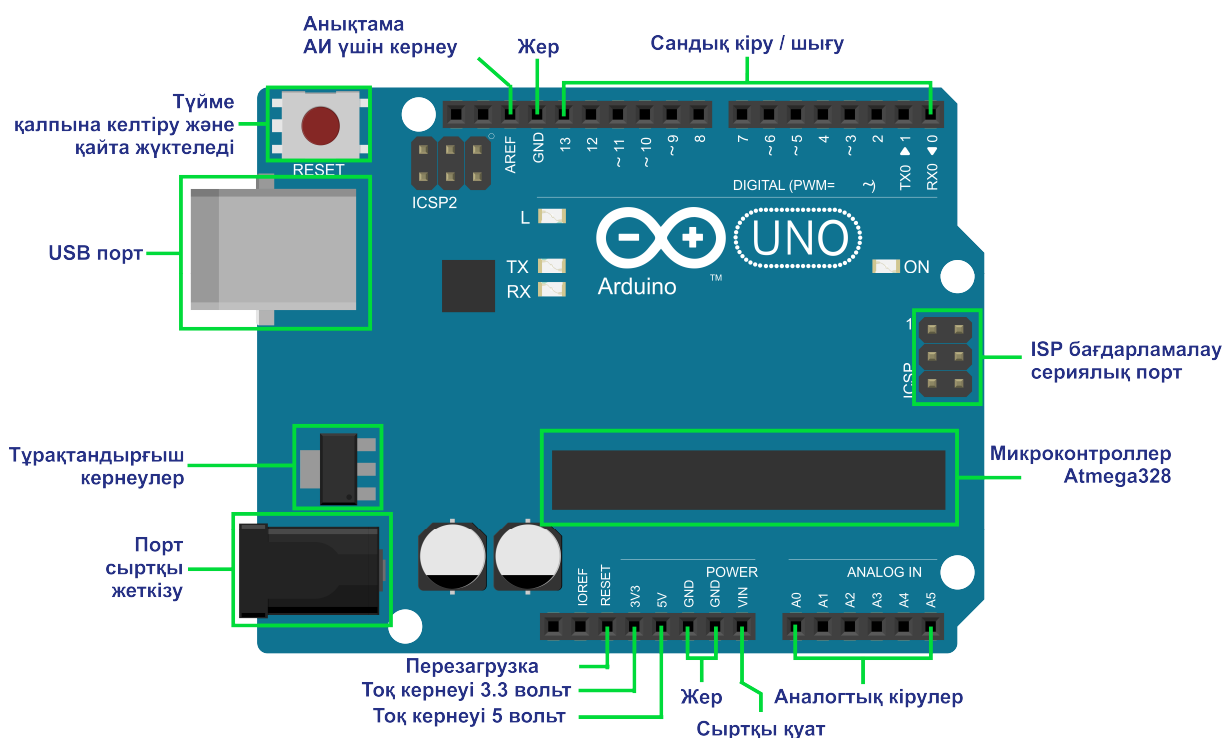
Барлық роботехника микроконтроллерлар көмегімен басқарылады, микроконтроллерлар оған салынған микросхемалар компьютер бағдарламасының көмегімен электрді басқара алады. Төмендегі фотографияда микроконтроллер көрсетілген.



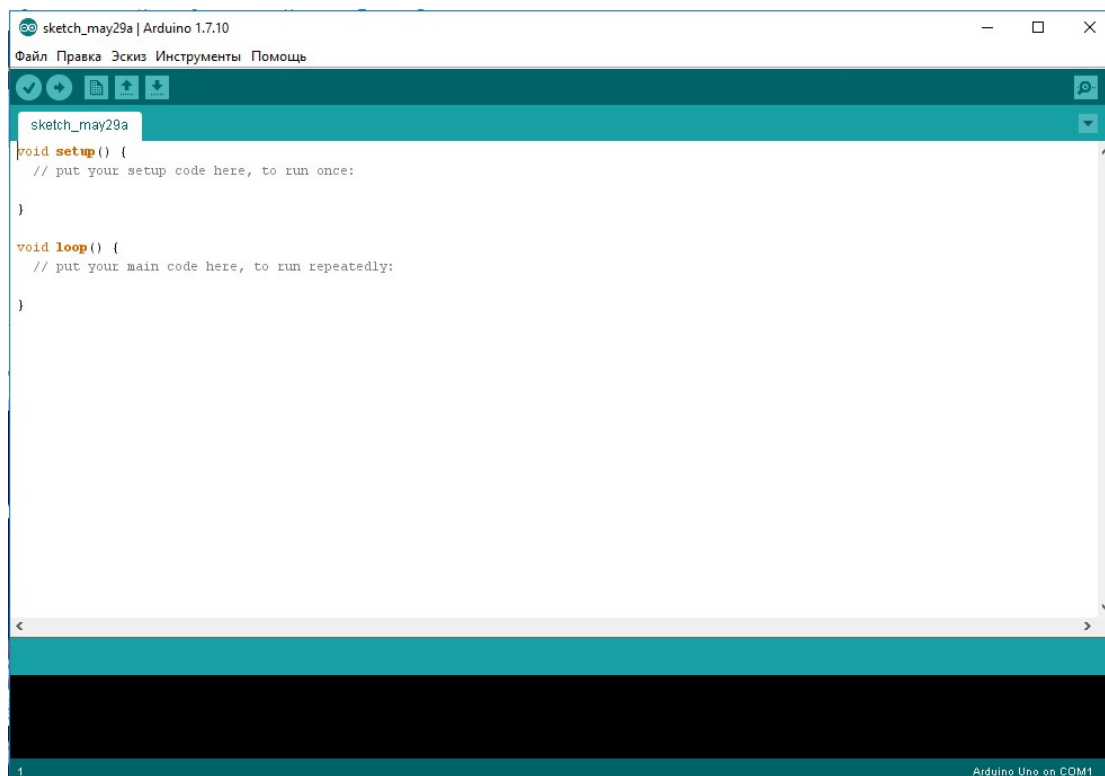
Осындай үлкен емес құрылымның көмегімен көптеген қызық заттарды жасауға болады, жарықты, өрт сигнализациясын автоматты басқарудан бастап. Пилотсыз ұшатын күрделі аппаратқа, тыңдауыш жучоктарға және көптеген басқаға дейін.

Егер оны жай компьютерге қоспай, кіші микросхемаға көп құрылымдарды қосуға және оны қалай бағдарлауға болады?

Ол үшін Arduino реттелген төлем әзірленген, бұл сол микроконтроллер, бірақ бізге үйреншікті алмалы-салмалы USB, коректену, сондай-ақ әртүрлі датчиктер мен құрылымдарды қосуға тұжырымдамалар. Arduino Италияның өзіндік өндірісі және басқа фирмалардың альтернативті ұқсастығы болады, алайда олар бір чипті пайдаланады, сондықтан оларда ешқандай айырмашылық жоқ, тек бағасында ғана. Төмендегі суретте осындай реттелген төлем көрсетілген.



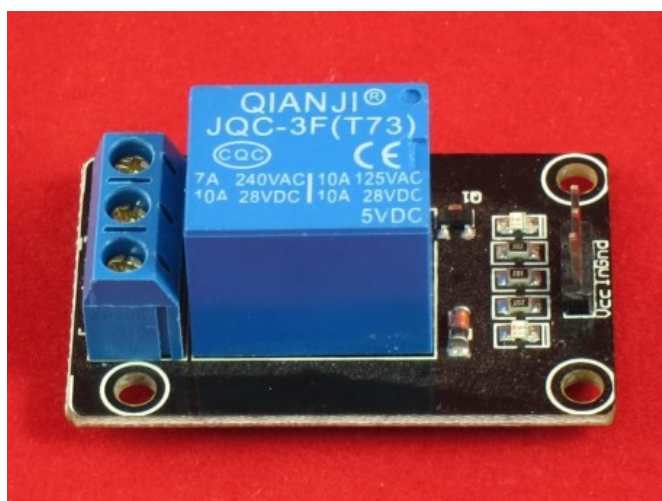
Ол компьютерге алмалы-салмалы USB көмегімен қосылып, Arduino IDE деп аталатын арнайы бағдарламаның көмегімен бағдарланады. Оны ресми сайттан <http://www.arduino.org/downloads> көшіріп, қондыруға болады.



Arduinoға 5 вольт сигналымен бақарылатын әртүрлі датчиктер мен құрылымдарды қосуға болады. Әдетте барлық датчиктар өте қарапайым принциппен жұмыс істейді. Мысалы жарық датчигі, егер жарық пайда болса датчик 5 вольтті көрсетеді, егер жарық болмаса, датчик ештеңе көрсетпейді, сонымен Arduino қазір қараңғы немесе жарық екенін түсінеді. Қозғалыс немесе от датчигімен де сондай, датчиктерден ақпаратты салыстырып оқылуға жоғары фотографияда көрсетілгендей Цифрлық порттар мен Ұқсас порттар керек. Осы порттардың әрбіреуіне датчикті қосуға болады, Arduino портқа қанша вольт берілгенін қарайды да, датчик жұмыс істей ме, істемей ме соны анықтайды.

Сонымен бірге Arduino құрылымдарды да басқара алады, мысалы оларды қоса да, сөндіре алады. Ол қалай болады? Ол үшін Arduino Цифрлық портқа 5 вольт береді, егер осы портқа светодиод қосылған болса, ол жанады. Біз 5 вольт берілуін басқара аламыз, нақты уақыт арқылы 5 вольтті беруімізге болады, немесе жарық датчигі жұмыс істеген жағдайда, не басқа жағдайларда. Бұл бағдарламаның көмегімен жасалады.

Тек светодиодтармен ғана емес, қуатты аспаптарды қалай басқару керек, мысалы егер оларға 5 вольт емес, 220 вольт керек болса. Олар үшін арнайы сөндіргіштер реле деп аталады, бірақ оларда әдеттегі сөндіргіштердегідей кнопкалар жоқ, кнопкалардың орнына бұл сөндіргішке 5 вольт беріледі де, ол құрылымды күштірек қосады, мысалы шәйнек, жылытқыш, желдеткіш, т.б. Реленің модулі төменгі фотографияда көрсетілген



Сол жақта күшті электр аспаптарын қосу контактылар орналасқан, ал оң жақта Arduino қосу үшін аяқтар орналасқан.

Arduino қандай портқа болмасын 5 вольт беру үшін не істеуі керек?

Ол үшін өз мұқтажыңа Arduino төлемін бағдарландыру қажет. Бағдарлама компьютерге түсінікті команда жиынтығынан тұрады. Бір қарағанда олар күрделі де түсініксіз болып көрінеді, егер осы командаларды ағылшыннан орыс тіліне аударсақ, олар есте сақтауға өте жеңіл.

Мысалы delay() командасы ағылшыннан “кідіріс” деп аударылады және бағдарламаны қандай да уақытқа тоқтату үшін қызмет етеді. Мысалы delay(1000) бағдарламаны 1000 миллисекундқа тоқтатады, яғни 1 секундқа.

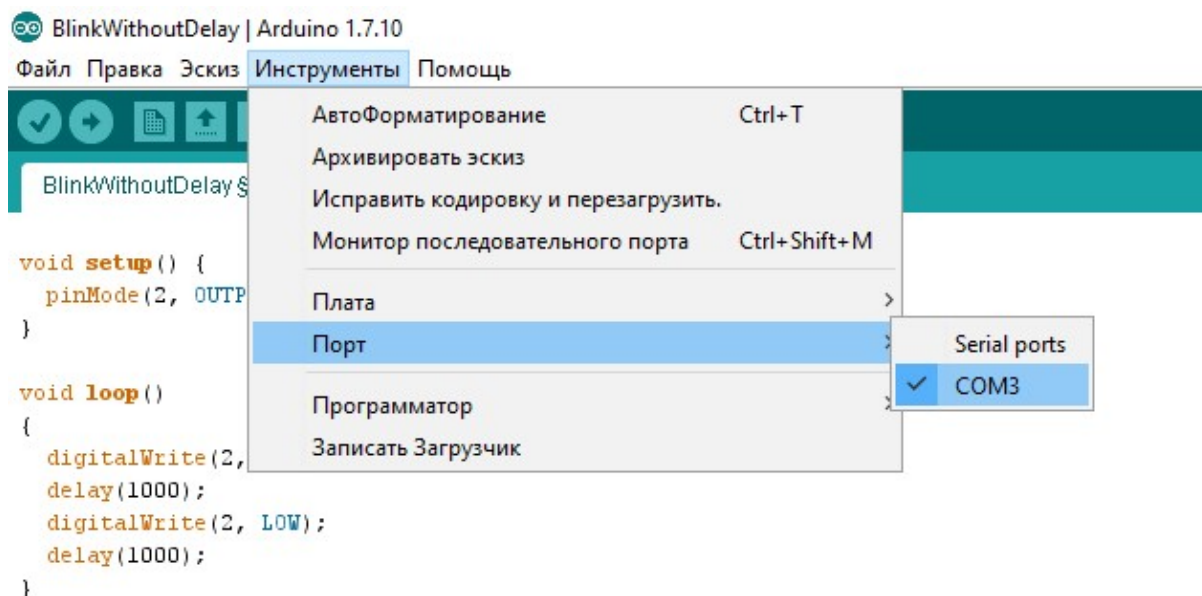
Светодиодты Arduino-ға, цифрлық №2 шығуға қосайық, Arduino-ға тестік бағдарламаны толтырайық. Ол үшін Arduino IDE ашамыз



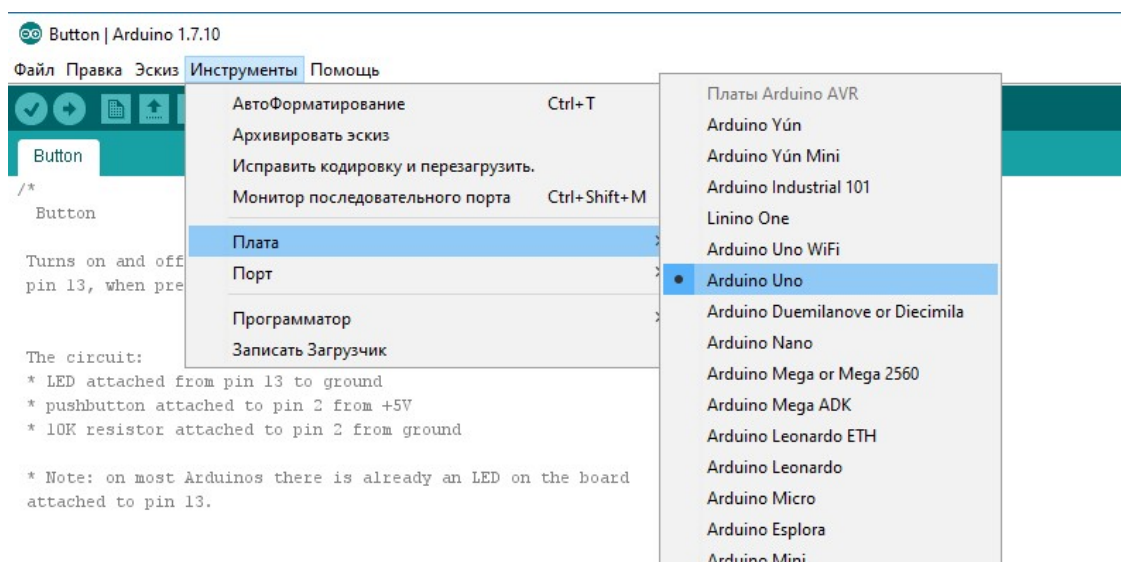
Мұнда жазылғанның бәрін өшірейік, төменде көрсетілген кодты жазайық

```
void setup() {  
  pinMode(2, OUTPUT);  
}  
void loop()  
{  
  digitalWrite(2, HIGH);  
  delay(1000);  
  digitalWrite(2, LOW);  
  delay(1000);  
}
```

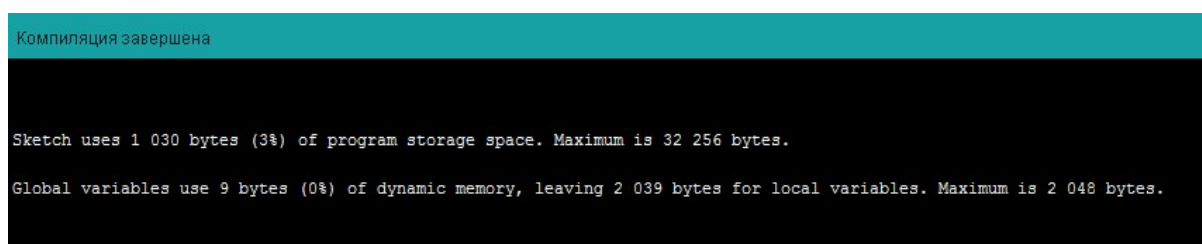
Енді Arduino-ны компьютерге қосайық, одан кейін **Инструменты -> Порт** менюіне кірейік, егер Arduino қосылған және драйверлер орнатылған болса, бізде COM порт көрінеді, ол COM1, COM2, COM3 және т.б. (әркімде әр түрлі), болып аталуы мүмкін, төменде көрсетілгендей, біз оны таңдаймыз.



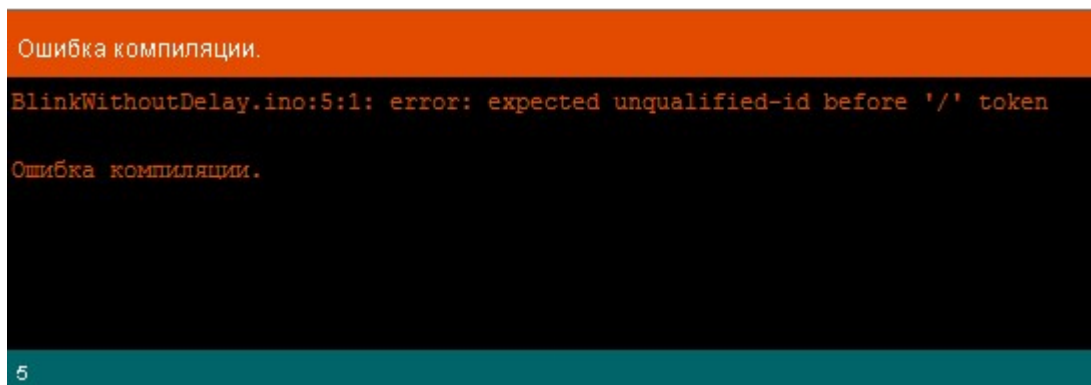
Енді Arduino түрін таңдау қажет, біздің сабақтарымызда Arduino UNO төлемі қолданылады, сондықтан **Инструменты** -> **Плата** менюі, төменде көрсетілгендей **Arduino Uno** таңдаймыз



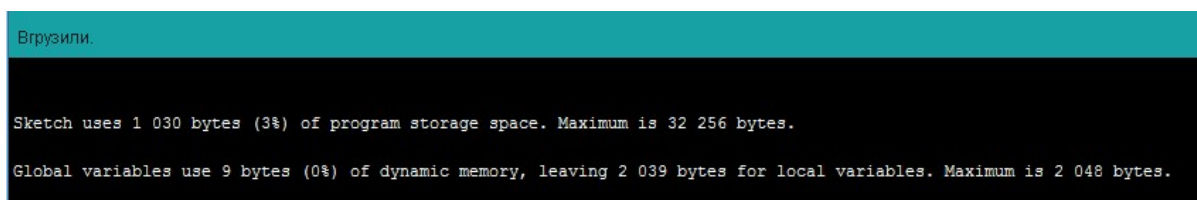
Енді біздің кодымызды тексеріп, Arduinoға салайық. Кодты тексеру үшін  галочкасы көрсетілген кнопканы басу қажет. Енді Arduino IDE біздің код қателерін тексереді, егер қате болмаса, біз қара өрісте төменде мынандай хабар көреміз: “Компиляция аяқталды”. Бағдарлама код қателерін тексеру үрдісі бағдарлауда “Компиляция” деп аталады, кодтың қателерін тексеру мен бағдарламаны жіберуге дайындау “Бағдарламаны өзі талдамай- талғамай компиляция жасау” деп аталады.



Егер кенеттен компиляция сәтсіз өтсе, кодта қателер болса, онда төменде “Компиляцияның қатесі” деген хабар шығады, төменде қызғылт сары мәтінде қандай қате және қандай жолда жіберілгені сипатталады.



Бұл жағдайда қате №5 жолда. Сіз әдейі қате жасап, компилятордың қалай жұмыс істейтінін тексеруге болады (аталады, бұл жағдайда Arduino IDE компилятор). Егер қате болмаса, кодты Arduino Unoға салуға болады. Ол үшін салу кнопкасын басу қажет, ол жасыл түсті,  онда оң стрелка бейнеленген. Кодты ойдағыдай салғаннан кейін төменде “Салынды” деген хабар шығады.



Құттықтаймыз, Сіздің бірінші кодыңыз Arduino Uno төлеміне салынды және жұмыс істеп жатыр. Егер қазір светодиодты №2 цифрлық кіруге қоссақ, ол секундына бір рет жыпылықтай бастайды. Неліктен секундына бір рет? Өйткені біз солай жаздық. Енді жазған кодты толығымен талдайық, ретімен жүреміз: Arduino кодында әрқашанда `setup()` және `loop()` блоктары бар, бұл блоктар функция деп аталады, олар былай жазылады:

```
void setup() { ЗДЕСЬ ЛЮБОЙ }  
void loop() { ЗДЕСЬ ЛЮБОЙ КОД }
```

Алдымен `void` сөзі жазылады, ол нені білдіретінін кейінірек талдаймыз, ары қарай `setup` сөзі жазылады, одан кейін міндетті екі жақша “()”, ашылатын және жабылатын, ішінде ештеңе жоқ, функция `loop` та солай жазылады, `loop` сөзінің алдында `void` сөзі ашық жермен жазылып, `loop` сөзінен кейін екі бос жақша қойылады “()”. Ары қарай байқағандай фигурлы жақшалар, ашылатын және жабылатын “{ }”. Бұл жақшалар ең маңызды! Біз оның ішіне әрқашанда біздің кодты жазамыз. **Loop** және **setup** түсініксіз функциялар неге керек?

Атаулардан **loop** “ілмек”, ал “**setup**” “**ікемдеу**” деп аударылатынын байқадық. Бұл функциялар өз атауларына сай. **Setup** ішіне біз жазатын код, яғни **setupтан()** кейін келетін фигурлы жақшалар { ФИГУРЛЫ ЖАҚШАЛАР ІШІНДЕ } Arduinoны біз қосқанда 1 рет қана орындалады. Мысалы, егер светодиодты қосатын және сөндіретін кодты жазып, оны **setupке()** { } орналастырып, Arduinoны қосқанда, светодиод жанады да сөнеді. **Loop()** ішіндегі код, яғни **loopтан** кейінгі фигурлы жақшалардың ішіндегі { ФИГУРЛЫ ЖАҚШАЛАРДЫҢ ІШІНДЕГІ } Arduino қосылып тұрған кезде, ол қайталана береді. Мысалы, егер Arduino қосатын және сөндіретін кодты **loopқа()** { } орналастырсақ, Arduino қосылып тұрғанда, светодиод үнемі қосылады да, сөнеді.

Кодты ары қарай қарастырайық. **Setup()** { } функциясының ішінде мынандай бағдарлама жаздық:

```
digitalWrite(2, HIGH);  
delay(1000);  
digitalWrite(2, LOW);  
delay(1000);
```

Енді оны рет-ретімен талдайық. Мұнда **digitalWrite()** командасы бар, ол Arduino цифрлық портының қоректенуіне команда береді немесе сөндіреді, мысалы команда **digitalWrite(2, HIGH);** №2 портқа 5 вольт береді, ал **digitalWrite(2, LOW);** командасы №2 порттық қоректенуін сөндіреді. Егер заңдылықтарға көңіл аударатын болсақ, командалар тек **HIGH** және **LOW** сөздерімен ерекшеленетінін байқаймыз, олар ағылшын тілінен **ЖОҒАРЫ** және **ТӨМЕН** деп аударылады. Демек **digitalWrite(2, LOW);** командасы 2 портқа төмен қысым береді, ал **digitalWrite(2, HIGH);** командасы №2 портқа жоғары қысым (5 вольт) береді. Мысалы, егер **digitalWrite(3, HIGH)** деп жазсақ, Arduino цифрлы №3 портына 5 вольт береміз, осылай кез келген 0 ден 13-ке дейін цифрлы портқа 5 вольт беруге болады. Функция **digitalWrite** атауын еске оңай сақтау үшін она аудару керек, **digital** - цифрлы, **Write** – жазу, яғни цифрлы портқа жазу. Бірақ неге жазу керек? Шынында цифрлы 5 вольтқа береміз, жазамыз, шығарамыз немесе 0 вольт деп жазамыз. Мысалы, кернеу №2 портқа 5 вольт берілеміз, әлде жоғын білгіміз келсе, онда біз **digitalRead(2)** деп жазамыз, бірақ оны біз кейінірек зерделейміз.

Цифрлы портқа 5 вольт беру үшін **digitalWrite()** функциясын қолдану қажеттігін есте сақтау керек.

Сонымен қатар **digitalWrite()** командасы функция деп аталатынын байқаған боларсыздар, бірақ ол фигурлы жақшаларсыз “{}” жазылады және мұнда **void** сөзі жоқ? Неге олай? Функцияны жақшамен не жақшасыз жазуды қалай айыруға болады, бұл функция ма әлде функция емес екендігін қалай анықтауға болады? Шынында дөңгелек жақшалар келетін барлық командалар мұнда функциялар деп аталады, мысалы **delay()**, **serialWrite()**, **blablabla()** – бұның бәрі функциялар. Мұндағы айырмашылық сол, **loop()** және **setup()** функцияларында олардың ішіне жазылатын өз кодымызды өзіміз ойлап табамыз, сондықтан фигурлы жақшаларды жазып, олардың ішіне команданы көрсету керек.

Ал **digitalWrite()** функциясы Ардуинода орналастырылған, оны Arduino әзірлеушілері жазған, біз оның ішіне өз кодымызды жазбаймыз, оларды тек пайдалануға болады, өзгертуге болмайтын Arduino орналастырылған барлық функциялармен мына сілтеме <http://arduino.ru/Reference> арқылы танысуға болады.

Ары қарай байқағандай кодта **digitalWrite()** функциясынан кейін - **delay()** функциясы жазылады;

```
digitalWrite(2, HIGH);  
delay(1000);  
digitalWrite(2, LOW);  
delay(1000);
```

Орысшаға аударғанда, **delay** сөзі **кідірісті** білдіреді. Яғни **delay()** функциясы бағдарламада пауза жасайды. Пауза ұзақтығы миллисекундпен беріледі, бір секундта 1000 миллисекунд екенін ескертеміз. Яғни бағдарламада бір секунд пауза жасау үшін **delay(1000)**, екі секундқа пауза жасауға **delay(2000)**, т.с.с. жазу керек.

Енді бұл командаларды біліп, **loop(){}** функциясында орындалатын кодты түсіндіре аламыз.

```
void loop()  
{  
  digitalWrite(2, HIGH);  
  delay(1000);  
  digitalWrite(2, LOW);  
  delay(1000);  
}
```

```
}
```

Алдымен **digitalWrite(2, HIGH);** деп жазамыз, бұнымен 2 портқа 5 вольт береміз, оған светодиод қосылған, одан светодиод жанады. Ары қарай біз 1 секундқа **delay(1000)** пауза жасап, одан кейін 2 портқа **digitalWrite(2, LOW);** төмен қысым береміз, ары қарай 1 секундқа **delay(1000)** пауза жасаймыз; код қайталанады, өйткені біз оны **loop()** функциясында жазғанбыз.

Енді төменгіні жасауды байқап көріңіз:

1. Жарты секунд сайын светодиодпен жыпылықтау
2. Әр 2 секунд сайын
3. Светодиод жарты секунд жанып, 2 секундке өшуін жасаңдар
4. Светодиод 500 миллисекундтан 2 рет жыпылықтап, үшінші рет 2 секунд жыпылықтап, осы әрекетті қайталауын жасаңдар

Сонымен біз талдамаған **pinMode(2, OUTPUT);** кодтың соңғы жолы қалды, сендер байқағандай, ол **setup(){ }** функциясында орналасқан, жоғарыда айтылғандай ол тек 1 рет қана Arduino қосқан кезде орындалады. Бұл функция цифрлы №2 портты жазу режиміне қосу үшін қызмет етеді, яғни №2 портқа ЖОҒАРЫ не ТӨМЕН қысым беруге, егер оны жасамасақ кернеу бере алмаймыз. Яғни **pinMode()** функциясы порттың жұмыс тәртібін белгілейді, **pin** - “порт”, **Mode** - “тәртіп” деп аударылады.

№3 портқа 5 вольтті шығару үшін **pinMode(3, OUTPUT);** №4 портқа шығару үшін **pinMode(4, OUTPUT);** және т.с.с. жазу керек. Заңдылықтарға жүгінетін болсақ, тек порттың нөмірі өзгертінін байқаймыз.

Портты жазу тәртібіне көшіруге бұл функцияны тек 1 рет орындау жеткілікті, одан кейін портқа **digitalWrite(2, HIGH);** функциясын беруге болады.

Біз жазған кодтың әр жолы нені білдіретінін енді білеміз, әр жолға тағы да зер салып, олардың міндетін өздерің түсіндіріңдер

```
void setup() {  
  pinMode(2, OUTPUT);  
}  
void loop()  
{  
  digitalWrite(2, HIGH);  
  delay(1000);  
  digitalWrite(2, LOW);  
  delay(1000);  
}
```

Енді кодпен эксперимент жасаңдар:

1. Светодиодты 3 пинге қосыңдар, светодиод жыпылықтау үшін кодты қайта жазыңдар
2. **pinMode();** жолын өшіріп көріңдер, не болатыны байқаңдар
3. **setup()** функциясынан **loop()** функциясына **pinMode();** функциясын көшіріп көріңдер. Не өзгерді? **pinMode();** функциясын қайда жаздық, айырмашылығы неде?
4. Светодиод секундына 50 рет жыпылықтайтын қылыңдар, оның жыпылықтауын сіздер мүлдем байқамауларыңыз керек.

№2 сабақ

Тақырыбы: Электроника және программалауға кіріспе

Электрон: Электр тоғы поляның қысым әсерімен өткізгіште электрондардың реттелген қозғалысы екенін өткен сағатта білдік. Электрон дегеніміз не? Бәріміз атом мен молекуладан тұрамыз, бұл өзара жалғасқан кішкене-кішкене бөлшектер, электрондар атомдардан да кіші, олар атом құрамына кіреді және оның ядросын айнала қозғалады, осылай атомды бейнелейді: ортасында атомның ядросы және оны айналатын электрондар



Ядроның айналасындағы сызықтар бұл жолдар (траекториялар), бұл жолдармен электрондар қозғалады. Бірақ бізге оларды әдетте суретте бейнелейді, шынында бұл сызықтар жоқ, электрондар шеңберде қозғалмайды, олар өте жылдам және бытыраңқы астан-кестен болған тәртіпте қозғалады, олар ары қарай қайда жыл-житынын есептеу және болжау мүмкін емес. Электрондар суреттегідей төртеу болу міндетті емес, әртүрлі атомдарда оның саны әртүрлі, мысалы оттегі атомында олар 8, темір атомында олар 26, алтын атомында олар тіпті 79! Бірақ электрондардан басқа атом протоннан да тұрады, протондар атомның ядросында болады.

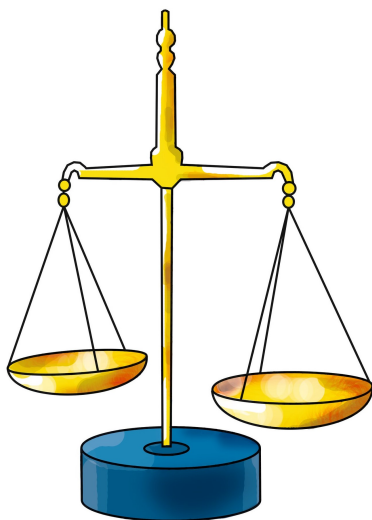
Сонымен бірге суретте электрондардың орналасуы теріс бейнеленген, атап айтқанда олар ядродан 100 000 рет алысырақ, егер біз оны суретте бейнелесек, онда ядро алыста болып, осындай кіші бейнелеуде біз оны көрмес едік және бейнелеуді соншалықты үлкейту керек болып, мына бетке сымас еді, бізге шамамен 30 шақырымға жететін бет-қағаз керек болар еді, сонда олардың алыс орналасқанын көреміз, сондықтан атомды осылай бейнелейді. Электрондар мен протондар қандай екенін ешкім де білмейді, өйткені осы кезеңде технологияның дамуы оларды көруге мүмкіндік бермейді.

Өткізгіштермен электрондар қалай жылжитынын талдайық, атомдар онда орнында қалады, сендер кабельдің ішіндегі металл электрмен бірге аққанын ешқашан көрмедіңдер ғой? Жоқ, атомдар әрқашан орнында қалады, электрондарға сыртқы поле әсер еткенде, олар атомның тартуынан шығып, ары қарай жылжиды, мәнінде атом оларды мүлдем жоғалтпайды, атомның жанында басқа да атомдар бар, олар да электрондарды жоғалтады. Олардан шыққан бос электрондар қозғалып, біздің атомның әсеріне түседі де, өзіне жетіспейтін электрондарды тартады.

Енді оның қалай болатынын ретімен талдайық. Сымды розеткаға қосқанда, розеткадағы электрондар атомнан электрондарды итере бастайды да, олардың орнына тұрады, итерілген электрондар ары қарай жылжи бастайды да, көрші атомдардан электрондарды итеріп, олардың орнына тұрады және солай жалғаса береді. Электрондар шамға жеткенде, ол жанады. Неліктен сым жанбайды, ал шам жанады? Электрондар қозғала бастағанда, олар жылу өндіреді, егер

электрондар әдеттегі сыммен қозғалса, онда жылу күшті болады, өйткені жуан сымды жылыту үшін ол өте аз, бірақ шамда арнайы металл жібі орнатылған, әрі өте жіңішке, ол күшті қызады да, жана бастайды. Қысым соншалықты көп болса, электрондар күшті қозғалады да шам жарығырақ жанады, егер қысым өте көп болса, шам күйіп кетеді, егер тым аз болса, шам күңгірт жанады. Қысым вольтпен өлшенетінін еске саламыз.

Тыныш жағдайда электрондар ешқайда шықпайды және сымда ток жүрмейді, электрондар ядроның тарту әсерінен шыға алмайды, өйткені ядрода протондар бар, олар электрондарды магнит сияқты тартады, бұл электрондарда теріс заряд (-), ал протондарда оң заряд (+) бар. Магниттің қарама-қарсы жақтары бір бірін тартатынын сендер эксперименттеген шығарсындар, сол сияқты протондар да электрондарды тартады. Тыныш жағдайда атомда электр заряды жоқ, бірақ мысалы оны қыздырса не ұйқаласа, онда атомнан электронды тартып алуға болады да, ол оң зарядталады, өйткені протондардың зарядтары (+) электронның зарядтарынан (-) көбірек болады, өйткені біз электронды жоғалттық та, тепе-теңдік сәл жағымды жаққа өзгерді. Бұл таразының оң мен сол жағына таразы тең болу үшін көп гирьді салумен тең (тыныш атом) сосын

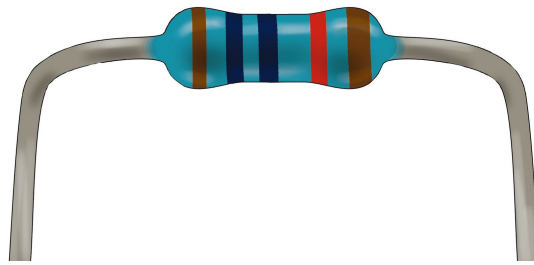


бір гирьді алсақ, таразы бір жағына қисая бастайды (оң зарядталған атом).

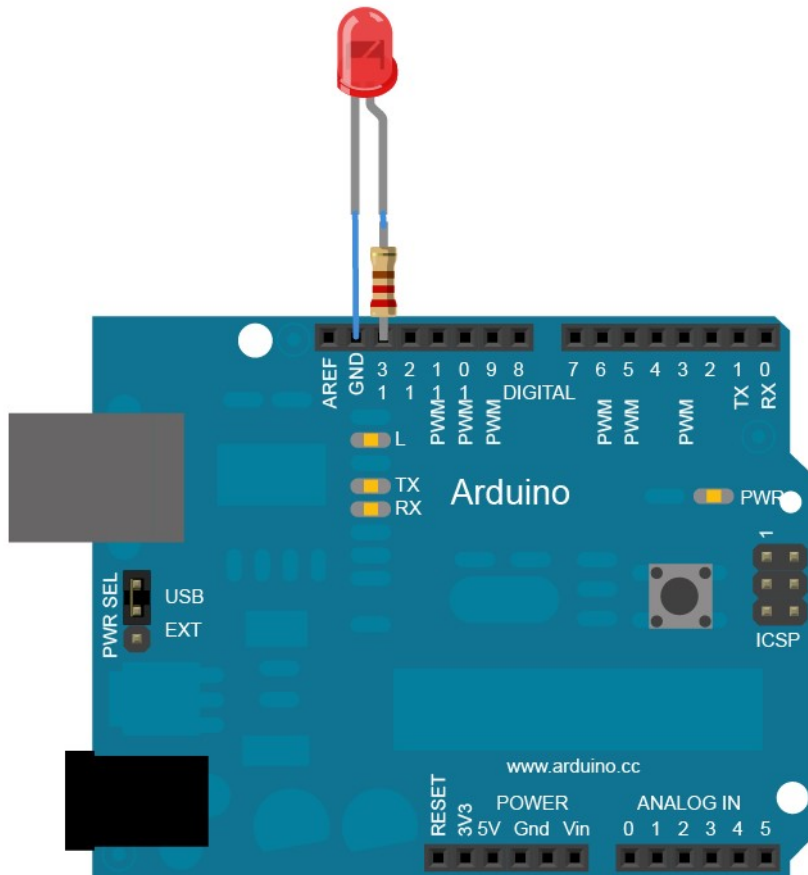
Егер атомға электрондарды қоссақ, онда оны тағы да тепе-теңдіктен шығарамыз, бірақ басқа жағына, онда электрондар көбірек болады да, атом теріс зарядталады, мұндай атом кішігірім әсерден барлық күшімен артық электроннан тепе-теңдікке жету үшін арылады, дәл осылай оң зарядталған атом барлық күшімен оған жетіспейтін электрондарды тартады. Мұны бітірдік, енді кейбір заттар тоқты жеңіл де тез өткізетіні туралы сөйлесейік, мысалы, темір, мыс, алюминий, алтын, т.б. Мысалы, ағаш, пластик, резеңке оларды өткізбейді.

Мұндағы іс неде: біз айтқандай әртүрлі заттарда атомдағы электрондар саны әр түрлі, сонымен бірге протондар саны да түрлі, егер атомда протондар көп, электрондар аз болса, онда протондар электрондарды көп тартады, егер біз алып, оған әсер етіп, одан электронды тартып алуға ұмтылсақ, одан ештеңе шықпайды және электр тоғы мұндай атомдар арқылы ақпайды, өйткені бірде-бір электрон орнынан қозғалмайды. Сымдар жасалған металлдар атомдағы шығатын электрондар саны көп, сондықтан олар арқылы ток жақсы өтеді. Металл еместе барлығы керісінше, қолайлы жағдайда шыға алатын электрондар мүлдем жоқ, сондықтан оларға әсер еткен кернеу оларда ток шығармайды. Сондықтан резеңкеге қапталған сым ток ұрмайды, электрондар атомнан шықпайды.

Кейде бізге қысымды төмендету талап етіледі, мысалы Arduino цифрлы шығуда 5 вольт береді, оған 3 вольтта жанатын бізге светодиодты қосу керек, егер біз оны 5 вольтқа қоссақ, ол күйіп кетеді. Бізге қосымды 3 вольтқа дейін төмендету керек. Оны резистордың көмегімен жасауға болады. Резистор шамамен мынандай:



Егер светодиодты Arduinoға тікелей қоспай, резистор арқылы қосса, төменде көрсетілгендей, онда кернеу төмендейді де, светодиод күйіп кетпейді.



Резистордан кейін қысым 3 вольтқа өзгереді де, светодиод ұзақ та мәнді жанады. Неліктен бұлай болады, неліктен резистордан кейін қысым 3 вольтқа дейін төмендеді?

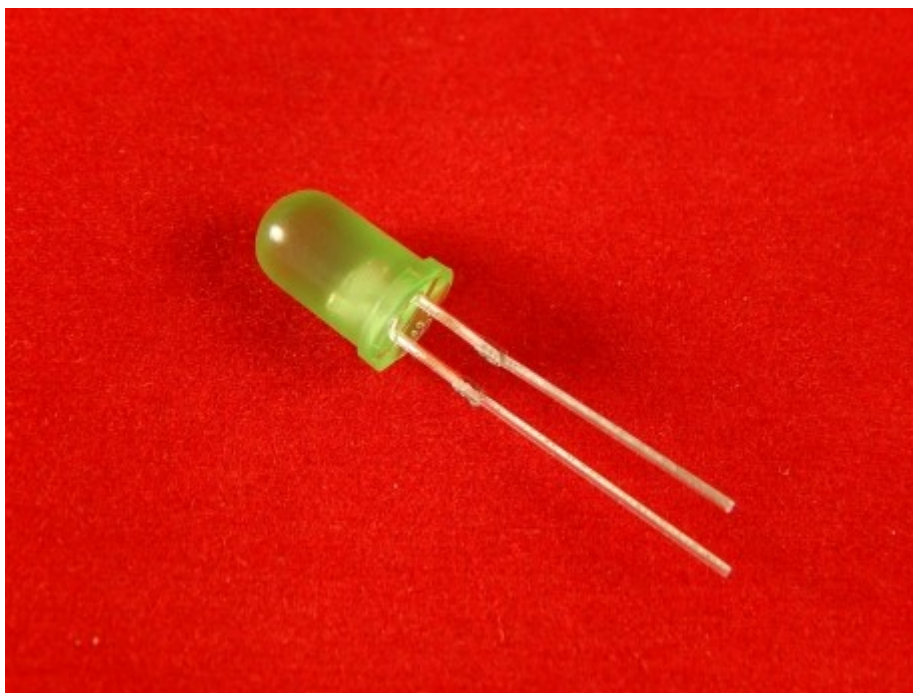
Атап айтқанда, резисторлардың іші металлдардан тұрады, олар арқылы электр тоғы өткенде қызады, қызғанда токтың бір бөлігін жұмсайды да, оны жылуға айналдырады, сондықтан резистордың шығуында ток кіруіне қарағанда аз болады. Резисторлардың басқа түрлері электрондарды жартылай блокадейтын, жартылай жіберетін арнайы заттардан тұрады, мұндай заттар жартылай өткізгіштер деп аталады, өйткені олар жартылай ток өткізеді. Сондықтан электрондар бөлігін блокадей, олар резистор қарсылық көрсетіп, токты шектейді де, қысым төмендейді. Резисторды тағы да басқаша кедергі деп атайды. Бірақ резисторлар әртүрлі болады, біреуі күшті қарсы тұрса, екіншісі нашар, жүздеген түрлі резисторлар бар, қандай бізге керек екенін қалай білуге болады? Мұнда бәрі қарапайым, мысалы сендер батарейкалар 1.2 вольт, 1.5 вольт, 3.7 вольт, 9 вольт болатынын, қандайын қайда қосудың білесіңдер және ешқашан шатастырмайсыңдар. Резисторлармен де солай, барлығында да өзінің қарсыласуы бар және ол Оммен өлшенеді, мысалы, біздің светодиод үшін қысымды 5 вольттан 2.8 вольтқа төмендетуге 220 Ом кедергісімен резистор керек.

Неліктен осындай кедергімен резистор керек, басқа светодиод үшін қандай резистор керек? Ол үшін кедергі есептеу формуласы бар, ол Ом заңына негізделген, әзірше заңды біз

сипаттамаймыз, формуланы талдап көрейік. Формула өте қарапайым: Кедергі = Кернеу/Ток күші (немесе $R = U / I$).

Енді осы формуланы қалай қолданып, тоқты төмендету үшін бізге керек кедергіні қалай есептеуге болады.

Мысалы, бізде мынандай жасыл түсті светодиод http://RadioMart.org/_904 бар деп болжайық.



Оны бейнелеуде жазылған

Цвет свечения: **зеленый**

Напряжение в прямом направлении (В): DC1.9 ~ 2,5

Прямой ток (мА): 4—14

Интенсивность освещения : 100-200mcd

Длина волны (нм) :520 - 530

Бұл сипаттамадан бізге Кернеу және Ток керек (сол және тоқтың күші). Неліктен бұл параметрлер? Өйткені формулада солар керек “*Кедергі = Кернеу/Ток күші (немесе $R = U / I$)*”.

Светодиодтың бізде кернеуі 1.9 вольт – 2.5 вольт, осының орташасын 2.2 вольтті аламыз. Бұл светодиодтың қолданатын тоқтың күші 4-14 мА (миллиамперге) тең, орташа 9 мА мәнін аламыз. Айтпақшы, бір амперде (А) 1000 миллиампер (мА).

Сонымен светодиодтың 9мА қолданатынын немесе басқаша 0.009 А және оның жұмыс істеуіне қажетті кернеу - 2,2 В.

Сонымен бірге Arduinoға шығуға 5 вольт берілетінін, сондықтан бізге кернеуді 2.2 вольтқа төмендету керектігін білеміз. Яғни бізге артық 2.8 вольтті кеміту керек (5 вольт – 2.2 вольт=2.8 вольт) біз формулаға мәндер қоямыз:

$Кедергі = Кернеу/Ток\ күші$	$Кедергі = 2.8\ вольт/0.009А$
------------------------------	-------------------------------

Егер 2.8-ді 0.009ға бөлсек, шамамен 311 болады – бұл резистордың кедергісі, бізге светодиод күйіп кетпеу үшін номиналы 311 Ом болатын резистор керек.

Енді сендер светодиод үшін керек резисторды есептеп шығуға аласыңдар. Әр кезде есептеуге уақыт кетірмеу үшін көптеген калькуляторлар шығарылған, оған тек светодиодтың параметрлерін және неше вольттан қоректену керектігін беру жеткілікті, калькулятор бәрін есептейді, осындай калькулятордың бірін мына <http://cxem.net/calc/ledcalc.php> сілтемеден табуға болады.

Напряжение источника питания (В):

Прямое напряжение светодиода (В): [выбрать]

Прямой ток светодиода (мА): [выбрать]

Количество светодиодов:

☒ Схема подключения
 ☐ Принципиальная схема

☐ Другое расположение светодиодов
☐ Сохранить вычисления на картинку
☐ Резистор 4 полосы

Мында қажетті параметрлерді толтыруымыз керек:

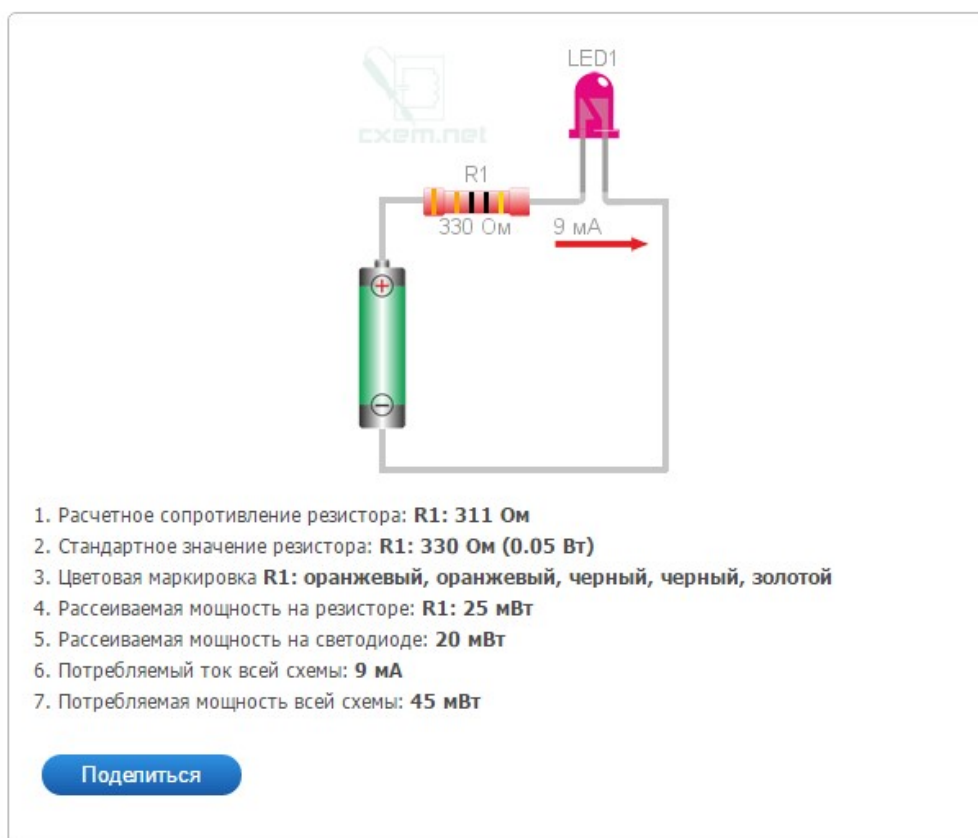
Қоректену көзінің кернеуі – бізде қоректену көзі Arduino, ол 5 вольт береді, сонымен мында 5 цифрын жазамыз

Светодиодтың тік кернеуі – светодиодтың кернеуі 2.2 вольт екенін RadioMart сайтындағы сипаттамадан білдік

Светодиодтың тік тоғы – светодиодтың 9 миллиампер қолданатынын сол сипаттамадан білдік, мында 9 цифрын жазамыз

Светодиодтар саны – бізде ол 1, сондықтан мында бірді жазамыз.

Енді есеп кнопкасын басамыз, одан біздің бетімізге келесі нәтиже шығарылады:

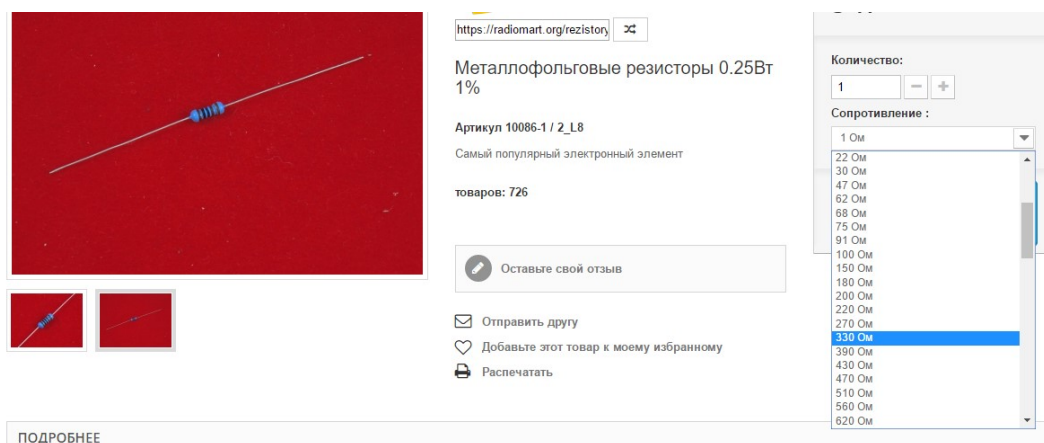


Біз есептегендей, нәтижеден резистордың есептік кедергісі дәл **311 Ом** екенін көреміз.

Мында резистордың стандарттық кедергісі **330 Ом-ға** тең деп жазылған, бұл нені білдіреді?

Зауыттарда резисторлардың барлық номиналдарын шығара бермейді, тек ең танымал, 311 Ом, 312 Ом, 313 Ом резисторларын шығарады, басқалар жоқ. Сондықтан егер сізге керек резистордың кедергісі болмаса, онда бар кедергіні үлкенірек алып, одан светодиода жамаң болмайды, ол күңгірт жанады, өйткені 330 Ом мен 311 Ом арасындағы айырмашылық өте аз, сондықтан күңгірттік мүлдем байқалмайды.

Тағы да осындай светодиод бар ма, әлде жоқ па, қалай анықтауға болады? Ешбір емес, жақын радиобөлшектер дүкеніне барып, сендерге керек резисторды сұрайсыз. Немесе ешқайда бармай http://RadioMart.org/_905 резисторлар бөліміне кіресіз де, қандай резисторлар бар екенін қарайсыз:



Номиналы кіші резисторды қосып көруге болады, мысалы 270 Ом, онда светодиод жарығырақ жанады, күймеуі де мүмкін, өйткені есептегенде кернеудің орта мәнін және тоқтың орта мәнін алдық, бізде кішігірім қор бар. Егер сендер светодиод барынша жарық жансын десеңіз, онда есептегенде оның максималды мәнін алу керек, мысалы жасыл светодиодаың сипаттамасына оралайық.

Цвет свечения: **зеленый**

Напряжение в прямом направлении (В): DC1.9 ~ 2,5

Прямой ток (мА): 4—14

Интенсивность освещения : 100-200mcd

Длина волны (нм) : 520 - 530

Осыдан есептеу үшін 2.2 вольт емес, 2.5 вольт алу керек, тоқты 9 емес, 14 мА алу қажет. Бізге керек кедергіні есептесек, бізде 180 Ом шығады. Біздің светодиода осындай резистормен күйіп кетпейді. Ал 270 Ом резисторы одан да артығырақ. Кернеудің ұсақ жыпылықтауында светодиод жанып кетпеу үшін номиналы сәл үлкен резисторды алған жақсы.

Резистордың кедергісін қалай таңдау керек және светодиода үшін неліктен 220 Ом қолданғанымызды енді сендер білесіндер.

Мультиметрмен танысу.



Бүгін біз мультиметр деп аталатын құрылыспен танысамыз. Мүмкін онымен таныс шығарсындар. Әртүрлі адамдар оны түрлі атайды: цэшка, прибор, тестер және т.б. Оның қалай аталатын маңызды емес, ең бастысы ол туралы екенін түсініңдер. Ол неге керек. Оның атауы МУЛЬТИ (әртүрлі) МЕТР (өлшеу), мультиметр әртүрлі көлемді өлшеуге жасалғанын аңғаруға болады. Оның көмегімен тоқтың тұрақты және айнымалы кернеуін, тұрақты тоқтың күшін өлшеуге, резистордың кедергісін білуге, тұрақты тоқтың тізбегінде плюс пен минусты анықтауға, қысқа тұйықталуды табуға және көп басқаларды табуға болады. Бүгін біз **DT-830** мультиметрмен танысамыз, ол шамамен мынандай:



Мультиметр дисплейі мен жұмыс режимін ауыстыруға бұратын тұтқа, оған 2 шұп қосылған, оның көмегімен өлшеу жүргізетін құрылғы. Бұл шұптар алуан түрлі, әдетте қара және қызыл.



Олардың айырмашылығы жоқ, қолайлық үшін түрлі түстерден жасалған. Әдетте қызыл түсті плюске, қараны – минуске қосады.

Сонымен, мультиметрде шұптарды қосатын 3 ұя бар, шұптар екеу ғана. Олар тек 2 төменгіге, қызыл ортаудағы ұяға, ал қара төменгіге қосылады. Онда жоғарғы ұя не үшін. Робототехника және микроконтроллерлармен жұмыс істегенде олар сирек қолданылады. Ол күшті тоқты өлшеу үшін қажет, 10 Амперге дейін, ондай жағдайлар сирек кездеседі, сондықтан оған кейін тоқталамыз. Сонымен рет-ретімен, тұрақты тоқтың кернеуін қалай өлшеуге болады?

Ол үшін мультиметрдің жұмыс кестесі бар, ол DCV (direct current voltage – тұрақты электр тоғының

кернеуі) деп аталатын топқа жатады, мультиметрде DCV жазуымен не V әрпімен, пунктир және түзу сызықпен белгіленеді. Мұнда бізде 5 режим (1000, 200, 20, 2000m, 200m), олар бір-біріне ұқсас, мысалы егер бізге 1.5 вольт батарейканың кернеуін өлшейтін болсақ, бұл режимге 2000m, 20, 200, 1000 жарайды, егер осы режимдердің біреуіне ауысып, щуптармен батарейканың плюсы мен минусына тисек, онда қанша вольт екенін көреміз. Бірақ бұл режимдерде айырмашылық бар. Режимдердің біреуі жоғары кернеу үшін, екіншісі төменгіге керек, режим **1000** 200 вольттан 1000 вольтқа дейін кернеуді өлшеуге қажет. Режим **200** 20 вольттан 200 вольтқа дейін кернеуді өлшеуге қажет, режим **20** 2 вольттан (2000 милливольт) 20 вольтқа дейін кернеуді өлшейді, режим **2000m** 0.2 вольттан (200 милливольт) 2 вольтқа дейін (2000 милливольт) кернеуді өлшейді, соңында режим **200m** 0 вольттан 0.2 (200 милливольт) вольтқа дейін кернеуді өлшейді.

Бұл режимдердің өз өлшеу диапазоны бар, неліктен 1.5 вольтті әртүрлі режимде өлшей аламыз, режимдер арасындағы айырмашылық олардың өлшеу дәлдігіне қатысты, мысалы, егер 1.5 вольтті **2000m** режимде өлшейтін болсақ, онда мультиметр бізге 1500 көрсетеді – егер батарейка толығымен зарядталған болса, неліктен 1500, бізде 1500 емес, 1.5 вольт ғой, бұл режим барлығын миллиамперде көрсетеді, оған (2000m) жазуы сай келеді. Егер 1.5 вольтті **20** режимде өлшейтін болсақ, онда мультиметр 1.50 көрсетеді, өйткені мұнда дәлдік жүздік дояда өлшенеді, мультиметрге басқа символдар сыймайды, **200** режимде 1.5 вольт көрінеді (мұнда нүктеден кейін тек бір цифр ғана сияды), ал **1000** режимінде 2 вольт көрінеді, неліктен 2, бізде бір жарым, ол режимде нүктеден кейін мүлдем цифр жоқ, мультиметр 1.5 мәнін 2-ге дейін жинақтайды.

Айырмашылық осында – қаншалықты режим жоғары болса, соншалықты жоғары кернеуді жақсы, төменгіні нашар өлшейді, сондықтан бір жарым вольт үшін нағыз оптималды режим **2000m** болады, 5 вольт үшін ең оптималды режим **20** болады, өйткені режим **2000m** 2 вольттан артық кернеуді өлшей алмайды, сондай режим **200m** 1.5 вольтті өлшей алмайды, өйткені оның максималды кернеуі – ол 0.2 вольт, ол 1.5 көп азырақ.



Мұнда ешқандай ерекшеліктер жоқ, 2щупты плюс пен минуске қосып, кернеуді өлшейсіңдер, бұл режим тек тұрақты токқа екенін ескертеміз (тұрақты ток бізде аккумуляторда, батарейкада, компьютерде, телефонның қоректену блогында).

Осы режимдерде сонымен бірге қарама-қарсылықты анықтауға болады, яғни плюс қайда екенін, минус қайда екенін білуге болады. Мұны жай мысалда түсіндіруге болады: егер қызыл щупты плюске, ал қараны минуске қосса, мысалы сол батарейкаға қосса, мультиметр 1.5 вольтты көрсетеді, егер оны керісінше жасаса, қызыл щупты минусқа, ал қараны плюске қосса, онда мультиметр -1.5 вольтты көрсетеді.

Бұдан ереже шығаруға болады - егер мультиметрде минус таңбасы көрінсе, онда қызыл щуп минуске қосылған, егер мультиметрде минус таңбасы көрінбесе, онда қызыл щуп плюске қосылған.

Ауыспалы токтың кернеуі дәл осылай өлшенеді, бірақ ол үшін ACV режимінің басқа топтары бар (alternating current voltage – ауыспалы токтың кернеуі), мультиметрде ол әдетте үш әріппен ACV немесе ирек сызықты V әрпімен белгіленген. Мұнда бізде екі режим 200 бен 750 бар, 200 200 вольтқа дейін кернеуді, ал 750 750 вольтті өлшейтінін аңғарған шығарсындар, бізде розеткаларда 200 вольттан асады (қалай болғанда да жақсы елді мекендерде), онда бізге тек 750 режимі дәл келеді. Режимді қайта қоямыз да, ауыспалы ток көздеріне щуппен тигіземіз, мысалы

щуптарды розеткаға қосамыз, мында қай щупты қайда екені маңызды емес, бізде қандай жағдай болсын 220 вольтті көрсетеді (өзгеруі, 200 ден 235-ке дейін болуы мүмкін). Осымен де барлығы.



Мультиметрмен тағы нені өлшеуге болады? Мысалы бізде бір жерде резистор жатқан, біз оның кедергісін білмейміз, мультиметрдің көмегімен оның кедергісін өлшей аламыз. Ол үшін таңбалы Ома – Ω режим топтары бар. Мұнда бізде 5 режим бар:



200 – 200 Ом –ға дейінгі кедергіні өлшейді

2000 - 2000 Ом- ға дейінгі кедергіні өлшейді (2кОм - 2 килоом)

20k (20 000) – 20000 Ом –ға дейінгі кедергіні өлшейді (20кОм)

200k (200 000) – кедергіні өлшейді (200кОм)

2000k (2 000 000) – 2 000 000 Ом –ға дейінгі кедергіні өлшейді (2МОм - 2 мегаом)

Резисторда қандай кедергі екенін анықтау үшін, бізге резистордың сол және оң аяғына щуптардың шеттерін қосу керек, одан соң мультиметр резистордың кедергісін, егер сендер дұрыс режим тандасандар көрсетеді, егер бұрыс режим тандасандар, онда мультиметр ештеңе көрсетпейді, не 1 цифрын сол жағында көрсетеді, бұл режимді көбейту керектігін және біздің кедергі көрінгенше арттыру керектігін білдіреді. Щуптардың металл шеттеріне тиіспеу өте маңызды, өйткені біздің денемізде де кедергі бар, резистордың кедергісін өлшегенде – мәндері дұрыс болмауы мүмкін.



Мультиметрдің көмегімен жасай алатын маңызды нәрсе, ол сылдырлату, бұл нені білдіреді? Егер мультиметрдің режимін дыбыстық толқын белгісіне ауыстырық щуптар бір-біріне тисе, онда мультиметр шиқылдай бастайды. Бұл электр тізбесін тұйықталғандығын білдіреді, бұл жағдайда электр тізбесінде тек щупалар.

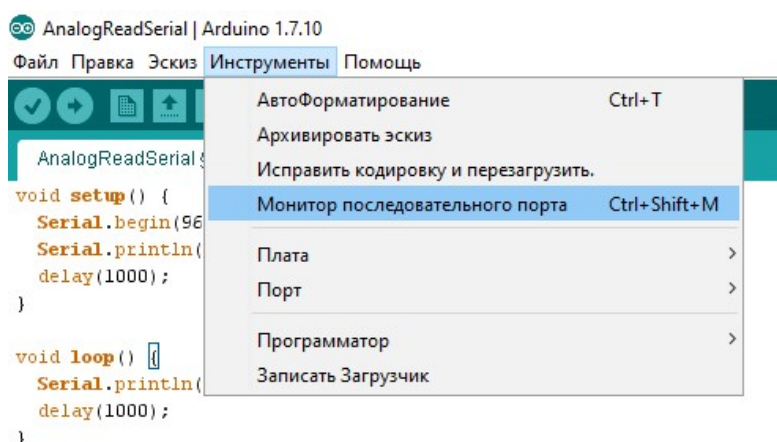
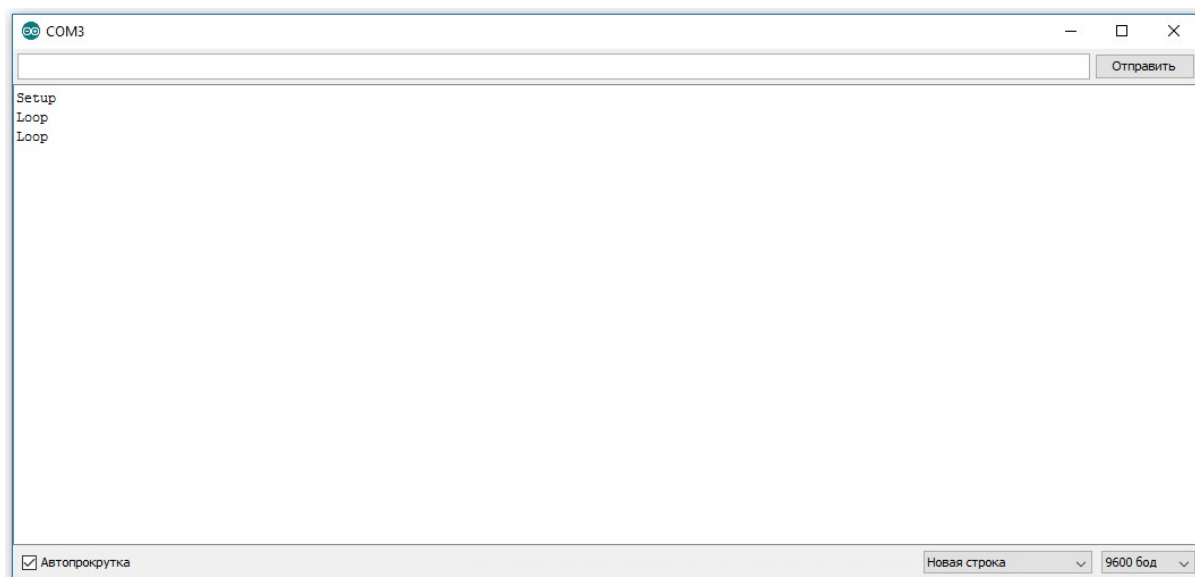
Осындай тәсілмен тұйықталу немесе сым қайда барады және контакт қандай екенін тексеруге болады, бізде 10 шатысып қалған сымдар делік, біз бір үшін таптық, екінші үшін бізге табу керек, ол үшін бір щупке сымның бір үшін ораймыз да, екінші үшімен мультиметр дыбыс сымға бергенше, барлық үшін кезекпен тиеміз.

Serial.read және Serial.print

Бүгінге теория жеткілікті, программалауға көшейік. Өткен сабақта светодиодпен жыпылықтауға үйрендік, енді компьютерден тікелей Arduinoға қалай команда беріп, оған қалай жауап алу керектігін қарастырайық. Қарапайым кодты жазайық және оны талдайық:

```
void setup() {  
  Serial.begin(9600);  
  Serial.println("Setup");  
  delay(1000);  
}  
void loop() {  
  Serial.println("Loop");  
  delay(1000);  
}
```

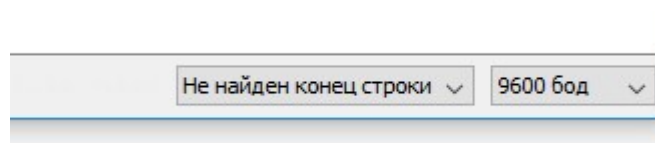
Енді осы кодты біздің Arduino платасына салайық, оны қалай жасауды өткен сабақта талқыладық. Программаны ойдағыдай салғаннан кейін менюге **Инструменты -> Жүйелі порт мониторуна кіріңіз** немесе **Ctrl+Shift+M** қызу клавишті басу керек. Осыдан кейін сізге терезе ашылады, оған Setup сөзі шығарылып, шексіз қайталана беретін Loop сөзі шығады.



Бұл кодта жазылған сөздер, егер біз басқа сөздер жазсақ, олар да шығады. Біз не жазғанымызды талдайық. Сендер түсінгендей, команда **Serial.println("Setup");** “Setup” жазуын шығарады, ал команда **Serial.println("Loop");** “Loop” сөзін шығарады. Егер тереңірек қарасақ, бұл командалар компьютердің экранына тек сөздер шығарып қоймай, Arduino қосылған олар компьютерге USB порты арқылы кез келген ақпаратты жібереді, ал компьютер бұл ақпаратты өңдеп, қандай да бір әрекет жасайды. Қазір біз жазуды программалауға шектелеміз және Arduino компьютерге шығарғанды

оқимыз. Сонымен Setup жазуы 1 рет шықты, өйткені біз оны `setup() { }` блогына шығардық, ал Loop жазуы үнемі шығады, өйткені `loop() { }` блогында. `Delay()` функциясы жай үзіліс жасайды, оны біз өткен сабақта өттік. Біз зерделемеген бір жаңа функция **Serial.begin(9600);** Ол `setup() { }` блогында шығарылады да, жөндеп күйге келтіруге қызмет ететінін аңғарған шығарсындар. Ол осылай – бұл функция мәліметтерді беру портын ашады, онсыз **Serial.print()** жұмыс істемейді. Параметр 9600 мәліметтер жіберетін порттың жылдамдығын білдіреді, бұл жылдамдық бодпен өлшенеді, демек біз 9600 бод жылдамдығымен мәліметтерді берудің сериялық портын аштық.

Бод дегеніміз не? Шынында **бод** бұл секундына жіберілетін символдар саны. Яғни 9600 бод жылдамдығымен біз секундына 9600 символдар жібере аламыз (әріптер, тыныс белгілері және т.с.с.). Біздің мақсат үшін бұл толығымен жеткілікті, бірақ жылдамдықтар жоғары да болады, мысалы 19200, 38400, 57600, 115200 және басқалар. Егер сендер жүйелі порт мониториянда сонымен бірге жылдамдықты күйге келтіру бар екенін байқаған шығарсындар:



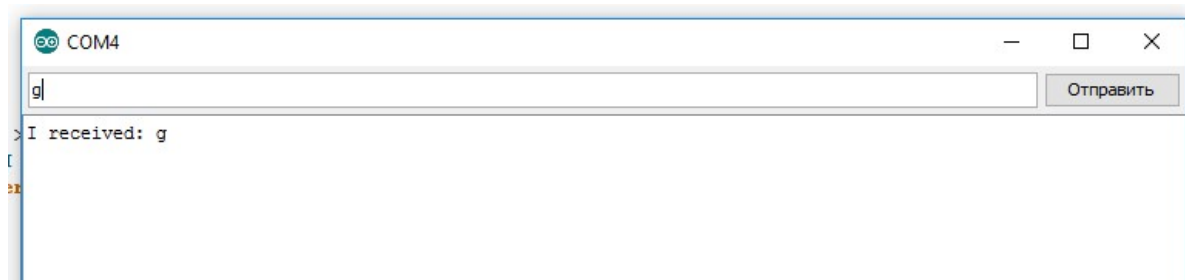
Егер біз программада қажетті жылдамдықты қойдық, ол енді бізге не керек? Біз программада Ардуинодан мәліметтерді беру жылдамдығын қойдық, ал порт мониториянда компьютердің мәліметтерді қабылдау жылдамдығын қоямыз. Бұл жылдамдық Ардуинодан мәліметтерді беру жылдамдығымен тура келу керек, әйтпесе мәліметтер бұрмаланады, және мүлдем ештеңе

алмайсыз, не “кракозябры” аласыз. Егер жылдамдық әртүрлі болса, компьютер керексіз кезде мәліметтерді салыстырып оқиды да, оларды толық қабылдауды үлгермейді. Оның қалай жұмыс істейтінін елестетіңіз, әр минут сайын басқа бөлмеде әріп жазылған қысқа хат пайда болады, оны сен алып, басқа бөлмеге столға кетесің, осылай бірінен соң бірінен сөз жасалады. Басқа бөлмеде әріптер минутына бір реттен жиі пайда болатынын елестетіндер, сендер оны білмей, әр минут сайын сол бөлмеге келіп, бір әріпті әкетесіз, ал сол минутта сендер алмаған басқа көп әріптер пайда болып жоқ болып кетті. Нәтижесінде сендерде түсініксіз сөздер жинала бастады. Дәл солай, компьютер символдарды оған жиі салыстырып оқу керектігін білу керек.

Serial.println(); функциясымен біз түсіндік. Компьютерге мәліметтерді шығарудан басқа, біз оларды компьютерден салыстырып оқи аламыз. Оған **Serial.read();** командасы бар: кодты жазайық, Arduinoға жүктейік те, оны тестілейік:

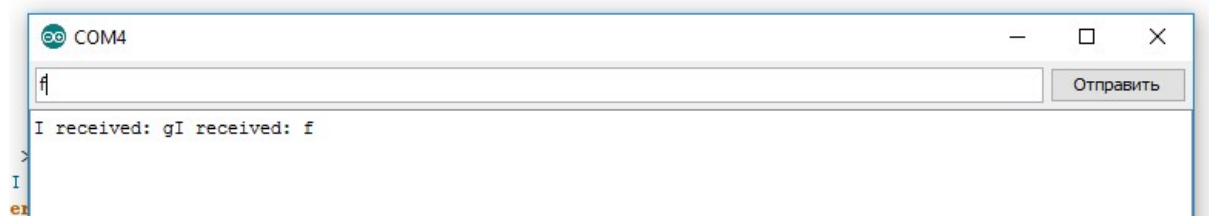
```
void setup() {
    Serial.begin(9600);
}
void loop() {
    if (Serial.available() > 0) {
        Serial.print("I received: ");
        Serial.write(Serial.read());
    }
}
```

Платоға кодты жүктегеннен кейін, жүйелі порттың мониторын ашыңдар, полеге бір символ, әріп не цифр жазып көріңдер де, **Жіберу** кнопкасын басыңдар, сендерде төменде "I received: " жолы көрініп, одан кейін біз жіберген символ көрінеді. Бұл біз не жазсақ, соны біздің код шығаратынын білдіреді.



Кодты жол сайын талдайық, **Serial.begin(9600);** біз білетіндей жүйелі портқа атайды. Ары қарай жол портқа қандай да ақпарат берілетінін тексереді **if (Serial.available() > 0) {**, бұл кодты біз келесі сабақта талдаймыз, сондықтан функция **Serial.available()** қандай болмасын ақпарат түсті ме, жоқ па, соны тексереді. Егер әлдеқандай мәліметтер жүйелі портқа берілсе, онда программа кодты ары қарай оқиды, егер жоқ болса, онда жанынан өтіп, тағы да мәліметтер берілген тексереді.

Функция **Serial.print("I received: ");** бұл функция "I received: " жазуын шығаратынын сендер аңғарған шығарсыңдар, бұл функция **Serial.println()** өте ұқсас, бірақ сәл одан ерекшеленеді. Функция **Serial.println()** жолды шығарғаннан кейін, курсорды келесі жолға ауыстырды, ал функция **Serial.print();** бір ұзын жол шығарып, келесі жолға ауыстырмайды, шынында **ln** жазуы бұл қысқаша **line**. Егер біз тағы бір символ жіберіп көрсек, "I received: " жазуы осы жолда жазылады:



Егер біз кодты сәл өзгертіп, тағы **Serial.println("");** жолын қосақ, онда сол қосқан жерде жол тасымалданады:

```
void setup() {
    Serial.begin(9600);
}
void loop() {
    if (Serial.available() > 0) {
        Serial.print("I received: ");
        Serial.write(Serial.read());
        Serial.println("");
    }
}
```

Платоға кодты жүктейміз, егер біз бірнеше символдар жіберсек, онда жазу әр кезде жаңа жолдан шығады.



Функция **Serial.read()** компьютерден жіберген мәліметтерді салыстырып оқиды, онда функция әр кез бір символды оқиды, яғни 10 әріптен тұратын сөз жіберсек, функция **Serial.read()** сөзді толығымен алғанша 10 рет жұмыс жасайды.

Ақырында функция **Serial.write();** - бұл функция **Serial.println();** мен **Serial.print();** функциялары атқарған істерді жасайды – олар мәліметтерді жүйелі порт арқылы жібереді, бірақ олардан айырмашылығы **Serial.write();** толық сөздерді жібере алмайды, ол тек бір символды жібереді.

Енді **Serial.write(Serial.read());** функциясының комбинациясы не істейтіні түсінікті, функция **Serial.read()** бір символ алады да, дереу оны **Serial.write()**, функциясына жібереді, ол компьютерге кері жібереді.

Жүйелі порт осылай жұмыс істейді, бұл білім бізге bluetooth арқылы мәліметтерді жібергенде керек болады, онда да **Serial** қолданамыз, бұл bluetooth көмегімен телефонда кез келген құрылғыны басқаруға көмектеседі.

№3 сабақ

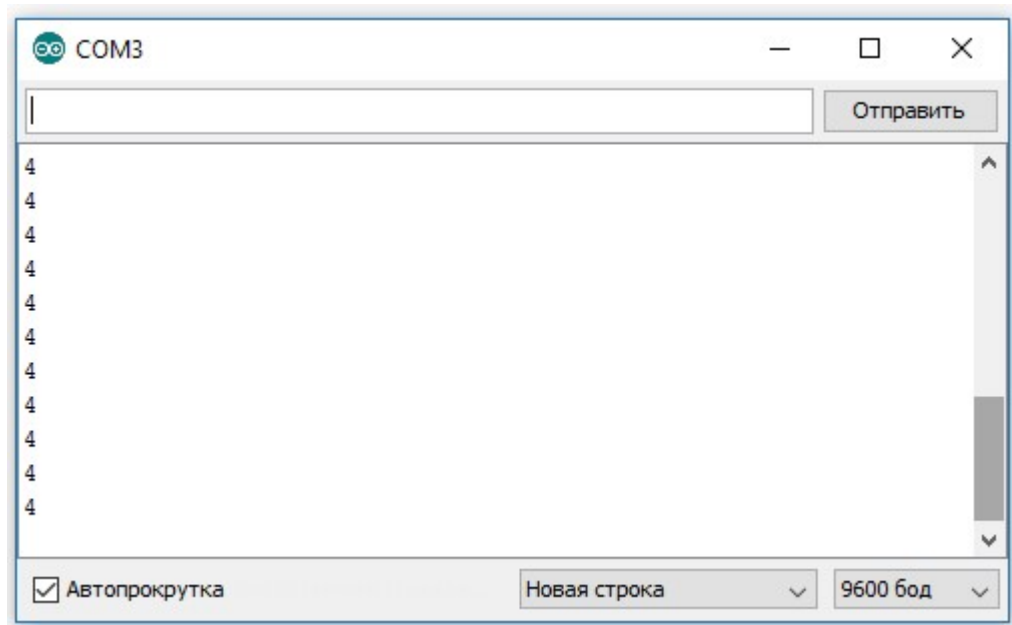
Тақырыбы: Математика операторлары, ауыспалы, шартты операторлармен таныстыру

Бұл сабақта микроконтроллерлерді программалауын зерделеуді сәл тереңдетеміз. Программалауда математикалық функциялар бар екенін бүгін білеміз, сонымен бірге жағдайларымен, өзгергіштерімен программалауда математика операторларынан, математикада сандармен қалай математикалық операциялар орындауға болатынын қараймыз: бөлу, көбейту, қосу және алу. Өткен сабақта Serial порт арқылы компьютерге Ардуинодан ақпаратты беруге үйрендік. Біз қарапайым мәтін жолдарын шығардық, жолдан басқа сандарды да шығаруға болады. Мысалы, **Serial.println(2);** жүйелі портқа 2 санын жібереді. Жүйелі портқа **Serial.println(2+2);** жіберіп көрейік те экранда не көрінетінін байқайық, сендер байқадыңдар деп ойлаймын.

```
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  Serial.println(2+2);  
  delay(1000);  
}
```

Енді бұл кодты Ардуиноға жіберейік:

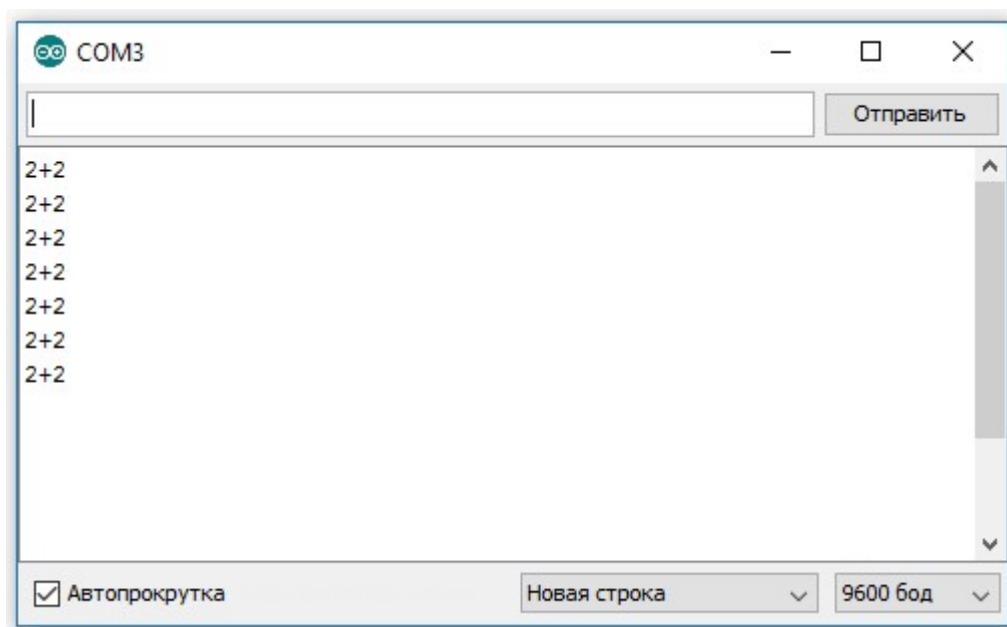
Жүйелі порттың (**ctrl+ shift + M**) мониторын ашамыз да, нәтижеге қараймыз:



Нәтижесінде 4 шығады – екі сандық 2+2 есебі, яғни программа қосындыны өзі есептеді де, бізге дайын жауап берді. 2+2 сандар қосындысын тырнақшасыз жазғанымызға назар аударындар. Өткен сабақта жолға **Serial.println("Loop");** тырнақшамен шығарғанымыз естеріңде шығар. “2+2” қосындысын тырнақшаға алып көрейік те, одан не шығаттынын қарайық:

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println("2+2");
  delay(1000);
}
```

Енді кодты платоға саламыз, порт мониторын жібереміз де, байқаймыз:



Математикалық теңдеулер тырнақшаға алынса, онда программа оны қалай болса, солай шығаратынын нәтижеден қорытынды шығаруға болады. Программа үшін тырнақшаға алынғанның бәрі әдеттегі жол болып табылады, ол қосуға, алуға, басқа да теңдеулерді орындауға тырмыспайды. Бұл арнайы жасалған, кенеттен оған санау программасы қажет емес, сен жай ғана жазғаныңды шығарғың келеді, онда тырнақшаға алу өте пайдалы.

Қосудан басқа, программа басқа математика операцияларын орындай алады. Ол үшін өзінің операторлары бар, олардың негізгісін біз төменгі кестеде көрсеттік:

+	қосу
-	азайту
/	бөлу
*	Көбейту

Әр оператормен теңдеулерді орындап көр де, нәтижесін шығар:

Serial.println(2); - мұнда экранда 3 саны шығады

Serial.println(6*2); - көбейту нәтижесі 18 санын көрсетеді

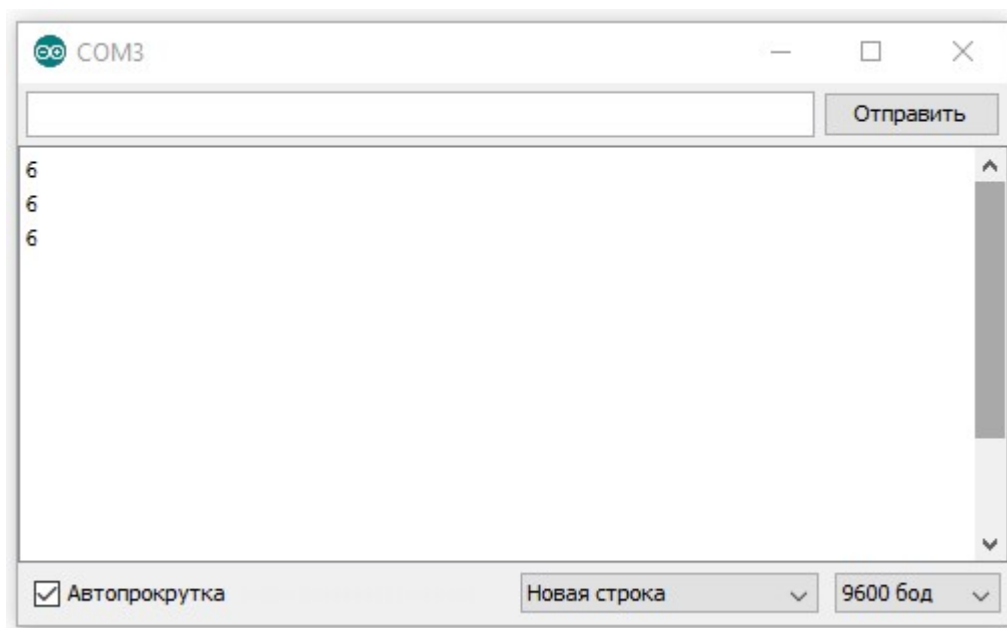
Serial.println(6-2); - мұнда да 4 цифры шығады.

Программалауда математикадағыдай әр операцияның өзіндік приоритеті бар, мысалы көбейту мен бөлу қосу мен азайтудан бұрын орындалады. Мына алдамшы математикалық есепті естіген шығарсыңдар: “Екіге екіні қосып, оны екіге көбейткенде неше болады, көбі қателесіп, жауабы 8 дейді, өйткені олар алдымен $2+2$ теңдеуін орындайды, сондықтан 4 шығады, содан кейін оны 2-ге көбейтеміз, $4 * 2 = 8$ шығады. Бұл дұрыс емес. Өйткені алдымен көбейтуді орындап, содан

кейін қосуды орындау керек. Дұрыс жауабы 6 сәйкес келеді, өйткені $2*2 = 4$ және $4 + 2 = 6$. Неліктен бұл айтылған – программа теңдеулердің орындау тәртібін әрқашанда дұрыс анықтайды, осыны тексеру үшін жүйелі портқа осы теңдеуді $2+2*2$ шығарайық, егер программа математикалық амалдарды есептесін десең, оларды тырнақшасыз жазу керектігін ескертеміз:

```
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  Serial.println(2+2*2);  
  delay(1000);  
}
```

Соңында бізде жауап шығу керек:



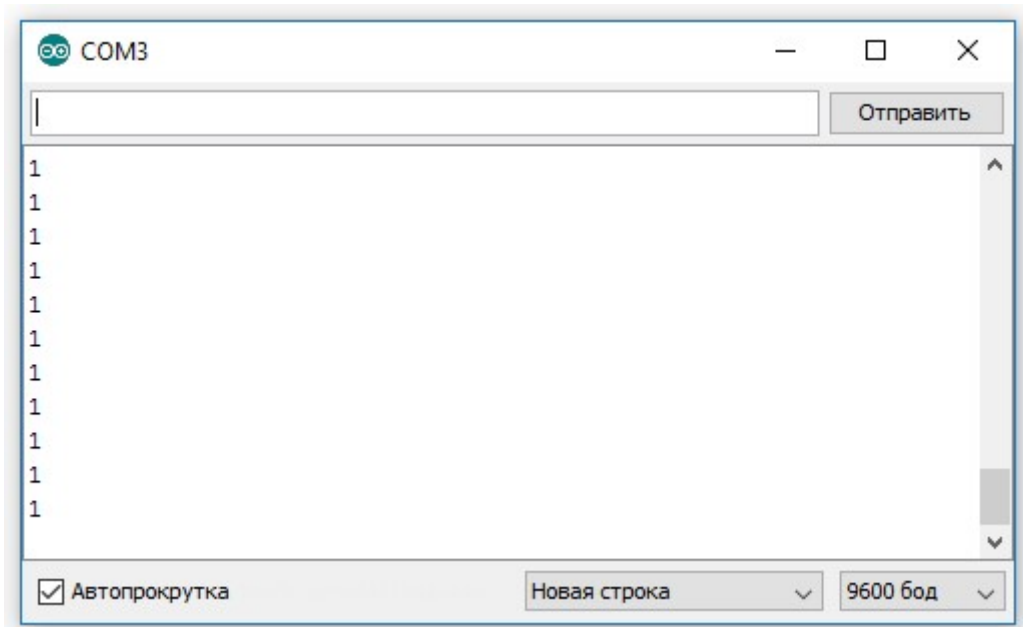
Программа математикалық операторлар дөңгелек жақшаларға алынғанын түсінеді, мысалы егер біздің амалдарды $(2+2) * 2$ жақшаға алса, онда программа мұны ескереді де, 8 көрсетеді, мұны өздердің жасап көріңдер:

```
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  Serial.println((2+2)*2);  
  delay(1000);  
}
```

Тағы да мынандай мысалды қарастырайық, $5/3$ бөліп көрейік, бізге белгілі, онда шексіз ұзын сандар $1,6666666666666666...$ және т.с. шығады. Егер біз программаны оны есептеуге мәжбүр етіп, жауап берсін десек, онда жауап басқаша болады:

```
Serial.println(5/3);
```

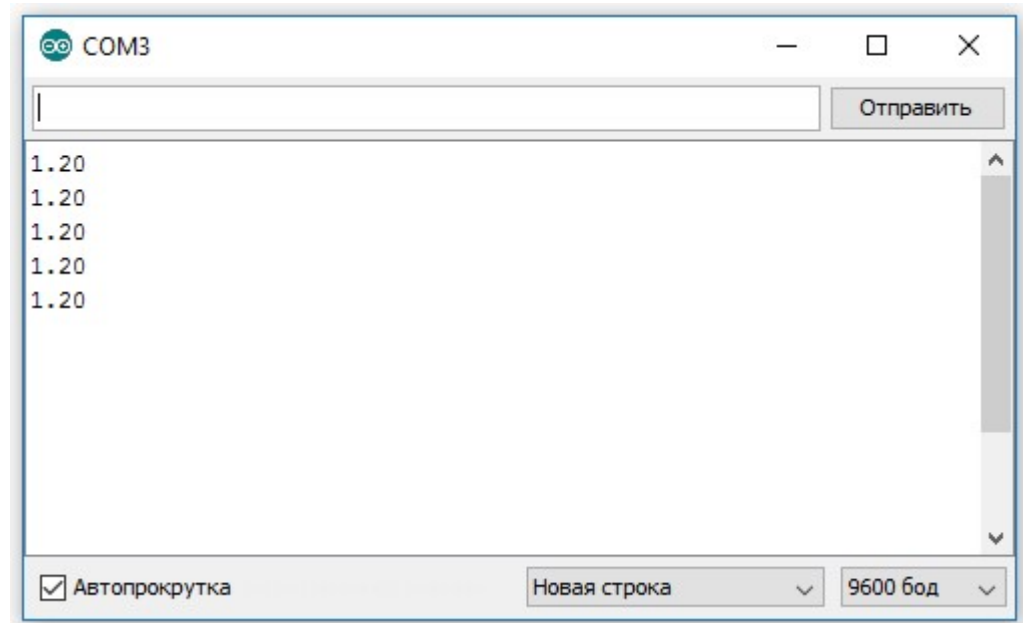
Онда жүйелі порттың мониторын ашсақ, жауап 1 болатынын біз көреміз:



Программа мұндай жауапты 1,6 1 дейін дөңгелектендірді (бөлшек бөлігін алып тастады), өйткені бүтін сандармен жұмыс жасайды және бөлшек бөлігі бізге керек емес деп ойлады. Программаға бізге бөлшек бөлігін қоса нақтырақ жауап керектігін түсіндіріп, теңдеуде бөлшек сандарды жазу керек, мысалы егер біз $2.4/2$:

```
Serial.println(2.4/2);
```

онда программа 1.20 жауабын береді.

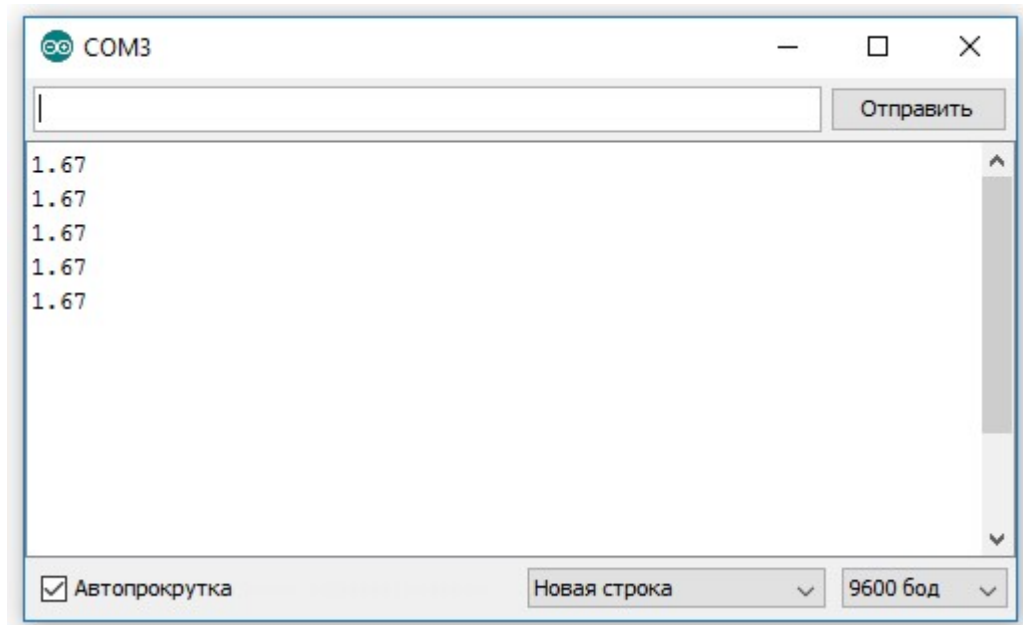


Егер де математикалық операцияға қатысатын сандардың бірі бөлшектік болса, онда программа бөлшектік санмен жауап береді. Біз $5/3$ бөлгенде, екі санда болған, бірақ жауап бөлшек болды. Бөлшек жауап алу үшін кез келген цифрдың жанына нүкте мен нөл қойған жеткілікті, мысалы $5.0/3$ не $5/3.0$, не $5.0/3.0$.

Енді ескі мысалды жаңаша қарап көрейік

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println(5.0/3.0);
  delay(1000);
}
```

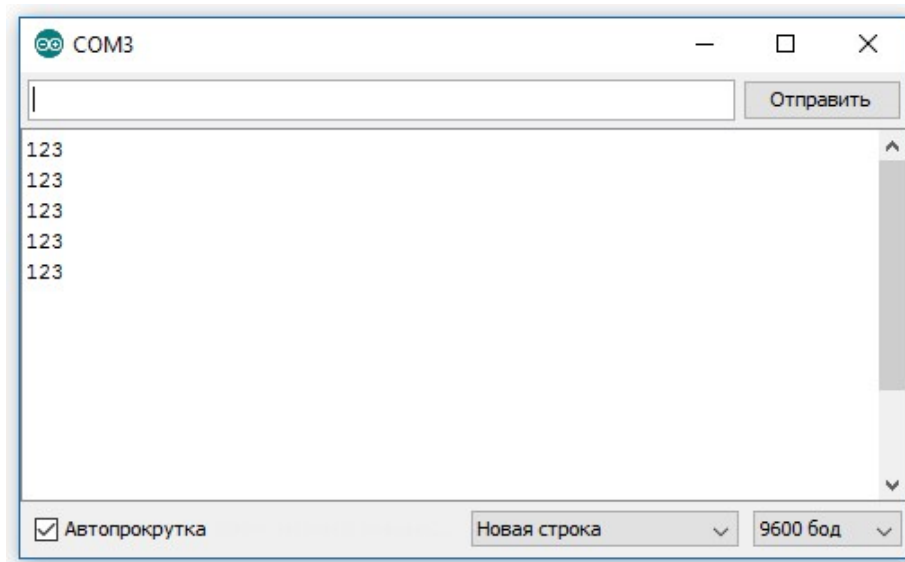
Программа аса нақты жауап бергенін көреміз:



Осымен әзірше операторларды зерделеуді тоқтатамыз да, өзгеріштерді зерделеуге көшеміз. Өзгергіш дегеніміз не және оны немен қолданады? Ең жақсы тәсіл – оны өз көзіңмен көру, келесі кодты жазып көрейік.

```
int korobka;
void setup() {
  Serial.begin(9600);
}
void loop() {
  korobka = 123;
  Serial.println(korobka);
  delay(1000);
}
```

Енді кодты Arduinoға қосамыз да, нәтижені қараймыз:



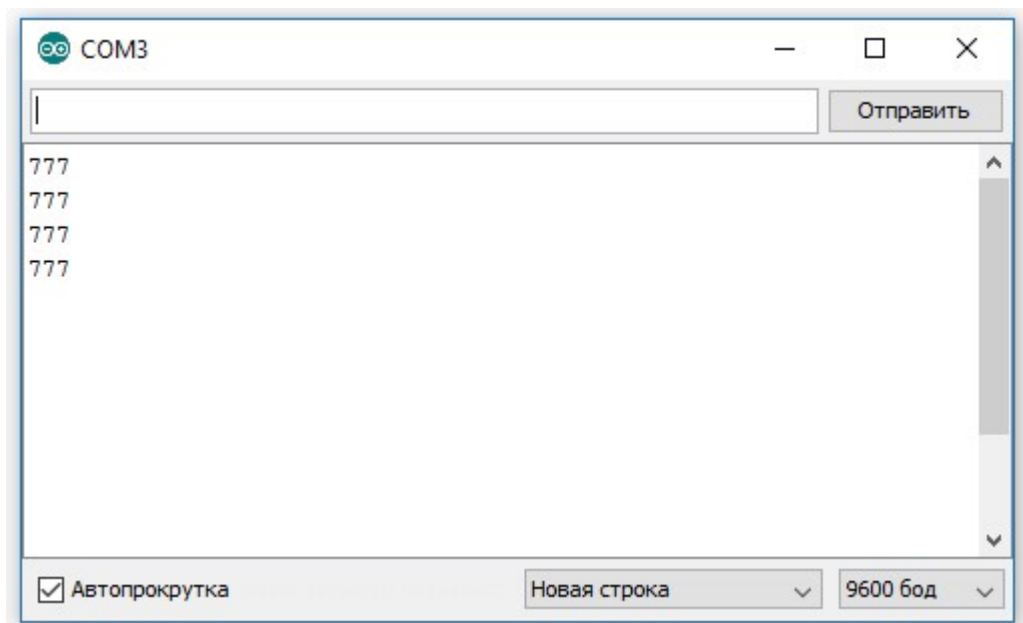
Біз көргендей, порт монитормына 123 саны шығады, егер код пен 123 санын басқамен өзгертсек, онда порт монитормына біз жазған сан шығады, мысалы жолды өзгертеміз

```
korobka = 123;
```

мынандай кодқа:

```
korobka = 777;
```

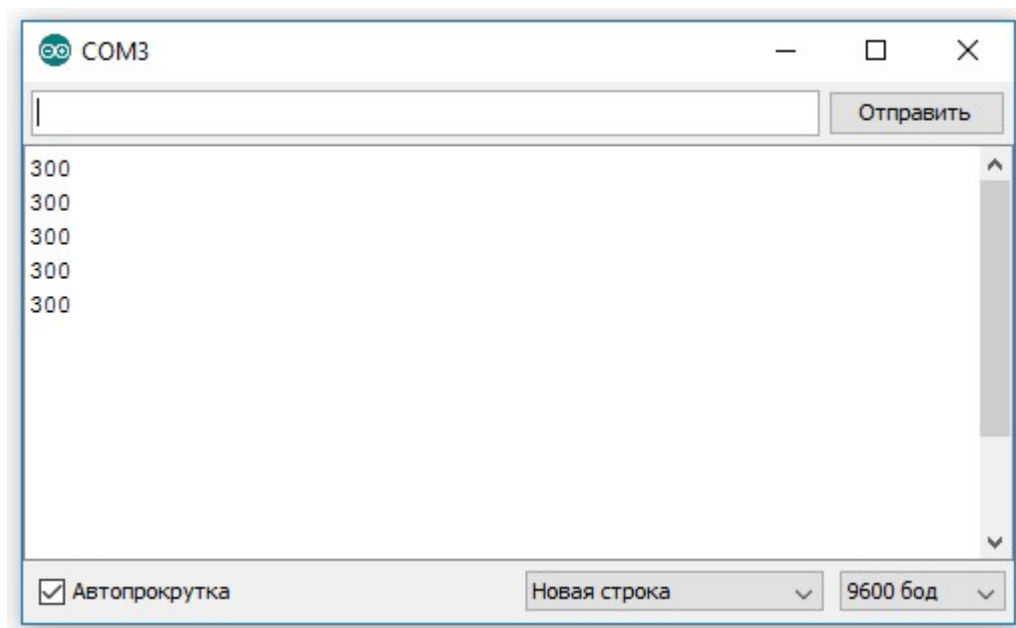
онда монитор портына үш жеті шығады:



Енді өзгергіш дегеніміз не екенін жақсы түсіндіруге болады, **korobka** – бұл өзгергіш. Программалауда өзгергіш бір қорап болып саналады, оған ақпаратты салуға болады, сосын оны қолдануға болады, программа өзгергіштен біз салған ақпаратты алады да, пайдаланады, біздің жағдайымызда программа бұл мәнді монитор сериалына шығарады. Егер өзгергішке (**korobka = 777;**) санын салсақ, онда программа өзгергіш **korobkaға** 777 саны деп оған қарайды. Егер біз **korobka + 1** деп жазсақ, онда программа бізге 778 жауабын береді. Яғни программа өзгергіште жатқан 1 цифрын қосады да, бізге жауап береді, эксперименттеп көрелік, келесі кодты жазайық:

```
int korobka;
void setup() {
  Serial.begin(9600);
}
void loop() {
  korobka = 600;
  Serial.println(korobka/2);
  delay(1000);
}
```

Аrduinoға салайық та, не болғанына қарайық:



Программа өзгергіште сақталған 600 санын 2 бөлді де, бізге жауап берді.

Өзгергіште санды сақтау үшін өзгергіштің атауын жазып, одан кейін теңдік “=” белгісін қойып, содан кейін онда сақтайтын санды жазу керек екенін түсіндіңдер деп ойлаймын.

Мүмкін сендер аңғарған шығарсыңдар `korobka` сөзі “ойынан алынған”, өйткені өзгергішті қай атаулармен атауға болады, бұл `korobka` сөзінің орнына қандай да басқа сөз жазуға болады, мысалы `box`, `container`, `blablabla`, `peremennaya1`, `peremennaya2`, `a`, `b`, `c`.... солай ары қарай, өзгергіш атауында қандай да латын (ағылшын) әріптерін, цифрлар мен сызу белгілерін “_” қолдануға болады, бірақ атаудың бірінші символында міндетті түрде әріп болу керек, мысалы, өзгергішті `box2` деп атауға болады, бірақ өзгергішті `2box` деп атауға болмайды. Жаңа атаумен жаңа өзгергіш жасауға сендер одан мәліметтерді салыстырып оқығанша, сақтағанша жазуларың (хабарлауың) керек. Өзгергіш өте жай жасалады, ол үшін кодтың басында `setup() { }` функциясының алдында **`int korobka;`** деп жазғанды байқаған шығарсың – бұл жол `korobka` өзгергішін жасады. Егер мұнда басқа өзгергіш басқа атаумен құрған болар едік.

Енді **`int`** командасы туралы толығырақ, **`int`** - бұл **`integer`** сөзінің қысқарған түрі (аударғанда – *бүтін сан*), бұл команда емес, **`int`** деп жазып, біздің өзгергішке бүтін сан типін тапсырдық. Бұл не? Бұл өзгергіште тек бүтін сандарды сақтай алатынды білдіреді, егер онда бөлшек санды дөңгелетеді де, бәрібір бүтін санды сақтайды. Егер өзгергіште бөлшек санды сақтағымыз келсе? Ол үшін **`float`** мәліметтер типі бар, яғни **`float korobka;`** өзгергіштер салсақ, онда бөлшек санын сақтай аламыз, оны жасап көрейік:

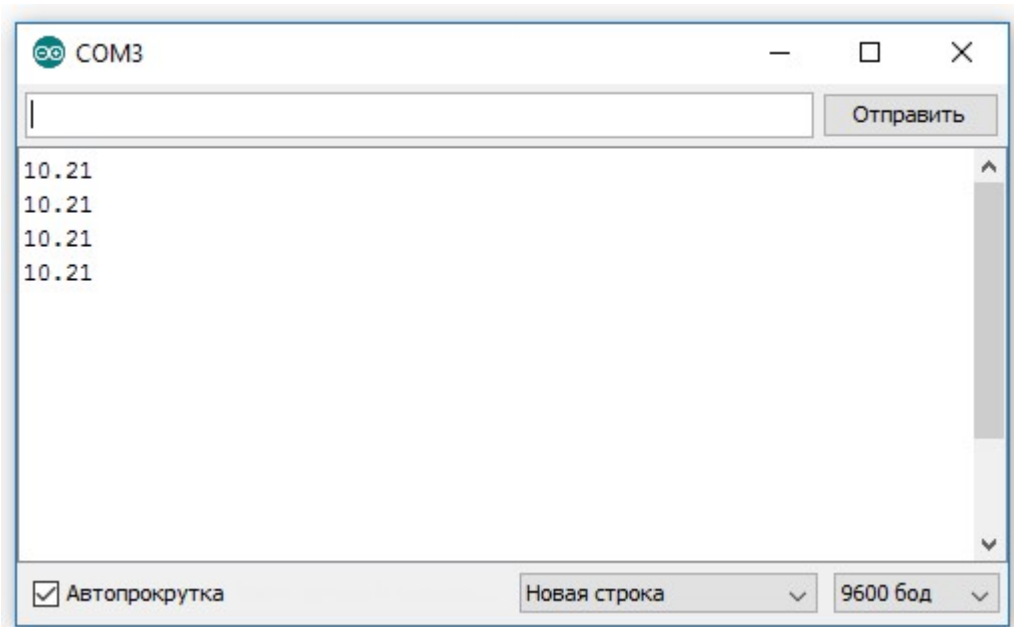
```
float korobka;
void setup() {
```

```

Serial.begin(9600);
}
void loop() {
  korobka = 20.42;
  Serial.println(korobka/2);
  delay(1000);
}

```

Программа бөлшек жауап береді, өйткені санның біреуі бөлшек болып табылады:



Өзгеріштермен байланысты тағы бір шектеу бар, **int** типті өзгеріште -32768-ден кіші, 32767-ден көп санды сақтауға болмайды. Кодты жазған кезде сіздер әрқашанда мұны ескеру керек, шектеуден мөлшері артық сандарды сақтау күтпеген нәтижелелерге әкелуі мүмкін.

int пен **float** – бұл мәліметтердің жалғыз типі емес, төменгі кестеде Arduinoда қолдануға болатын бірнеше типтерін көрсеттік:

Мәліметтер типі	Мән диапазоны
int	-32768-ден 32767 дейін
unsigned int	0-ден 65535 дейін
long	-2 147 483 648-ден 2 147 483 647 дейін
unsigned long	0-ден 4 294 967 295 дейін
float	-2 147 483 648.0-ден 2 147 483 647.0 дейін

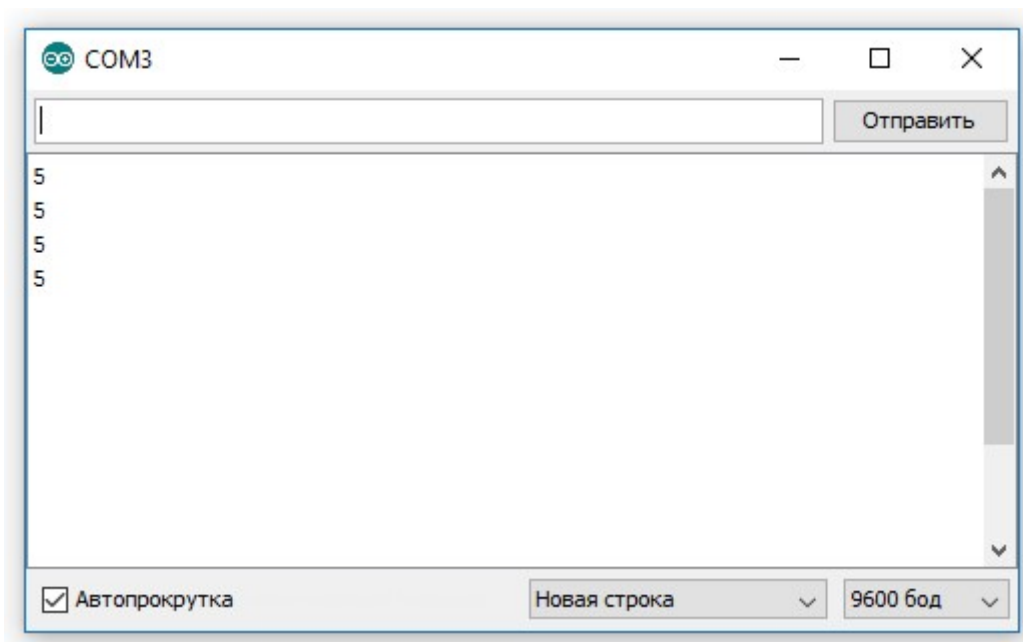
Кейбір атауларда **unsigned** сөзі кездесетінін байқадыңдар – аударғанда “таңбасыз” дегенді білдіреді, мәліметтердің мұндай типінде минус таңбасын қолдануға болмайды. Бұл мұндай өзгергіш үшін максималды мәнін көбейтуге мүмкіндік береді.

Мәліметтердің осындай типінің әрқайсымен өзгергіш жасап, сақтап, максималды мәнін шығарып көріңдер.

Өзгергіштің тағы да бір қасиетін қарастырып көрейік. Мысалы, егер кодта былай деп жазсақ, не болады:

```
int korobka;
void setup() {
  Serial.begin(9600);
}
void loop() {
  korobka = 20;
  korobka = 5;
  Serial.println(korobka);
  delay(1000);
}
```

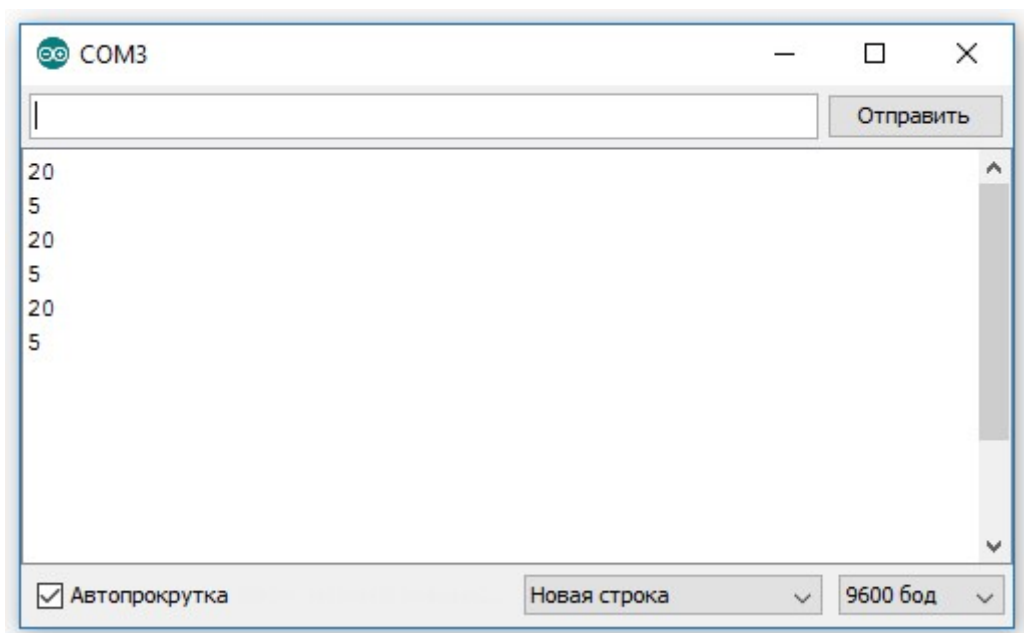
Жүйелі порт мониторында программа қандай сан көрсетеді деп ойлайсындар?



Біз көргендей 5 саны шықты, біз өзгергіште 20 санын сақтадық, ол қайда кетті? Егер біз өзгергіште санды сақтайтын болсақ, онда алдындағы мәліметтер одан лезде қашықтайды. 20 саны korobka өзгергішінде болды, бірақ ол онша ұзақ болған жоқ. Егер біз korobka өзгергішінде 20 санын лезде сақтағаннан кейін оны жүйелі портқа шығарып көрсек, онда өзгергіште сан бар екенін көретін едік, байқап көрейік:

```
int korobka;
void setup() {
  Serial.begin(9600);
}
void loop() {
  korobka = 20;
  Serial.println(korobka);
  korobka = 5;
  Serial.println(korobka);
  delay(1000);
}
```

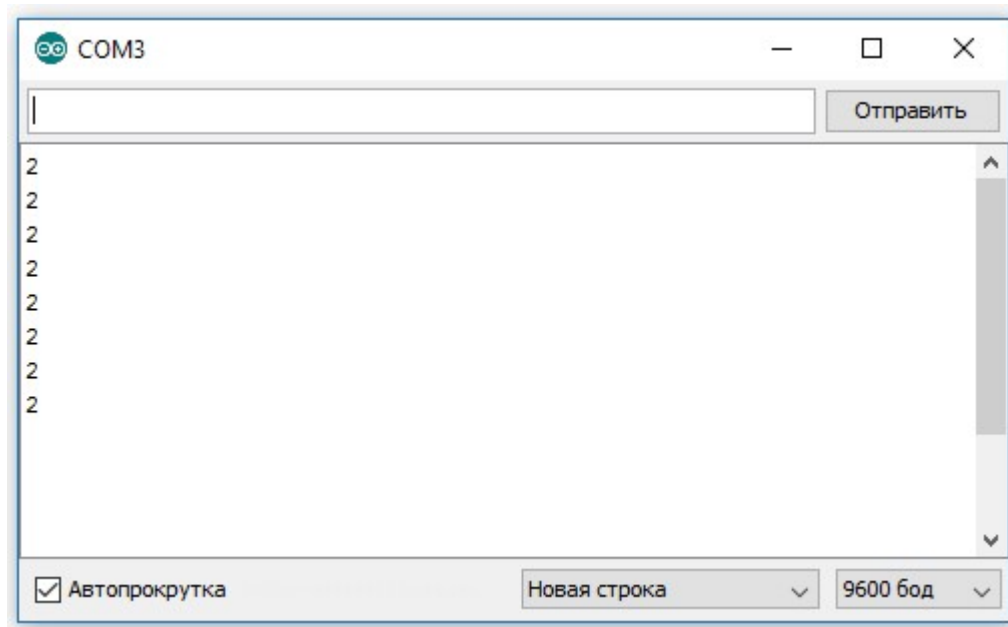
Сол бір өзгергішті түрлі уақытта шығарсақ та, оның мәні әртүрлі, экранға 20 саны да, 5 саны да шыққанын көреміз.



Сондықтан өзгергішті “Өзгергіш” деп атайды, өйткені оның мәндері өзгереді, ауысады. Жоғарыда айтқандай, өзгеріштермен сандар сияқты сол математикалық амалдарды орындай аламыз, мысалы 2 өзгергіш жасайық, оларға 2 санын сақтайық та, бір өзгергішті екінші өзгергішке бөлейік те, нәтижесіне қарайық:

```
int box1;  
int box2;  
int x;  
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  box1 = 20;  
  box2 = 10;  
  x=box1+box2;  
  Serial.print(box1);  
  Serial.print("+");  
  serial.print(box2);  
  serial.print("=");  
  serial.println(x);  
  Serial.println(box1 /box2);  
  delay(1000);  
}
```

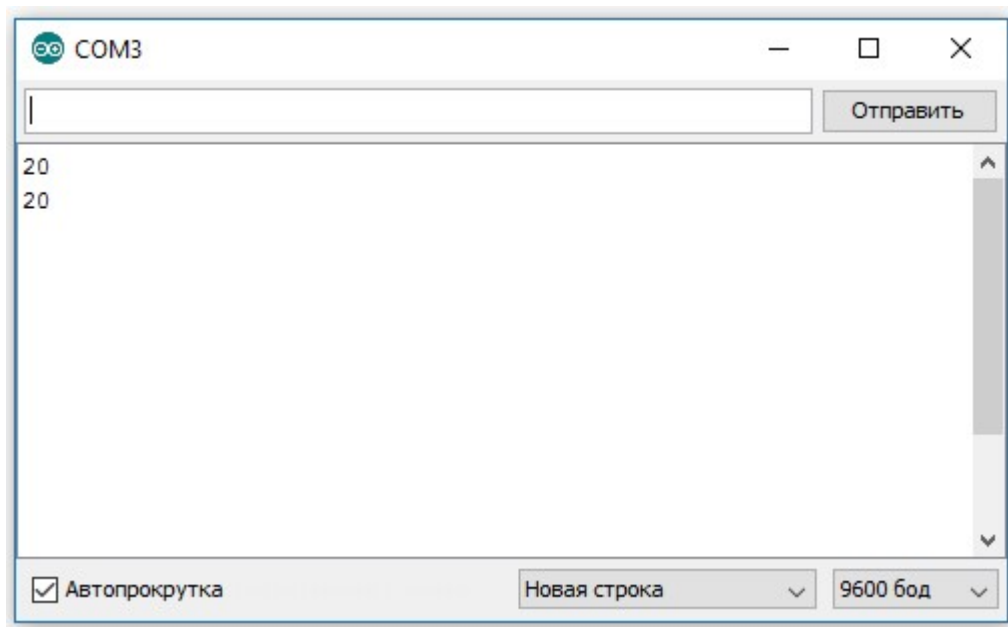
Программа box1өзгергіштен 20 мағынасын оқығанын көреміз, ал box1 өзгергішінен 10 мағынасын көреміз, бөлуді орындап, 2 жауабын шығарды:



Өзгергіште басқа да өзгергіштерді сақтауға болады, мысалы

```
box1 = 20;  
box2 = box1;  
Serial.println(box2);
```

Бұл мысалда box1 өзгергішінде 20 санын сақтадық, содан кейін box2 өзгергішінде box1 өзгергішін сақтадық, аяғында box2 өзгергішінде бізде 20 саны болады, өйткені box2 = box1 деп жазғанда, программа box1 өзгергішінде не сақталғанын қарап, оны сан деп есептеп, оны box2 өзгергіште де 20 саны, Serial.println(box2); жүйелі портының мониторында шығарғанда, бізде 20 саны көрінеді.

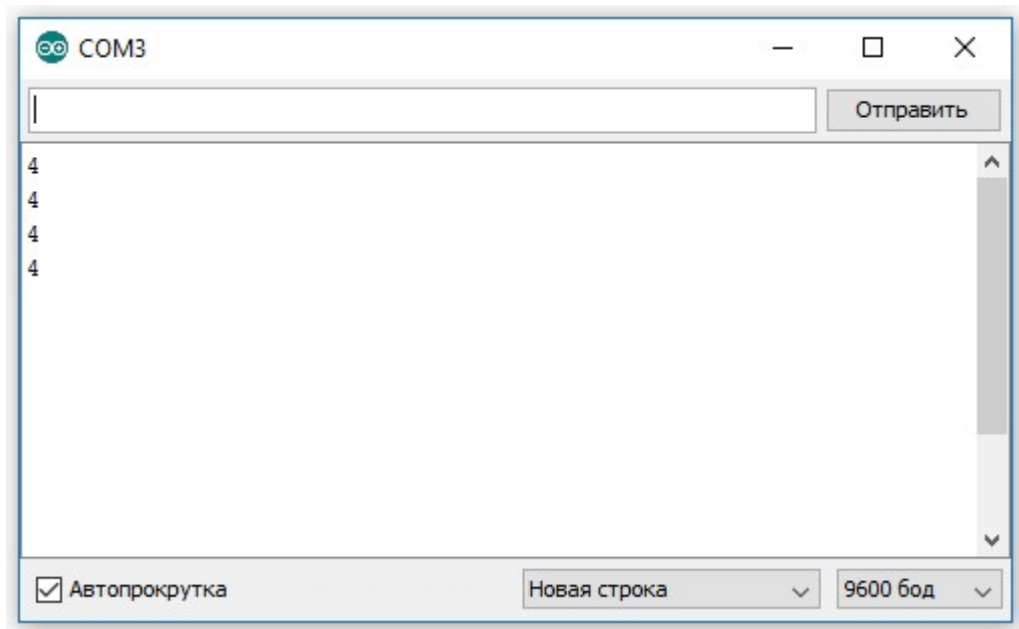


Өзгергіште өзін-өзі сақтауға болады, мысалы

```
box1 = 2;  
box1 = box1 + box1;
```

```
Serial.println(box1);
```

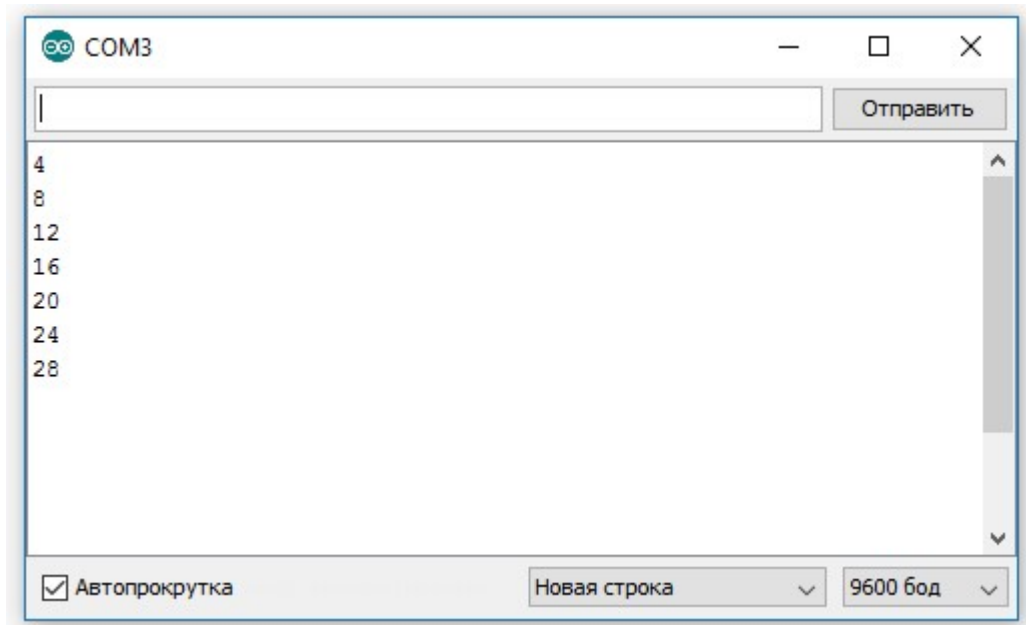
Бұл мысалда box1 өзгергіште алдымен 2 санын сақтадық, одан кейін box1 өзгергішінде box1 + box1 сақтаймыз, сонда программа не жасайды? Алдымен box1 өзгергішінде 2 саны жатыр, ары қарай box1 = box1 + box1 деп жазғанда, программа алдымен жақшаның оң жағындағы кодты орындайды, яғни ол box1 + box1 деп жазғанда – бұл бәрі бір, егер 2 + 2 деп жазғандай. Программа box1 + box1 4 болады деп есептейді және төртті кері box1 өзгергішіне сақтайды, енді box1 өзгергішінде 4 саны жатыр, Serial.println(box1); командасы порт мониторына 4 санын шығарады.



Тағы бір мысал қарастырайық, осындай код жазып, оның не істейтініне қарайық:

```
int box1;  
void setup() {  
  Serial.begin(9600);  
  box1 = 0;  
}  
void loop() {  
  box1 = box1 + 4 ;  
  Serial.println(box1);  
  delay(1000);  
}
```

Монитор портының платасына программа жүктегеннен кейін әркез 4 көбейтілген сан шығады



Неге бұлай болатынын түсінбедіңдер, рет-ретімен талдайық, ең басында `setup() { box1 өзгергішінде 0 санын сақтаймыз. Ары қарай box1 = box1 + 4; береміз, өткен мысалдағыдай программа алдымен теңдік таңбасынан кейін барлық формуланы есептейді, яғни box1 + 4, қазір өзгергіште 0 саны сай, box1 + 4 сондықтан 4 тең болады, бұл санды кері box1 өзгергішінде қайта сақтаймыз, енді box1 өзгергішінде 4 саны. Ары қарай бұл санды порт монитормына шығарып, 1 секундқа үзіліс жасаймыз. Ары қарай loop циклы қайталанады, қайтадан box1 = box1 + 4; оқылады, бірақ бізде қазір box1 өзгергішінде 0 емес, 4, онда box1 + 4 8 тең болады және 8 саны кері box1 өзгергішінде сақталады, енді box1 өзгергішінде 8 саны және келесіде box1 + 4 12 тең болады және т.с.с.`

Осылай азайтумен, көбейтумен не бөлумен жасауға болады, мысалы:

```
box1 = box1 - 1;  
box1 = box1 * 2;  
box1 = box1 / 3;
```

Осыны өздерің мына мысалдармен жасап көріңдер.

Программалауда мұндай тәсілдер жиі қолданылады, бір өзгергішке үнемі бір сан қосып отырады, сондықтан сол үшін өте қысқа жазу ойлап табылған, өткен мысалда біз былай жаздық:

```
box1 = box1 + 4;
```

Программалауда мұндай амалды едәуір қысқа жазуға да болады:

```
box1 += 4;
```

Екі жазу да дұрыс және ол тіпті бір амалды орындайды, `box1` өзгергішіне 4 санын қосады; Дәл осылай бөлуді де, азайтуды, көейтуді орындауға болады, мысалы, мынандай жазу:

```
box1 = box1 - 1;  
box1 = box1 * 2;  
box1 = box1 / 3;
```

және мұндай жазу бір сол амалды орындайды:

```
box1 -= 1;  
box1 *= 2;  
box1 /= 3;
```

Осы мысалдарды өздерің қайталап көріңдер.

Ақырында, программалауда 2 оператор бар, олар қосу мен азайтуға мүмкіндік береді. ++ и --; Оператор ++ бірді қосады, ал оператор – бірді алады, мысалы:

```
box1 = 6;  
box1++;
```

Box1 өзгергішінде біз алдымен 6 санын сақтадық. Одан кейін box1++ деп жаздық, енді біздің өзгергішімізде 7 саны болады, өйткені оператор ++ бірді қосады.

Тағы бір мысал:

```
box1 = 6;  
box1--;
```

Box1 өзгергішінде алдымен 6 сақтаймыз, одан кейін box1—командасын шақырамыз, біздің өзгергіштен бірді азайтады, енді box1 өзгергіште 5 қалады.

Осы мысалдарды өздерің орындап көріңдер.

Бүгін осымен өзгергіштер туралы әңгіме жеткілікті. Бізде тағы бір маңызды тақырып бар – олар шартты операторлар.

Шартты операторлар.

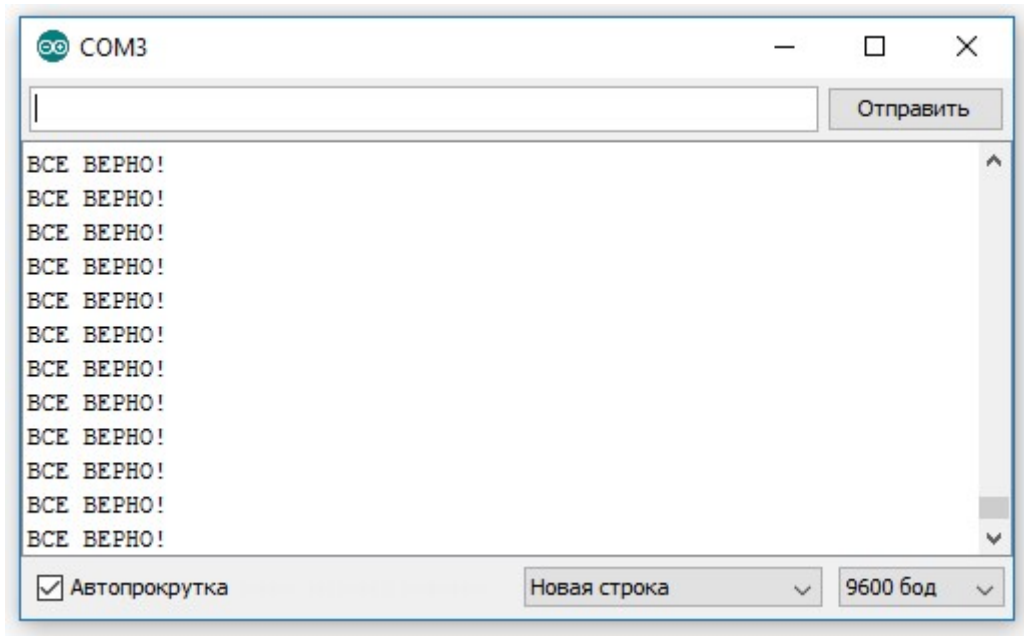
Шартты операторлар дегеніміз не? Тағы қандай жаңа және түсініксіз анықтама?

Нақты өмірде жағдай дегеніміз не екенін білесіңдер? Мысалы, ертеде ата-аналарың: “Егер сабағыңды орындап болсаң, ойнауға барасың!” деп, не мектепте мұғалімдер “Егер үй тапсырмасын орындамасаңдар, онда екі аласыңдар!” және т.с. айтатын.

Программалауда да жағдай бар, онсыз программалау мүлдем пайдасыз болар еді, онда өмірдегідей программаға жағдай берер едіңдер, мысалы “Егер қозғалыс датчигі жұмыс істесе, сигнал беруді қосу!” не “Егер қазір 22.00, онда ауладағы шамды жақ!”, бірақ жағдаят жай сөйлемдермен жазылмайтын түсіндіңдер, ол үшін арнайы шартты оператор бар () **if{ }**. **If** сөзі ағылшыннан “егер” болады. Дөңгелек жақшаның ішіне тексергіміз келген жағдайды жазамыз, ал фигуралы жақшаның ішіне программаны орындайтын кодты жазамыз, егер жағдай орындалған болса, мысалы

```
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  if(5 > 0){  
    Serial.println("БӨРІ ДҰРЫС!");  
  }  
}
```

Бұл кодта жүйелі порт мониторуна БӨРІ ДҰРЫС! Жазуын сол жағдайда шығарамыз, егер 5 0-ден көп болса, 5 қалай да 0 көп екенін білеміз (*барлық әріптер мұнда ағылшынша жазылғанына - bce berpho! назар аудар, тек жоғары регистрде (CAPS LOCK басу керек)*)



Егер 5 0-ден кіші деп жазсақ, онда жағдай орындалмас еді, тексеріп көрейік

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  if(5 < 0){
    Serial.println("БӘРІ ДҰРЫС!");
  }
}
```

Енді, егер программаны жіберсек, онда порт мониторын ашсақ, онда ештеңе көрмейміз, өйткені жағдай дұрыс емес, 5 ешқашан 0-ден кіші болмайды және программа ешқашан шығармайды, **Serial.println("БӘРІ ДҰРЫС!");** амалын орындамайды.

Екі санды салыстыру үшін, математикадағыдай олардың арасына көп не аз салыстыру белгісін қояды, математикада программалауда да басқа салыстыру символдар бар:

Салыстыру символы	Ол нені білдіреді
>	көп – жағдай дұрыс, сол жақтағы цифр оң жақтағыдан көп
<	аз – жағдай дұрыс, егер сол жақтағы цифр оң жақтағыдан аз
==	тең – жағдай дұрыс, екі сан да тең
!=	тең емес – жағдай дұрыс цифрлар тең емес
<=	аз не тең – жағдай дұрыс, сол жақтағы цифр оң жақтағы цифрдан аз не тең
>=	көп не тең – жағдай дұрыс, сол жақтағы цифр оң жақтағы цифрдан көп не тең

Мысалы, егер кодты жазсақ:

```
if(5 == 5){  
    Serial.println("ДҰРЫС!");  
}
```

Онда экранға ештеңе ДҰРЫС! жазуы шығады, өйткені 5 5-ке тең, егер кодты жазсақ:

```
if(5 != 5){  
    Serial.println("ДҰРЫС!");  
}
```

Онда экранға ештеңе шықпайды, өйткені 5 беске тең, тек 5!=5 жағдайында жазуды шығаруға команда береміз, ондай болу мүмкін емес, сондықтан экранға ештеңе шықпайды.

Егер жағдай орындалмаса программалауда кодты орындауға программа беруге болады. Мысалы, бір кезде бізге “Егер жақсы тамақтансаң, үлкен де күшті боласың, әйтпесе кішкентай боласың” деген болатын.

Егер орындалатын жағдай дұрыс болмаса, программалауға кодты беруге болады, ол үшін **ELSE{ }**, операторы бар, ағылшыннан – басқаша деп аударылады. Оның қалай жұмыс жасайтынын қарастырайық:

```
void setup() {  
    Serial.begin(9600);  
}  
void loop() {  
    if(5 < 0){  
        Serial.println("ДҰРЫС!");  
    } else {  
        Serial.println("ДҰРЫС ЕМЕС!");  
    }  
}
```

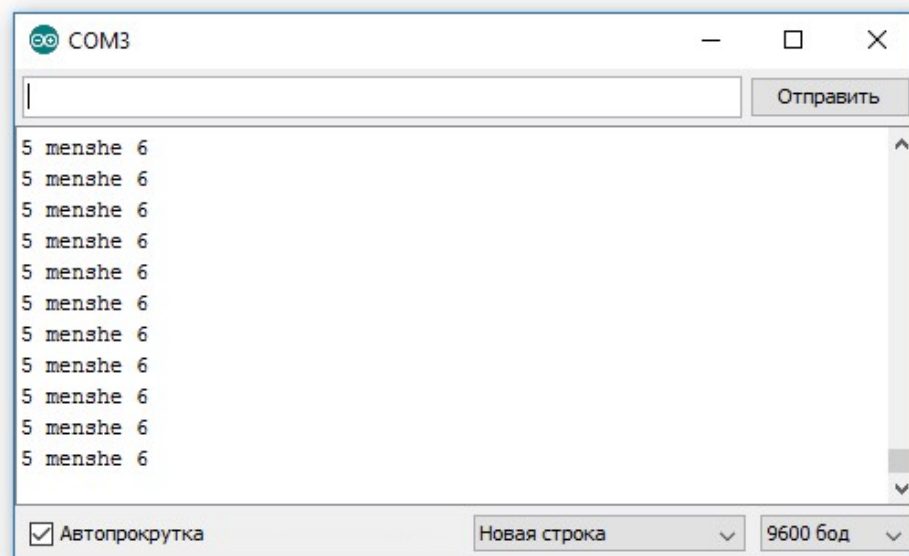
Фигуралы жақшада **else** операторынан кейін жазылған код сол жағдайда орындалады, егер **if()** жазылған жағдай дұрыс болмаса, яғни қазір экранға “ДҰРЫС ЕМЕС!” жазуы шығу керек, өйткені жағдайда 0 5-тен кіші деп тұр, өйткені 5 көп, демек жағдай дұрыс емес және **else-де** жазылған **{ Serial.println(“ДҰРЫС ЕМЕС!”); }** код орындайды.

If операторында тағы да өте қолайлы қасиеті - **else if(){ }** бар. Егер **if()** жазылған жағдай дұрыс болмаған кезде, ол **else{ }** сияқты орындалады. Бірақ **else{ }** айырмашылығы, **else if(){ }** өзінің қосымша жағдайынды жазуға болады. Мысалдар келтіреміз:

```
void setup() {  
    Serial.begin(9600);  
}  
void loop() {  
    if(5 < 0){  
        Serial.println("5 menshe 0");  
    } else if( 5 < 3) {  
        Serial.println("5 menshe 3");  
    } else if( 5 < 6) {  
        Serial.println("5 menshe 6");  
    } else if( 5 < 10) {  
        Serial.println("5 menshe 10");  
    }  
}
```

```
}
```

Енді егер осы кодты салсақ, тек соңғы "5 menshe 6" жазуы шығатынын көреміз, өйткені $5 < 6$ жағдайы $5 < 10$ дұрыс жағдайына қарамастан орындалды, "5 menshe 10" жазуы бәрібір serial шықпайды, өйткені программа оның біріншісі орындалған соң, жағдайды тексеруді тоқтатады.



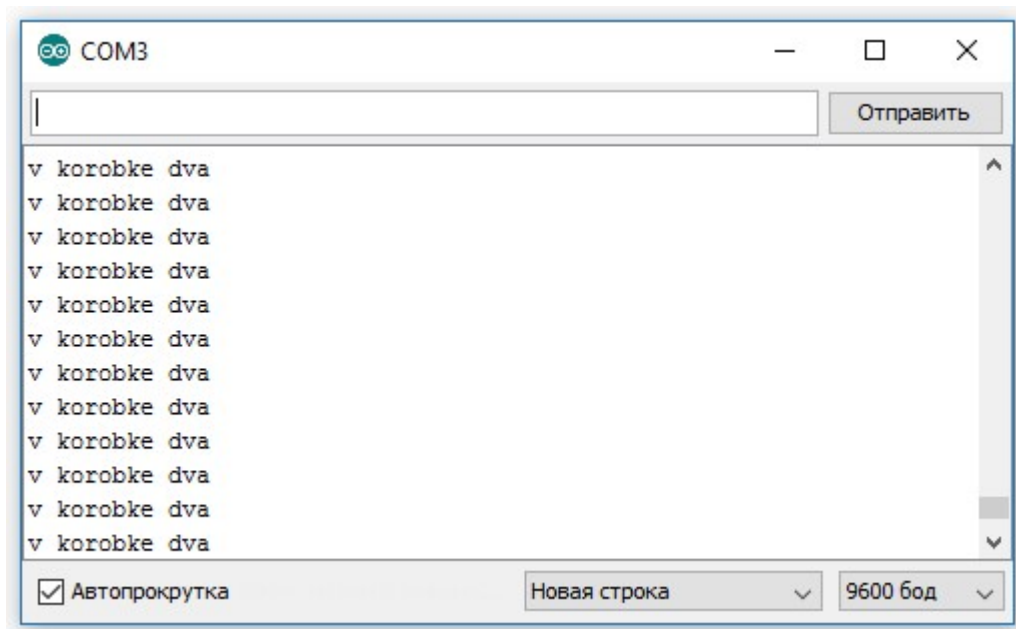
Ал егер бізге 2 жағдайды тексеру керек болса ше.

Шартты **if** операторынан басқа **switch() { case: break;}** операторы бар.

Ол **if** операторы не істесе соны істейді, бірақ азырақ басқаша, код мысалын жазайық, оны орындайық, сонда бізге **switch()** қалай жұмыс жасайтыны түсінікті болады.

```
int korobka;
void setup() {
  Serial.begin(9600);
}
void loop() {
  korobka = 2;
  switch(korobka){
    case 1:
      Serial.println("v korobke odin");
      break;
    case 2:
      Serial.println("v korobke dva");
      break;
    case 3:
      Serial.println("v korobke tri");
      break;
  }
}
```

Программаны Arduinoға қосамыз, порттың мониторын ашып, бізде не болғанын қараймыз:

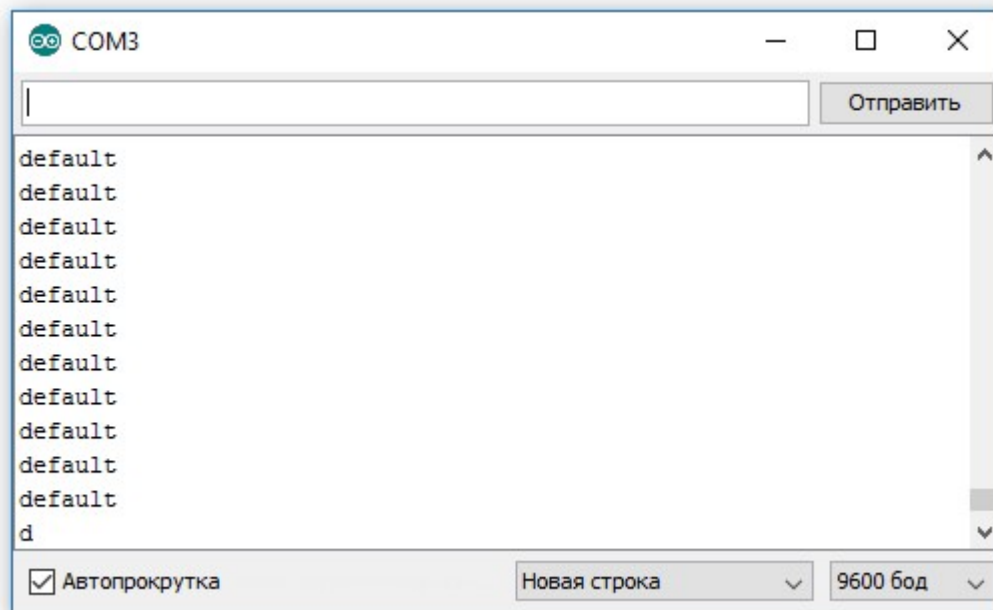


Экранға “v korobke dva” жазуы шыққанын көреміз, ол программамен шықты, өйткені **korobka** өзгергішінде 2 саны сақталған болатын. Біз korobka өзгергішіне switch жағдайын қойдық, **switch операторы case** блогынан біздің өзгергішті әр мәнімен салыстырды, өйткені **case**нің бірінші блогында 1 цифры болды, программа өзгергіштегі 1 цифрының мәні тең бе, соны тексерді, ол тең емес болып шықты да, программа ары қарай кетті. Келесі **case**де бізде 2 цифры, программа тағы да өзгергіште жатқан сан 2 цифрына тең бе, соны тексерді, **korobka** өзгергішінде 2 саны жатқандықтан, тексеру ойдағыдай өтті де, екінші кейсте болған код орындалды. **Break;** командасы фигуралы жабылатын жақшаның “}” істейтінін **if** жағдайында істейді, ол **case** блогын жабады, яғни “**case...break;**” бұл сол сияқты “{...}”, тек ұзынырақ жазылады.

If жағдайындай, бізде **else{...}** бар, сондай **switch** те бар, **default:...break;** жағдайы бар, егер бірде-бір **case** жұмыс істемеген кезде ол орындалады, мысалы, егер біз алып, korobka өзгергішіне 5 санын берсек, онда бірде-бір **case** жұмыс істемейді, егер **default** қоссақ, онда нақ ол жұмыс жасайды:

```
int korobka;
void setup() {
  Serial.begin(9600);
}
void loop() {
  korobka = 5;
  switch(korobka){
    case 1:
      Serial.println("v korobke odin");
      break;
    case 2:
      Serial.println("v korobke dva");
      break;
    case 3:
      Serial.println("v korobke tri");
      break;
    default:
      Serial.println("default");
      break;
  }
}
```

Егер біз шілтені салсақ, онда экранда “default” жазуы шығады:



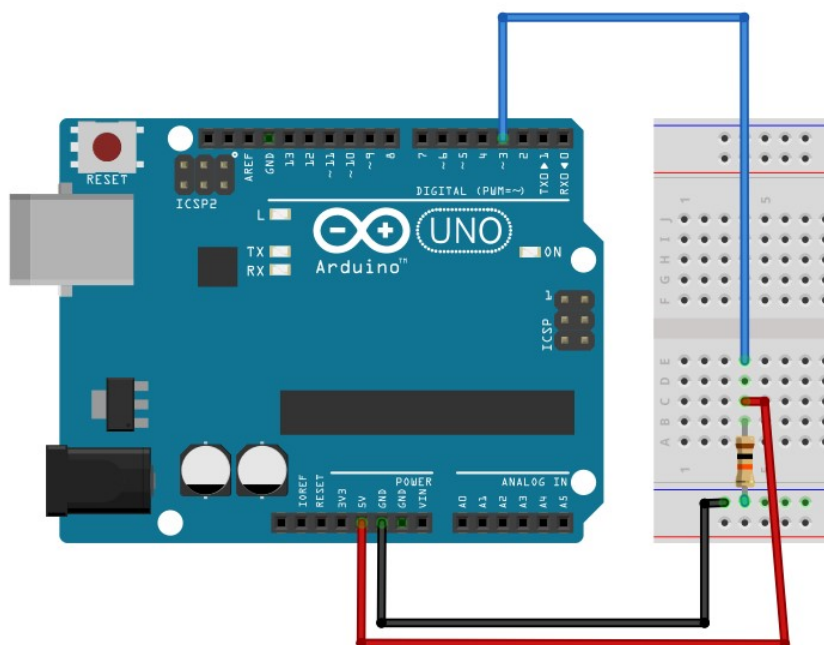
Switch шартты операторы **if** сияқты жиі қолданбайды, сондықтан оған көп көңіл аудармаймыз.

№4 сабақ

Тақырыбы: Енгізу және шығару порттарымен жұмыс, константалар, ауыспалы булвалар

Бүгін біз Arduino цифрлы портынан мәліметтерді салыстырып үйренеміз. Кнопканы қосамыз да, оның көмегімен әртүрлі амалдарды басқарамыз. Цифрлы порттан мәліметтерді салыстырып оқу үшін, Arduinoда арнайы **digitalRead()**; функциясы бар. Төменде суретте көрсетілгендей плата макетіне Arduino қосып көрейік:

Arduino порты	Плата макеті
+5V	Резистор1 10кОм
GND	Резистор1 10кОм
3 цифрлы порт	+5V Arduino



ina

Аяғында біз 5V Arduinoдан тікелей 3 цифрлы портқа қостық, сонымен бірге 3 цифрлы портты резистор арқылы жермен жалғадық, неге бұлай жасадық, кешірек түсіндіремін. Енді келесі кодты жазайық:

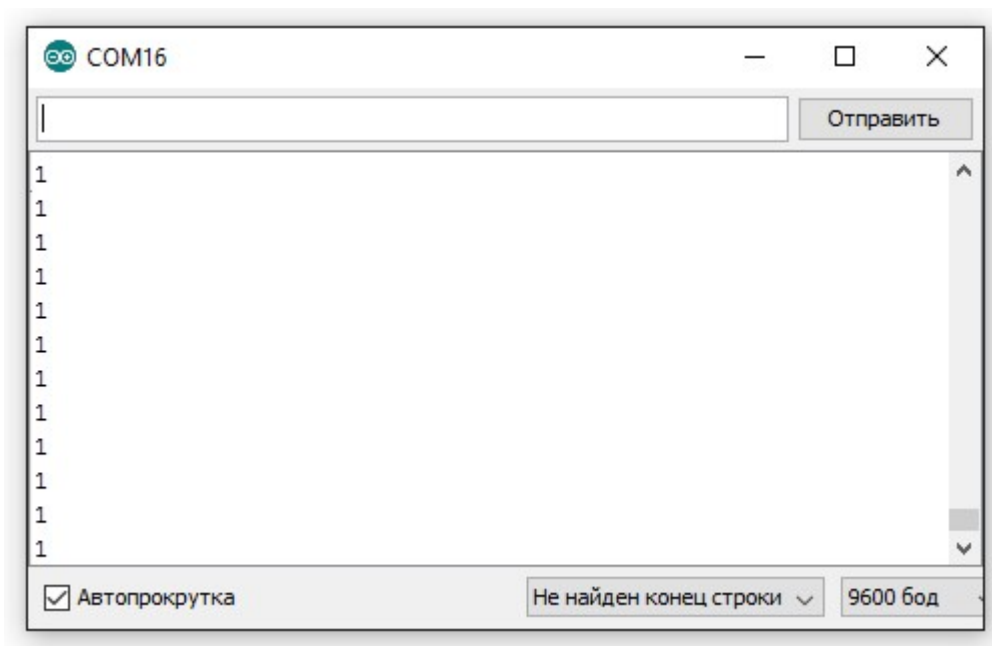
```
int pin;
void setup() {
  Serial.begin(9600);
  pinMode(3, INPUT);
}
void loop() {
  pin = digitalRead(3);
  Serial.println(pin);
}
```

Енді кодты әр жол бойы талдайық.

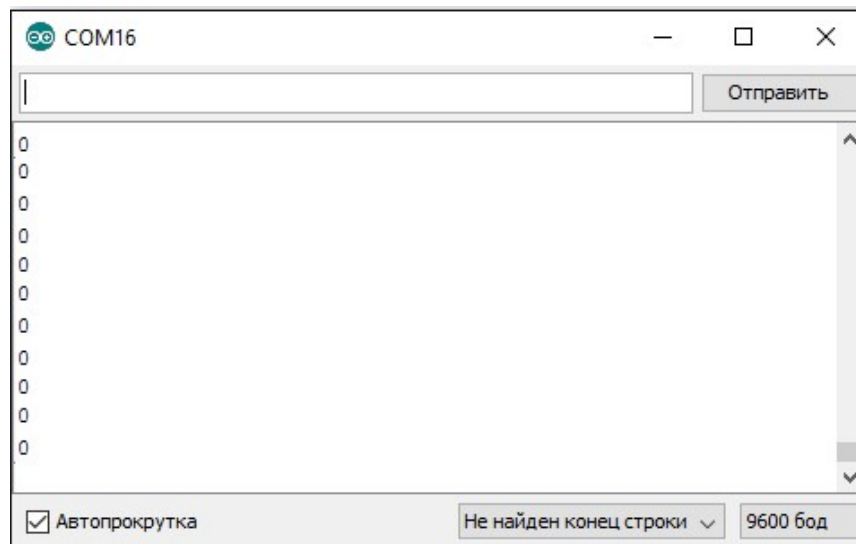
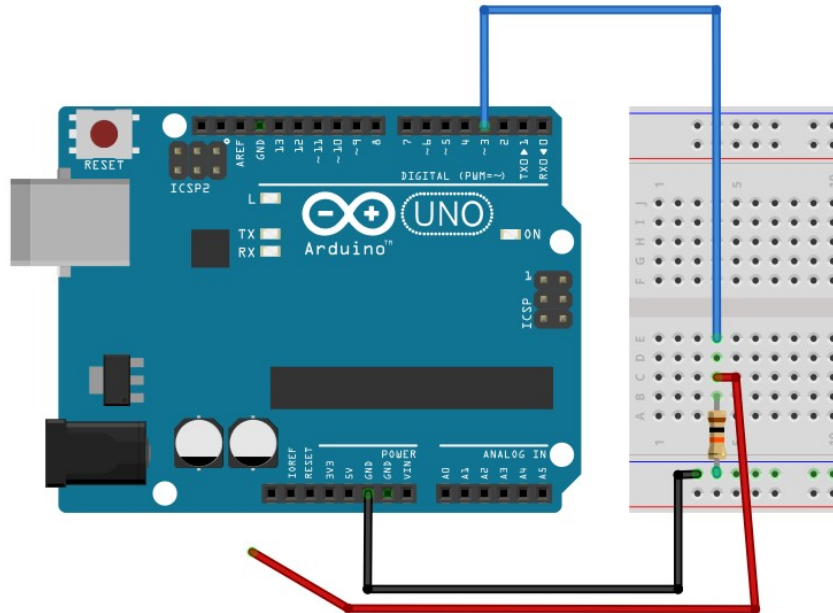
Setup() біз **pinMode(3, INPUT);** деп жаздық, **pinMode** командасы порттың қосу не шығару жұмысының тәртібін белгілейді, светодиодты қосқанда **OUTPUT** шығару режиміне тіркедік. Енді біз керісінше порттан кернеуді салыстырғымыз келеді, бізге **INPUT** режимін орнату керек. Команда **pinMode(3, INPUT);** салыстырып оқуға режиміне 3 портты бекітеді, біз үшінші портқа 5 не 0 вольтті беруге болады және мәліметтерді салыстырып оқиды.

Loop() блогында бізге таныс емес жаңа команда - **digitalRead(3);** бар, өткен сабақтарда **digitalWrite(3, HIGH);** ұқсас команданы қолданған болатынбыз, естерінде шығар, оның көмегімен Arduino портына 5 вольт не 0 вольт береміз. Функция **digitalRead();** керісінше порттан кернеуді салыстырып оқиды. Егер портқа кернеу берілсе, онда функция 1 береді, егер кернеу берілмесе 0 болады. Параметр ретінде кернеуді салыстырып оқитын порттың нөмерін көрсетеміз.

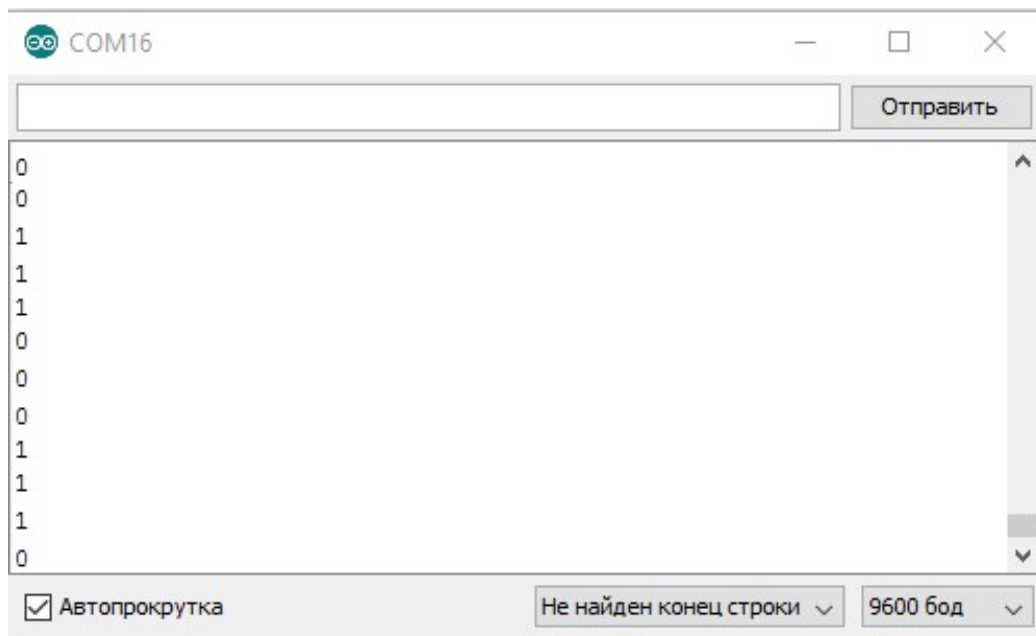
Егер қазір 3 портқа 5 вольт берсек, онда **digitalRead(3)** функциясы 1 беретінін білдіреді. **Pin = digitalRead(3);** жолына сәйкес **pin** өзгергішінде 1 санын сақтау керек. Егер біз порт монитормына **Serial.println(pin);** шығарсақ, онда сақталған мәнді көреміз. Платоға біздің кодты саламыз және не болғанын қараймыз:



Жйелі мониторға бір цифры шығады, өйткені 3 портты Arduinoны 5V қостық. Егер қазір біздің сымды 5 вольттан ағытып тастасақ, порт монитормына 0 шығады:

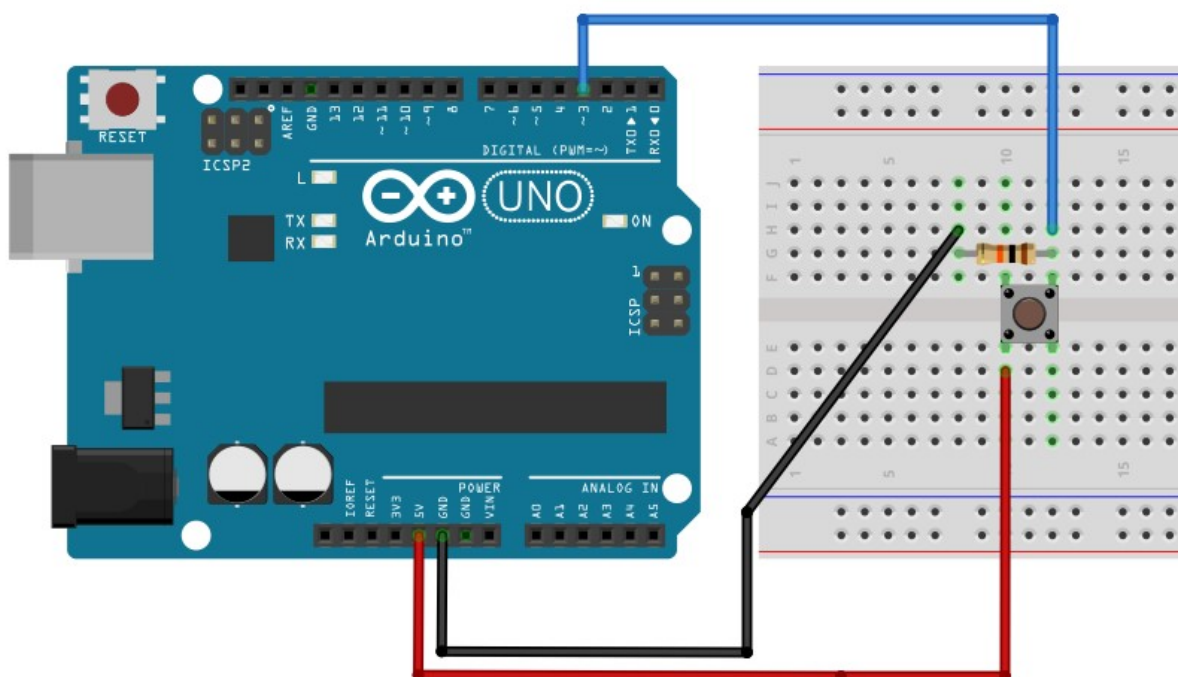


Енді 3 портқа ештеңе берілмейді, онда **digitalRead(3) pin** өзгергішінде 0 мәні сақталады; Енді резисторға оралайық, ол мұнда жай емес, сымды 5V алып тастайық, біздің порт мониторуна нөл беріледі, ал енді резисторды макет платосынан шығарып көрейік, сендер жүйелі порт мониторуна астан-кестен болған бір мен нөлдер шыққанын көресіңдер:

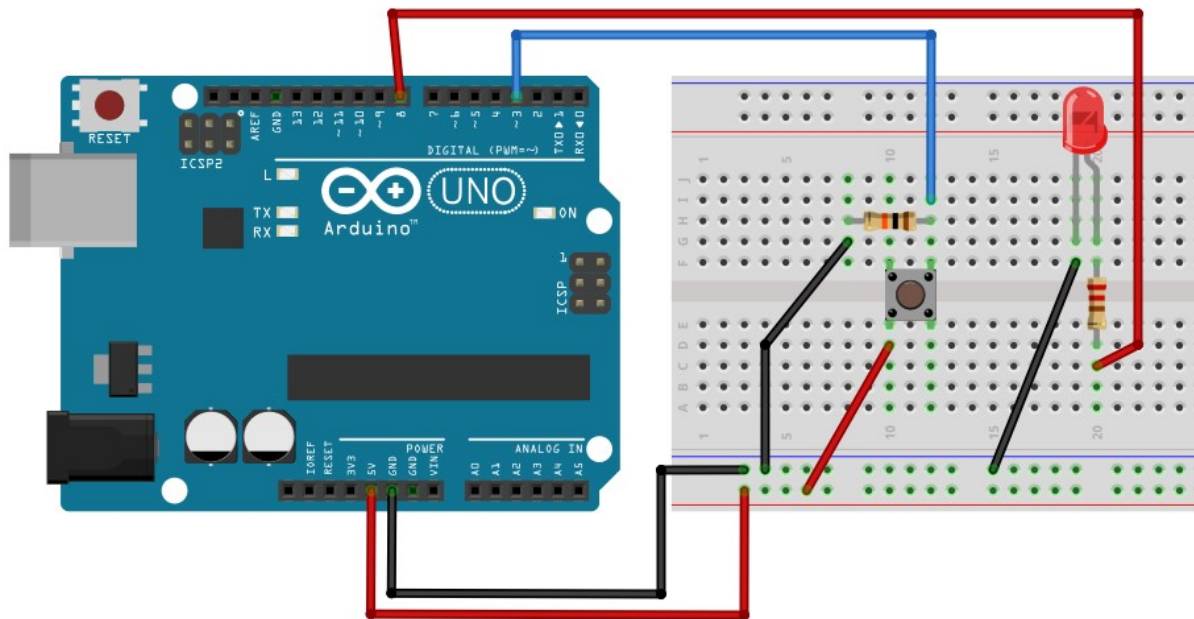


Қазір біз arduinoның 3 портына 5 вольт бермейміз, бірақ бірлер экранда бәрібір көрініп жатады, өйткені біздің айналамызда радиотолқындар, электр сымдарынан нысаналау, статика-электр сияқты электр бөгеттері көп, және Arduino оларды 3 порт арқылы аулайды, радио радиотолқынды қалай аулайды да, бәрін біздің мониторға береді, оған Arduino осындай бөгетті елемеу үшін біздің портқа резистор арқылы жерді міндетті түрде қосу керек, сонда бөгеттен алынатын электрондардың бәрі регистр арқылы кетеді. Егер порттың осындай астан-кестен тәртібін көрсендер, резисторды жерге қосу қажеттілігін еске түсіріңдер. Резистордың кедергісі қандай болу керек?

Кедергі 5кОм артық болу керек. Өйткені резистордың кедергісі Arduino портының кедергісінен артық болу керек, әдетте 10кОм номиналды резисторларды қолданады. Мұндай резистор **тұтастыру** деп аталады – олар электрондарды минусқа тұтастырады. Егер резисторды плюсқа қоссақ, оны **жеткізу** деп атайды. Енді цифрлы портқа қоректену берілгенде, бұл әртүрлі датчиктер мен кнопкаларды қосуға мүмкіндік береді, қарапайымнан бастап такті кнопканы мына схемамен қосайық.



Егер кнопканы бассак, шынжыр тұйықталады да, бізде порт мониторында сондай 1 цифры шығады - кнопканың басылуын анықтай аламыз. Өздерің көргендей, кнопка да резистор арқылы қосылады. Светодиод кнопканы басқанда жанатын етіп жасайық, ол үшін біздің схемаға светодиод пен оны Arduino 8 портына қосамыз, светодиодты міндетті түрде 220 Ом резисторы арқылы, ал кнопканы 10кОм резисторы арқылы қосамыз:



Енді светодиодты кнопкамен қосу үшін кодты жазамыз. Кнопканы басқанда **digitalRead()** функциясы бірді қайтаратынын біз білеміз, оны біле тұра біз жағдай қоямыз, егер **digitalRead()** бірден үлкен болса, онда светодиодты жағамыз, басқаша – светодиодты сөндіреміз.

```
int knopka;
void setup() {
  pinMode(3, INPUT); // 3 порт салыстырып оқу режиміне (қосу) орнатамыз
  pinMode(8, OUTPUT); // 8 портты шығару режимін орнатамыз (шамды жағу үшін орнатамыз)
}
void loop() {
  knopka = digitalRead(3); // 3 порттан өзгергішке кнопканың басылуын санаймыз
  if(knopka == 1){          // Егер кнопка қазір басылған болса
    digitalWrite(8, HIGH); // Светодиодты жағамыз
  } else {                  // Егер кнопка басылмаған болса
    digitalWrite(8, LOW);  // Светодиодты өшіреміз
  }
}
```

Бұл кодтағы барлық жолдар бізге таныс, онда күрделі ештеңе жоқ, егер Сізге олардан бірдеңе түсініксіз болса, онда өткен сабақтарды қайталаңыз. Енді кнопка басылғанда светодиод жанады.

Енді мына тапсырмаларды орындап көріңдер:

1. Керісінше жасаңдар, кнопка басылмағанда – светодиод жанып тұрсын, ал басылғанда сөніп тұрсын.
2. Қысқа басқанда светодиод 3 секунд жанатындай, лезде сөніп қалмайтындай жасаңдар.

3. Светодиод кнопкаға қысқа басудан кейін лезде қосылмай, 5 секундтан кейін қосылып, 3 секунд жанатындай жасаңдар.

4. Кнопканы қысқа басқанда светодиод 1 секундтан 2 рет жыпылықтайтын қылыңдар.

Тағы бір мысалды қарастырайық та, кнопканы қысқа басқанда светодиод қосылсын да, өшірілмесін, ол үшін кнопканы басқанда өзгергіште мысалы 1 цифрын сақтау керек. Ал светодиодты өзгергіште 1 цифры болғанда жандыру керек. Кнопканы жібергенде өзгергіштің мәнін өзгертпейміз, ал светодиод жануын жалғастырады, кәне байқап көрейік:

```
int knopka;
void setup() {
  pinMode(3, INPUT); // 3 порт салыстырып оқу режиміне (қосу) орнатамыз
  pinMode(8, OUTPUT); // 8 портты шығару режимін орнатамыз (шамды жағу үшін
орнатамыз)
}
void loop() {
  if(digitalRead(3) == 1){ //Егер кнопка қазір басылған болса
    knopka = 1;           //Біздің өзгергіште 1 цифрын сақтаймыз
  }
  if(knopka == 1){        //Егер өзгергіште 1 цифр
    digitalWrite(8, HIGH); //Светодиодты жағамыз
  } else {                //Егер өзгергіште басқа цифр
    digitalWrite(8, LOW);  //Светодиодты өшіреміз
  }
}
```

Енді кнопканы басқанда және оны жібергенде, светодиод сөнбейді, жана береді. Енді екінші басқанда светодиод өшетіндей етейік. Оны қалай жасауға болады? Қазір бірінші басу ма, әлде жоқ па, қалай айқындауға болады?

Егер біз бір рет кнопканы бассак, онда knopka өзгергішінде 1 саны болатынын білеміз, өйткені оны сонда сақтадық. Жағдайын өзгерту керек, егер кнопканы бассак, өзгергіште бір бар, онда ол бірді 0-ге айырбастаймыз, мысалы жағдай

```
if(knopka == 1){
  digitalWrite(8, HIGH);
}
```

Жұмыс жасамай, elseде бар код істейді,

```
else {
  digitalWrite(8, LOW);
}
```

Егер нопканы басып, өзгергіште бір жоқ болса, онда светодиодты қосамыз, кәне байқап көрейік:

```
int knopka;
#define pin =7
void setup() {
  pinMode(3, INPUT); // 3 порт салыстырып оқу режиміне (қосу) орнатамыз
  pinMode(pin, OUTPUT); // 8 портты шығару режимін орнатамыз (шамды жағу үшін
орнатамыз)
```

```

}
void loop() {
  if(knopka != 1 && digitalRead(3) == 1){ //Егер бұл бірінші басу
    knopka = 1;           //біздің өзгергішке 1 цифрын сақтаймыз
    delay(1000);
  } else if(knopka == 1 && digitalRead(3) == 1){ //Егер бұл екінші басу
    knopka = 0;           //біздің өзгергішке 0 цифрын сақтаймыз
    delay(1000);
  }
  if(knopka == 1){        //Егер өзгергіште 1 цифр
    digitalWrite(pin, HIGH); //Светодиодты жағамыз
  } else {                //Егер өзгергіште басқа цифр
    digitalWrite(8, LOW);  //Светодиодты өшіреміз
  }
}
}

```

Енді бізде светодиод уақытша басқанда жанады және өшеді, бізде нағыз жарық сөндіргіш болды, әрекетті есіне сақтайтын кнопка фиксациялы кнопка деп аталады, бізде фиксациялы кнопка пайда болды, өйткені қысқа басқанда, светодиод әрқашанда қосылып қалады да, екінші қысқа басқанда әрқашанда өшіріліп қалады. **If(knopka != 1 && digitalRead(3)==1)** тексереді “Егер өзгергіште әлі бір жоқ болып, кнопка басылған болса”, өзгергіште бір жоқ болғандықтан, біз кнопканы бірінші рет басқанымызды білдіреді. Өйткені кнопканы бірінші рет басқанда, онда өзгергіште 1 санын **knopka = 1;** сақтаймыз. Бұл жағдай бұдан былай жұмыс істемейді, онда программа екінші жағдайды **else if(knopka == 1 && digitalRead(3) == 1)** тексереді, мұнда knopka өзгергіші бірге тең бе, соны тексереміз, егер кнопканы 1 рет бассак, онда өзгергіште дәл бір болады, ал егер баспасак болмайтынын білеміз. Демек бұл жағдай тек кнопканы екінші рет басқанда істейді.

Константалар

Енді кнопкадан басқа тақырыпқа ауысайық, константалар дегеніміз не, соны талдайық. Біз сендермен өткізгіштерді өттік, ол қорапқа ұқсас бірдене екенін білеміз, тек программада, өзгергішке мәнді салуға болады, мәнді алуға, ауыстыруға болады.

Константа дегеніміз не? Константа өзгергішке өте ұқсас, онда мәнді де сақтауға, алуға болады, бірақ өзгергіштен айырмашылығы константада мәнді ауыстыруға болмайды, егер 1 рет программада бір мәнді константаға салсақ, онда бұл мәнді басқаға ауыстыра алмаймыз. Константаның осындай үлкен кемшілігімен бізге керек және оның маңызын екінші рет өзгерте алмаймыз?

Іс мынада, константаны Arduino оперативті жадында сақтамайды. Егер өзгергіште 1 рет мәнді сақтау керек болса, онда оны константада сақтаған жақсы. Бұл константалар қалай көрінеді және қалай қолдануға болады?

Константа жасаудың 2 тәсілі бар:

```
const int BOX = 2;
```

Және

```
#define BOX 2;
```

екі команда да бір нәрсе жайды, ол BOX атаудағы константаны жасап, онда 2 мәнін сақтайды. Одан кейін бұл мәнді біз өзгерте алмаймыз. Константаны **const** арқылы хабарлаған болсақ, онда өзгергіштің **int**, **float** және т.с. типтерін көрсету қажет байқаған боларсыңдар. Егер константаны **#define** көмегімен хабарласақ, онда өзгергіштің типін көрсету керек емес. Сендер осы нұсқалардың кез келгенін қолдана аласыңдар.

Өзгергіштегі сияқты константа атауында латын әрпін, цифрлар мен сызу таңбаларын, цифрлар бірінші болмау керек, таңбалардың қайсысын болса да тапсыруға болады.

Әдетте констант атауында бас (үлкен) әріптерді қолданамын, кодта константа қайда, өзгергіш қайда екені түсінікті болу үшін және қайсында мәнді өзгертуге, қайсында өзгертпеуге шатаспас үшін.

Arduinoның өз константтары бар, мысалы, **HIGH** және **LOW**, **HIGH** константасында 1 саны, ал **LOW** – константасында 0 саны сақталады. Мысалы, егер біз жазсақ

```
digitalWrite(8, HIGH);
```

және жазамыз

```
digitalWrite(8, 1);
```

онда мына 2 команда біреуді істейді, олар 8 портқа жоғары кернеу береді.

Өткен мысалдағы барлық константаларды HIGH 1-ге, ал LOW константаларды 0-ге өзгертіп көріңдер де, код бұрынғыдай жұмыс жасай ма, тексеріп көріңдер.

Константаларды не үшін қолдануға болады, оларды тек 1 рет ғана белгіленсе?

Мысалы, өткен мысалда светодиодты қосу үшін 8 портты жиі қолдандық, егер светодиодты 9 портқа қосқымыз келсе, онда барлық жерде кодты өзгерту керек, ал егер 8 саны сақталатын константаны сақтаймыз, онда кодты тек 1 рет өзгертеміз, біздің кодты өзгертіп, константаны қолданып көрейік.

```
int knopka;
#define PIN 8;
void setup() {
  pinMode(3, INPUT); // 3 порт салыстырып оқу режиміне (қосу) орнатамыз
  pinMode(PIN, OUTPUT); // 8 портты шығару режимін орнатамыз (шамды жағу үшін
орнатамыз)
}
void loop() {
  if(knopka != 1 && digitalRead(3) == 1){ //Егер бұл бірінші басы
    knopka = 1; //біздің өзгергішке 1 цифрын сақтаймыз
    delay(1000);
  } else if(knopka == 1 && digitalRead(3) == 1){ //Егер бұл екінші басы
    knopka = 0; //біздің өзгергішке 0 цифрын сақтаймыз
    delay(1000);
  }
  if(knopka == 1){ //Егер өзгергіште 1 цифр
    digitalWrite(PIN, HIGH); //Светодиодты жағамыз
  } else { //Егер өзгергіште басқа цифр
    digitalWrite(PIN, LOW); //Светодиодты өшіреміз
  }
}
```

Біз PIN константасын онда 8 мәнін сақтап жасадық. Барлық жерде 8 портты қолданып, 8 цифрының орнына біз PIN константасын жазамыз. Егер біз светодиодты басқа портқа тез қосқымыз келсе, онда **#define PIN 8;** кодын жай ғана **#define PIN 9;** кодына өзгертеміз; константа PIN қолданылған барлық жерде 8 саны емес, 9 саны шығады, бұл едәуір жылдамырақ, барлық жерде 8 сонын 9 ауыстырғаннан.

Өзгергіш булевалар

Константаларды аяқтадық, енді өзгергіш булеваларды қарастырайық.

3 сабақта біз өзгергіштің әртүрлі типін зерделедік: **int**, **float**, **long**, **unsigned long** және **unsigned int**. Ардуинода, нақтырақ C++ тілінде (өйткені Arduino атап айтқанда осы тілде) программаланады, тағы да мәліметтерді бір типі бар және ол **boolean** деп аталады – мәліметтердің булевалық типі. Бұл типтің өзгергіштері **bool** сөзінің көмегімен жасалады. Бұл типтің өзгергіштері тек 2 мәнді **true** мен **false** (шындық және өтірік) сақтай алады.

Мысалы:

```
bool box;  
box = true;  
box = false;
```

true и false – шынында сол константалар, trueде 1 сақталады, ал falseде 0 сақталады; Сондықтан егер кодты жазсақ:

```
bool box;  
box = 1;  
box = 0;
```

Онда жоғарыдағыны істейміз. Тек 2 мәнді сақтайтын өзгергіштер неге керек, бізде мысалы **int** өзгергіші бар, онда тұтас 65536 әртүрлі мәндерді сақтауға болады.

Дұрысында **int** типтегі өзгергіштер оперативті жадында 2 байтты алып жатыр, ал **bool** типтегі өзгергіштерде - 1 байт, сондықтан **bool** өзгергішін мүмкіндік бар жерде қолданып, оперативті жадын үнемдеп жұмсайсыңдар.

Bool өзгергішін қайда қолдануға болады? Мысалы, тағы да біздің мысалымызда korobka өзгергішінде тек 2 мәнді 1 мен 0 сақтадық. **Int** өзгергішінің орнына **bool** өзгергішін қолдану үшін біздің кодты өзгертіп көрейік:

```
bool knopka;  
#define PIN 8;  
void setup() {  
  pinMode(3, INPUT); // 3 порт салыстырып оқу режиміне (қосу) орнатамыз  
  pinMode(PIN, OUTPUT); // 8 портты шығару режиміне орнатамыз (шамды жағу үшін  
  орнатамыз)  
}  
void loop() {  
  if(knopka != 1 && digitalRead(3) == 1){ //Егер бұл бірінші басу  
    knopka = 1; //біздің өзгергішке 1 цифрын сақтаймыз  
    delay(1000);  
  } else if(knopka == 1 && digitalRead(3) == 1){ //Егер бұл екінші басу  
    knopka = 0; //біздің өзгергішке 0 цифрын сақтаймыз  
    delay(1000);  
  }  
  if(knopka == 1){ //Егер өзгергіште 1 цифр  
    digitalWrite(PIN, HIGH); //Светодиодты жағамыз  
  } else { //Егер өзгергіште басқа цифр  
    digitalWrite(PIN, LOW); //Светодиодты өшіреміз  
  }  
}
```

Мұнда korobka өзгергішінің типін өзгерттік, егер программаны жіберсек, онда ештеңе

өзгермейді, бірақ бос оперативті жад көбірек болды. Енді `korobka` өзгергішінің барлық жерінде салыстырып оқитын не сақталған мәндерді 1 мен 0-ді **true** мен **false**ге ауыстырайық.

```
bool knopka;
#define PIN 8;
void setup() {
  pinMode(3, INPUT);    // 3 порт салыстырып оқу режиміне (қосу) орнатамыз
  pinMode(PIN, OUTPUT); // 8 портты шығару режимін орнатамыз (шамды жағу үшін
орнатамыз)
}
void loop() {
  if(knopka != true && digitalRead(3) == 1){ //Егер бұл бірінші басу
    knopka = true ;           //біздің өзгергішке 1 цифрын сақтаймыз
    delay(1000);
  } else if(knopka == true && digitalRead(3) == 1){ //Егер бұл екінші басу
    knopka = false;
    //біздің өзгергішке 0 цифрын сақтаймыз
    delay(1000);
  }
  if(knopka == true){           //Егер өзгергіште 1 цифр
    digitalWrite(PIN, HIGH); //Светодиодты жағамыз
  } else {                      //Егер өзгергіште басқа цифр
    digitalWrite(PIN, LOW);  //Светодиодты өшіреміз
  }
}
```

Егер программаны жіберсек, ештеңе өзгермегенін көреміз, сондықтан `bool` типтегі өзгергіште не сақтайтының маңызды емес, 1 немесе 0 сақтауға болады, `true` не `false` сақтауға болады, мұнда ешқандай айырмашылық жоқ, егер келешекте кодта `true` не `false` көрсендер, оны 1 не 0 екенін біліңдер.

№5 сабақ

Тақырыбы: Ұқсас порттармен жұмыс, жарық датчигінен, есептуішінен мәліметтерді салыстырып оқу

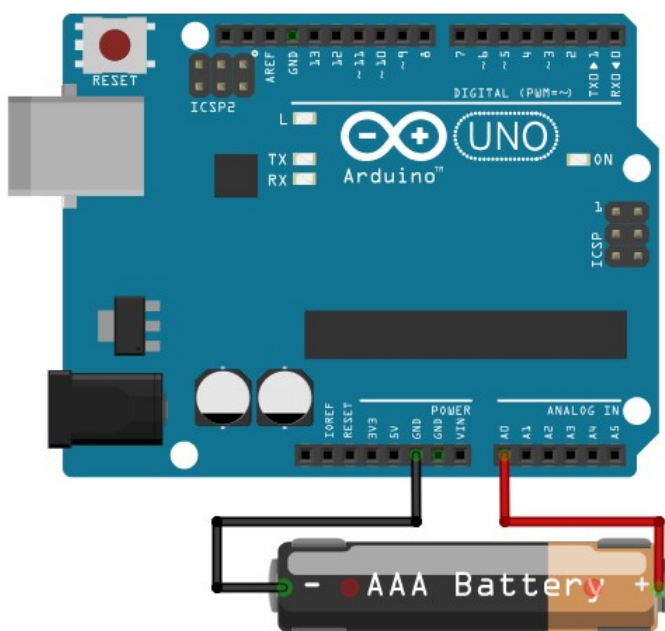
Ақырында датчиктер жұмысына келдік, бүгін біз жарықталғандық деңгейін анықтау үшін фоторезисторды қосамыз. Бірақ ол үшін Arduinoның ұқсас порттарын мәліметтерді салыстырып оқуды үйренуіміз қажет.

Бірінші сабақта Arduinoның ұқсас порттары цифрлардай емес, аса нақты диапазонда 0-ден 5-ке дейін кернеу деңгейін салыстырып оқи алатынын айттық.

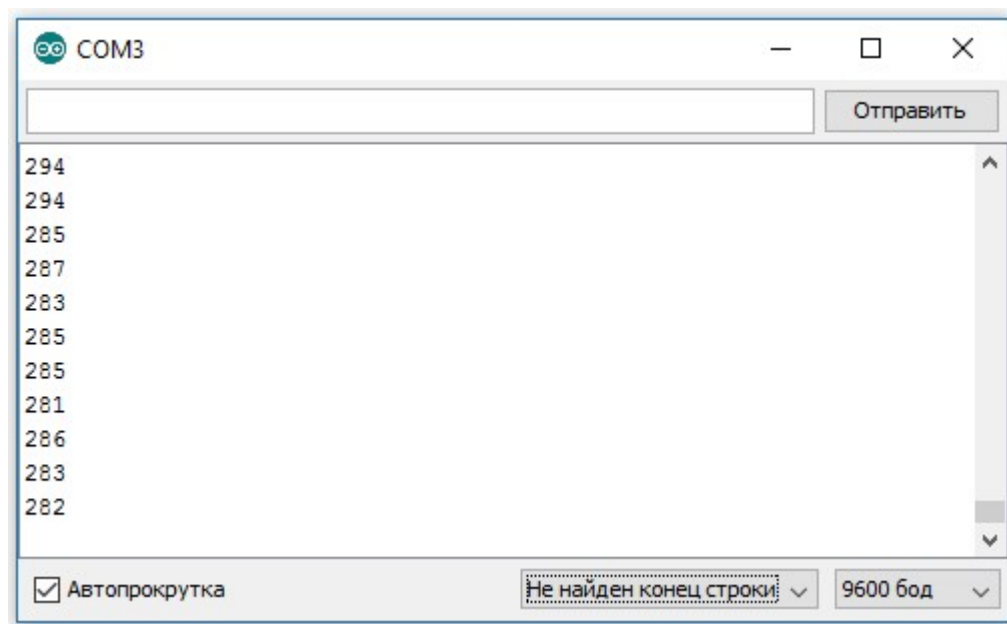
Arduino ұқсас портына берілетін кернеу деңгейін есептеу үшін **analogRead()**; функциясы бар. Функция параметрлерінде A0-ден A5-ке дейін (Arduino UNO) ұқсас порттың нөмірін жазамыз. Бұл функция 0-ден 1023-ке дейінгі сандық мәнін қайтарады, портқа қанша вольт берілуіне байланысты. Егер 0 вольт болса, функция 0-ді қайтарады, егер 5 вольт болса, онда функция 1023 қайтарады. Қалған басқа сандарды кернеу 0 мен 5 вольт аралығында болғанда қайтарады. A0 портынан кернеуді салыстырып оқитын келесі кодты жазайық та, жүйелі порт монитормына не шығатынын көрейік:

```
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  Serial.println(analogRead(A0));  
  delay(100);  
}
```

Енді Arduinoға тігуді салайық та, оған 1.5 вольтті батареяканы қосайық, A0 ұқсас портына плюс батареяканы қосындар, ал батареякадан минусы Arduino минусымен біріктіріңдер.

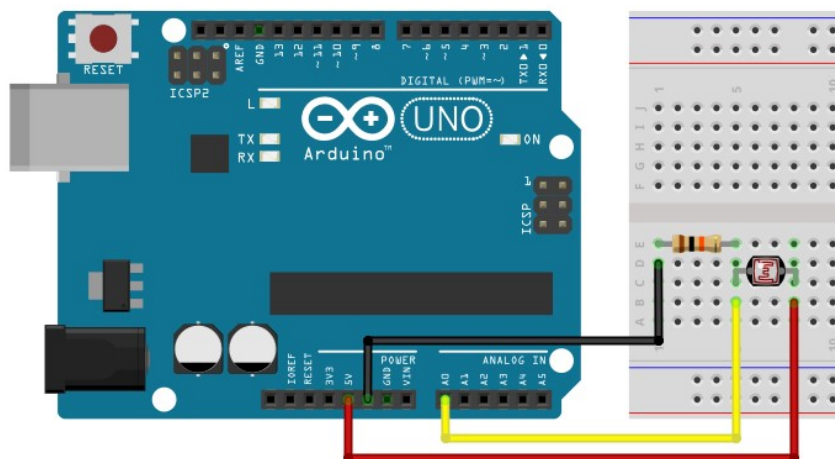


Енді жүйелі порт монитормын ашсақ, онда ұқсас портқа қандай мән берілгенін көреміз:



Сендерде батареикалардың қаншалықты зарядталуына байланысты басқа мән болуы мүмкін. Егер программаны осылай ұзақ уақытқа қалдырсақ, мәnnің біртіндеп азайғанын көреміз, бұл батареиканың отырғанын білдіреді, сондықтан оның кернеуі түседі, сондықтан **analogRead()** функциясы бұрынғыдан аз мән береді.

Фоторезисторды Arduinoға қосайық. Фоторезистор қазір қанша жарық екенін анықтау үшін керек. Фоторезистор әдеттегі резисторға ұқсас. Ол резистор сияқты токқа кедергі болады, бірақ жай резисторға қарағанда ол оған қанша жарық түскендігіне байланысты өз кедергісін өзгертеді, фоторезисторға жарық көп түссе, сонша оның кедергісі аз, егер оны Arduino +5V қоссақ, онда шығуда резистор 5 вольттан аз болады да, айнала жарық аз түсіп, қараңғырақ болса, соншалықты шығуда кернеу аз болады, analogRead() функциясының көмегімен бұл мәліметтерді санап аламыз да, бөлмедегі жарықтанудың деңгейін анықтаймыз. Оны біздің Arduinoға қосайық, мысалы A0 ұқсас портына:



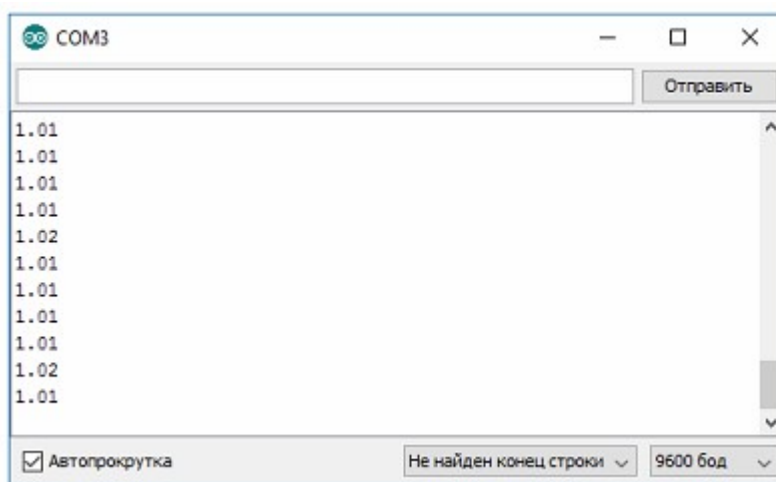
Егер сол тігумен жүйелі порт мониторын ашайық, онда батареикамен сияқты диапазонда 0-ден 1023-ке дейін сандар шыққанын көреміз, егер фоторезисторды бірдеңемен жауып көрейік, оған жарық түспес үшін, онда порт мониторына шығарылған мән азайған, өйткені фоторезистор өзінің кедергісін көбейтіп, портқа берілген кернеу төмендеді. Егер керісінше фоторезисторға ашық бірдеңемен жарық берсек, онда шығарылатын мән көбейеді, өйткені фоторезистордың кедергісі азайды да, портқа берілетін кернеу өсті.

Айтпақшы, портқа қанша вольт берілгенін санауымызға болады.

Егер портқа 5 вольт берсек, **analogRead()** 1023 санын көрсететінін білеміз. Бір вольтқа қандай цифр сай келетінін есептейміз, ол үшін $1023 / 5$ бөліп, 204.6 аламыз. Егер экранға кодта есептелген санды турасынан шығармай 204.6 бөлсек, онда портқа берілетін кернеуді аламыз, мысалы:

```
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  Serial.println(analogRead(A0)/204.6);  
  delay(100);  
}
```

Енді портқа 0-ден 5-ке дейін вольт кернеуі шығады, қазіргідей қараңғыға байланысты, әдетте мән ешқашан дәл 5 вольт не дәл 0 вольтқа жетпейді, мұндай болу үшін өте қараңғы немесе өте жарық болу керек.



Егер біздің батареяканы кері қоссақ, онда қандай кернеу екенін көреміз. Портқа берілетін кернеуді есептеудің тағы бір тәсілі бар, ондай мақсат үшін Arduinoда **map()** функциясы бар, егер **map()** функциясын мынандай параметрлармен: **map(analogRead(A0), 0, 1023, 0, 5)** шақырсақ, онда ол автоматты түрде порттан алған мәндерді 0-ден 5-ке дейін санға айналдырады. Функцияның 5 параметрі бар екенін көреміз, бірінші параметр айналдыруға қажет саны, екінші параметр – бұл санның минималды мүмкін мәні, біз үшін ол 0, өйткені **analogRead()** функциясы 0 қайтарады, егер портқа ештеңе берілмесе. Функцияның екінші параметрі бұл санның минималды мүмкін мәні, ол айналдыратын сан, ол бізде 1023. Үшінші параметр – бұл санның минималды мүмкін мәні, ол айналдыратын сан, яғни 0 вольт, соңында төртінші параметр – бұл біз айналдырғымыз келетін максималды мүмкін мәні, біз үшін ол 5 вольт, бұл функцияны қолданып көрейік:

```
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  Serial.println(map(analogRead(A0), 0, 1023, 0, 5));  
  delay(100);  
}
```

Нәтиже өткен мысалға ұқсас болады.

Енді кішігірім тапсырманы орындайық:

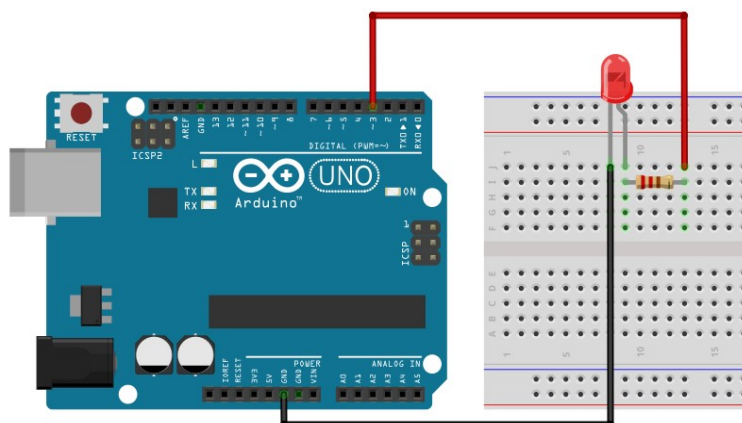
1. *Arduinoға светодиодты қосыңдар да, оны қараңғы болғанда қосылаттындай, жарық болғанда сөнетіндей етіңдер.*

AnalogRead() функциясының көмегімен ұқсас порттардан мәндерді салыстырып оқыдық, **analogWrite()** функциясының көмегімен кейбіреулерде кернеуді бір қалыпты ауыстыра аламыз. Егер портқа 4 вольттан аз кернеу берілсе, онда порт мониторына: “Low power” ескерту жазылатындай жасаңдар, бұл бізде батарейдің заряды төменді білдіреді.

1. *Схемаға кнопканы қосайық та, одан тек тәуліктің қараңғы кезінде светодиод қосылаттындай жасаңдар.*

AnalogWrite() функциясы

Енді **analogWrite()** функциясын қарастырайық, бұл функцияның **analogRead()** функциясына қарама-қарсы функция екенін сендер сезген шығарсыңдар, бірақ бұл ондай емес. Егер Arduino порттарынан **analogWrite()** функциясы 0-ден 255 дейін мәндерін қабылдаса, егер **analogWrite(3, 0)** шақырсак, онда 3 портқа 0 вольт беріледі, егер **analogWrite(3, 255)** шақырсак, онда 3 портқа 5 вольт беріледі, 0-ден 255 дейін мәндерді беріп, біз 0-ден 5 вольтқа дейін кернеуді бір қалыпты өзгертеміз. Келесі мысалды жүзеге асырып көрейік, ол үшін 3 портқа светодиодты қосамыз:



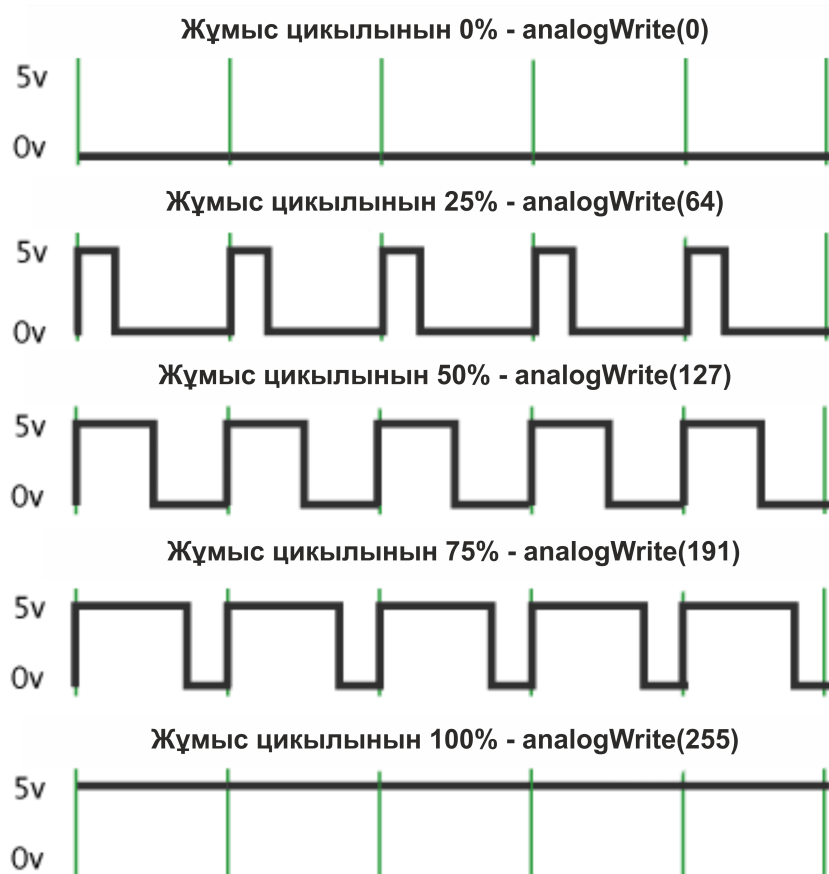
Енді келесі кодты жазып, оны Arduinoға саламыз:

```
void setup() {  
}  
void loop() {  
  analogWrite(3, 0);  
  delay(500);  
  analogWrite(3, 127);  
  delay(500);  
  analogWrite(3, 255);  
  delay(500);  
}
```

Кодты салғаннан кейін светодиод алдымен жанбайды, сосын күңгірт, сосын толық жарықтанады.

Оның толықтығына берілсек, онда **analogWrite()** функциясымен 3 портта кернеуді өзгертпейміз, кезекпен оған 5 және 0 вольтті тез береміз. Бұл өте тез болады, секундына 490 рет (490 герц), бір уақытта жоғары және төмен кернеу берілгенде, светодиод жартылай жанады, егер 5 вольт ұзақ уақыт берілсе, онда светодиод ашық жанады, солай максимумға дейін, **analogWrite(3, 127)** деп жазсақ, онда портқа бірдей бір уақытта 5 және 0 вольт беріледі. Егер **analogWrite(3, 255)** деп жазсақ, онда портқа тек 5 вольт беріледі, егер **analogWrite(3, 0)** деп жазсақ, онда портқа тек 0 вольт беріледі. Оны өте көрнекті <http://arduino.ru/Tutorial/PWM>

алынған суреттен көруге болады:



бұл графикте **`analogWrite()`** функциясы әртүрлі мәндермен жоғары мен төмен сигналдың ұзақтығы өзгертетінін көреміз. Сондықтан жоғары кернеудің (5 вольт) уақыт аралығы ұзағырақ болса, светодиод жарқырап жанады, ал аралық қысқа болса, күңгірт жанып, толығымен сөнуге ейін - **`analogWrite(3,0)`** болады. Барлық порттарда емес, тек арнайы ШИМ порттарда жарықтықты басқара аламыз, олар Arduinoда ирек сызықпен “~” белгіленген. ШИМ дегеніміз не? ШИМ – бұл ендік импульсті модуляция. Егер кезекпен тез төмен мен жоғары кернеу берілсе, бұл импульсті тоқ деп аталады, өйткені жоғары, содан кейін төмен кернеудің қысқа сигналы (импульс) беріледі.

Ағылшыннан ШИМ PWM (pulse-width modulation) деп аударылады. Arduinoда әдетте мынандай жазу бар:



Енді **`analogWrite()`** және **`analogRead()`** функцияларының арасында жалпы ештеңе жоқ екендігін қорытындылай аламыз, **`analogWrite()`** тіпті ұқсас порттармен жұмыс істемейді.

Енді бірнеше тапсырмаларды орындап көрейік:

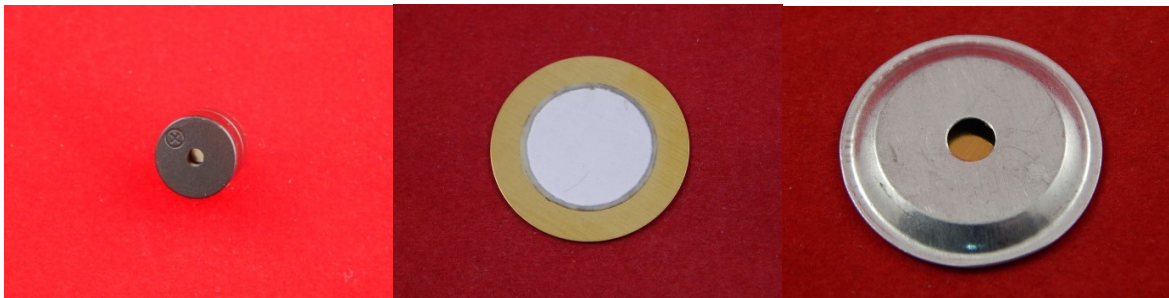
1. *Светодиодтың жарықтығы бір қалыпты көбейіп, қалай максималды мәнге жеткенде, азаятындай етіп жасаңдар. Жалпы алғанда светодиод бір қалыпты жыпылықтауы қажет.*
2. *Arduinoға қосыңдар, оны басқанда светодиодтың жану жарықтығы өзгерсін, жарықтықтың барлығы 3 деңгейі болу керек: светодиод өшірілген, светодиод жартылай жарықтықта жанып тұр және светодиод толық жарықтықта жанып тұр.*

№6 сабақ

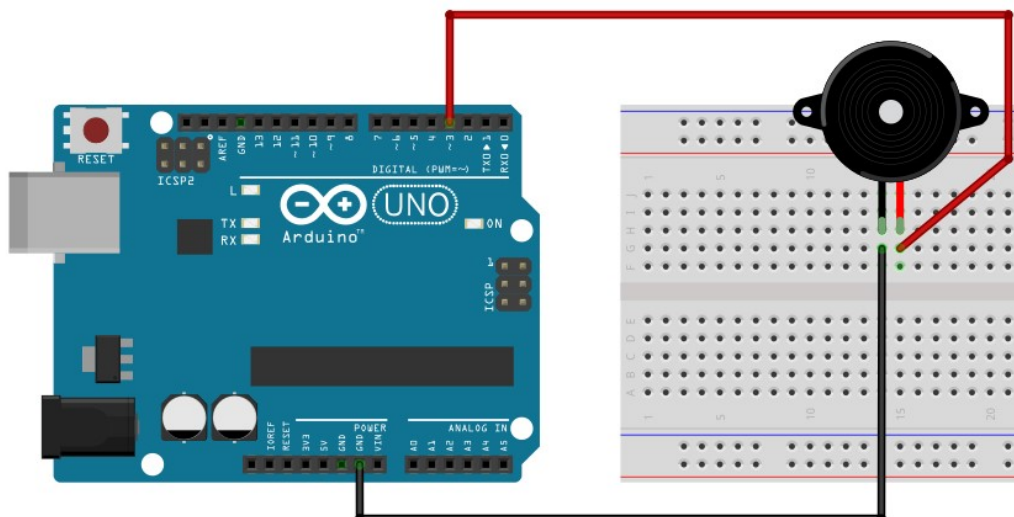
Тақырыбы : Ұқсас порттармен жұмыс, үндесті динамикті қосу, температура датчигімен жұмыс

Өткен сабақта ұқсас порттармен және кең импульсивті модуляциямен (ШИМ) таныстық, бұл сабақта осы тақырыпта практикалық сабақтар орындауды жалғастырамыз, біз Arduinoға тоналді динамик пен температура датчигін қосып, қалай жұмыс істейтінін қарастырамыз.

Тоналді динамик не оны басқаша пьезо-сыбызғы деп атайды, бұл динамик 2-4кГц жиілігінде дыбыстар шығарады, ол 4-8 вольтта жұмыс жасайды. Мүмкін сендер өмірде осындай динамикалармен музыкалық ашық хаттарда, 00 жылдардағы танылған ойында – тетрис, дыбыс шығаратын қытай ойыншықтарда. Олар әртүрлі болады, бірақ мынандайлар жиі кездеседі:



Бүгін біз бірінші нұсқамен жұмыс жасаймыз. Тоналді динамикке импульсивті ток берсек ол дыбыс шығарады, өткен сабақта analogWrite() функциясы ШИМ (PWM) порттары арқылы импульсивті ток бере алатынын, параметрлерде 0 ден 255 дейін мән беріледі, мән қаншалықты жоғары, уақыт бойынша жоғары кернеу импульсі ұзынырақ және уақыт бойынша төмен кернеу импульсі қысқарақ. Бұл егер сыбызғыны кез келген ШИМ портына қоссақ, мысалы 3-ке, және analogWrite(0) командасын жазсақ, онда динамик мүлдем шырылдамайды, ал егер analogWrite(255) деп жазсақ, онда динамик максималды қатты шырылдайды. Arduinoға динамикті қосайық, біздің динамиктің 2 аяғы бар, қысқа да ұзын, олар к + и – сәйкес қосылады. Біз 3 цифр портына плюсті, ал минусты GND-ға қосамыз.



Енді мынандай код жазайық:

```
void setup() {  
}  
void loop() {  
  analogWrite(3, 0);  
  delay(500);  
  analogWrite(3, 100);  
}
```

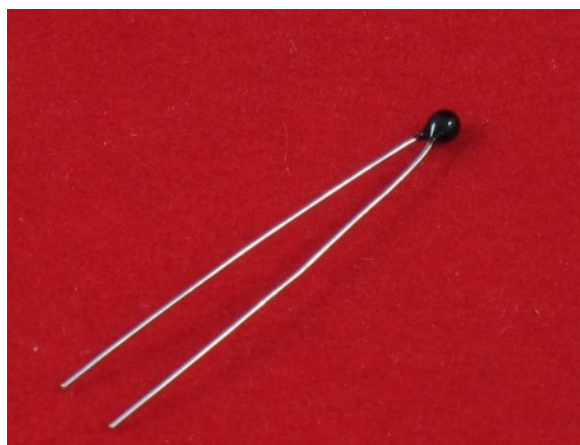
```
delay(500);  
analogWrite(3, 150);  
delay(500);  
analogWrite(3, 200);  
delay(500);  
analogWrite(3, 255);  
delay(500);  
}
```

Өткен сабақта ұқсас программаны қолданғанымыз естерінде шығар, светодиод әр ретте жарығырақ жану үшін. Сол код біздің динамикті қатты одан сайын қатты шырылдауға мәжбүр етеді. Arduino көмегімен дыбыстар шығару қиын емес екенін көрдік, бірнеше тапсырмаларды орындайық:

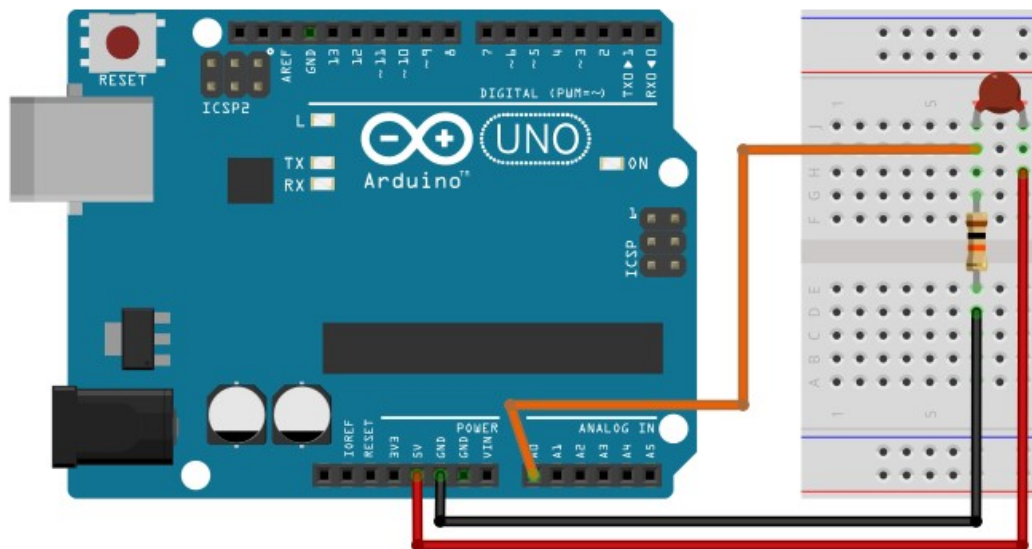
1. Өткен сабақтағы мысалды еске түсірейік, динамикті бір қалыпты қатты және қаттырақ, максималды күшті шырылдау үшін, содан кейін бір қалыпты төмен де төменірек шырылдауды жасаңдар. Мұны `analogWrite()` функциясының көмегімен жасаңдар
2. Терменвокс жасайық, терменвокс бұл музыкалық аспап, ол қолдың оған салуына байланысты дыбыс шығарады <https://ru.wikipedia.org/wiki/Терменвокс>, біз оны жарық түсіру датчигінің көмегімен жасаңдар, жарық датчикке көбірек түссе, динамик қаттырақ естіледі, оны `map()`, `analogRead()` және `analogWrite()` функцияларының көмегімен жасауға болады.
3. Өзіміздің синтезаторды жасайық, Arduinoға 3 кнопканы қосайық, әр кнопканы басқанда дыбыс шығатындай болсын.

Температура датчигі, термистор

Енді термистормен танысайық, өткен сабақта фоторезисторды зерделедік, ол оған түсетін шамның жарықтығына байланысты өз кедергісін өлшеді. Термистор температураға байланысты солай жұмыс істейді, ол өз кедергісін өзгертеді, температура жоғары болса, оның кедергісі төмен, ал температура төмен болса, кедергі жоғары. Термистор 2 аяқ шығатын кішкене қара тамшы түрінде көрінеді, термисторда резистордағыдай плюс те, минус те жоқ. Бұл қара тамшының ішінде кіші зат кесегі бар, ол температура өзгергенде өзінің кедергісін өзгертеді. Біз MF52 термисторын қолданамыз, оның кернеуі 0-ден 10кОмға дейін өзгереді. Ол төменгі фотода бейнеленген:



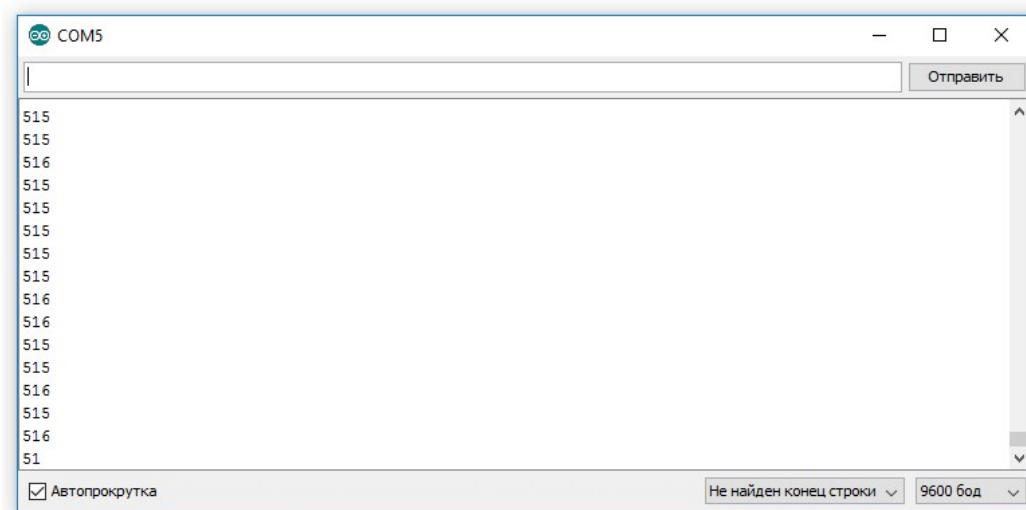
Енді оны Arduinoға қосайық, оны қосу схемасы жарық датчигі сияқты, бір аяғы плюске қосылады, екіншісі ұқсас портқа қосылып, жерге 10кОм резисторымен тартылады.



Енді жүйелі порт мониторына ұқсас порттан шыққан мәнді шығарайық:

```
int temp;
void setup() {
  Serial.begin(9600);
}
void loop() {
  temp = analogRead(A0);
  Serial.println(temp);
}
```

Егер біздің кодты салып, жүйелі мониторын ашсақ, онда бізде порттан салыстырып оқылатын мәндердің температураға байланысты өзгеретінін көреміз



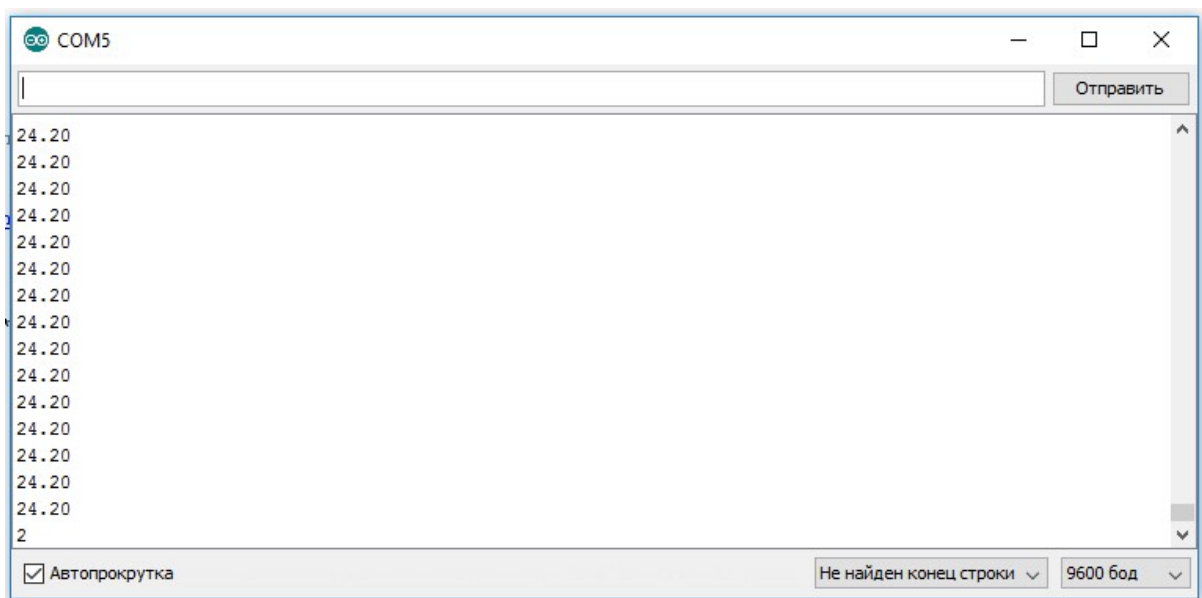
Енді термисторға қолымыздың саусағын тигізейік, портта кернеудің өскенін көреміз, дәл қара тамшыға тигізіндер, контактыларға тиюге болмайды, әйтпесе өлшем тура болмайды. Стаканға ыстық су құйып, термисторға жақынырақ әкеліп көріндер, тіпті тигізуге болады.

Википедияда осы термистордың кедергіге байланысты температурасын есептеу формуласы бар, оны мында <http://en.wikipedia.org/wiki/Thermistor> көруге болады, біз оның нақтылығына берілмейміз, өйткені бүкіл сабақты алады, сабақ логарифмдер мен сандардың дәрежесін табумен мүмкін таныс емес оқырмандарға арналған. Сондықтан бірден дайын функцияны

жаздық, ол формуламен біздің термистордың температурасын есептейміз, оны getTemp() деп атадық, оның ішіндегісін нақтыламай, оны жай көшіріңдер де, программаны жіберіңдер. Бұл функция Arduinoның ұқсас портынан мәнді алады да, лезде температураны береді.

```
float temp;
void setup() {
  Serial.begin(9600);
}
void loop() {
  temp = getTemp(analogRead(A0)); //Температураны есептейміз
  Serial.println(temp);           //Температураны экранға шығарамыз
}
// Формуласы бойынша жазылған http://en.wikipedia.org/wiki/Thermistor
double getTemp(int analogVal) {
  double t;
  t = log(((10240000/analogVal) - 10000));
  t = 1 / (0.001129148 + (0.000234125 * t) + (0.0000000876741 * t * t * t));
  t = t - 273.15; // Цельсияда Кельвинді конверсиялаймыз
  return t;
}
```

Енді программаға саламыз, жүйелі порт мониторын жібереміз, ағымдағы температура лезде шығады:



getTemp(analogRead(A0)) функциясы ұқсас порттан мәнді салыстырып оқиды, оның температурасын Цельсия градусына айналдырады да temp өзгергішінде сақтайды, ары қарай бұл өзгергішті жүйелі портқа шығарамыз. Енді датчикке қолыңызды тигізіңіз, оның температурасы тез өзгергенін көресіңдер. Бұл температура қазір дәл емес, өйткені макет платасы мен сым арқылы термисторды қосамыз, олардың өз кедергісі бар, аса нақты мәндер алу үшін термисторды тікелей Arduinoға дәнекерлеу қажет еді.

Термистормен жұмыс істеуде қиындық жоқ, енді бірнеше тапсырмаларды орындайық:

1. Температурасы 30 градус Цельсиядан асқанда динамик шыңғырлай бастап, ал төмендегенде тоқтайтындай жасаңдар.
2. Кнопканы қосыңдар және қандай болмасын өзгергіштің кнопкасын басқанда температура сақталып, бұл өзгергіш жүйелі порттың мониторына шығатындай етіп жасаңдар.

№7 сабақ

Тақырыбы: Мәліметтер массивімен танысу, циклдар, массивтерді сұрыптау, потенциометрмен жұмыс

Бүгін бізде қиын тақырып, біз массивтерді зерделейміз. Массивтер дегеніміз не? Жай сөзбен айтқанда – бір өзгергішке біріккен өзгергіштер үймесі.

Мысалы, біз 10 өзгергіш жасау керек деп елестетейік және онда сандарды сақтау керек. Мұны онда былай жазар едік:

```
int box1;  
int box2;  
int box3;  
int box4;  
int box5;  
int box6;  
int box7;  
int box8;  
int box9;  
int box10;
```

Бұл көп орын мен уақыт алатынына келісіндер, егер әр өзгергіште бір сан сақтағымыз келсе, онда бізге әр өзгергішті санап қою қажет:

```
box1 = 5;  
box2 = 7;  
box3 = 6;  
box4 = 1;  
box5 = 4;  
box6 = 80;  
box7 = 5;  
box8 = 21;  
box9 = 40;  
box10 = 0;
```

Массивтермен едәуір қарапайым, жай командамен өзгергіштер массивін жаза аламыз, мысалы, егер 10 өзгергіштен массив жасағымыз келсе, онда бізге былай жазу жеткілікті:

```
int box[10];
```

Енді бізде 10 өзгергіштен тұратын массив бар, басқаша айтқанда 10 элементтен тұратын массив деп аталады, әр өзгергішке (элементке) өткен мысалдағыдай мәндерді сақтай аламыз:

```
box[0] = 5;  
box[1] = 7;  
box[2] = 6;  
box[3] = 1;  
box[4] = 4;  
box[5] = 80;  
box[6] = 5;  
box[7] = 21;
```

```
box[8] = 40;  
box[9] = 0;
```

Яғни квадратты жақшада біздің өзгергіштің нөмірін көрсетеміз, программалауда нумерациялау нөлден басталатынын ескертеміз, сондықтан box массивінің бірінші элементінде мәнді сақтағымыз келсе, онда box[0] = 5 деп жазамыз.

Бірақ мәліметтерді былай сақтау ыңғайсыз, бұл көп орын алады, массивтерде тағы бір плюс бар, массив жасағанда лезде барлық мәндерді сақтай аламыз, ол мынандай болып көрінеді:

```
int box[10] = {5,7,6,1,4,80,5,21,40,0};
```

Біз элементтің нөмірін көрсетпей бос квадрат жақшаларды қоямыз да, массивке фигуралы жақшада қойылған барлық керекті сандарды үтір арқылы береміз. Енді біздің код әлдеқайда аз орын алады, 20 жолдың орнына 2 жол. Бұрынғыдай массивтің әр элементіне жекелей қарауымызға болады, мысалы егер код жазсақ:

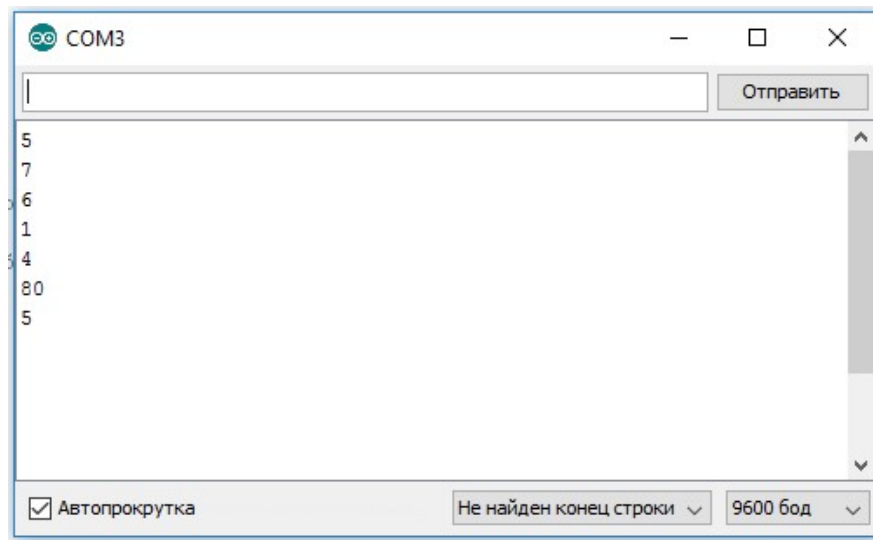
```
int box[10] = {5,7,6,1,4,80,5,21,40,0}; //10 элементті массив жасаймыз  
                                     //Және онда элементтерді бірден сақтаймыз  
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  Serial.println(box[0]); //Массивтің нөл элементін экранға шығарамыз, яғни 5 саны  
}
```

Ақырында экранға 5 саны шығады, өйткені ол массивтің нөл элементі болып табылады.

Сондай-ақ элемент нөмірін ашық көрсетпеуге болады, өзгергіште элемент нөмірін сақтай аламыз, сосын элемент нөмірінің орнына өзгергішті қоямыз, мысалы, егер int i=0 өзгергіші сақтасақ, содан соң экранға box[i] шығарсақ, онда экранға 5 саны да шығады. Өзгергішті көбейтіп, массивтің барлық элементтерін кезекпен шығара аламыз, мысалы:

```
int box[10] = {5,7,6,1,4,80,5,21,40,0}; //10 элемент мөлшеріндегі массив жасаймыз  
                                     //Онда лезде элемент сақтаймыз  
int i=0;  
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  Serial.println(box[i]); //Экранға массивтің нөл элементін шығарамыз, яғни 5 саны  
  i++; //i өзгергішінің 1 көбейтеміз  
  if(i==10){ //Егер i 10-ға тең болса, онда массив мөлшерін көбейтпес үшін оны нөлдейміз  
    i=0;  
  }  
  delay(1000);  
}
```

Енді экранға кезекпен массивтің барлық элементтері шығады:



Өзгергіші 9-дан асып кеткенде, оны нөлдейміз, өйткені бізде массивте оныншы элемент жоқ, егер оның мәнін алуға талаптансақ, онда қате мәліметтер береді, біздің массивте 10- ақ элемент бар, 0-ден 9-ға дейін, онда 10 санынан артық санды сақтай алмаймыз, ол үшін оны жасағанда массивтің размерін үлкейту керек.

Массивтің барлық элементтерін сұрыптау және оларды экранға шығару үшін, оларды тональды динамикке шығара аламыз, мысалы, егер дыбыстау деңгейін кезекпен жазсақ, онда өз мелодиямызды жасай аламыз. Немесе светодиодпен нақты жүйеде жыпылықтай аламыз.

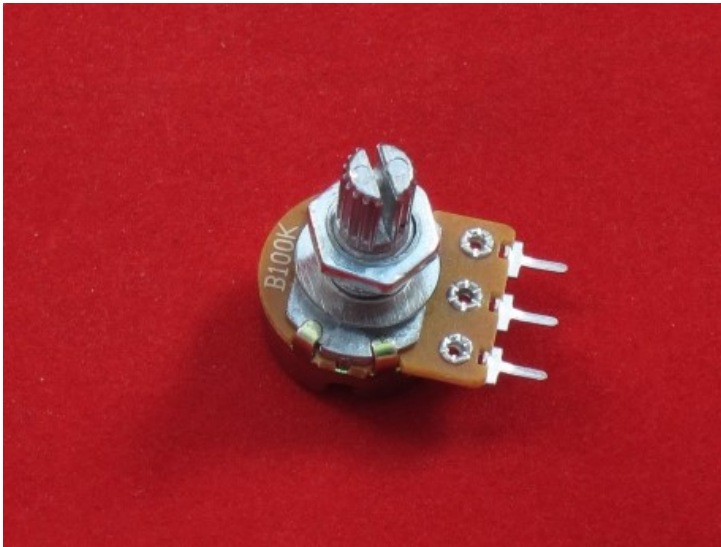
Бекіту үшін тасырмаларды орындандар:

1. *Қандай да мелодияны динамик өндіру үшін мысалы сигнал SOS <https://ru.wikipedia.org/wiki/SOS> ол 3 қысқа сигнал, 3 ұзақ сигнал және тағы 3 қысқа сигнал естіледі. Ол үшін массивті қолданыңдар, онда динамиктің 0-ден 255 дейінгі тональдығын көрсетіңдер және тағы бір массивте үн шығару ұзақтығын миллисекундта көрсетіңдер, delay() командасы көмегімен үн шығару ұзақтығын тапсырыңдар, үн шығару арасындағы үзіліс бірдей*
2. *Светодиодтың 3 режим жыпылықтауын жасаңдар, мысалы: SOS, нормалы және тез, кнопканы басу арқылы режимді ауыстырыңдар. Ол үшін 3 массивті қолданыңдар, онда светодиодтар үшін үзіліс, тапсырыңдар да, кнопканы басып, массивті өзгертіңдер.*

Мысал үшін төменде код келтірілген, ол “Калинка-малинка” өлеңіне кіріспе, аянышты ұқсастықты көз алдыңа елестетеді.

```
int box[11] = {150,125, 75,100,125, 75,100,125, 100,75,50}; //Динамиктің тональдығы
int pause[11] = {200,300, 100,100,200, 100,100,200, 100,100,100}; //Дыбыс ұзақтығы
int pause2[11] = {200,100, 150,150,200, 150,150,200, 150,150,1000}; //Үзілістер
int i=0; //Мұнда біз ағымдағы тональды нөмірді сақтаймыз
void setup() {
}
void loop() {
  analogWrite(3,box[i]); //Массивтен дыбыс шығарамыз
  delay(pause[i]);       //Pause массивінен ұзақ дыбыс шығарамыз
  analogWrite(3,0);
  delay(pause2[i]);      //Pause2 массивінде көрсетілген үзілісті жасаймыз
  i++;                  //i өзгергішін 1-ге арттырамыз
  if(i==11){ //Егер i өзгергіші 11 тең болса, онда өзгергішті нөлдейміз
    i=0;
  }
}
```

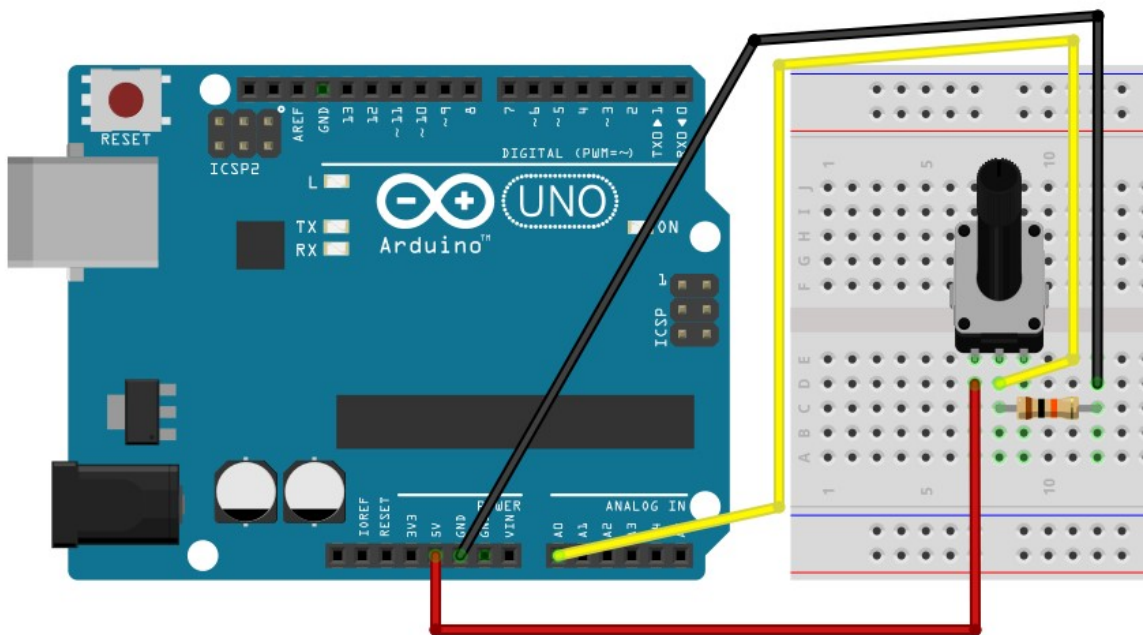
Потенциометр



Енді потенциометрмен танысайық. Потенциометр дегеніміз кедергіні өзгерту үшін бұратын тұтқасы бар резистор, сендер мұндай потенциометрді дауыс қаттылығының регуляторында, динамикаларда кездестірдіңдер, мысалы не тоңазытқыштың температурасын өзгерту үшін пайдаланасындар.

Мұндай потенциометрлер әртүрлі кернеуде болады, мысалы, суретте B100K потенциометрі бейнеленген, бұл оның кедергісін 0-ден 100кОм-ға дейін тұтқаны айналдырып, өзгерте аламыз. Потенциометрде 3 аяқ бар, әдетте екеуі ғана қолданылады: ортадағы және шеткі. Үшінші

не үшін керек? Егер потенциометрде максимум бұрсақ, онда 1 және 2 аяқтың арасында максималды кедергі болады, ал егер минимум бұрсақ минималды болады. 2 мен 3 аяқтың арасында керісінше егер потенциометрді максималды бұрса, онда 2 мен 3 аяқтың арасында минималды кедергі, егер минималды болса, онда максималды кедергі болады. Arduinoны шеткі аяқтардың біріне қосқанда, әдетте плюс беріледі, ал ортадан көрсеткіштер салыстырылып оқылады, яғни ортаны A0 ұқсас портына қосамыз, мысалы, біздің потенциометрді қосайық:

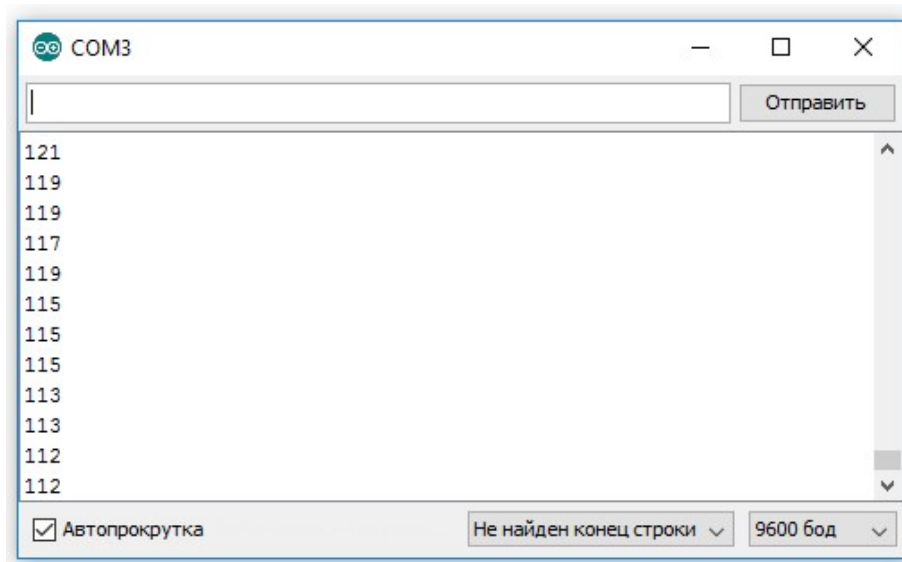


Ұқсас A0 портынан салыстырып оқитын және көрсеткіштерді шығаратын кодты жазайық:

```
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  Serial.println(analogRead(A0));  
}
```

Енді жүйелі порттың мониторын ашайық та, потенциометрдің тұтқасын бұрсақ, экранға

шығатын мәндердің қалай өзгеретінін көреміз:



Енді бірнеше тапсырмаларды орындап көрейік:

1. *Потенциометрдің көмегімен светодиодтың жарықтығына программалық басқару жасаңдар. Ол үшін `analogRead()` пен `analogWrite()` функцияларын қолданыңдар, онда светодиодтың орнына тоналды динамикті қосыңдар да, онымен соны жасаңдар.*
2. *Өткен сабақта нақты температураға (мысалы, 40 градус Цельсия) жеткенде динамик шырылдай бастайтынын жасадық. Енді соны тағы да жасаңдар, потенциометр көмегімен температура шегін 0 ден 60 градус Цельсия диапазонында өзгертуге болатындай жасаңдар, ол үшін `map()` функциясын қолданыңдар.*
- 3.

For() циклы

Программалауда циклдар сияқты түсінік бар – бұл программаның қайталанатын блоктары. Ардуинода `loop()` циклымен таныссыңдар, ол шексіз орындалады. Бірақ біз өз циклымызды жасай аламыз. Ол үшін **for()** командасы бар, ол толық келесі түрде жазылады:

```
for (int i=0; i < 255; i++){  
  //Біз қайталағымыз келетін біздің код  
}
```

Енді ретімен **for()** талдайық, фигуралы жақшаның ішінде бәрі түсінікті онда әрқашан біздің код жатады, бұл жағдайда ол қайталанады, дөңгелектердің ішінде сәл қызығырақ, алдымен жақша ішінде **int i** өзгергішін хабарлаймыз да, онда 0 санын сақтаймыз. Одан кейін нүкте үтір қойылып, **i < 255** жағдайы жазылады. Не үшін циклда жағдай керек, бұл **if()** емес қой?

Жағдай мұнда цикл қайталану үшін керек, егер бұл жағдай дұрыс болса, онда цикл жағдай теріс болғанша қайталана береді, әр орындаудан кейін цикл бұл жағдайды тексереді, егер ол дұрыс болса, онда цикл қайтадан барлық кодты қайталайды, бірақ өзгергіш **i** бізде нөлге тең, бұл жағдай **i < 255** әрқашанда дұрыс болады да цикл мәңгі орындалады. Ол үшін жағдайдан кейін көбейтуге команда **i++** беріледі, соның арқасында біздің өзгергіш көбейеді, бұл команда әр циклдан кейін жұмыс жасап, фигуралы жақшаларды біздің кодты орындағаннан кейін, цикл қайтадан бастайды және біздің өзгергішті **i** тағы бірге көбейтеді, аяғында циклдың 255 жұмыс жасау арқылы 255 көбейеді де, жағдай істеуді тоқтатады. Осы **for()-дың loop()** циклынан негізгі айырмашылығы, мұнда біз циклды орындаудың бізге керекті санын тапсыра аламыз, **i** өзгергішін қалай атасақ та, жағдайда да қандай таңба **>**, **<**, **==**, **!=**, **<=** не **>=** қойсақ та, біз өзгергіш **i**, көбейтуге, азайтуға, оған басқа өзгергіш беруге және т.с. жасауға болады. Мысалы, келесі цикл 255 рет орындалады, тек бұл кезде **i** 255-ке тең болады және оны азайтамыз:

```
for (int i=255; i > 0; i--){  
  //Мұнда қайталағымыз келетін біздің код  
}
```

Біз циклды **loop()** блогында да, сондай **setup()** блогында да қолдана аламыз. Светодиодтың бір қалыпты жыпылықтау сол кодын жазып көрейік, бірақ оны **setup()** блогында жасаймыз.

```
void setup() {  
  for (int i=0; i < 255; i++){  
    analogWrite(3, i);  
    delay(10);  
  }  
}  
void loop() {  
}
```

Өте тез де қолайлы екеніне келісіңдер, ешқандай жағдай мен өзгергіштер керек емес, қолданылатын жалғыз өзгергіш-бұл өзгергіш **i**.

Енді келесі тапсырмаларды орындайық:

1. Светодиод тағы да бір қалыпты сөнуге кодты солай жазыңдар, бірақ онда **if()** жағдайын қолданбаңдар. Светодиод бір қалыпты жанып және сөну үшін не істеу керек? Светодиод үнемі **setup()** жанып және сөну үшін не істеу керек?
2. Санды дәрежеге шығарып және жауабын қайтаратын программа жасаңдар, оны циклдың көмегімен жасау қажет, мысалы 2 саны 3 дәрежеде бұл $2 * 2 * 2$ мен тең. Осы ережені қолданып, циклдың көмегімен 2 саны 9 дәрежеде қанша болатынын есептеңдер.

№8 сабақ

Тақырыбы: Кітапханаларды қосу, ультрадыбысты дальномермен таныстыру

Бұл сабақта ара қашықты өлшеу ультрадыбысты модульмен (дальномермен), Arduinoға кітапханалар кодын қосудымен танысамыз.

Дальномерді HC-SR04 модулі мысалында зерделейміз. Ара қашықтықты ультрадыбыс сигналын жібереді (40кГц жиілігі) одан кейін оның жаңғырығын тыңдайды, сигнал жіберіледі, қандай да бір затқа соғылады да, кері қайтады, сигналды жіберу мен жаңғырықты алу арасындағы уақытты белгілеу арқылы объектіге дейінгі ара қашықтықты білуге болады. Осындай принциппен дельфин объектілеріне дейін ара қашықты өлшейді.

HC-SR04 модулі 4-тен 200 см дейін диапазонда ара қашықтықты өлшейді. Мұнда өлшеу -+1см -+3 см дейін, қашықтыққа байланысты түрленеді. Бұл модульдің кемшілігі, ол дыбысты жұтатын не жіберетін объектіге дейін, жалпы дыбыс жібермейтін заттарға дейінгі ара қашықты дұрыс анықтамайды. Мысалы, қалың жүн немесе марля матасының жұқа қабаты.

Бірақ бұл модуль қолдануда өте қарапайым, модульда барлығы 4 ажырау:

VCC	Модульдің қорлану порты, модуль 5 вольттан қоректенеді.
TRIG	Бұл портқа +5V сигнал беру керек, өйткені ультрадыбысты сигнал жіберу үшін, беруді 10 микросекунд (0,01 миллисекунд) арасында жүзеге асыру қажет.
ECHO	ЖАҢҒЫРЫҚ дальнометр жаңғырықты алғанда, оған импульс беріледі, жаңғырық ұзақ уақыт болса, импульс ұзағырақ ара қашықта алысырақ, мұны біле отырып, объектіге дейінгі ара қашықты есептеуге болады.
GND	Жер

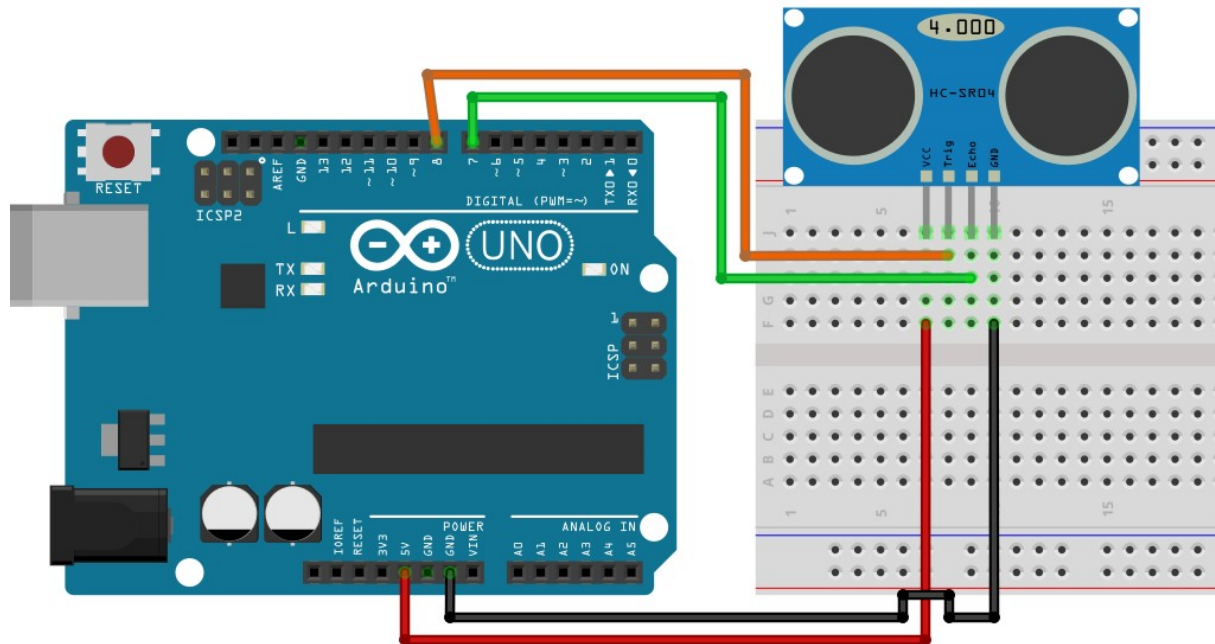
Сонымен, объектіге дейінгі ара қашықты өлшеу үшін, алдымен импульсті жіберу қажет, сосын ғана сенсордан алынған импульс ұзындығын есептеу керек. Импульстың ұзындығын білу үшін, біз **pulseIn()** функциясын қолданамыз, бұл функцияда 2 параметр: порт нөмірі мен HIGH не LOW сигнал деңгейі, одан өлшеуді санау басталады, мысалы егер **pulseIn(9, HIGH)** деп жазсақ, онда **pulseIn()** функциясы 9 портқа HIGH (5 вольт) сигналы берілгенше басады, одан кейін портқа төмен сигнал LOW (0 вольт) берілгенше уақытты өлшеп, функция оны микросекундта қайтарады. Сонымен trig портына 5 вольт береміз, 10 микросекунд тосамыз да, echo портынан импульс ұзындығын есептеуді бастаймыз. Бірақ 10 микросекундты қалай тосуға болады? Delay() функциясында минималды мән 1 миллисекунд, ал бұл 10 микросекундтан 100 есе көп.

Мұнда бізге басқа **delayMicroseconds()** функциясы көмектеседі, бұл функция delay() функциясы сияқты программаны орындауда пауза жасайды, бірақ delay()-дан айырмашылығы 1 ден 1000-ға дейін уақытты микросекундта қабылдайды. Бір миллисекундта - 1000 микросекунд. Бұл **delayMicroseconds(1000)** delay(1) не істесе, соны істейтінін білдіреді.

Сонымен дыбысты жіберіп **pulseIn()** функциясымен импульстің ұзындығын есептедік, енді импульстің ұзындығын біле отырып, затқа дейінгі ара қашықты білу керек. Ол үшін импульс уақытын 58 бөлу (бұл санға датчиктердің өндірушісін бөлуге ұсынады) жеткілікті, онда біз объектіге дейінгі ара қашықты сантиметрде аламыз. 58 саны осы модуль үшін есептелген қарапайым константа. Ұзындықты дюймада есептеу үшін 148 санына бөлу қажет.

Ары қарай цикл қайталанады және дыбыс жіберуге болады, және ара қашықты қайтадан өлшеуге болады. Ультрадыбысты жіберу арасында паузаны 50 миллисекундтан кем жасамау өте дұрыс, әйтпесе өлшеу дәл болмауы мүмкін.

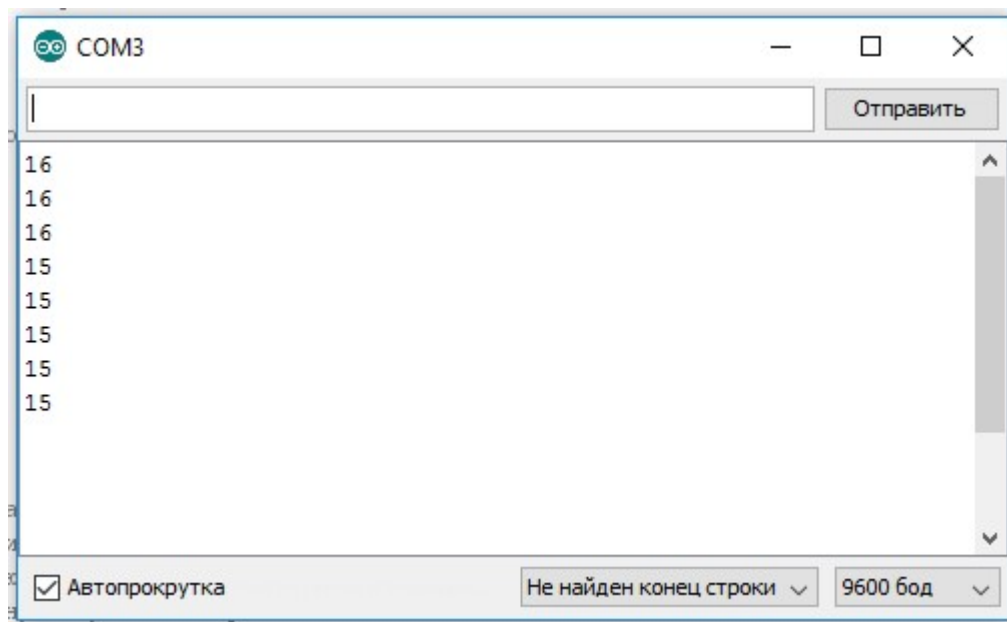
Енді соңында біздің модульді қосып, оның жұмысын тестілейік:



Енді келесі программаны жазып, оны жүктейік:

```
const int trig = 8; // trig қосылған порттың нөмірімен константа
const int echo = 7; // echo қосылған порттың нөмірімен константа
void setup()
{
  pinMode(trig, OUTPUT); //trig портына біз кернеу береміз
  pinMode(echo, INPUT); //INPUT портынан салыстырып оқимыз
  Serial.begin(9600);
}
unsigned int timeSignal = 0;
unsigned int distanceSm = 0;
void loop()
{
  digitalWrite(trig, HIGH);           // TRIG ажырауына сигнал береміз
  delayMicroseconds(10);              // 10 микросекунд ұстап тұру
  digitalWrite(trig, LOW);            // Сигналды үзіп тастаймыз
  timeSignal = pulseIn(echo, HIGH);    // ЕCHO портынан импульстың ұзындығын өлшейміз
  distanceSm = timeSignal / 58;        // Сантиметрді санап шығамыз
  Serial.println(distanceSm);          // Портқа шығарамыз
  delay(100);
}
```

Енді біздің программаны жіберіп, жүйелі порт мониторын ашамыз. Мұнда біз жақын объектіге дейінгі ара қашықта сантиметрмен көреміз, дальномерді әртүрлі заттарға бағыттап, оларға дейінгі ара қашықты өлшеп көріңдер.



Осылай аса күрделі емес тәсілмен объектілерге дейінгі ара қашықты анықтауға болады.

Тағы да аса қарапайым тәсіл бар, өзің барлық кодты жазбай, код кітапханасын қолдануға болады. Код кітапханасы бұл әртүрлі кодтар жиынтығы, оны нөлден бастап өзің жазбай, осы кітапхананы жай қосып, қолдану, онда өздері барлық есептеуді жасайтын дайын функциялар бар, сендерге тек сәл өз параметріңді өзгертіп, кодты жүктеп, құрылғының жұмысына рақаттану.

Біздің ультрадыбыс дальномерінде да кітапхана бар. Оны келесі сілтемеден көшіріп алуға болады https://radiomart.org/index.php?controller=attachment&id_attachment=572

Немесе осы модульді сипаттау бетінде https://RadioMart.org/_975.

Енді біздің кітапхананы Arduinoға орнату керек, ол үшін көшірілген архивтің бумасын шешеміз, онда Ultrasonic папкасы бар

Загрузки > ultrasonic-HC-SR04			Поиск: ultrasonic-HC-SR04
Имя	Дата изменения	Тип	
Ultrasonic	15.02.2012 14:22	Папка с файлами	

Бұл папка бізге керек. Бұл папканың ішінде біздің кітапхананың файлдары сақталған.

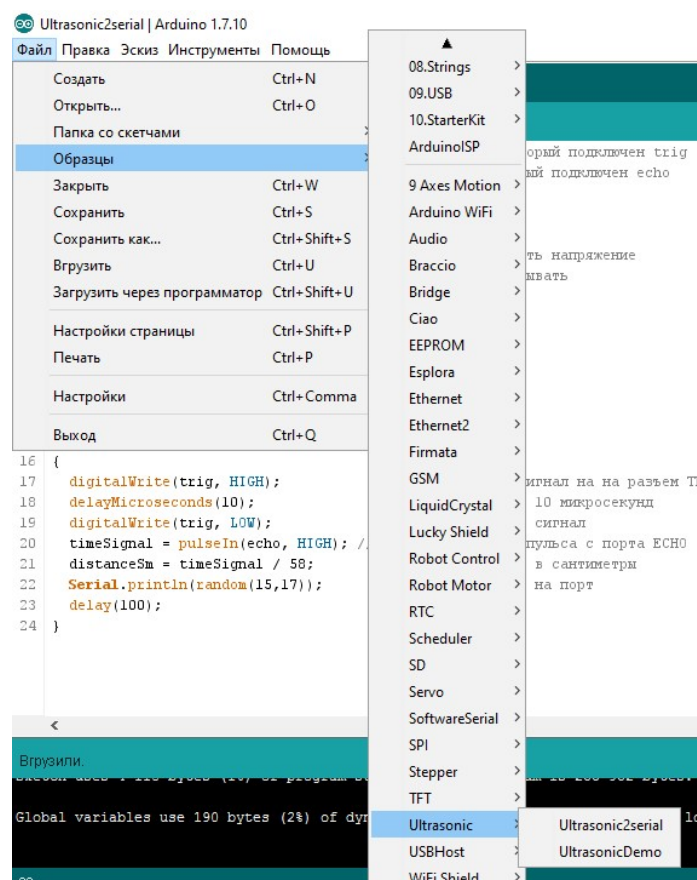
ultrasonic-HC-SR04 > Ultrasonic			Поиск: Ultrasonic
Имя	Дата изменения	Тип	
examples	15.02.2012 14:26	Папка с файлами	
keywords.txt	24.04.2010 20:55	Текстовый докум...	
Ultrasonic.cpp	15.02.2012 14:33	Файл "C++"	
Ultrasonic.h	15.02.2012 14:25	Файл "H"	

Кітапхананы Arduinoға орналастыру үшін Ultrasonic папкасын Arduino кітапханалар папкасына көшіріп алу керек. Ол Arduino IDE орнатылған папкада.

Бұл папка **C:\Program Files (x86)\Arduino\libraries**, сендерде бұл басқа папка болуы мүмкін, Arduino IDE сендер қайда сәйкес орнаттыңдар. Бұл папкада көптеген кітапханалар бар. Сонда біздікін де көшірейік:

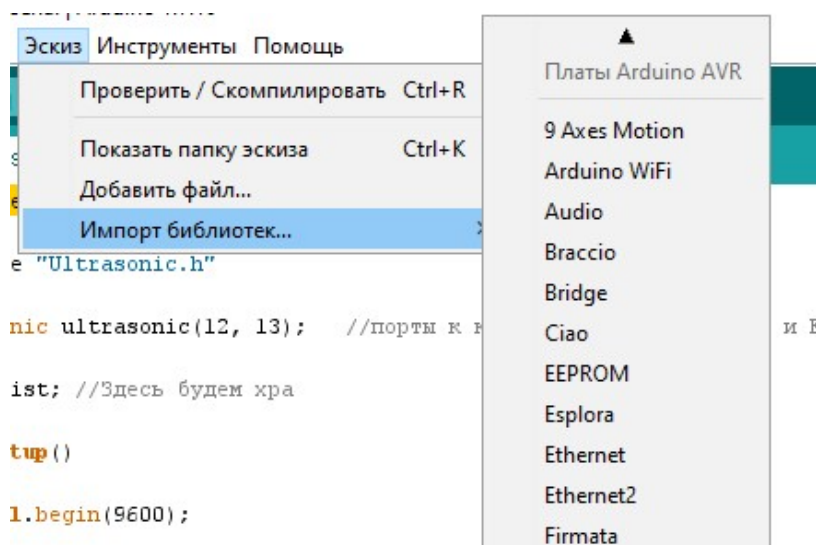
6) > Arduino > libraries		Поиск
Имя	Дата изменения	
NAxesMotion	29.05.2016 21:27	
Robot_Control	29.05.2016 21:27	
Robot_Motor	29.05.2016 21:27	
RobotIRremote	29.05.2016 21:27	
RTC	29.05.2016 21:27	
Scheduler	29.05.2016 21:27	
SD	29.05.2016 21:27	
Servo	29.05.2016 21:27	
SpacebrewYun	29.05.2016 21:27	
Stepper	29.05.2016 21:27	
Temboo	29.05.2016 21:27	
TFT	29.05.2016 21:27	
Ultrasonic	22.06.2016 5:49	
USBHost	29.05.2016 21:27	
WiFiShield	29.05.2016 21:28	

Егер программа бізде ашық болса, онда Arduino IDE-ні қайтадан жіберу. Бізде менюде **Файл -> Үлгілер -> Ultrasonic** жаңа кітапхана көрінуі керек.



Құттықтаймыз! Кітапхана орнатылды, енді оны пайдалана аламыз, кітапхананы пайдалану үшін оны программаға қосу керек, ол үшін мынандай команда: **#include "Ultrasonic.h"** немесе **#include <Ultrasonic.h>** қолданылады, олардың арасында айырмашылық жоқ, екі команда да "Ultrasonic" кітапханасын қосады. Әдетте кітапханалар программаның жоғарында қосылады,

#include командасы ағылшын тілінен қосу деп аударылады. Arduino IDE-де кітапханаларды қосуға қолайлы меню бар, қосу командасын қолмен жазбас үшін **Эскиз -> Импорт библиотек...** менюіне жай ауысу керек те сендер қосқыларың келетін кітапхананы қосу автоматты түрде сенің кодында жазылады.



Енді кітапхананы пайдаланып, біздің программаны жазып көрейік:

```
#include "Ultrasonic.h" //Кітапхананы қосамыз
Ultrasonic ultrasonic(8, 9); // TRIG пен ECHO қосылған порттар
float dist; //Бұнда ара қашықтықты сақтаймыз
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  dist = ultrasonic.Ranging(CM); //Ара қашықты сантиметрде аламыз
  Serial.println(dist); //Порт мониторуна шығарамыз

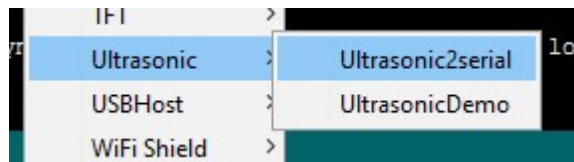
  delay(100);
}
```

Біздің кодты Arduinoға салайық та, жүйелі порттың монитормын ашайық, ол өткендегідей жұмыс істейтінін көреміз, тек бізге ультрадыбысты қолмен бермей, 10 микросекунд тосып, содан кейін импульсті салыстырып оқиды да, оны 58 бөледі. Біздің кітапхана **ultrasonic.Ranging(CM)** функциясының көмегімен бәрін өзі істейді.

Бұл функция барлық есептеуді өзі жасайды және объектіге дейін ара қашықты сантиметрде қайтаратынын аңғарған шығарсыңдар. Егер біз функция параметрінде **CM** емес, **INC** көрсетсек, мысалы: **ultrasonic.Ranging(INC)**, онда бұл функция затқа дейінгі ара қашықты дюймде қайтарады. Сендер таныс емес **Ultrasonic ultrasonic(8, 9);** кодын байқаған боларсыңдар. Бұл код модуль мен кітапхананы жұмысқа инициалайды (дайындайды), 8 бен 9 цифрлары бұл Arduino порттарының нөмірлері, оған TRIG пен ECHO модулінің пиндары қосылған.

Енді біздің программа едәуір аз код алады.

Тағы 1 кішкене құпия айтамыз, бұл барлық кодты қолмен жазбас едік, әдетте кітапханада мысалдар бар, онда дайын кодтар бар, ал сендерге тек бір параметрлерді өзгерту керек. Ондай мысалдарды қалай алуға болады? Әдеттегідей, ол үшін **Файл -> Үлгілер -> Ultrasonic** менюіне кіру керек, онда кодтың 2 үлгісін **Ultrasonic2serial** және **UltrasonicDemo** көресіндер



Мысалы бірінші бойынша шақырыңдр, сендерде дайын программа ашылады, онда кодты түсіндіретін, бұл код не жасайтыны туралы түсінік бар. Комментарий ағылшын тілінде, егер сендер ағылшын тілін білмесендер, онда әрқашан аудармашыны пайдалануға болады, мысалы <https://translate.google.com/>. Код дәл біздікіндей, тек порттардың нөмірлері ерекшеленеді, мұнда олар 12 мен 13. Біздің модульді осы порттарға қоса аламыз, не программада біздің модуль қосылған портты қалай істегіміз келеді, солай өзгертеміз. Аяғында программаны қосамыз, ол жұмыс жасайды, барлығына екі минут кетті, егер модуль ішінен қалай жұмыс істейтінін білмесек те.

```
#include "Ultrasonic.h"
// sensor connected to:
// Trig - 12, Echo - 13
Ultrasonic ultrasonic(12, 13);
void setup()
{
  Serial.begin(9600); // start the serial port
}
void loop()
{
  float dist_cm = ultrasonic.Ranging(CM); // get distance
  Serial.println(dist_cm); // print the distance

  delay(100); // arbitrary wait time.
}
```

Кітапхананың көмегімен кодингті осылай жеңілдетуге болады. Келешекте кітапханаларды өте жиі пайдаланамыз, сондықтан оларды қалай қосу керегін жақсы есте сақтаңдар.

Енді келесі тапсырмаларды орындап көрейік:

1. Өткен сабақтарда терменвокс жасағандарың есте шығар, динамик ақырын не қатты естілу үшін қолымызды жақын не алыс жақындаттық, бірақ оны шам датчигінің көмегімен жасадық. Енді соны дальномермен жасап көрейік, зат алысырақ болса, тоналды динамик жоғары шырылдасын, ал жақын болса, ақырын шырылдасын.
2. Динамик затқа жақындағанда тез де тез шырылдай бастауын, автомобильдің парковка датчиктері сияқты кедергі жақындағанда тезірек шырылдауы сияқты жасаңдар.
3. Шамы қосылған қозғалыс датчигін жасаңдар, ол үшін светодиодты қолданыңдар. Дальномерді жылжымайтын жайға орналастырыңдар, ал кодта бірінші жібергенде жақын объектіге дейін ара қашық есептелінсін, ал егер ара қашық өзгерсе, демек қасынан бірдеңе не біреу өтті, бізге 5 сек, светодиодты жолды жарықтандыру үшін қосу керек

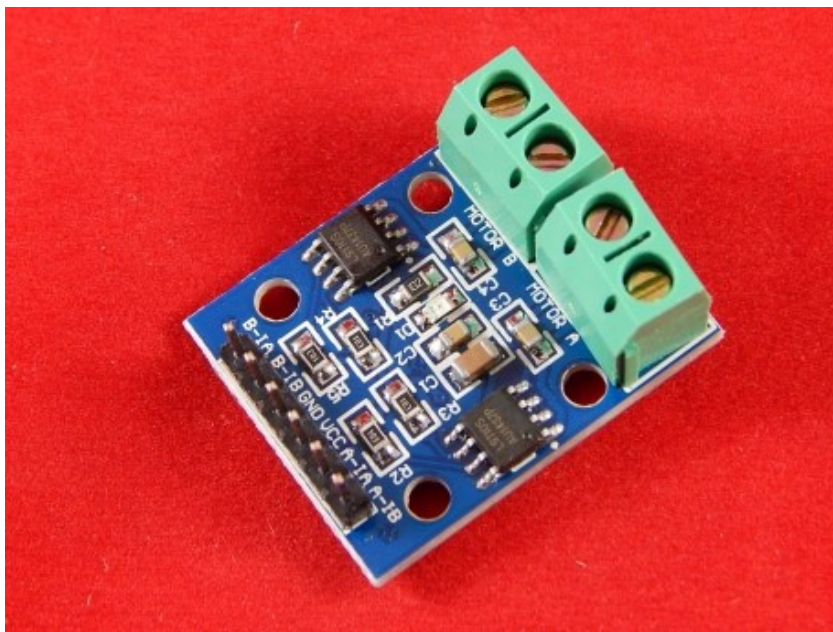
№9 сабақ

Тақырыбы: Arduino арқылы ДС двигателін басқару

Сонымен Arduino көмегімен двигательді басқаруға үйренетін сабаққа да жеттік. Біз 6V тұрақты ток двигательі 1:120 редукторы мысалында үйренеміз.



Бұл двигатель жүктемесіз минутына 100 айналым береді, минимум 60mA жұмсайды, ал егер двигательге жүктеме берсек, онда жұмсау одан көп болады. Arduino порты максимум 40mA ток береді. Бұл бізге двигательдерді толық күшіне жіберуге ток жетпейді, сондықтан бұл двигательді Arduino портына қосуға болмайды. Не істеу керек? Осы мақсат үшін двигательдің драйверлері жасалған, бұл құрылғылар двигательді басқарып, олардың бағыттарын, айналым жылдамдығын реттей алады, онда меншіктегі, жеке қоректенуді қолданып, ал драйвердің басқару, оған команда беру Arduinoның көмегінде. Драйверлер дегеніміз не? Олар әрқашан двигательдің типіне, оның күшіне байланысты және т.с. әртүрлі болып көрінеді. Біз L9110S драйвермен жұмыс істейміз



Бұл қымбат драйвер емес, біздің мақсатымызға сай келеді, ол бір уақытта 2 двигательді басқара алады және 1 двигательге бере алатын максималды ток 800mA құрайды, ал ұстайтын кернеу 5 - 12 вольт, бұл біздің двигательге өте жеткілікті.

Бұл двигатель драйвері айналым бағытымен де, оның жылдамдығын да басқара алады. Төменгі

кестеде көрсетілгендей, драйверді қосу үшін 6 конакталар бар:

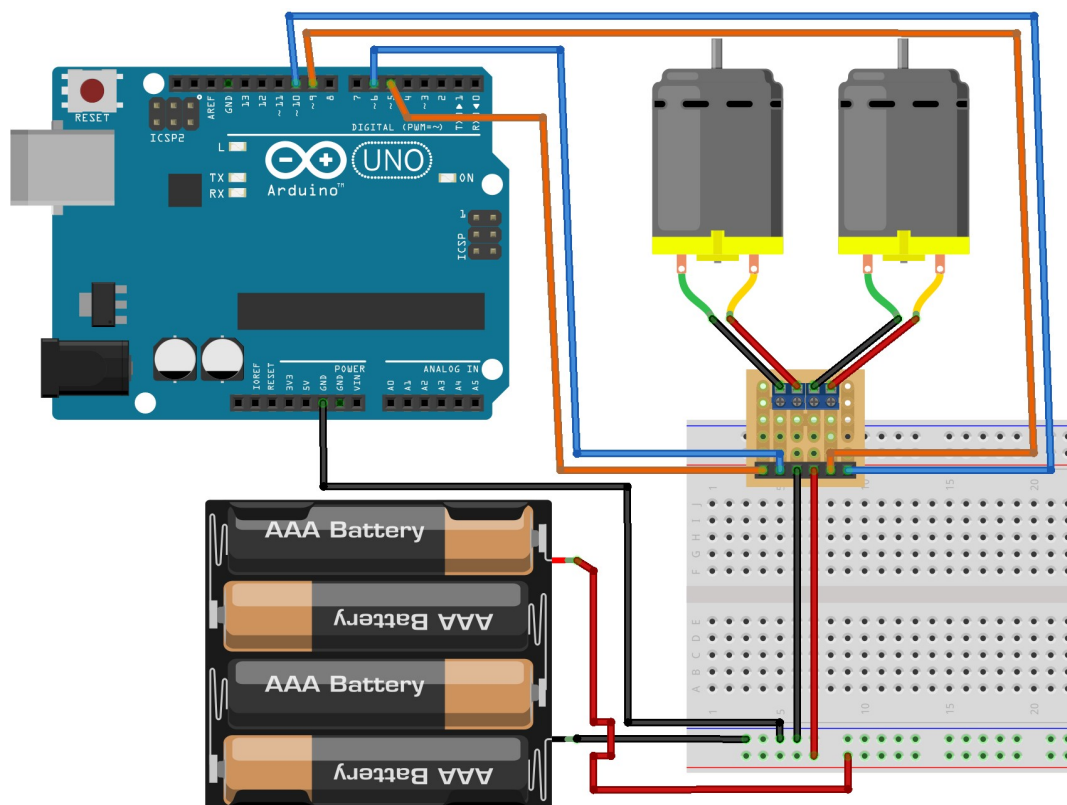
Атауы	Қайда қосамыз
B-IA	Arduиноның цифрлық портына қосылады, осы порттың көмегімен Motor B клеммасына қосылған двигательдің айналым бағытын өзгертуге болады.
B-IB	Arduino цифрлық портына қосылады, осы порттың көмегімен Motor B клеммасына қосылған двигательдің айналым бағыты мен жылдамдығын өзгертуге болады.
GND	Жер қоректенудің минусына (-) қосылады
VCC	Модульдің қоректену порты +2.5-нан +12 вольтқа дейін қоректенуге қосылады
A-IA	Arduino цифрлық портына қосылады, осы порттың көмегімен Motor A клеммасына қосылған двигательдің айналым бағытын өзгерте алады.
A-IB	Arduino цифрлық портына қосылады, осы порттың көмегімен Motor A клеммасына қосылған двигательдің айналым бағыты мен жылдамдығын өзгертуге болады.

Сонымен двигательде 2 клемма бар, двигательдердің өзін қосу үшін клеммаларда Motor A және Motor B деп жазылған. Бұған біз двигательдің плюс пен минусын қосамыз, қайда двигательдің плюсын, қайда минусын қосамыз, айырмашылық жоқ, бірақ екі двигатель де бірдей қосқан дұрыс, мысалы екеуінде де солында плюс, оңынды минус немесе керісінше. Әйтпесе, егер оларды әртүрлі қоссақ, онда кодта оларды басқару қолайсыз болады. Біздің двигатель драйверімен жұмыс істеу де қолайсыз.

Оны басында Arduino мен қоректенуге қосу қажет. Arduinoдан драйверге қоректенуді тікелей қосуға болмайды, өйткені Arduino стабилизаторына біздің двигательді айналдыруға күші жетпейді. Біздің двигательдер 6 вольттан бастап жұмыс істейді, демек біздің драйверге VCC портына двигатель жұмысының максималды жылдамдығы үшін 6 вольттан кем емес беруіміз керек. Ол үшін 1.5 вольтті саусақ батареикасының 4-еуін пайдалануға болады. DURACELL типіндегі қымбат батареикаларды қолданған дұрыс, өйткені қытайдың сілтілі-қышқыл батареикаларынан двигательдерді жіберуге күші жетпейді. Не 2 құйма-ионды 3.7 вольтті батареикаларды пайдалануға болады, онда 2 осындай батареикалар 7.4 вольт береді, бұл бізге керектен сәл артығырақ.

Бірақ біздің жағдайымызда ешқандай қорқыныш жоқ, двигательдер бұндай кернеумен жұмыс істейді. Сонымен драйверден плюске VCC біздің қосымша қоректену көздеріне қостық. Алдымен біздің драйверді Arduinoға қоспастан, GND драйверінің минусын Arduinoға қосу керек. VCC Arduinoға қосу керек емес! Тек GND. GND порты бір уақытта батареикалардың минусына және Arduinoның GND қосылады. Егер 2 құрылым бір-біріне қосылып және оларда әртүрлі қоректену көздері (мысалы Arduino USB арқылы, ол драйвер батареикалардан жұмыс істейді), онда олардың минусы (-) ӘРҚАШАН бірігуі керек, әйтпесе құрылғы дүрілдеуі мүмкін, немесе мүлем саптан шығып қалуы мүмкін.

Сонымен VCC плюске қостық, ал GND минуске және Arduinoны қостық.



Енді бір моторға B_IA мен B_IB сондай A_IA мен A_IB екінші моторға басқару контакталарын қосу қалды. Оларды кез келген цифрлық порттарға қосуға болады, мысалы үшін оларды келесі түрде қосайық.

Драйвер порты	Arduino порты
A_IA	D9
A_IB	D10
B_IA	D5
B_IB	D6

Сонымен бәрі қосылған, біздің двигательдерді қалай басқару керек. Сонымен мұнда схема күрделі емес, алдымен бір двигательді басқаруды қарастырайық. Мысал үшін Motor A қосылған двигательді алайық. Оны айналдыру үшін A_IA портына HIGH жоғары сигналын беру қажет, ал A_IB портына - төмен LOW. Онда двигатель максималды жылдамдықта бір жаққа айнала бастайды. Егер біз керісінше жасап, A_IA LOW төмен сигналын берсек, ал A_IB HIGH жоғары, онда двигатель басқа жаққа айнала бастайды. Айналымды тоқтату үшін A_IA портымен A_IB портына LOW төмен сигналлы жіберу керек. Мұндай режимде двигательдер тоқтап қолады, бірақ бәрібір батарея энергиясын кішкене жұмсайды. Двигательді толығымен өшіруге A_IA порты мен A_IB портына жоғары HIGH сигналын беру керек. Бұл комбинацияны жеңіл есте сақтау үшін төмендегі кестеге қарайық:

A_IA-дағы кернеу	A_IB-дағы кернеу	Алынатын керек
LOW (0 вольт)	LOW (0 вольт)	Motor A двигателі тоқтатылған
HIGH (5 вольт)	HIGH (5 вольт)	Motor A двигателі

		ажыратылған
HIGH (5 вольт)	LOW (0 вольт)	Motor A двигателі бір жаққа айналады
LOW (0 вольт)	HIGH (5 вольт)	Motor A двигателі басқа жаққа айналады

Motor B клеммасына қосылған двигателімен де осылай

В_ІА-дағы кернеу	В_ІВ-дағы кернеу	Алынатын керек
LOW (0 вольт)	LOW (0 вольт)	Motor B двигателі тоқтатылған
HIGH (5 вольт)	HIGH (5 вольт)	Motor B двигателі ажыратылған
HIGH (5 вольт)	LOW (0 вольт)	Motor B двигателі бір жаққа айналады
LOW (0 вольт)	HIGH (5 вольт)	Motor B двигателі басқа жаққа айналады

Сонымен жұмыс принципі бізге түсінікті, енді тым болмаса 1 моторды айналдырып көрейік, ол үшін жоғарыда көрсетілген кестеде А_ІВ әрқашан жоғары кернеу беру керек, егер А_ІА портына төмен LOW сигналын берсек, онда двигатель айналады, егер HIGH жоғары сигналын берсек, онда двигатель тоқтайды:

```
#define A_IA 9 //Константа - A_IA қосылған порт нөмірі
#define A_IB 10 //Константа - A_IB қосылған порт нөмірі

void setup()
{
  pinMode( A_IA, OUTPUT ); //Шығару режиміне цифрлық порттар орнатамыз
  pinMode( A_IB, OUTPUT ); // Шығару режиміне цифрлық порттар орнатамыз

  digitalWrite( A_IA, HIGH ); //Екі портқа да жоғары кернеу береміз
  digitalWrite( A_IB, HIGH ); //Двигательдерді ажырату үшін
}

void loop()
{
  digitalWrite( A_IA, LOW ); //Двигательді бір жаққа айналдырамыз
  delay(3000);

  digitalWrite( A_IA, HIGH ); //Двигательдің айналуын ажыратамыз
  delay(3000);
}
```

Енді двигатель әр 3 секунд сайын айналады, сосын 3 секунд айналмайды. Енді двигатель 3

секунд сайын кері жағына айналатын етіп жасайық. Ол үшін жоғарыда айтылғандай A_IA портына керісінше әрқашан жоғары кернеу беру керек, онда A_IB портына егер төмен кернеу берсек, онда двигатель кері жағына айналады:

```
#define A_IA 9 //Константа - A_IA қосылған порттың нөмірі
#define A_IB 10 //Константа - A_IB қосылған порттың нөмірі

void setup()
{
  pinMode( A_IA, OUTPUT ); //Шығару режиміне цифрлық порттар орнатамыз
  pinMode( A_IB, OUTPUT ); // Шығару режиміне цифрлық порттар орнатамыз

  digitalWrite( A_IA, HIGH ); //Екі портқа да жоғары кернеу береміз
  digitalWrite( A_IB, HIGH ); //Двигательді ажырату үшін
}

void loop()
{
  digitalWrite( A_IB, LOW ); //Двигательді кері жағына айналдырамыз
  delay(3000);

  digitalWrite( A_IB, HIGH ); //Двигательдің айналуын ажыратамыз
  delay(3000);
}
```

Бізде қосымша қоректену көздері – батареялар бар болғандықтан, Arduinoның USB-дан қоректенусіз жұмыс істеуін жасай аламыз. Ол үшін USB-ны Arduinoдан ажыратамыз да, Arduinoның VIN портына батареялардың (+) қосамыз, ал батареяның минусын (-) Arduinoдағы GND минусымен қосамыз, бірақ жерді сабақтың басында біріктірдік, тек бізге плюсті VIN-ге қосу қалды, біздің Arduino компьютерден қоректенбей, автономно жұмыс істейді.

НАЗАР АУДАРЫҢДАР! Arduinoға бір уақытта батарея мен USB-дан қоректенуді қоспандар, бұл Arduinoны да, Сіздің компьютеріңізді де саптан шығарады. Сондықтан тігу алдында, VIN-НЕН ҚОРЕКТЕНУДІ МІНДЕТТІ ТҮРДЕ АЖЫРАТЫҢДАР.

Ал енді келесі тапсырмаларды орындаңдар:

1. Двигатель алдымен бір жағына, сосын келесі жағына кезекпен айналуға, мысалы әр 3 секунд сайын айналатынын жасаңдар, машинканы еденге қойыңдар да, ол қалай істейтінін қараңдар.
2. Екінші двигательді қосыңдар да, екі двигатель алдымен бір жаққа айналып, содан кейін екінші жаққа айналатындай жасаңдар, машинканы еденге қойыңдар да, ол қалай істейтінін қараңдар.
3. Двигательдер қарама-қарсы жақтарына айналысын жасаңдар, машинканы еденге қойыңдар да, ол қалай істейтінін қараңдар.
4. Дальномерді қосыңдар, егер ара қашық жақын объектіге дейін 40 см аз болса, онда машинка ара қашық 40 см көп болғаниа артқа жүрсін, ал егер жақын объектіге дейін ара қашық 60 см көп болса, онда машинка ара қашық 60 см аз болғаниа алға жүретіндей жасаңдар.

Біз осылай двигательдің жылдамдығын басқара аламыз, ол үшін тек A_IA, A_IB, B_IA және A_IB басқарушы порттарын ШИМ портына қосу керек. Онда одан да аз болады, өйткені бізге pinMode(); командасын жасау керек емес. Тағы да бірінші мысалды жасайық, онда Motor A қосылған двигатель 3 секунд айналады, сосын 3 секундқа тоқтайды, енді оны тек analogWrite() функциясының көмегімен жасаймыз:

```

#define A_IA 9 //Константа - A_IA қосылған порттың нөмірі
#define A_IB 10 //Константа - A_IB қосылған порттың нөмірі

void setup()
{
  analogWrite( A_IA, 255 ); //Екі портқа жоғары кернеу береміз
  analogWrite( A_IB, 255 ); //Двигательдерді ажырату үшін
}
void loop()
{
  analogWrite( A_IA, 0 ); //Двигательді бір жаққа айналдырамыз
  delay(3000);
  analogWrite( A_IA, 255 ); //Двигательдің айналуын ажыратамыз
}

```

Бізде ештеңе өзгерген жоқ, двигатель бұрынғыдай біресе айналады, біресе айналмайды. Тек бұл ШИМ сигналының көмегімен жасалады, A_IA портына 0 және A_IB портына 255 берсек, онда двигатель ажыратылады. Кодты өзгертіп көрейік те, A_IA портына 0 емес, мысалы 127 берейік:

```

#define A_IA 9 // Константа - A_IA қосылған порттың нөмірі
#define A_IB 10 //Константа - A_IB қосылған порттың нөмірі
void setup()
{
  analogWrite( A_IA, 255 ); //Екі портқа жоғары кернеу береміз
  analogWrite( A_IB, 255 ); //Двигательдерді ажырату үшін
}
void loop()
{
  analogWrite( A_IA, 127 ); //Двигательді бір жаққа айналдырамыз, бірақ баяу
  delay(3000);
  analogWrite( A_IA, 255 ); //Двигательдің айналуын ажыратамыз
  delay(3000);
}

```

Енді двигатель 2 рет баяу айналады, егер біз санды 200-ге ауыстырсақ, онда ол одан да баяу айналады, сонымен қаншалықты көбірек берсек, соншалықты ол баяу айналады, егер 255 берсек, онда ол тоқтайды, өйткені екі портта да жоғары кернеу болса, онда двигатель ажыратылады.

ШИМ порттың көмегімен двигательді кері жағына айналууды мәжбүр ету үшін, бізге дәл солай 0 A_IA емес, A_IB беру керек, ал A_IA әрқашан 255 беру керек, егер біз сандарды A_IB 0-ден 254-не дейін өзгертсек, онда двигатель жылдам жылдамдықтан баяуға басқа жаққа айналады:

```

#define A_IA 9 // Константа - A_IA қосылған порттың нөмірі
#define A_IB 10 // Константа - A_IB қосылған порттың нөмірі
void setup()
{
  analogWrite( A_IA, 255 ); //Екі портқа жоғары кернеу береміз
  analogWrite( A_IB, 255 ); //Двигательдерді ажырату үшін
}
void loop()
{
  analogWrite( A_IB, 127 ); //Двигательді кері жағына айналдырамыз, бірақ баяу
}

```

```
delay(3000);  
  analogWrite( A_IB, 255 ); // Двигательдің айналуын ажыратамыз  
delay(3000);  
}
```

Енді келесі тапсырмаларды орындандар:

1. *Екі двигательдің analogWrite() көмегімен айналуын жасаңдар*
2. *Екі двигатель максимум 2 рет баяу айналуын жасаңдар*
3. *1 двигательді максимумде, басқасының жартылай айналуын жасаңдар, машинканы еденге қойып, не болатынын қараңдар*
4. *Екі двигатель бір қалыпты жылдамдық алып, содан кейін бір қалыпты жылдамдықты төмендетуін жасаңдар.*

№10 сабақ

Тақырыбы: Өз функцияларын жасау

Бүгін біз өз функцияларымызды жасауды зертейміз, ол сендер жазған бір де сол код кесектерін қолайлы пайдалану үшін қажет. Біз сендермен Arduinoның біраз стандартты функцияларымен таныстық, олар: `delay()`, `delayMicroseconds()`, `digitalWrite()`, `digitalRead()`, `analogWrite()`, `analogRead()` және басқалар.

Өз функциянды жазуға үйрену кезеңі келді. Функцияда өзгергіштер сияқты мәліметтер типі бар, мысалы, `int`, `float`, `bool` және басқалар. Егер өзгергіште мәліметтер типін өзгергіште қандай мәліметтер сақталатынын түсіндіру үшін көрсетеміз, ал функцияда мәліметтер типін тапсырамыз, функция қандай мәндерді қайтаратынын көрсету үшін функцияларды жасау өте қарапайым, олар `setup()` { } пен `loop()` { } шегінен тыс жасалады және оларға өте ұқсас, өз функциямызды жасау мысалын қарастырайық.

```
int myFunction(){
  return 123;
}
void setup(){
}
void loop(){
}
```

Біз өз функциямызды **myFunction** атауымен жасадық, бұл функцияның ішінде тек бір команда **return 123;** `return` – ағылшын тілінен аударғанда «қайталау» дегенді білдіреді. Аңғарғандай, бұл команда әрқашан функциядан 123 санын қайтарады. Сондықтан біздің функцияның типін **int** деп көрсетейік, өйткені сан бізге бүтін сан болып қайтарылады. Біз функцияны **myFunction** деп атадық, бұл еркін атау, сендер өз атауларыңызды қоюға болады, өзгергіш атауындағыдай ереже, тек латын әріптері, цифрлар мен сызу таңбалары функцияның атауы цифрдан басталуы мүмкін емес және басқа функциялардың, командалардың, константалар мен өзгергіштердің бар атауларымен тура келмейді.

Біз өз функциямызды жаздық, бірақ ол өз-өзімен ештеңе жасамайды, егер жай жазылған болса, енді оны пайдалану қажет, программалауда функцияны қолдану функцияны шақыру деп аталады. Енді бізге кез келген жерде **setup()** немесе **loop()** өзінің жалғыз да пайдасыз әрекеті – 123 санын қайтару үшін біздің функцияны шақыру жеткілікті. Осыны жасайық:

```
int myFunction(){
  return 123;
}
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.println(myFunction());
}
```

Енді бізде жүйелі порт мониторуна біздің 123 саны шығады. Біз функцияның қайтарған санын өзгергіште сақтай аламыз, жалпы алғанда жай сандар сияқты әрекет жасаймыз, өйткені оның мәліметтер типі **int**. Мысалы, мына кодты өзгергіште xxx 123 санын сақтау керек:

```
int xxx;
xxx = myFunction();
```

Келесі код функция қайтарған санды 2-ге көбейтеді де, өзгергіште 246 санын сақтайды.

```
int xxx;  
xxx = myFunction()*2;
```

Яғни программа біздің функциямен дағдылы сан немесе **int** типтегі өзгергіш сияқты әрекет жасайды, бірақ өзгергіштен айырмашылығы – функция **return** командасының көмегімен мәнді қайтарады.

Егер өзгергішті пайдаланатын болсақ, неге жеке функция жазуымыз керек?

Әрине, егер біздің функцияда тек **return 123** командасымен жол болса, ондай функциядан мән аз, одан өзгергішті қолданған қарапайым. Бірақ әдетте функцияларда өте күрделі код. Мысалы:

```
int myFunction(){  
  int box=2;  
  for(int i=0; i<10; i++){  
    box = box * 2;  
  }  
  return box;  
}
```

Бұл кодта біздің функция бокс өзгергішін 10 рет циклында 2-ге көбейтеді де алынған мәнді қайтарады, яғни біз 2-ні 10 дәрежеге көтердік. Егер біздің функцияны шақырсақ, онда ол 1024 санын қайтарады (яғни ол 2 санын 2 бүтінге 10 рет көбейтті, бұл 2 саны 10 дәрежеге тең):

```
int myFunction(){  
  int box=2;  
  for(int i=0; i<10; i++){  
    box = box * 2;  
  }  
  return box;  
}  
void setup(){  
  Serial.begin(9600);  
}  
void loop(){  
  Serial.println(myFunction());  
}
```

Егер бізге бір кодты бірнеше рет қолдансақ, онда оны функцияға жай салуымыз керек. Сонымен қай жерде біздің код қажет – біздің функцияны шақыру керек, ол өз кезегінде біздің кодты орындап, нәтижені қайтарады.

Мысалы:

```
int myFunction(){  
  int box=2;  
  for(int i=0; i<10; i++){  
    box = box * 2;  
  }  
  return box;  
}  
int aaa;
```

```

void setup(){
  aaa = myFunction(); //Біздің 1024 мәнін aaa өзгергішінде сақтаймыз
  Serial.begin(9600);
}
void loop(){
  //Бұл код жүйелі порт мониторуна сандардың қосындысын береді
  // 1024+1024+1024, яғни экранға 3072 саны шығады.
  Serial.println(myFunction() + myFunction() + myFunction());
}

```

Бұл кодта біздің функцияны бірнеше рет қолданамыз.

Сендер байқағандай, біздің функция ішінде box өзгергішін жасадық, функция ішінде кез келген өзгергішті жасай аламыз. Әрі бұл өзгергіштерді программа функцияның сыртынан көрмейді. Мысалы, келесі код қатені көрсетеді, өйткені box өзгергішін біздің myFunction() функциясында жасаймыз, loop() функциясында шығарамыз, сондықтан Arduino IDE box өзгергіші жасалмаған қателігін көрсетеді:

```

int myFunction(){
  int box=2;
}
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.println(box);
}

```

Ошибки компиляции.

Ultrasonic2serial.ino:10:18: error: 'box' was not declared in this scope

Ошибки компиляции.

Егер **loop()** ішінде немесе **setup()** ішінде өзгергішті дәл осылай жасасақ, ал оны біздің функция ішінде қолдануға талаптансақ, бұл да қатені көрсетеді:

```

int myFunction(){
  box=2;
}
void setup(){
  Serial.begin(9600);
}

void loop(){
  int box=2;
  Serial.println(box);
}

```

Ошибка компиляции.

Ultrasonic2serial.ino:2:3: error: 'box' was not declared in this scope

Ошибка компиляции.

Егер біз box өзгергішін функцияның сыртынан жасасақ, онда біздің функцияда **loop()** { } та да, **setup()** { } те де пайдалана аламыз. Мысалы, былай:

```
int box;
int myFunction(){
  box=2;
}
void setup(){
  Serial.begin(9600);
}

void loop(){
  box=5;
  Serial.println(box);
}
```

Енді біздің мысалға оралайық, онда екіні 10 дәрежеге көтердік. Сонымен сендер біздің функцияны `int myFunction(){ }` программаның ең жоғары жағында жасағанымызды байқаған шығарсындар, бұл міндетті емес, оны ортада, ең төменгі жағында жасауға болады, ең бастысы мұны басқа функциялардың сыртында істеу керек, басқа функцияның ішінде функция жасауға болмайды. Шақыруға болады, жасауға болмайды!

Мысалы, келесі үш мысалдың барлығында программа мониторға 123 санын шығарады, программаға функция қайда жасалғаны маңызды емес, программаға оның қайда шақырылатыны мәнді, ол қай жерде болса да жасалуы мүмкін, бірақ ол қайда шақырылды, тек сол жерде істейді:

```
int myFunction(){
  return 123;
}
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.println(myFunction());
}
```

```
void setup(){
  Serial.begin(9600);
}
int myFunction(){
  return 123;
}
void loop(){
  Serial.println(myFunction());
}
```

```
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.println(myFunction());
}
int myFunction(){
  return 123;
}
```

Үш мысалдың барлығы тіпті бірдей, біздің функция әртүрлі орындарда жасалғанына қарамастан, ол әрқашан бір - **loop()** те шақырылады. Функцияда біз параметрлерді жібере аламыз, онымен түйістік, қашан **delay(1000)** шақырғанда, оған параметрді жібереміз – миллисекунд саны – 1000.

DigitalWrite(3, HIGH) функциясын шақырғанда, біз оған 2 параметрді жібереміз, 3 – нөмірі және HIGH – портқа берілетін сигналдың деңгейі.

Біздің функцияға параметрлерді жіберуді жасауға болады, екіні 10 – дәрежеге көтеру мысалымызды қайтадан қарастырайық:

```
int myFunction(){
  int box=2;
  for(int i=0; i<10; i++){
    box = box * 2;
  }
  return box;
}
void setup(){
  Serial.begin(9600);
}

void loop(){
  Serial.println(myFunction());
}
```

Box-та екі емес, басқа мән, мысалы, 5 сақталуын болжайық. Box мәнін өзгерткіміз келгенде, әр рет біздің функцияны өзгертпес үшін, оған біз параметрді жібере аламыз, бұл қалай жасалады? Иә, өте жай, дөңгелек жақшада функция жасағанда өзгергіш жасау керек, ол біздің параметр болады, қараңдар:

```
int myFunction(int param1){
  int box=param1;
  for(int i=0; i<10; i++){
    box = box * 2;
  }
  return box;
}
```

енді егер біздің функцияны шақырсақ, жақшада мысалы үшті көрсетсек:

```
myFunction(3);
```

Онда бұл үштік **param1** өзгергішінде автоматты сақталады, ары қарай оны **int box=param1** сақтаймыз, демек box өзгергішінде де 3 саны, ары қарай циклда box біздің өзгергіште сан 2 көбейтіле бастайды, ақырында біздің функция бұрынғыдай 1024 санын емес, **59049** санын қайтарады. Бұл сан **int** мәліметтер типі өзінде сақтай алатыннан көп, сондықтан біздің функцияның мәліметтер типін өзгерту керек, әйтсе де **long int** жасасақ, онда төменде көрсетілген код порт монитормына 59049 санын шығарады:

```
long int myFunction(int param){
  int box=param;
  for(int i=0; i<10; i++){
    box = box * 2;
  }
  return box;
}
void setup(){
  Serial.begin(9600);
}

void loop(){
  Serial.println(myFunction(3)); //Біздің функцияны 3 параметрімен шақырамыз
}
```

Ал егер функцияға 2 параметрді жіберуді қаласақ? Бәрі жай, функция жасағанда жақшада тағы бір өзгергішті көрсетіңіздер, мысалы, 2 санды көбейтіп, жауабын беретін функцияны жасайық:

```
int myFunction(int param1, int param2){
  return param1 * param2; //Бір өзгергішті екіншіге көбейтеміз де, нәтижесін береміз
}
void setup(){
  Serial.begin(9600);
}

void loop(){
  Serial.println(myFunction(3,10)); // 3 және 10 параметрінде біздің функцияны шақырамыз
}
```

Шақырғанда біздің функцияға 2 параметр жібереміз: 3 пен 10, функция оларды көбейтіп, бізге нәтижесін береді, оны порттың монитормына шығарамыз. Сөйтіп қандай да күрделі функцияларды жазуға болады, оны сосын жиі өз кодында ұзын код кесектерін қайталамай қолдануға болады.

Ал егер ештеңе есептемейтін, бізге жауап бермейтін кодты функцияға жай орналастырсақ, мысалы, машинаны алға жүруге мәжбүр ететін кодты орналастырсақ?

Оны істеуге болады, онда бізге функцияның мәліметтер типі керек емес, ол ешқандай мәнді қайтармайды, ол жай двигательді айналдырады, осымен болды. Ол үшін мәліметтер типінің орнына **void** сөзін көрсету керек, енді мысалды қарайық:

```
void goBack(){
  digitalWrite( 9, LOW ); //Двигательді кері жағына айналдырамыз
  delay(3000);
  digitalWrite( 10, HIGH ); //Двигательдің айналуын ажыратамыз
  delay(3000);
}
```

```

void setup(){
  pinMode( 9, OUTPUT ); //Шығу режиміне цифрлық порттарды орнатамыз
  pinMode( 10, OUTPUT ); //Шығу режиміне цифрлық порттарды орнатамыз

  digitalWrite( 9, HIGH ); //Екі портқа жоғары кернеу береміз
  digitalWrite( 10, HIGH ); //Двигательді ажырату үшін
}

void loop(){
  goBack();
}

```

Біз двигательді 3 секунд кейін айналдырып, сосын 3 секунд пауза жасайтын функция жасадық. Енді двигатель осыны үнемі жасау үшін, бізге **loop(){ } goBack()** функциясын шақыру жеткілікті. Осылай роботты тура, кейін, солға, оңға қозғалуына мәжбүр ететін функция жасауға, сосын оны әртүрлі жағдайға шақыруға болады.

Егер сендер байқасаңыздар, біз бірінші сабақтан бастап **setup() { }** және **loop() { }** функцияларын жасадық, олар екеуі де **void** типті, өйткені мәндерді қайтармайды, тек оларды ешқашан шақырмаймыз, ал олар бәрі бір орындалады, оны Arduinoның өзі автоматты түрде шақырады. Бұл жалғыз Arduino өзі жасайтыны, қалғанын біз өзіміз жазамыз. Тек солай бірінші сабақтан бастап функцияларды жаза білдік, бірақ оны білген жоқпыз.

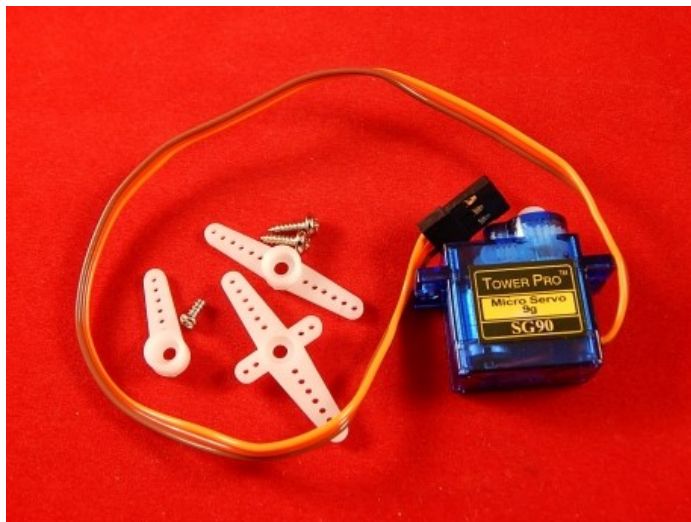
Енді тақырыпты бекітуге тапсырмалар:

1. Роботты алға және артқа қозғалуына мәжбүрлейтін функция жасаңдар, оларды кезекпен **loop()** шақырыңдар.
2. Роботты солға және оңға бұрылуға мәжбүрлейтін функция жасаңдар.
3. Роботты алға және артқа қозғайтын функцияны қайта жасаңдар, себебі олардың қозғалу жылдамдығын 0-ден 255-ке дейін біз бере аламыз.
4. Енді осы функцияларды оларға 0-ден 10-ға дейін жылдамдықты тапсыра алатындай қайта жасаңдар. Оны **tar()** функциясының көмегімен жасаңдар.

№11 сабақ

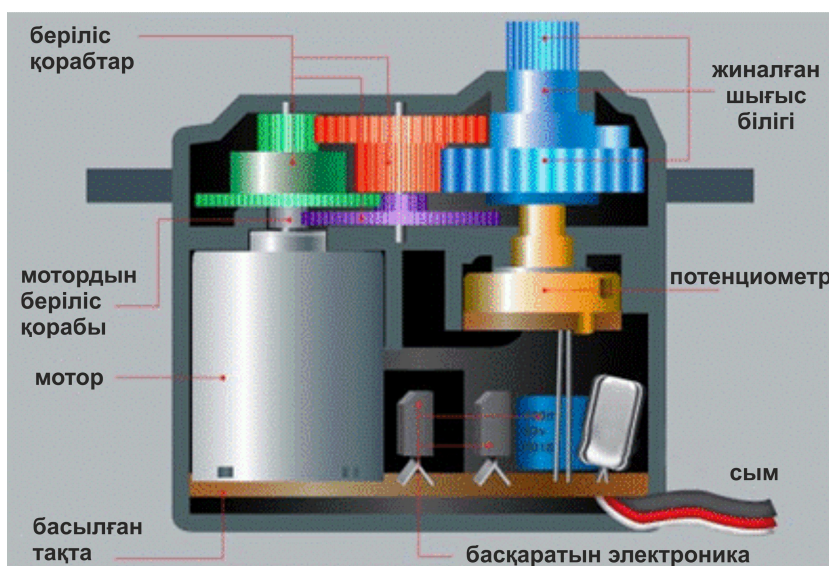
Тақырыбы: Серводвигателмен таныстыру

Бұл сабақта серводвигатель SG90 үлгісінде серводвигательдің жұмысымен танысамыз.



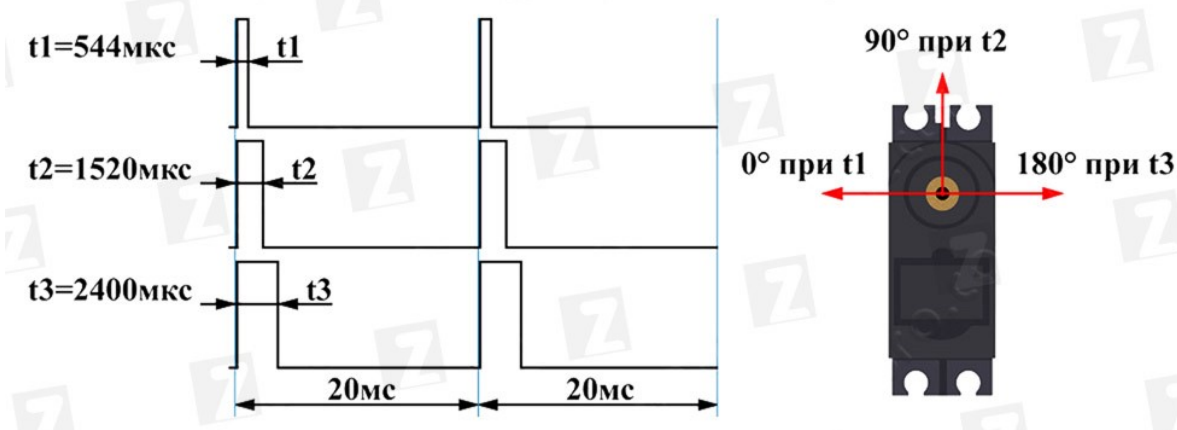
Серводвигатель әдеттегі двигательден айырмашылығы сол, оны әрқашан нақты бұрышқа бұруға болады. Сервоприводтың бұл моделі 180 градусқа бұрылады, әртүрлі серводвигательдерде әртүрлі айналым бұрыштары, сонымен бірге 360 градусқа айналатын серводвигательдер бар. Серводвигательдерді кейде сервомотор, сервомашинка немесе жай серво деп атайды. Атауы оның маңызды емес, ең бастысы сөздің түбірінде серво бар, сөз атап айтқанда сондай двигательдер туралы. Мұндай двигатель іші қалай жасалған және қалай жұмыс істейді, соны қарайық.

Серводвигательдің көрнекті құрылымы төменгі суретте көрсетілген: пластик корпусының ішінде двигатель орналасқан, бұл двигатель тез айналады, оның валында шестерня отыр, ол басқа шестернямен қосылып редуктор жасайды, ол соңында двигательдің жылдамдығын азайтады, двигатель тез айналады, бірақ вал жылдамдығы ақырында баяу, бұл валдың бұрылу күшін көбейту үшін жасалған, вал баяу айналсын, бірақ осының әсерінен оның күші көп. Двигатель басу платасына жапсырылған, сондай оған біздің двигательді басқаратын барлық электроника дәнекерленген. Ең басты элемент, оның арқасында двигатель қандай бұрышқа бұрылғанын түсінетін элемент-потенциометр, бір шестерня потенциометрге қосылғанын көріп отырсындар, двигатель айналғанда потенциометр валмен бірге бұрылады, кедергі өзгереді, потенциометрдегі кедергіге байланысты двигатель нақты бұрышқа бұрылғанын түсінеді де, тоқтайды.

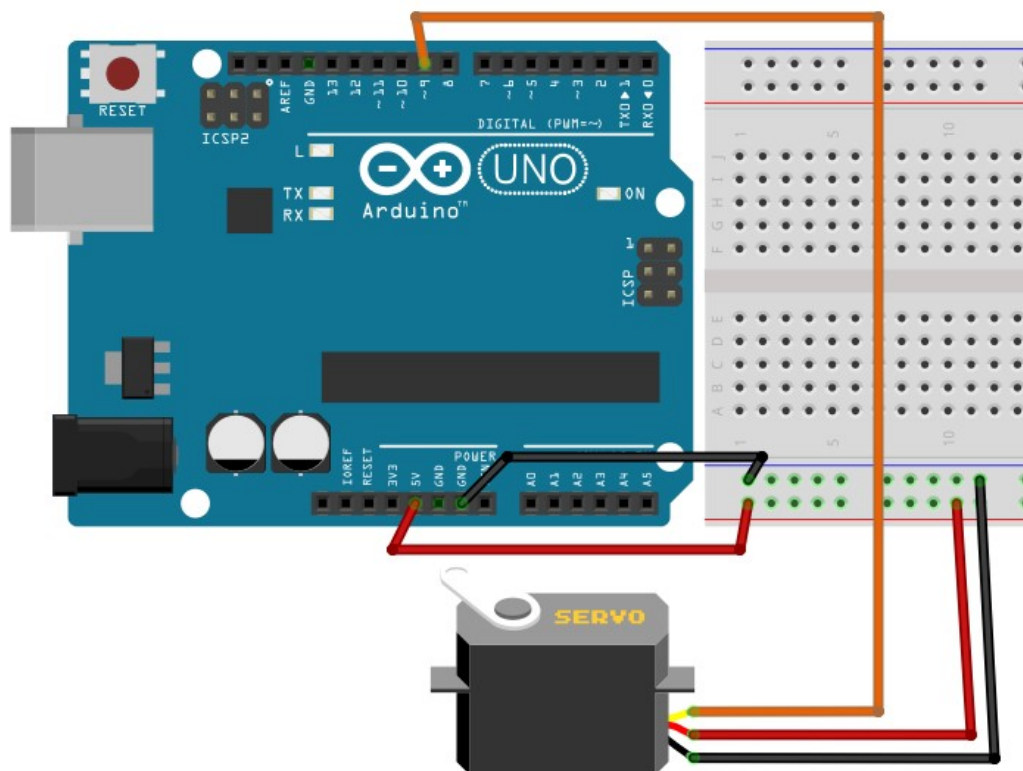


Енді серводвигательді қалай басқару керектігін қарайық. Барлық серводвигательдерде әдетте 3 контакт, бұл плюс, минус және басқару. Плюс пен минус түсінікті, оған двигательдің қоректенуі жіберіледі, оған қандай кернеу керектігіне байланысты біздің серводвигательге Arduino алынатын 5 вольт жеткілікті, ол онша күшті емес. Плюс пен минус дағдылы қызыл мен қара сымға сәйкес келеді. Бірақ плюс пен минус қоректенуді жай беріп, двигательді бұрылуға мәжбүрлемейді. Сол үшін үшінші басқару сымы керек, ол әдетте сары, қызғылт сары немесе ақтүспен белгіленеді.

Серводвигательді бұру үшін порттың шимы пайдаланады. Сервоприводтың бұрылу бұрышы импульстің ұзындығына байланысты, төмендегі www.zelectro.cc: сайтынан алынған суретке қараңдар:

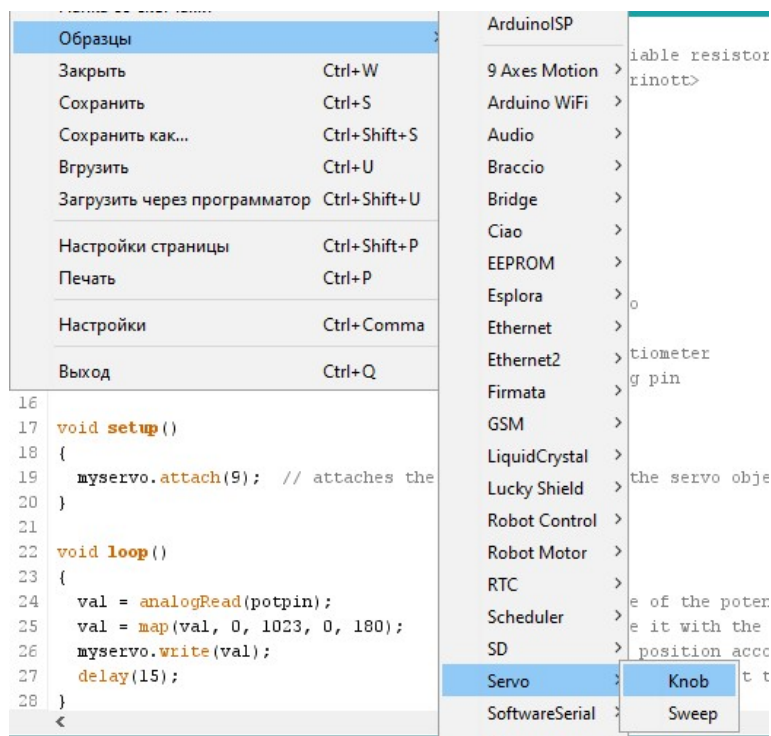


Суретте микросекунд ұзағырақ, импульс жалғасады, двигательдің бұрылу бұрышы көбірек, диапазоны шамамен 500 - 2500 микросекунд. Әртүрлі серводвигательдердің диапазондары ерекшеленеді. Сонымен серводвигательді мынадай қарапайым схемамен біздің Arduinoға қосайық:



Біз сервоприводты қадамдап басқаруды тереңдетпей, оны басқару үшін кітапхананы қосамыз, ол үшін Arduinoға Servo аталатын стандартты кітапхана орналастырылған, бұл кітапханада кодтың жұп мысалдары бар, оларды ашып, талдайық, **Файл -> Образцы -> Servo -> Knob**

менюін ашыңдар.



Бізде робот үшін Servo кітапханасымен код үлгісі ашылады. Сонымен бұл код не істейді, ол біздің сервопроводты A0 ұқсас портынан алынған мәнге байланысты нақты бұрышқа бұрады, ол map функциясының көмегімен 0-ден 1023 дейінгі 0-ден 180-ге дейін (бұрылу градусының бұрышы) айналдырады. Кітапхананың өзінде **myservo.write()** функция бар; бұл функция біздің сервоприводты параметрлері тапсырылған бұрышқа бұрады, мысалы **myservo.write(90)** командасы біздің сервоприводты 90 градусқа бұрады. **Servo myservo** командасы Servo кітапханасынан инициалдайды, **#include <Servo.h>** командасы біздің кітапхананы қосады. Болды. Бұл барлық командалар, бұл сервоприводты басқару үшін тіпті жеткілікті. Ешқандай күрделі емес? Енді практика, келесі тапсырманы орындандар:

1. Потенциометрдің бұрылысынан сервопривод бұрылуын жасаңдар.
2. Дальномерге затты жақындатқанда роботтың басы басқа жаққа теріс айналуын жасаңдар.
3. Енді радар жасап көрейік, біз роботтың басын айналдырамыз да, дальномердің метр радиусында кедергі тапқан жағдайда, динамикпен қысқа сигналдар береміз.
4. Енді алынған кодтан ары қарай пайдалану үшін бір немесе бірнеше функциялар жасаңдар.

№12 сабақ

Тақырыбы: Серводвигателмен жобалар

Бұл сабақта бізде минимум теория, максимум практика, серводвигательдермен жұмысты жалғастырамыз. Бастама үшін бір қызық функция туралы айтқымыз келеді. Ол **random()** деп аталады, бұл функция кездейсоқ сандарды генерациялауға арналған, оның 2 параметрі бар: бұл бастапқы және соңғы, сандар мәнінің ішінен ол кездейсоқ мәнді таңдап алу керек. Мысалы, келесі код, экранға 0-ден 10-ға дейінгі кездейсоқ сандар шығарылады, өйткені параметрде 0-ден 10-ға дейін **random(0, 10)** сандарын көрсеттік.

```
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  Serial.println(random(01,0));  
}
```

Параметрлерде біз **long int** шегінде кез келген сандарды көрсете аламыз. Мысалы, **random(3852, 2000000)**. Біз осылай бір параметрмен **random()** функциясын шақыра аламыз, мысалы **random(777)**, егер осылай жасасақ, онда функция 0-ден 777-ге дейінгі кез келген санды қайтарады, яғни біз бір параметрді көрсетпесек, онда функция автоиатты түрде оның орнына 0 қояды. Бұл бүгінгі барлық теория.

Енді практикаға көшейік:

1. Біздің роботтың басы 0-ден 180 градус шегінде кездейсоқ бұрышқа бұрылатындай етіп жасаңдар, ол үшін **randot()** функциясын қолданыңдар.
2. «Шындық пен өтірік» әзіл программасын жасаңдар, ол үшін кнопканы басқанда программа 0-ден 1-ге дейін генерирлаулап, егер нәтижесінде 0 шықса, робот 3 рет басын изейді (бұл өтірікті білдіреді), егер бір шықса, онда робот динамикпен шырылдасын (бұл шындықты білдіреді). Сендер роботқа сұрақ қойып, кнопканы басыңдар, ал ол сендер шындықты айттыңдар ма, әлде жоқ па. Әрине, ол оны кездейсоқ түрде жасайды, оны сізден басқа ешкім білмейді.
3. Енді робот үнемі жүрсін, ал басы тура қарасын, егер жақында кедергі пайда болса, онда тоқтау керек, басын бір жаққа бұрып, егер онда кедергі болмаса, онда бұрылып, сол жаққа жүру, егер онда кедергі болса, онда басын басқа жаққа бұрып, сол жаққа жүру, егер ол жақта да кедергі болса, онда кейін шегініп, жан-жағына қарап, әрекетін қайталау.

№13 сабақ

Тақырыбы: Бекітуге жобалар

Бүгін бізде тағы да минимум теория максимум практика. Екінші сабақты еске түсірейік, онда жүйелі порттың жұмысын өткенбіз. `Serial.print` пен `Serial.println` командасын сендер жақсы естеріңде сақтадыңдар деп ойлаймын. Ал **`Serial.read()`** пен **`Serial.write()`** ұмытқан шығарсыңдар. Сонымен еске түсіреміз. **`Serial.read()`** функциясы компьютерден жүйелі портқа берілген мәліметтерді салыстырып оқиды. Бірақ **`Serial.read()`** функциясын орындамастан бұрын қандай болмасын мәліметтер беріліп жатыр ма, әлде жоқ па, тексеру керек, ол үшін **`Serial.available()`** функциясы бар, егер бұл функция 0 ден көп сандарды қайтарса, онда мәліметтер беріліп жатыр, егер жоқ болса, онда жоқ. Демек, компьютерден мәліметтерді санау үшін, келесі кодты орындау жеткілікті:

```
void setup() {
    Serial.begin(9600); //Жүйелі портты ашамыз
}
void loop() {
    if (Serial.available() > 0) { //Мәліметтер беріле ме, тексереміз
        Serial.read(); //Берілген символдарды салыстырып оқимыз
    }
}
```

Демек, алдымен мәліметтер беріліп жатыр ма, тексереміз, егер берілсе, онда `Serial.read()` функциясы бір символды салыстырып оқиды. Егер бір символ ғана емес, 2, одан да көп, `Serial.read()` бәрібір бір символды салыстырып оқиды, келесі циклда тағы да басқа символды оқиды, осылай жалғасады, біз жіберген символдардың бәрін санап болғанша. `Serial.read()` функциясы тек символды салыстырып қоймай, ол оны қайтарады. Яғни, біз мысалы мынадай жағдай жаза аламыз:

```
if(Serial.read() == 'Z'){ //Егер Z символын біз есептесек
    digitalWrite(2, HIGH); //Екінші портқа - 5 вольт беру }
```

Яғни біз берілген мәліметтерді тексере аламыз және одан әртүрлі амалдар жасаймыз, мысалы, егер 'Z' әрпі берілсе, светодиодты жағамыз. Тексерілетін символ қос тырнақшаға емес “”, апострофқа “” алынғанын назарға алыңыздар.

Біз бұлай есептелген символды өзгергіште солай сақтаймыз. Өзгергіштің типі қандай болу керек? Өйткені бізде алынған символ сан емес қой. Символдар үшін өзгергіштердің арнайы типі бар. Ол **`char`** деп аталады (ағылшыннан - символ). `Char` типтегі өзгергіште біз бір символды сақтай аламыз. Байқап көрейік:

```
char box;
void setup() {
    Serial.begin(9600); //Жүйелі портты ашамыз
}
void loop() {
    if (Serial.available() > 0) { //Мәліметтер беріле ме, тексереміз
        box = Serial.read(); //Берілген символды оқимыз да boxта сақтаймыз
        if(box == 'Z'){ //Егер өзгергіште Z символы
            digitalWrite(2, HIGH); //Екінші портқа - 5 вольт береді
        }
    }
}
```

Дәл осылай алған символдарды сақтап, өңдей аламыз. Осымен бүгін жүйелі портты аяқтаймыз. Тағы бір кішкентай функция, оны білу сендерге пайдалы, бұл **millis()**. Бұл функция жібергеннен бастап немесе Arduinoны салғаннан кейін миллисекунд санын қайтарады. Ол миллисекундты қайтарады, Arduino ұзақ жұмыс істесе, оған қайтарылатын сан көбірек болады, санның максималды шегі бұл **unsigned long**, үшінші сабақтан білеміз, бұл типтегі мәліметтердің максималды мәні 4 294 967 295 құрайды. Бұл 49.7 күнде миллисекундты есептеуге жетеді. Одан кейін **millis()** функциясы қайтадан есептеуді бастайды. Біздің проектiлерге мұндай үлкен сандар әбден жеткілікті. Бұл функцияның көмегімен секундомер жасай аламыз. Мысалы, бір өзгергіште секундомерді жарысқа шыққанда, ағымдағы миллисекундты сақтай аламыз, ал қанша миллисекунд өткенін білгіміз келсе, онда біз ағымдағы мәннен **millis()** сақтағанды жай аламыз. Мысалы, біз Arduino 20345 миллисекунд жұмыс істегенде біздің секундомерді жібердік, бұл мәнді сақтаймыз, сосын біздің секундомерді тоқтатқанда, Arduino 25350 миллисекунд жұмыс істеді деп болжайық, онда біз 25350 - 20345 шегереміз, онда 5005 миллисекунд аламыз, демек біздің секундомер 5 бүтін секунд пен тағы 5 миллисекунд есептеді. Мысалы:

```
unsigned long start; //Бұл өзгергіште миллисекундтарды сақтаймыз
void setup() {
  Serial.begin(9600);
  start = millis(); // Arduinoны жарысқа қосқанда миллисекундты белгілеу
}
void loop() {
  Serial.println( millis() - start ); //Сақтаудан кейін қанша уақыт өткенін
                                   // start өзгергішіне
}
```

Осындай жай программа жүйелі порт монитормына Setup() функциясын орындағаннан кейін қанша уақыт өткенін шығарады. Тағы да Arduinoда бір функция бар, ол **micros()** деп аталады, ол Arduino жібергеннен кейін микросекундтарды есептейді, бір секундта миллион микросекунд екенін ескертеміз. Бұл функция **unsigned long** типтегі, ол да сандарды қайтарады, мұнда сандар **millis()** функциясына қарағанда 1000 рет көбірек болады, онда бұл функцияда барлығы 71.5 минут жұмыс уақытын есептеу үшін жетеді, одан кейін ол нөл болып қалады да, қайтадан есептеу басталады. Теория бойынша бүгін бітті.

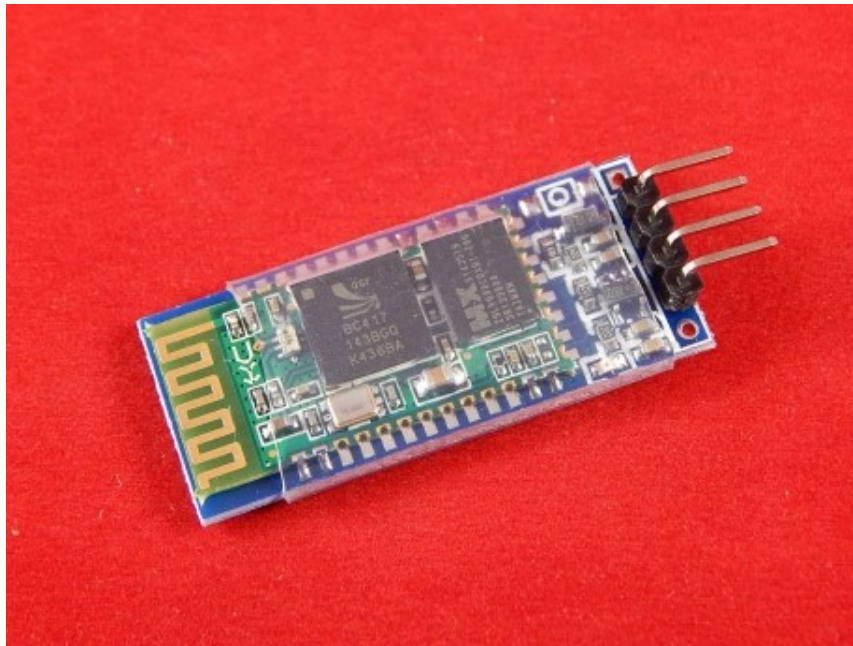
Келесі тапсырманы орындаңдар:

1. *Serial портқа мәліметтерді жіберу көмегімен светодиодты қосу мен өшіруін жасаңдар. Мысалы, 'A' әрпін жібергенде светодиод қосылысын, ал 'a' әрпін жібергенде – светодиод өшірілсін.*
2. *Компьютерден Serial портына жіберілген цифрларға байланысты (0-ден 9-ға дейін) серводвигательдің бұрылысын жасаңдар, мысалы, егер 0 жіберсек, онда двигатель 0 градусқа бұрылады, 9 жіберсек - 180 градусқа бұрылады және т.с. Ол үшін tar() функциясын қолданыңдар.*
3. *Реакцияның жылдамдығын анықтайтын құрылғы жасаңдар. Құрылғы динамик пен кнопкадан тұру керек. Құрылғы кнопканы басу арқылы жіберіледі. Ары қарай біраз ара уақыттан кейін динамик дыбыс шығарады, солай кнопканы басыңдар. Динамик шырылдай бастағанда, Arduino уақытты белгілейді, кнопканы қалай бастыңыз, Arduino реакцияға жұмсаған уақыт санының нәтижесін береді. Осыдан кейін, программа кнопканы басуды, қайтадан бастау үшін күтеді.*
4. *Есепті күрделендірейік, енді программа екі адам реакциясының жылдамдығын анықтайтын болыңдар, енді бізге 2 кнопка керек. Кнопкалардың бірін басып, программаны жібереміз, ары қарай егер динамик дыбыс шығарса, онда уақыт есептеуді бастаймыз. Бір кнопканы басқан кезде, оның уақытын сақтаймыз, екінші кнопканы басқан кезде екінші уақытты сақтаймыз. Осыдан кейін нәтижелерді салыстырамыз да, кім жеңді және реакцияның жылдамдығы қандай екенін жазамыз.*

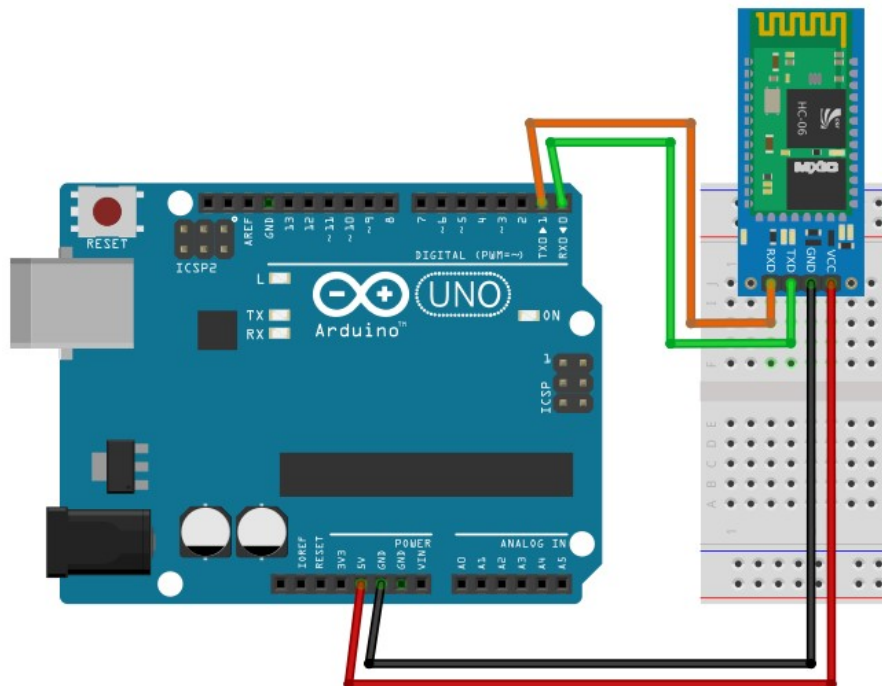
№14 сабақ

Тақырыбы: Bluetooth модулімен таныстыру

Ақыры біз bluetooth модуліне жеттік. Arduinoға bluetooth модулін құрылғы мәліметтерін және bluetooth технологиясын қолдайтын құрылғылардан, олар ноутбук, телефон, немесе қандай да басқа құрылғы болуы мүмкін, беру үшін қосуға болады. Біз HC-06 модуль жұмысын қарастырамыз.



Оны қосу схемасы өте қарапайым:



Біздің модульде 4 контакта бар екенін көрдік, олар VCC, GND, RX және TX. GND мен VCC контактарымен бәрі жай, оларға жер мен +5V сәйкес қосылады. RX пен TX контактары ақпаратты беру мен алу үшін жасалған. Біздің модульде RX контактасы Arduinoдан алынған мәліметтерді оқу үшін, ал TX контактасы Arduinoға мәліметтерді беру үшін пайдаланылады. Біздің Arduino платасында да осындай контактар бар, олар 0 және 1 цифрлы портында орналасқан. Олардың міндеттері де осындай: RX мәліметтерді оқиды, TX

береді. Arduino мәліметтерді сәйкес оқу үшін оны bluetooth модулімен TX аяғын RX қосу керек. Bluetooth модулі Arduinoдан мәліметтерді оқу үшін Arduinoдағы TX контактасын bluetooth модуліндегі RX контактасына қосу керек. Қосу схемасы мынадай болады:

Arduino контакталары	HC-06 модулінің контакталары
5V	VCC
GND	GND
RX (0 цифр порты)	TX
TX (1 цифр порты)	RX

Қай аяқтың мәліметтерді қабылдап, қайсысы қабылдайтынын жақсы еске сақтау үшін, T (Tx) әрпі - Transceive бұл қысқарту, ағылшыннан жіберу, ал R (Rx) әрпі - Receive қысқарту, ағылшыннан алу не қабылдау, оны Read - оқу ретінде де есте сақтауға болады. Сонымен модульді біз қостық, қалай енді мәліметтерді беруге болады?

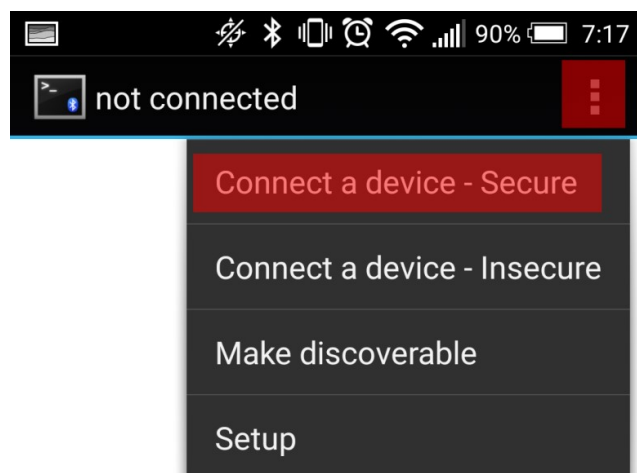
Мұны ОС Android орнатылған телефонға және телефоннан мәліметтерді беру мысалында қарастырайық. Бастапқыда Google Play Marketке кіру керек те, қандай болсын bluetooth терминалынан көшіру керек, олар көп, олар бір-бірінен аз ерекшеленеді. Біз Qwerty әзірлемесінің терминалын қарастырамыз. Оны келесі сілтемеден көшіруге болады <https://play.google.com/store/apps/details?id=Qwerty.BluetoothTerminal>

Сонымен біздің HC-06 модулін енді Arduinoға қосыңдар. Arduinoға қорек беріндер, сендердің смартфондарыңда bluetooth қосыңдар да, жаңа орнатылған bluetooth терминал қосымшасын ашыңдар.

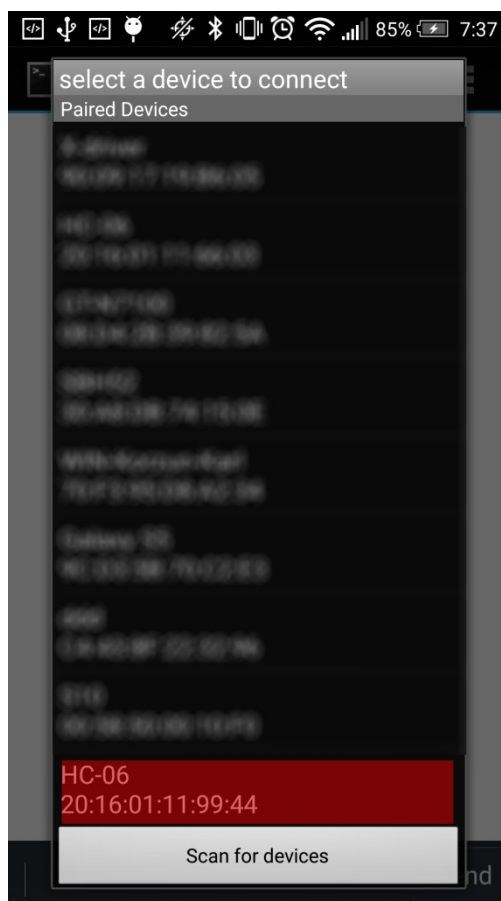
Үндемеу бойынша жоғарғы жолда not connected жазуы болады,



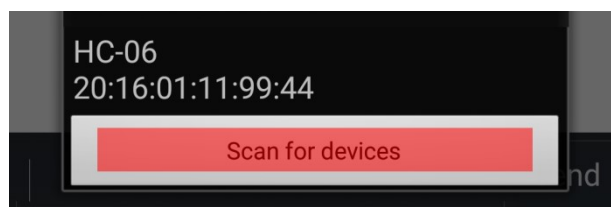
менюді басыңдар да, онда **Connect a device - Secure** (құрылғыға паролмен қосылу) пунктін таңдаңдар, бұл біздің bluetooth модуліне қосылу үшін қажет.



Меню пунктін басқаннан кейін сендерде жақын құрылғы bluetooth тізімімен терезе ашылады, сендер тізімнен **HC-06** тауып, оны таңдауларың керек.



Егер тізімде ол жоқ болса, онда **Scan for devices** (құрылғыны іздеу) кнопкасын басындар.



Тізімде біздің HC-06 модулі пайда болғанда қосылу үшін оны басу керек, егер модуль табылмаса, модульдің Arduinoға дұрыс қосылғанын тексеріңдер, оны қосқанда онда светодиод жануы керек. Өшіріп көріңдер де, Arduino қоректенуін қосындар, телефонда bluetooth өшіріңдер де қосындар.

Біздің модульге қосылғанда, сенен пароль сұралады, **HC-06** модулінің **стандартты паролі 1234**.

Ойдағыдай қосылғаннан кейін жоғарғы жол **connected** жазуына ауысады.

Енді тек Arduinoдан мәліметтерді алу немесе беру кодын жазу ғана қалды.

Оны жасау өте қарапайым, сендерге басқа кітапханалар керек емес, сендер барлық командаларды білесіңдер, Serial порт арқылы мәліметтерді беру үшін пайдаланылатын сол кодтар.

Яғни, телефонға қандай да мәтінді жіберу үшін, саған жазу жеткілікті:

```
void setup() {  
  Serial.begin(9600); //Жүйелі портты ашамыз  
}  
void loop() {
```

```
Serial.println("Text"); //Жолды ауыстырып Text сөзін жібереміз  
}
```

Телефоннан алынған символдарды есептеу үшін, жазу жеткілікті:

```
char box;  
void setup() {  
    Serial.begin(9600); //Жүйелі портты ашамыз  
}  
void loop() {  
    if (Serial.available() > 0) { //Мәліметтер берілгенін тексереміз  
        box = Serial.read(); //Берілген символды салыстырып оқимыз да, boxта сақтаймыз  
    }  
}
```

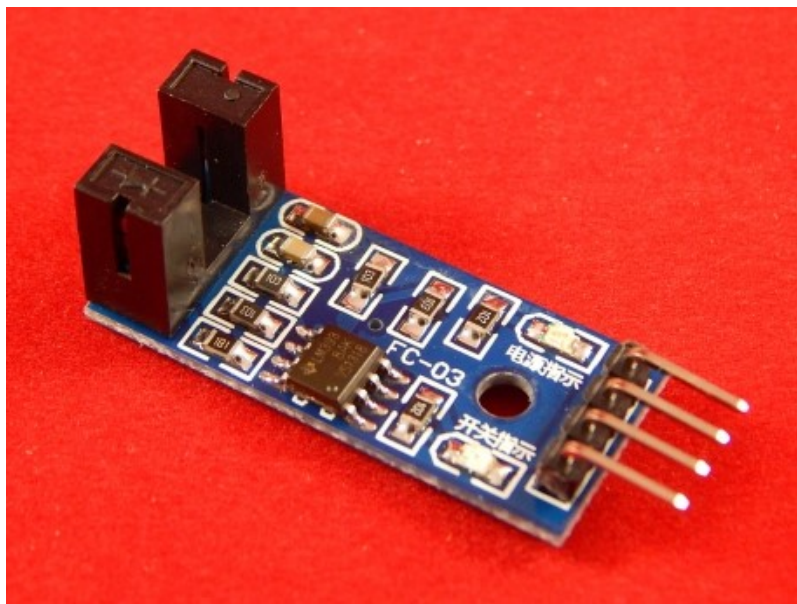
Міне, барлығы, бұлардың барлығын өткен сабақтарда өттік. Енді практикадан өтейік, келесі тапсырмаларды орындайық:

1. *Светодиодты телефоннан сөндіру мен қосу үшін программа жасаңдар.*
2. *Динамиктің тональдығын басқару үшін, оған дыбыс биіктігін 0-ден 9-ға дейін тапсыруға, 0 - динамик мүлдем шырылдамайды, ал 9 - ол максималды биіктігіне дыбыс шығаратын программа жасаңдар.*
3. *Температура датчигін қосып, бөлменің температурасы телефонға берілетіндей жасаңдар.*
4. *Arduino F әрпі берілгенде біздің робот алға жүретіндей, B - әрпін жібергенде - кейін, S әрпін жібергенде тоқтайтындай жасаңдар.*

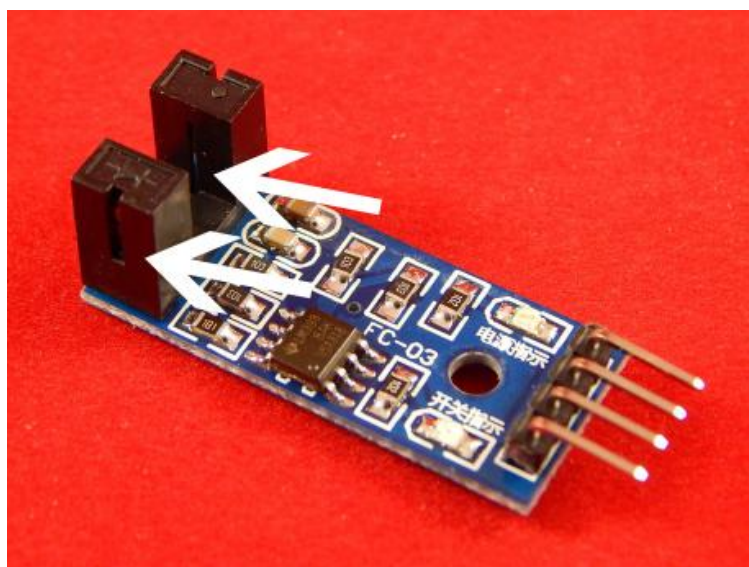
№15 сабақ

Тақырыбы: Цифрлық тахометрмен таныстыру

Бүгін біз FC-03 цифрлық модуль тахометр көмегімен доңғалақтың айналым жылдамдығын анықтау мен түзетуге үйренеміз:



Бұл тахометрдің жұмыс істеу принципі өте қарапайым, оның екі жағында пластик бекіткіштер, оның ішінде жарық көздері (светодиод) және фотоқабылдағыш (жарықтың датчигі) бар:



Светодиод жұқа саңылау арқылы фотоқабылдағышқа жарық түсіреді, тахометрде жасыл светодиод бар, ол да фотоқабылдағышқа жарық түсіреді.

Егер кенет жарық көзі мен фотоқабылдағыштың арасында жарықты жабатын қандай да затты қойсақ, онда жасыл светодиод сөнеді. Егер затты алып тастасақ, онда көзден жарық тағы да датчикке түседі де, жасыл светодиод тағы да жанады.

Бұл тахометрдің 4 аяғы бар, олардың жүктелген міндеттері төменгі кестелерде көрсетілген:

Тахометр портының атауы:	Жүктелген міндеті:
VCC	Қоректену: +5V
GND	Қоректену: GND

D0	Сигналды цифрлы порт
A0	Сигналды ұқсас порт

Сендер байқағандай, тахометрде D0 және A0 порттары бар, жарық датчикке түскенде, онда D0 портында 0V – LOW кернеудің төменгі деңгейі болады, ал егер жарық үзілсе онда D0 портына +5V – HIGH кернеудің жоғары дәрежесі беріледі. A0 ұқсас порты кернеудің өте нақты мәнін көрсетеді. Датчикке жарық жарқырап түссе, онда кернеу көп те, аз да болады және керісінше. Егер байқаған болсаңдар, цифрлы тахометрдің жұмыс принципі линия датчигінің жұмысына өте ұқсас.

Доңғалақ айналымының жылдамдығын қалай анықтауға болады. Ол үшін бізге тесіктері бар диск, мынау сияқты керек болады:



Бұл дискіні двигатель осіне бекітеміз, ал тахометрдің дискі фотоқабылдағыш пен жарық көздерінің арасында болу керек. Егер двигатель дискіні айналдырса, дискідегі тесіктердің арқасында, тахометрдегі жарық біресе үзіледі, біресе қайтадан басталады. D0 портына кезекпен 5 не 0 вольт беріледі, диск тез айналса, мәндер де тез өзгереді. Бұл дискте 20 тесік, бұл доңғалақтың бір айналымына D0 портына 5 пен 0 вольт кернеуі 20 рет беріледі.

Егер Arduino көмегімен бұл импульстерді есептесек және олардың арасындағы уақытты белгілесек, онда доңғалақ минутына немесе сағатына қанша айналым жасайтынын есептей аламыз және мысалы, сәйкес жылдамдықты білеміз.

Доңғалақ айналымын есептеуден бастайық. Сонымен тахометрді Arduinoға қосудан бастаймыз, D0 контактысын қандай да цифрлық портқа қосамыз, A0 контактысын қосып керегі жоқ.

```
void setup() {
  Serial.begin(115200); //Жүйелі порт мониторын ашамыз
  pinMode(2, INPUT); //Портты салыстырып оқуға орнатамыз
}
```

Бұл жағдайда D0 контакты Arduinoның D2 цифрлық портына қосылған. Arduinoның жұмысын баяулатпас үшін жүйелі портты 115200 бод жылдамдығымен аштық, бұл маңызды, өйткені диск тез айналым уақытын өлшейміз. Енді тахометр импульстерінің есептегішін жасаңдар. Кезекті жарықты тахометрдің жанынан өткендегі кезеңде сандарды сақтайтын алдымен өзгергіштерді жасайық. Сонымен бірге өзгергіш – ауыстырып қосқыш жасау қажет, онда жарық пайда болғанда мысалы 1, ал кедергі пайда болғанда 0 сақтаймыз. Бұл сол бір жарықты есептегіш бір реттен арттық жұмыс істемеу үшін керек:

```
int counter=0; //айналым есептегіші  
byte impulse=0; //тесікті ауыстырып қосқышы|кедергі
```

Енді бізге порттан сигналды салыстырып оқу керек, онда төмен деңгей – LOW пайда болғанда есептегішті көбейту керек және біздің өзгергішке impulse=1 ауыстырып қосқышын беру қажет. Ал кедергі пайда болғанда портқа HIGH сигналы беріледі,біз оны білеміз де, біздің өзгергішті кері 0-ге ауыстырып қосамыз:

```
void loop() {  
  //Егер портқа LOW берілсе және ауыстырып қосқышта 0 сақталса  
  if(digitalRead(2)==0 && impulse==0){  
    //Есептегішті кеңейтеміз  
    counter++;  
    //Есептегіш артық жұмыс істемес үшін ауыстырып қосқышта 1сақтаймыз  
    impulse=1;  
    //Егер портқа 1 берілсе, демек кедергі пайда болды  
  } else if(digitalRead(2)==1 && impulse==1){  
    //Біздің ауыстырып қосқышты нөлдейміз  
    //есептегіш жұмыс істеу үшін  
    impulse=0;  
  }  
}
```

Енді біздің есептегіш жұмыс істейді, егер есептегіште 20 саны көрінсе, демек біздің доңғалақ 1 айналым жасады. Біз есептегіш 0-ге тең кедергі micros() функция көмегімен үлкен нақтылық үшін немесе millis() функциясының көмегімен уақытты еске сақтаймыз. Оны есептегіште 20 саны пайда болғаннан кейін салыстыру керек, содан кейін біздің есептегішті нөлдейміз. Уақытты салыстырып, 1 айналымға қанша миллисекунд кеткенін білеміз. Ол үшін бізге өзгергіш керек, онда салыстыру уақытын сақтаймыз:

```
unsigned long time1Turn; //Мұнда 1 айналымның уақытын сақтаймыз
```

Енді белгілейтін уақыттың кодын жазамыз:

```
//Егер есептегіш 0-ге тең болса  
if(counter==0){  
  //уақытты сақтаймыз  
  time1Turn = millis();  
}  
//Егер есептегіш 20-ға тең болса  
if(counter==20){  
  //Ағымдағы уақыттан сақталғанды аламыз  
  //және экранға шығарамыз  
  Serial.println(millis()-time1Turn);  
  //Есептегішті нөлдейміз  
  counter=0;  
}
```

Енді біз білеміз, 1 айналымға қанша миллисекунд жұмсалғанын, бір минутта 60000 миллисекунд, онда минутына қанша айналым жасайтынын біз білеміз, ол үшін мәнді сақтауға өзгергішті жасаймыз:

```
float RPM=0; //Минуттағы айналым саны
```

Енді біз айналым санын есептей аламыз:

```
//Егер есептегіш 20 тең болса
if(counter==20){
  //Ағымдағы уақыттан сақталғанды аламыз
  //Секундтағы миллисекунд санын біздің бөлеміз
  //белгіленген уақыт
  RPM = 60000/(millis()-time1Turn);
  //және экранға шығарамыз
  Serial.println(RPM);
  //Есептегішті нөлдейміз
  counter=0;
}
```

Енді минутына біздің доңғалақ жасайтын айналым саны экранға шығарылады.

Сағатына айналым санын шығарайық, ол үшін бізге бір сағаттағы миллисекунд санын 1 айналымға жұмсалған уақытқа бөлеміз.

Бір минутта 60000 миллисекунд, бір сағатта 60 минут, демек біз $60 \cdot 60000$ көбейтіп, бір сағатта 3600000 миллисекунд аламыз, өзгергіш жасаймыз, онда айналым/сағат сақтаймыз:

```
float RPH=0; //Сағаттағы айналым саны
```

Есептейміз:

```
//бір сағаттағы айналымды есептейміз
RPH = 3600000/(millis()-time1Turn);
//және экранға шығарамыз
Serial.println(RPH);
```

Енді біз сағатына доңғалақ айналымының санын білеміз, бірақ жылдамдықты қалай білуге болады? Ол үшін доңғалақ 1 айналымда қанша ара қашық өтетінін білуіміз керек. Доңғалақ бір айналымда осы доңғалақ шеңберінің ұзындығына тең ара қашықта өтеді. Шеңбер ұзындығын қалай білуге болады? Мұнда 2 нұсқа бар, не өлшеу керек, не 5 сыныпта математиканы оқығандар шеңбер ұзындығы Π **3.14** саны доңғалақ диаметріне көбейтілгенге тең екенін біледі. UnoCraft машинкасы доңғалағының диаметрі шамамен 70 мм құрайды, яғни 7 см:



Мұндай доңғалақты 1 айналымда өтетін ара қашықтығы $7 \cdot 3,14 = 21,98 \text{ см}$ құрайды, егер біз ара қашықты метрде білгіміз келе, онда бұл мәнді 100ге бөлеміз, өйткені метрде 100 сантиметр, егер ара қашықты километрде білгіміз келсе, онда бұл мәнді 100 000 болу керек. Енді 1 сағаттағы айналым санын білсек, біз 1 сағаттағы жылдамдықты километрмен есептеуіміз керек:

```
//Жылдамдықты КМ/Ч есептейміз де, экранға шығарамыз  
Serial.println(RPM*7*3.14/100000.0);
```

Кодтың бәрі толығымен былай көрінеді:

```
int counter=0; //айналым есептегіші  
byte impulse=0; //жарық ауыстырып қосқышы/кедергі  
unsigned long time1Turn; //Мұнда 1 айналымның уақытын сақтаймыз  
float RPM=0; //Минуттағы айналым саны  
float RPH=0; //Сағаттағы айналым саны  
void setup() {  
  Serial.begin(115200); //Жүйелі порттың монитормын ашамыз  
  pinMode(2, INPUT); //Салыстырып оқу портын орнатамыз  
}  
void loop() {  
  //Егер портқа LOW берілсе және ауыстырып қосқышта 0 сақталса  
  if(digitalRead(2)==0 && impulse==0){  
    //Есептегішті көбейтеміз  
    counter++;  
    //Есептегіш енді жұмыс істемес үшін ауыстырып қосқышта 1 сақтаймыз  
    impulse=1;  
    //Егер портқа 1 берілсе, демек кедергі пайда болды  
  } else if(digitalRead(2)==1 && impulse==1){  
    //Келесі жарықта, ауыстырып қосқышты нөлдейміз  
    //есептегіш жұмыс істеуге  
    impulse=0;  
  }  
  //Егер есептегіш 0-ге тең  
  if(counter==0){  
    //Уақытты сақтаймыз  
    time1Turn = millis();  
  }  
  //Егер есептегіш 20 тең  
  if(counter==20){  
    //Ағымдағы уақыттан сақталғанды аламыз  
    //Секундтағы миллисекунд санын біздің бөлеміз  
    //белгіленген уақыт  
    RPM = 60000/(millis()-time1Turn);  
    //Сағаттағы айналымды есептейміз  
    RPH = 3600000/(millis()-time1Turn);  
    //және экранға шығарамыз  
    //Serial.println(RPM);  
    //Жылдамдықты КМ/сағ. есептейміз де, экранға шығарамыз  
    Serial.println(RPH*7*3.14/100000.0);  
    //Есептегішті нөлдейміз  
    counter=0;  
  }  
}
```

}

Міне, біздің доңғалақтардың жылдамдығын есептеуге үйрендік.

Тапсырмаларды орындауға көшеміз:

1. Дәл нақтылық үшін уақыт миллисекундта емес, микросекундта есептелу үшін кодты ауыстырыңдар.
2. Екі доңғалақты да қосыңдар және әрқайсының жылдамдығы *bluetooth electronics* программасына шығатындай жасаңдар
3. Егер біздің доңғалақтар әртүрлі жылдамдықпен айналса, онда тез айналатын доңғалақтың жылдамдығы екінші доңғалақ айналымынан соншалықты баяу қылыңдар. Немесе артсын, егер артатын орын болса.
4. *Bluetooth electronics* программасы арқылы доңғалақ айналымына 0.5 км/сағ. 5 км/сағ. дейін жылдамдық тапсыра алатындай жасаңдар. Яғни, егер программада 3км/сағ. жылдамдық қойсақ, онда доңғалақтар да сол жылдамдықпен айналады.