**ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ ԿՐԹՈՒԹՅԱՆ, ԳԻՏՈՒԹՅԱՆ, ՄՇԱԿՈՒՅԹՒ ԵՎ ՍՊՈՐՏԻ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆԸ**

**ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ**

**ՄԱԳԻՍՏՐՈՍԱԿԱՆ ԿՐԹԱԿԱՆ ԾՐԱԳԻՐ**

**« Միկրոկոնտրոլերի հենքով հրամանների կոնվեյերային ճարտարապետության մշակումը և իրագործումը »**

**Արկադի Գագիկի Հակոբյան**

**«Համակարգչային ճարտարագիտություն»**

**Մասնագիտությամբ ճարտարագիտության մագիստրոսի որակավորման աստիճան հայցելու ատենախոսություն**

**ԵՐԵՎԱՆ 2022ՀԱՍՏԱՏՄԱՆ ԹԵՐԹ**

**ԹԵՄԱ` « Միկրոկոնտրոլերի հենքով հրամանների կոնվեյերային ճարտարապետության մշակումը և իրագործումը »**

Հեղինակ՝ Ա․ Գ․ Հակոբյան

Ատենախոսության ղեկավար՝ Ա․ Կ․ Սաղաթելյան  
 տ․գ․թ․, դոցենտ

Գրախոս՝ Խ․Գ Շարոյան

տ․գ․թ․, դոցենտ

Ամբիոնի վարիչ` Գ.Տ. Կիրակոսյան

տ.գ.դ., պրոֆեսոր

Ինստիտուտի տնօրեն` Ս.Ա. Մանուկյան

տ.գ.թ., դոցենտ

**ԿԵՆՍԱԳՐԱԿԱՆ ՏՎՅԱԼՆԵՐ**

***Մագիստրանտ`***  Հակոբյան Արկադի Գագիկի

***Աստիճանը***՝ «Համակարգչային ճարտարագիտություն»

մասնագիտության մագիստրանտ

***Տարեթիվը՝*** 2022

***Ծննդյան տարեթիվը՝*** 11.05.1996 թ.

***Մինչ մագիստրոսական***  Ճարտարագետ բակալավր

***որակավորումը՝***

***Մասնագիտությունը՝***  « Ինֆորմատիկա և հաշվողական տեխնիկա»

***Հրատարակված***

***աշխատանքներ` ---***

**ՀԱՄԱՌՈՏԱԳԻՐ**

***Թեմա`* «Միկրոկոնտրոլերի հենքով հրամանների կոնվեյերային ճարտարապետության մշակումը և իրագործումը»**

IoT համակարգերի արագ տարածումը, ինտեգրումը շատ ոլորտներում կասկած չի հարուցում որ դրանք դեռևս իրենց կարևորության գագաթնակետին չեն հասել։ Այս համատեքստում կարևոր է նկատել, որ IoT համակարգերը պետք է ունենան տվյալների հետ աշխատանքի բարձր հաճախականություն սակայն միարժամանակ պահպանեն տվյալների ամբողջականությունը այդ տվյալների ստեղծման աղբյուրից մինչև նպատակակետ։ Վերջին նպատակը, այն է տվյալների ամբողջականությունը պահպանելը, IoT համակարգում միկրոկոնտրոլերների աշխատանքով ծանրաբեռնվածության հետ պայմանավորված հանդիպում է խնդիրների։ IoT համակարգում տարբեր սարքերի համատեղ աշխատանքը ասինխրոնություն է ստեղծում, որը իր հետ բերում է մի շարք խնդիրների, ինչպիսիք են տվյալների տարընթերցումը, ընդհանուր համակարգի արագագործության անկումը և այլն։ Այս աշխատանքում կուսումնասիրենք և կիրականացնենք կոնվերային ճարտարապետությամբ հրամանների կատարում, որը կլուծի համակարգում կարևոր օղակների միջև սինխրոնիզացիայի և տվյալների տարընթերցման հետ կապված խնդիրները։

***Առանցքային բառեր։ IoT, միկրոկոնտրոլեր, սենսոր, կոնվերացում*m building tools investigation and mobile cross-platform creation»**

Learning platforms are very useful in the field of learning, they can help in everyday learning and also in the case of distance learning. Today, distance learning is evolving rapidly. More and more educational organizations are introducing distance learning technologies into the education system. And the most preferred equipment for teaching at present are mobile phones. To do all this will need to use cloud services and databases. In the work we will study all the necessary technologies for development then we will build a mobile cross-platform platform for them.

***Key Words:*** *Cloud, PaaS, SaaS, API, Database, HTTP requests*

ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ՏՀՏէ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ԻՆՍՏԻՏՈՒՏ

Քոմփյութերային համակարգերի և ցանցերի ԱՄԲԻՈՆ

Մասնագիտություն «Համակարգչային ճարտարագիտություն» դասիչ 061103.00.7

Թիվ ՄՏՏ020 ակադեմիական խմբի \_\_\_\_\_\_Հակոբյան Արկադի Գագիկի \_\_\_\_\_\_\_\_\_\_\_\_ (ուսանողի ազգանուն, անուն, հայրանուն)

**Մ Ա Գ Ի Ս Տ Ր Ո Ս Ա Կ Ա Ն Թ Ե Զ Ի Ա Ռ Ա Ջ Ա Դ Ր Ա Ն Ք**

*Աշխատանքի թեման`* «Միկրոկոնտրոլերի հենքով հրամանների կոնվեյերային ճարտարապետության մշակումը և իրագործումը»

*Առաջադրանքը հաստատված է՝*  15.02.2021 թ. Թիվ 01-11/78 հրամանով

*Ինստիտուտի տնօրեն՝* Մանուկյան Ս.Ա. տ.գ.թ., դոցենտ\_\_\_\_\_

(Ա.Ա.Հ., ստորագրություն, ամսաթիվ)

*Ամբիոնի վարիչի* ` Կիրակոսյան Գ.Տ. տ.գ.դ.,պրոֆեսոր\_\_\_

(Ա.Ա.Հ., ստորագրություն, ամսաթիվ)

2. *Աշխատանքի նախնական տվյալները*` Ապարատածրագրային միջոցներ, վերծանիչ, խմբագրիչ

3. *Հաշվեբացատրագրի բովանդակությունը (բաժինների և մշակման ենթակա հարցերի թվարկմամբ)՝*

**Գլուխ1․** **Միկրոկոնտրոլերների ճարտարապետությունների հետազոտումը և բուֆերիզացման մեխանիզմների ուսումնասիրումը։**

**Գլուխ2. Անհրաժեշտ սարքավորումների ընտրությունը, կոնվերային ճարտարապետության կազմակերպման սկզբունքները:**

**Գլուխ3. Սարքավորումների միացումը եվ IoT համակարգի իրագործումը ։**4. Աշխատանքի կատարման օրացույցային պլան

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| թիվը/կ | Աշխատանքի կատարման փուլերը | | | Ծանոթություն |
| Անվանումը | կատ. ժամկ. | հաշվ. ձևը |
| 1 | Միկրոկոնտրոլերների ճարտարապետությունների հետազոտումը և բուֆերիզացման մեխանիզմների ուսումնասիրումը: |  |  |  |
| 2 |  |  |  |  |
|  | I ատեստավորում |  |  |  |
| 3 | Անհրաժեշտ սարքավորումների ընտրությունը, կոնվերային ճարտարապետության կազմակերպման սկզբունքները**:** |  |  |  |
| 4 |  |  |  |  |
|  | II ատեստավորում |  |  |  |
| 5 | Սարքավորումների միացումը եվ IoT համակարգի իրագործումը ։ |  |  |  |
|  |  |  |  |  |
|  | III ատեստավորում |  |  |  |
|  | Նախնական պաշտպանություն |  |  |  |

5. *Աշխատանքի պաշտպանության օրը* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. *Ամբիոնի վարիչի՝* Կիրակոսյան Գ.Տ., տ.գ.դ., պրոֆեսոր \_\_\_\_\_\_\_\_

(Ա.Ա.Հ., ստորագրություն, ամսաթիվ)

7. *Աշխատանքի ղեկավար՝* Սաղաթելյան Ա․Կ տ․գ․թ․, դոցենտ\_\_\_\_\_\_\_

(Ա.Ա.Հ., ստորագրություն, ամսաթիվ)

8. *Աշխատանքի առաջադրանքը ստացա* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ուսանողի ստորագրություն, ամսաթիվ)

ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆ

[ՆԵՐԱԾՈՒԹՅՈՒՆ 9](#_Toc103292002)

[ԳԼՈՒԽ 1. ՄԻԿՐՈԿՈՆՏՐՈԼԵՐՆԵՐԻ ՃԱՐՏԱՐԱՊԵՏՈՒԹՅՈՒՆՆԵՐԻ ՀԵՏԱԶՈՏՈՒՄԸ և ԲՈՒՖԵՐԻԶԱՑՄԱՆ ՄԵԽԱՆԻԶՄՆԵՐԻ ՈՒՍՈՒՄՆԱՍԻՐՈՒՄԸ 12](#_Toc103292003)

[1.1 Միկրոկոնտրոլերների տեսակները և առանձնահատկությունները 14](#_Toc103292004)

[1.2 Բուֆերիզացման մեխանիզմների ուսումնասիրումը 18](#_Toc103292005)

[1.3 Ժամանակակից, առավել օգտագործվող միկրոկոնտրոլերները 23](#_Toc103292006)

[1.4 Միկրոկոնտրոլերի ընտրման չափորոշիչները 25](#_Toc103292007)

[1.5 Թեզի կատարման հիմնավորումը 29](#_Toc103292008)

[Առաջին գլխի եզրահանգում 30](#_Toc103292009)

[ԳԼՈՒԽ 2․ ԱՆՀՐԱԺԵՇՏ ՍԱՐՔԱՎՈՐՈՒՄՆԵՐԻ ԸՆՏՐՈՒԹՅՈՒՆԸ, ԿՈՆՎԵՐԱՅԻՆ ՃԱՐՏԱՐԱՊԵՏՈՒԹՅԱՆ ԿԱԶՄԱԿԵՐՊՄԱՆ ՍԿԶԲՈՒՆՔՆԵՐԸ 31](#_Toc103292010)

[2.1 Միկրոկոնտրոլերի տպասալի ֆիզիկական կոմպոնենտները 31](#_Toc103292011)

[2.2 Սենսորների և շիֆթ ռեգիստրների ճարտարապետության ընտրությունը 38](#_Toc103292012)

[2.3 Կոնվերային ճարտարապետության սկզբունքները, իրականացումը IoT համակարգում 44](#_Toc103292013)

[Երկրորդ գլխի եզրահանգում 51](#_Toc103292014)

[ԳԼՈՒԽ 3. ՍԱՐՔԱՎՈՐՈՒՄՆԵՐԻ ՄԻԱՑՈՒՄԸ ԵՎ IOT ՀԱՄԱԿԱՐԳԻ ԻՐԱԳՈՐԾՈՒՄԸ 52](#_Toc103292015)

[3.1 IoT համակարգի շղթայի նկարագիրը 52](#_Toc103292016)

[3.2. Համակարգի բաղկացուցիչ մասերի կարգավորումները և միակցումը 54](#_Toc103292017)

[3.3 Միկրոկոնտրոլերի հենքով կոնվերային ճարտարապետությամբ IoT համակարգի իրականացումը 59](#_Toc103292018)

[Երրորդ գլխի եզրահանգում 66](#_Toc103292019)

[ԵԶՐԱԿԱՑՈՒԹՅՈՒՆ 67](#_Toc103292020)

[ՏԵՐՄԻՆԱԲԱՆՈՒԹՅՈՒՆ 68](#_Toc103292021)

[ԳՐԱԿԱՆՈՒԹՅԱՆ ՑԱՆԿ 68](#_Toc103292022)

[ՀԱՎԵԼՎԱԾ 69](#_Toc103292023)

# ՆԵՐԱԾՈՒԹՅՈՒՆ

**Աշխատանքի նպատակը** ՝ IoT համակարգերի զարգացման հետ կապված, դրա արդիականությունը հաշվի առնելով, միկրոկոնտրոլերի տվյալների մշակման ընթացքում առաջացող մի շարք խնդիրներից տարընթերցումը հախթահարելու համար ուղղված լուծումներն են այս թեզի շրջանակներում ընդգրկված։ Այն աշխատանքներում, որտեղ տվյալների ամբողջականությունը հանդիսանում է կարևորագույն խնդիր, անհնար է դառնում առանց կողմնակի միջամտության ապահովել դա։ Այս թեզի հիմքում ընկած է այդպիսի միջամտություն, որը որպես հավելում IoT համակարգին դարձնում է այն տվյալների կորստի տեսանկյունից առավել հուսալի:

**Խնդիրները** ՝ կոնվերային իրականացում ապահովելը այս աշխատանքում ներկայացված է որպես տվյալների տարընթերցման և առավել հուսալի IoT համակարգ ստանալու նպատակին հասնելու հիմնական մեխանիզմ։ Այդ մեխանիզմը կառուցելու համար ստեղծվել են միկրոկոնտրոլերի, սարքի միջավայրի ուսումնասիրության կոնվերային իրագործման և ծրագրի մշակման, այնուհետև ընդհանուր IoT համակարգի կոնվերային ճարտարապետության իրագործման խնդիրները։

**Արդիականությունը** ՝ IoT համակարգերը իրենց դերը և նշանակությունը տեխնոլոգիական այս դարաշրջանում էլ ավելի են ամրապնդվում զուգահեռ ինտեգրվելով տարբեր ոլորտներում։ Ժամանակակից մի քանի տեխնոլոգյաներ ինչպիսիք են 5G տեխնոլոգյան, արհեստական բանականությունը ինտեգրում են իրենց աշխատանքի հիմքերում IoT համակարգերը ինչը նշանակում է, որ այս համակարգերը դեռևս չեն հասել իրենց տարածվածության գագաթնակետին։ Ինչպես հաղորդվում է <<Gartner - ի>> միջազգային IT սեգմենտի վերլուծության և կանխատեսումների կազմակերպություն կողմից ՝ 2021 թվականին գլոբալ ծախսերը 5G ցանցի ենթակառուցվածքների ստեղծման համար ավելացել են 19-բիլիոն դոլարով` տարեկան 39 % - աճի տեմպերով։ Այս թվերը հավանաբար կշարունակեն ավելանալ 2022 թվականին։ IoT համակարգերի դերը կշարունակվի մեծանալ 5G ցանցի տեխնոլոգիաի և արհեստական բանականության, ավտոմատացծամ տեխնիկաների կատարելագործման հետ զուգընթաց մի շարք պատճառներով, որոնցից են ՝ ծրագրային իրականացման խոցելիության, կիբեր հարձակումների, տարբեր տվյալների փոխանցման արագության սահմանափակումների հախթահարման համար։ Այս աշխատանքում ներկայացված IoT համակարգի տվյալների ստացման , մշակման և փոխանցմանն ուղղված «բարելավումը» կարելի է համարել արդի, քանի որ IoT համակարգերը տարեցտարի ավելի մեծ նշանակություն են ունենալու տարբեր ոլորտներում ՝ աշխատանքների կազմակերպման գործում և հետևաբար տվյալների կոնվերային կերպով մշակումը հնարավորություն կտա պահպանել յուրաքանչյուր տվյալը այն աշխատանքների համար, որոնց համար դա կունենա առավել մեծ նշանակություն։

**Աշխատանքի կառուցվածքը** ՝ մագիստրոսական թեզը բաղկացած է 3 գլխից։

1. **Գլուխ 1**-ում ուսումնասիրված են միկրոկոնտրոլերների տեսակները, դրանց առանձնահատկությունները, կիրառական ոլորտները, ընտրման չափորոշիչները ՝ կոնվեյերային ճարտարապետության մշակման համար։ Արդի IoT համակարգերում առաջացող խոչնդոտը և դրա լուծմանն ուղղված ակնարկ։
2. **Գլուխ 2**-ում ներկայացված են սարքի իրագործման միջավայրերի ուսումնասիրումը, կոնվերային կառուցվածքային սխեմայի և ծրագրի մշակումը սենսորների, բուֆերային ռեգիստրների տեսակների ուսումնասիրությունները ինչպես նաև կոնվերային աշխատանքի գաղափարը և նշանակությունը տվյալ խնդրի շրջանակներում։
3. **Գլուխ 3**-ում ներկայացված են կոնվերային ճարտարապետությամբ IoT համակարգի իրագործումը, շղթայի նկարագիրը, մշակված է համակարգի ծրագրային և կառուցվածքային նկարագրությունները։

**Հետազոտման օբյեկտը** ՝ միկրոկոնտրոլերն է։ Միկրոկոնտրոլերը, որպես ժամանակակից համակարգիչների կենտրոնական հաշվողական մարմին հանդիսացող պրոցեսորներին հիմք դրած սարք, այժմ հանդես է գալիս որպես IoT համակարգերի կենտրոնական , ղեկավարող սարք, որի աշխատանքի ընթացքում տվյալների մշակման, ամբողջականության պահպանման հետ կապված կհետազոտենք, կհամեմատենք հավասար պայմաններում պրոցեսորների աշխատանքների հետ։

**Հետազոտման մեթոդը** ՝ մագիստրոսական թեզի շրջանակներում, հետազոտման համար օգտագործված են Arduino UNO տպասալը, Arduino IDE խմբագրիչը և Proteus մոդելավորման ծրագիրը։

# ԳԼՈՒԽ 1. ՄԻԿՐՈԿՈՆՏՐՈԼԵՐՆԵՐԻ ՃԱՐՏԱՐԱՊԵՏՈՒԹՅՈՒՆՆԵՐԻ ՀԵՏԱԶՈՏՈՒՄԸ և ԲՈՒՖԵՐԻԶԱՑՄԱՆ ՄԵԽԱՆԻԶՄՆԵՐԻ ՈՒՍՈՒՄՆԱՍԻՐՈՒՄԸ

**Ինչ է միկրոկոնտրոլերը**

Միկրոկոնտրոլերը սարք է ինտեգրալ սխեմայի վրա նախատեսված կառվարելու էլեկտրական սարքերի այլ հատվածներ՝ շնորհիվ ներկառուցված պրոցեսորի, հիշողության, հաղորդակցության սարքերի։ <<Microcontroller>> - բառը իրապես բնութագրում է այս սարքին, <<Micro>> - վերաբերում է փոքրությանն, իսկ <<Controller>> - վերաբերվում է սարքի կառավարելու հնարավորությանը։ Միկրոկոնտրոլերների տեսակների , առանձնահատկությունների և կիրառման մասին ավելի մանրամասն տեղեկատվություն ներկայացված է այս աշխատանքում։

**Կիրառման ոլորտները**

Միկրոկոնտրոլերները ամենուր են։ Դրանցով են ստեղծվում շատ էլեկրոնային սարքեր ինչպիսիք են համակարգչային ստեղնաշարեը, տպասարքերը, միկրոալիքային վառարանները, ականջակալները, հեռուստացույցի հեռակարավարման վահանակները և այլն։ Ինչպես կարող ենք ենթադրել, միկրոկոնտրոլերները կարող են կիրառվել բոլոր այն ոլորտներում, որտեղ կարելի ավտոմատացնել որոշակի գործառույթներ։ Տեխնոլոգիական այս դարաշջանում գրեթե անհնար է դարձել գտնել մի այնպիսի ոլորտ, որը անմասն է մնացել իր գործառույթները մասամբ կամ ամբողջովին ավտոմատացնելու գաղափարից, դա նշանակում է որ միկրոկոտրոլերները ոչ միայն շարունակելու են կիրառվել այլ նաև գրավել նորանոր ոլորտներ որտեղ նրանք կավտոմատացնեն բազմաբնույթ գործառույթներ։ Միկրոկոնտրոլերը, սակայն հանդես չի գալիս միայնակ, նրա հետ աշխատանքի համար անհրաժեշտ են հատուկ միջավայրեր, սարքեր որոնք կապ կհաստատեն միկրոկոնտրոլերի հետ և կապահովեն տվյալների փոխանցման կամ ստացման հնարավորությունները։ Այդպիսի համակարգերը կոչվում են IoT համակարգեր և միկրոկոնտրոլերնեը հանդիսանում են այդ համակարգերի կենտրոնական և հիմնական մասը։

Միկրոկոնտրոլերը բաղկացած է ՝

1. Կենտրոնական պրոցեսորային միավորից (CPU)
2. Հիշողությունից
3. Հաղորդակցության սարքերից (Peripherals)

**Կենտրոնական պրոցեսորային սարք (CPU)**

CPU-ն իրականացնում է թվաբանական, տվյալների հոսքի կառավարման գործողություններ ինչպես նաև գեներացնում է կարգավորումների համաձայն կատավարող ազդանշաները։ Շնորհիվ ծրագրային ինտեգրացիայի այժմ CPU-ով թվաբանական գործողություն կատարելու հրահանգներ կարելի է գրել բարձր դասի ծրագրավորման լեզուներով, ինչպիսին է օրինակ C -ն։

**Հիշողություն**

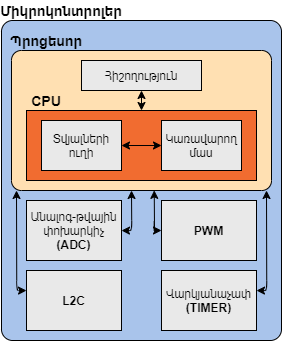
<<Nonvolatile>> - Էներգիայից կախում չունեցող հիշողությունը օգտագործվում է միկրոկոնտրոլերի ծրագիրը պահպանելու համար։ Հետագայում միկրոկոնտրոլերների տեսակները ուսումնասիրելուց կհանդիպենք <<Ֆլեշ>> - հիշողությանը, որը նույնպես համարվում է nonvolatile հիշողություն։

<<Votaile>> - հիշողությունը ՝ էներգիայից կախում ունեցող հիշողությունն է, որը ևս օգտագործվում է միկրոկոնտրոլերներում ժամանակավոր տվյալների պահպանման համար։ Այս տվյալները անհետանում են հոսանքի սնուցման անջատմանը զուգընթաց։

**Հաղորդակցության սարքեր (Peripherals)**

<<Peripherals>> - այս բառը օգտագործվում է ներկայացնելու համար այն բոլոր օժանդակող ֆիզիկական սարքերը, որոնք այս կամ այն կերպ ապահովում են կապը արտաքին աշխարհի հետ։ **Տվյալային փոխարկիչ** – անալոգային տվյալներից փոխարկում թվային տվյալների, **ժամանակի գեներատոր**, **ժամանակի հաշվարկ** (Timing), **անալոգային ազդանշանի մշակում** (Analog signal processing), **մուտք/ելք** (Input/Output), **հաջորդական կապ** (Serial communication) – UART, SPI, l2C, USB։

Տարբեր տեսակի միկրոկոնտրոլերներ օգտագործվում են արդյունաբերության, ավտոմեքենաների, էլեկտրոնային բժշկական սարքերի և այլ շատ ներկառուցված համակարգերում։ Դրանք հիմնականում նախնտրելի են իրենց կոմպակտ չափսերի, բարձր էներգիախնայողության աստիճանի և ցածր գնի համար։ Նախքան անդրադառնալը միկրոկոնտրոլերների առանձնահատկություններին, տեսակներին և արդի օգտագործվող միկրոկոնտրոլերների տարբերություններին անհրաժեշտ է ծանոթանալ դրանց կառուցվածքին և հնարավոր ծրագրավորման միջոցներին։



Նկ. 1.1 Միկրոկոնտրոլերի ֆիզիկական մասերը

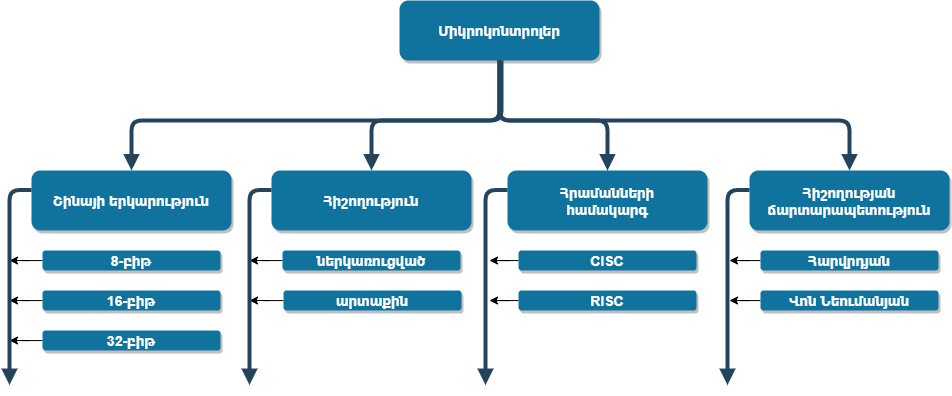
## 1.1 Միկրոկոնտրոլերների տեսակները և առանձնահատկությունները

Միկրոկոնտրոլերի տեսակ ասելով հասկանում ենք այն հրամանի համակարգը որով իրականացվում է տվյալ միկրոպրոցեսորի աշխատանքը։ Այդ տեսանկյունից գոյություն ունեն երեք հիմական հրամանային համակարգեր, որոնք օգտագործվում են միկրոկոնտրոլերներում։

1. CISC – կոմպլեքս հրամանների համակարգ
2. RISC - կրճատված հրամանների համակարգ
3. EPIC – հստակ զուգահեռականությամբ հրամանների համակարգ

Միկրոկոնտրոլերները կարելի է դասակարգել հիմնական մի քանի հատկանիշների հիման վրա։

1. Շինայի երկարության
2. Հիշողության
3. Հրամանների համակարգով
4. Հիշողության ճարտարապետության



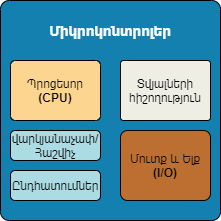
Նկ. 1.2 Միկրոկոնտրոլերների առանձնահատկությունների դասակարգման աղյուսակ

**Դասակարգում ըստ շինայի երկարության**

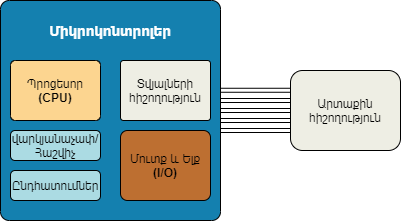
Շինան ՝ վերաբերվում է միկրկոնտրելերում տարբեր բաղադրիչների կապերի զուգահեռ միացմանը։ Այն փոխանցում է հրահանգները և տվյալները պրոցեսորի, հիշողության և տարբեր մուտքի , ելքի պորտերի միջև։

**Դասակարգում ըստ հիշողության**

Տարբերակում ենք ներկառուցված և արտաքին հիշողությամբ միկրոկոնտրոլերներ։  
Ներկառուցված հիշողությունը մնացած ֆունկցիոնալ բլոկերի հետ կազմում են մեկ մարմին , գտնվում են միևնույն չիպի ներսում ի տարբերություն արտաքին հիշողության որը գտնվում է չիպից դուրս և պահանջում է արտաքին միացություններ որպեսզի կապ հաստատի չիպի ներսում գտնվող ֆունկցյոնալ բլոկերի հետ։



Նկ. 1.3 Ներկառուցված հիշողությամբ միկրոկոնտրոլեր



Նկ. 1.4 Արտաքին հիշողությամբ միկրոկոնտրոլեր

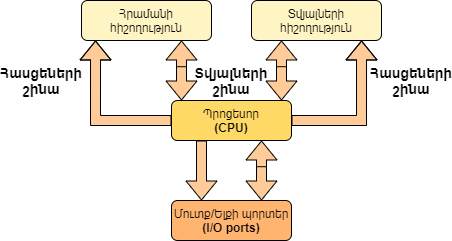
**Դասակարգում ըստ հրամանների համակարգի**

Հրամանների համակագը դա միկրոկոնտրոլերի մաս է որը համակարգում է այն կատարելու հատուկ ֆունկցիաներ։ Այն ներառում է իր մեջ հասցեավորման տեսակները, կարգավորումներ, տվյալների տեսակներ, ռեգիտրներ, ընդհատումներ և արտաքին մուտք/ելք։ Օրինակ՝ read, write, copy/move, add, subtract, multiply, divide, increment, decrement, compare, call to address location և այլն։

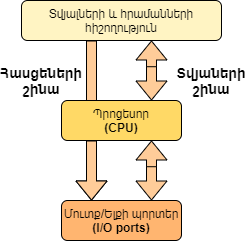
Ըստ հրամանների համակարգի բաժանում ենք հիմնական երկուսի։ CISC (կոմպլեկս հրամանի համակարգով համակարգիչներ) և RISC (կրճատված հրամանի համակարգով համակարգիչներ)։

**Դասակարգում ըստ ճարտարապետության**

Ըստ ճարտրապետության միկրոկոնտրոլերները բաժանում ենք հիմնական երկուսի։ Հարվրդյան ճարտարապետության և Վոն Նեումանյան ճարտարապետության միկրոկոնտրոլերներ։



Նկ. 1.5 Առանձնացված հիշողություն <<Հարվրդյան Ճարտարապետություն>>



Նկ. 1.6 Մեկ ամբողջական հիշողություն <<Վոն Նեումանյան ճարտարապետություն>>

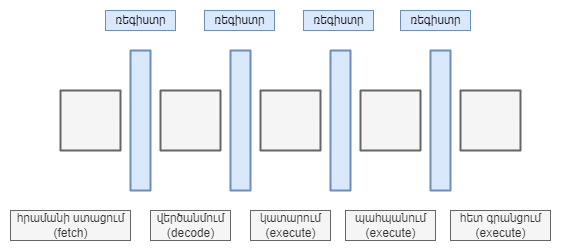
**Հարվրդյան ճարտարապետությամբ** միկրոկոնտրոլերները հիմնված են ֆիզիկապես առանձնացված հիշողության սարքերից ՝ հրամանների և տվյալների պահպանման համար։ Հետևաբար դրանք ունեն նաև առանձնացված ուղիներ շինայի համար և կարող են հասանելիություն ապահովել միաժամանակ։ Կարելի է եզրակացնել, որ Հարվրդյան ճարտարապետությամբ միկրոկոնտրոլերներում կարելի է հրամանը կատարել մեկ մեքենայական ցիկլի ընթացքում։

**Վոն Նեումանյան** ճարտարապետությամբ միկրոկոնտրոլերները ունեն մեկ հիշողության համար առանձնացված ֆիզիկական տարածք, որը նախատեսված է ինչպես հրամանների այնպես էլ տվյալների համար։ Այս ճարտարապետությունը, որի գաղափարը 1945թ․-ին առաջարկվեց մաթեմատիկոս Վոն Նեումանի կողմից մինչ այսօր էլ ամենաշատն է օգտագործվում համակարգչաշինության մեջ։

## 1.2 Բուֆերիզացման մեխանիզմների ուսումնասիրումը

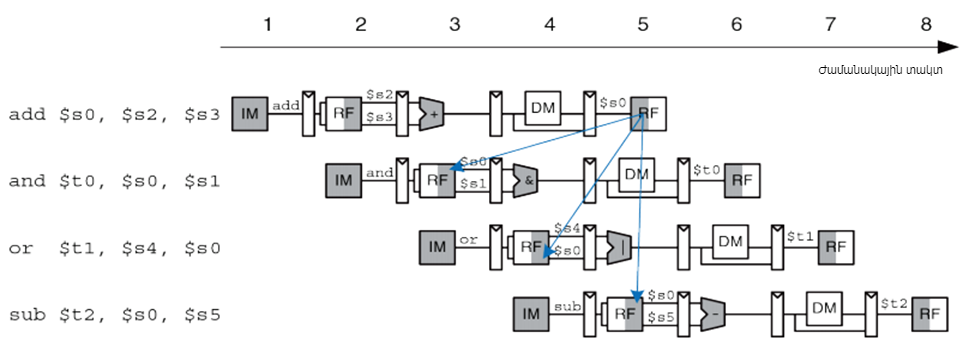
Մագիստրոսական աշխատանքի կատարման հիմքում, որպես բանալի ընկած է բուֆերային հիշողության ստեղծման գաղափարը։ Ժամանակակից համակարգչային տեխնոլոգյաները անհնար է պատկերացնել առանց ռեգիստրների, որոնք կենտրոնական մաթեմատիկական սարքերի աշխատանքի կատարման ընթացքում չեն պահում աշխատանքին անհրաժեշտ տվյալներ։ Առանց բուֆերային հիշողությունների անհնար է պատկերացնել նաև զուգահեռ պրոցեսների կատարման ճարտարապետությունները։

Կոնվերային աշխատանքի կազմակերպումը որպես զուգահեռականության հասնելու այդպիսի մի ճարտարապետություն, որը միտված է մեկ հրամանի ընթացքում զուգահեռացման հասնելու և միարժամանակ յուրաքանչյուր տակտում , ազատ մնացած ֆունկցյոնալ բլոկերին տվյալներ տրամադրելու այլ հրամանների կատարման համար, իր աշխատանքում հասնում է այդ նպատակին օգտագործելով բուֆերներ, որոնք պարզապես տեղակայվում են յուրաքանչյուր ֆունցկյոնալ բլոկերի միջանկյալ հատվածներում։



Նկ. 1.7 Բուֆերների տեղակայումը Վոն Նեումանյան ճարտարապետությամբ պրոցեսորում

Յուրաքանչյուր ֆունկցյոնալ բլոկ, իր գործողության կատարման ավարտին փոխանցում է ստացած տվյալները մոտակա բուֆերային հիշողության մեջ, այդ տվյալներi ՝ պահպանման համար , իսկ միարժամանակ այլ հրաման, որը իր գործողության համար ցանկանում է օգտագործել այդ տվյալները, դիմում է կատարում և ընթերցում է դրանք բոֆերներից ՝ այդպիսով խնայելով ժամանակ և հնարավորություն ստեղծելով յուրաքանչյուր ֆունկցյոնալ բլոկին անկախ աշխատանք կատարելու ՝ ընդհանուր հրամանի կատարման ավարտից։ Այժմ դիտարկենք բուֆերիզացման արդյունքում ստեղծված հրամանների կատարման օպտիմիզացիայի մեկ օրինակ։



Նկ. 1.8 Կոնվերային եղանակով հրամանների կատարումը

Նկար 1․8 – ում պատկերված է 6 հրաման, որոնք կոնվերային եղանակով անցնում են պրոցեսորի յուրաքանչյուր ֆունկցյոնալ բլոկով։ Ուսումնասիրելով պարզ է դառնում որ այս եղանակով՝ երբ յուրաքանչյուր հաջորդ տակտում ընթերցվում է մեկ նոր հրաման, <<and>>, <<or>> և <<sub>> հրամանները կատարում են գործողություններ այնպիսի օպերանդների հետ որոնք դեռևս չեն ստացել իրենց վերջնական արդյունքները, և հետևաբար դրանց աշխատանքը չպետք է թույլատրվի [[8](#table_of_content)]։ Առանց կոնվերային լուծման այս խնդիրը պետք է հախթահարել հապաղման միջոցով, այսինքն <<and>>, <<or>> և <<sub>> հրամանների կատարումը դադարեցնել այնքան ժամանակ քանի դեռ անհրաժեշտ օպերանդները չեն արժեքավորվել։ Հրամանի վերադասավորման արդյունքում, վերոնշյալ օրինակում 4 հրամանների կատարման համար կպահանջվի 17 ժամանակային տակտ (clock cycles) իսկ բուֆերիզացման օգնությամբ, <<and>>, <<or>> և <<sub>> հրամանները կստանան իրենց անհրաժեշտ տվյալները անմիջապես երբ դրանց արդյունքը գրանցվի մոտակա ռեգիստրի մեջ։ Այս կերպ, կոնվերային ճարտարապետությունը օպտիմիզացնում է պրոցեսորի հրամանների հաջորդական կատարումը և արդյունքնում ստանում ենք վերոնշյալ օրինակի համար 8 ժամանակային տակտ, ինչը նշանակում է որ 4 հրամանների համար պրոցեսորի կողմից դրանց կատարման ժամանակը կրճատվել է 47 % -ով։

Բոֆերային այսպիսի մոտեցումները ժամանակակից համակարգիչների ճարտարապետությունների ստեղծման անբաժանելի մասն են կազմում, դրանք իրենց մեջ դասակարգում են հրամանների կատարման հերթականություններ, տալիս որոշակի առաջնահերթություններ։ Որպես օրինակ, կարող ենք մեջ բերել այժմյան օպերացիոն համակարգերի կարևորագույն գործիք հանդիսացող ընդհատումների (interrupts) գաղափարը որը ոչ այլ ինչ է քան բուֆերային հիշողություն՝ ունակ պահպանելու ազդանշաններ, կապված համակարգչի առաջ դրված գործողությունների կատարման համար հաջորդականություններ։

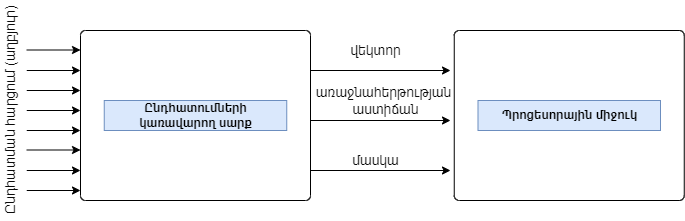
Ընդհատումները ժամանակակից համակարգիչների ճարտարապետություններում իրական օրինակ են հանդիսանում բուֆերային ինտեգրացիայի արդյունավետության։ Ժամանակակից օպերացիոն համակարգերը պրոցեսների վերահսկման, կառավարման գերազանց մշակված մեխանիզմներով են առաջնորդվում։ Այժմ ուսումնասիրենք ընդհատումները, իրենց իրագործման փուլերը։ Գոյություն ունեն երեք ընդհատումների տեսակներ ՝ ապարատային, ծրագրային և մասնավոր։

1. Ապարատային ընդհատումներ։ Սրանք այն ազդանշաներն են որոնք առաջանում են սարքավորումների կողմից (օրինակ պերիֆերիկ սարքերի, հիշողության քարտերի և այլն)։
2. Ծրագրային ընդհատումներ։ Սրանք առաջանում են ծրագրերի կողմից, աշխատանքի ընթացքում երբ վերջինս ցանկանում են ստեղծել պրոցեսորի համար ազդանշան որը իր վրա կվերցնի կատարման համար օպերացիոն համակարգը։
3. Մասնավոր ընդհատումներ։ Սրանք գեներացվում են հենց պրոցեսորի կողմից, նպատակ ունենալով ազդարարել որոշակի խափանման մասին և որի հետևանքով օպերացիոն համակարգի միջամտություն պետք է առաջանա։

Կարևոր գաղափարը բոլոր այս ընդհատումների աշխատանքի մեջ այն, որ պրոցեսորը վերաբերվում է յուրաքանչյուրին նույն կերպ, անկախ թե որտեղից է գեներացվել այդ ընդհատումը։ Այն աշխատում է ռեգիստրների հետ, վերծանում դրանց մեջ գտնվող տվյալները , հասցեները և կատարում անհրաժեշտ գործողությունները։

Այդպիսի մեկ ռեգիստր, որի հետ աշխատանքի մեջ է գտնվում պրոցեսորը, պահպանում է իր մեջ տվյալ պահին կատարվող գործողության հրամանների բլոկի հասցեների ցուցիչները։ Այդ ռեգիստրի տվյալները տեղակայվում են ամեն անգամ, երբ փոխանցվում է նոր հրաման։

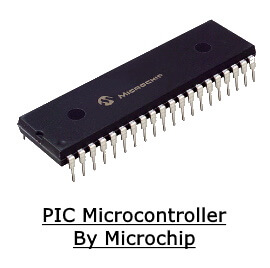
Հաջորդ ռեգիստրը, որի հետ աշխատանքի մեջ է գտնվում պրոցեսորը պահում է իր մեջ ցուցիչ, որը ցույց է տալիս պրոցեսորի ազդանշանների, ընդհատումներ կառավարող աղյուսակների վրա։ Այս ռեգիստրի արժեքը փոխվում է ամեն անգամ երբ օպերացիոն համակարգը բեռնվում է համակարգի միացման սկզբում։



Նկ. 1.8 Ընդհատումների մեխանիզմը պրոցեսորում

Միավոր պրոցեսորը կատարում է ժամանակի յուրաքանչյուր պահին իրեն փոխանցված հրամանի գործողությունները։ Առանց ռեգիստրների, որոնք իրենց մեջ գրանցում են անհրաժեշտ հրամանները իրենց առանահերթություններով, պրոցեսորը կկատարեր իրեն ֆիզիկապես հատկացված ունակությունների հնարավորինս առավելագույնի չափով տվյալների հետ աշխատանք, մեծ կլիներ խափանումների հավանականությունը և յուրաքանչյուր այդպիսի համակարգի օգտագործող պետք է նկատի ունենա, որ իրեն կողմից տրված հրամանի կատարման համար նա պետք է սպասի քանի դեռ պրոցեսորային միջուկը հաջորդաբար կատարելով վերջացնի բոլոր նախորդ հրամանների կատարման գործողությունները, նոր միայն սկսի կատարել օգտագործողի տրված հրամանը։

## 1.3 Ժամանակակից, առավել օգտագործվող միկրոկոնտրոլերները



Նկ. 1.9 PIC միկրոկոնտրոլեր

PIC միկրոկոնտրոլեր։ Այս միկրոկոնտրոլերները օգտագործվում են տրանսպորտային միջոցների, ռոբոտաշինության, բժշկական սարքերի, վաճառքի մեքենաների և շատ այլ ուրիշ ներկառուցված սարքերի մեջ։ Այն հեշտությամբ կարող է ծրագրավորվել Assembly, C, Basic C ծրագրավորման լեզուներով։

8051 միկրոկոնտրոլեր։ Այս միկրոկոնտրոլերը ամենահայտնին է երբևիցե ստեղծված Intel-ի կողմից, այն ունի հրամանի համակարգ առավել հաճախ այս միկրոկոնտրոլերը օգտագործվում է ուսուցման նպատակով տարբեր էլեկտրական սարքավորումներում նախագծվելով հատուկ նշանակության առաջադրանքների համար, օրինակ՝ հակահրդեհային անվտանգության համակարգերում, ջերմաչափման և լուսազգայուն սարքերում։

AVR միկրոկոնտրոլեր։ Այս միկրոկոնտրոլերները ունեն կատարելագործված Հարվրդյան ճարտարապետության հրամանների համակարգեր, որոնք առաջարկում են առանձնացված հիշողություն ծրագրային մասի և տվյալների համար։ Գոյություն ունեն մի շարք միկրոկոնտրոլերների տեսակներ՝ թողարկված մի քանի ընտանիքներով (TinyAVR, MegaAVR, XMEGAAVR, Application-specific-AVR, 32-bit AVR), որոնցից առավել հայտնի է MegaAVR-ը։

ARM միկրոկոնտրոլեր։ Կարելի է գտնել և Վոն Նեումանյան և Հարվրդյան ճարտարապետությամբ միկրոկոնտրոլերներ։Լայնորեն օգտագործվում են էլեկտրոնային սարքերում ինչպիսիք են մոբայլ հեռախոսները, պլանշետները, նվագարկիչները և այլն։ Առավել օգտագործվում են Cortex-M պրոցեսորով միկրոկոնտրոլերները։

RENESAS միկրոկոնտրոլեր։ Այս միկրոկոնտրոլերներին հատուկ է արագագործությունը և բարձր էներգախնայողությունը։ Առավել հայտնի RENESAS միկրոկոնտրոլերները RL78 & RX ընտանիքին պատկանող միկրոկոնտրոլերներն են որոնք ունեն CISC ճարտարապետություն։

**Առավել օգտագործվող միկրոկոնտրոլերների համեմատությունը աղյուսակով**



Նկ. 1.10 <<PIC>>,<< 8051>>,<< ATmega32>>, << ARM Cortex-M0>> միկրոկոնտրոլերների համեմատություն աղյուսակը

## 1.4 Միկրոկոնտրոլերի ընտրման չափորոշիչները

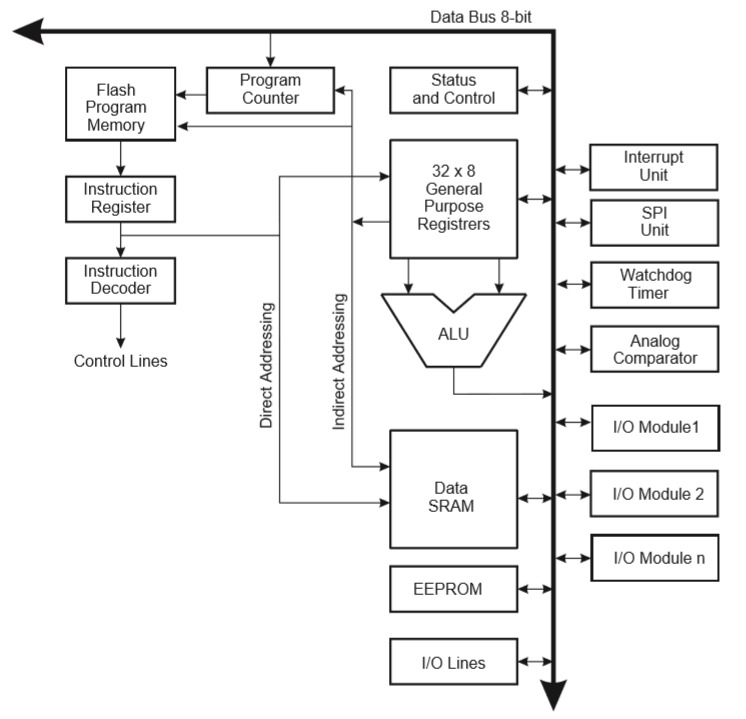
AVR ընտանիքի միկրոկոնտրոլերները կարելի է համարել արդի ամենաօգտագործվողներից և դրանց կարելի է հանդիպել գրեթե բոլոր սարքերի մեջ։ Շնորհիվ RISC հրամանի համակարգի և Հարվրդյան ճարտարապետության ստեղծվել են արագագործության համար նպաստավոր հիմքեր, որոնք իհարկե ավելի բարձր են գնահատվում ժամանակակից տեխնոլոգիական մարտահրավերների հախթահարման համար։ Օգտագործելով համացանցում բաց աղբյուրներից տրամադրվող անվճար ծագրային փաթեթներ կարելի է համարել որ ընտրելով AVR միկրոկոնտրոլեր համատեղում ենք մատչելիությունը արագագործության հետ ինչպես նաև շնորհիվ ինտեգրացված ծրագրերի կարելի է օգտագործել բարձր մակարդակի ծրագրավորման լեզուներ, այդպիսով կոնվերային միկրոկոնտրոլերի ծրագրավորումը դարձնելով էլ ավելի դյուրին։ ARM միկրոկոնտրոլերները էլեկտրոնային շղթաներ են, որոնք կարող են ծրագրավորված լինել ՝ բարդ առաջադրանքներ իրականացնելու համար: Դրանք հանդիպում են էլեկտրոնային սարքերի մեծ մասում, ինչպիսիք են ազդանշանային համակարգերը, համակարգչային կառավարման համակարգերը, հեռախոսները, իրականում գրեթե ցանկացած էլեկտրոնային սարքեր: Գոյություն ունեն ARM միկրոկոնտրոլերների բազմաթիվ տեսակներ։

Այժմ ավելի մանրամասն ուսումնասիրենք AVR ATmega328 միկրոկոնտրոլերը։

**ATmega328 միկրոկոնտրոլերների ուսումնասիրություն**

ATmega328 – ը հանդիսանում է արագագործ միակազմ միկրոկոնտրոլեր, որը թողարկվել է Atmel կազմակերպության կողմից, որպես megaAVR սարքավորումների ընտանիքի անդամ (հետագայում Microchip Technology – ն ձեռք բերեց Atmel - ը): Միկրոկոնտրոլերը ունի մոդերնիզացված 8 բիթանոց AVR RISC միջուկ` հիմնված Հարվարդյան ճարտարապետության վրա: ATmega328 – ն օժտված է 32 ԿԲ(որից 2 ԿԲ – ը օգտագործվում է բեռնիչի կողմից) ISP ֆլեշ (նախատեսված է ծրագրի պահպանման համար) հիշողությամբ` գրանցման ընթացքում ընթերցման հնարավորությամբ, 1 ԿԲ EEPROM (նախատեսված է սարքը` հոսանքի աղբյուրից անջատելուց հետո տվյալների պահպանման համար)` միայն ընթերցման համար նախատեսված և 2 ԿԲ SRAM (նախատեսված է տվյալների (փոփոխականների) պահպանման համար) հիշողություններով, 23 ընդհանուր նշանակության մուտք/ելք – երի հնարավորությունով, 32 ընդհանուր նշանակության աշխատանքային ռեգիստրներով, երեք ճկուն (երկու 8 բիթանոց և մեկ 16 բիթանոց) հաշվիչներով` համեմատման ռեժիմների, ինչպես նաև ներքին և արտաքին ընդհատումների առկայությամբ, ծրագրավորվող USART – ով, բայթային կողմնորոշմամբ 2 շինայական ինտերֆեյսով, SPI սերիական փորթով, 6 ալիքային 10 բիթանոց A/D փոխարկիչով (8 ալիք TQFP և QFN/MLF փաթեթներում), ներքին օսիլատորով և հինգ ընտրովի ծրագրային էներգախնայման ռեժիմներով ծրագրավորվող դիտման հնարավորությամբ ժամանակաչափերով, չունի USB – ի միջոցով հաղորդակցության հնարավորություն, գործողությունների առավելագույն հաճախականությունը սահմանափակվում է 20 ՄՀց – ով, սենսորային կապուղիների քանակը` 16, աշխատանքային հնարավորություն արտաքին ջերմաստիճանի -40 – ից մինչև 85 աստիճան ըստ Ցելսիուսի պայմաններում: Միկրոկոնտրոլերը գործում է լարման 1.8 – 5.5 Վ միջակայքում: Գործողությունների իրականացման 1 MIPS/MHz արագության հետ մեկտեղ, հավասարակշռում է էլեկտրաէներգիայի սպառումը և վերամշակման արագությունը: շ

**Ճարտարապետությունը**։ ATmega328 միկրոկոնտրոլերները հիմնված է Հարվարդյան ճարտարապետության վրա, այսինքն տվյալների և հրամանների պահպանումը իրականացվում է առանձնացված հիշողություններում (համապատասխանաբար SRAM և EEPROM): Հարվարդյան ճարտարապետության հիման վրա ներկայումս թողարկվում են միկրոկոնտրոլերների մեծամասնությունը: ATmega328 միկրոկոնտրոլերի ճարտարապետությունն ունի հետևյալ տեսքը.



Նկ. 1.11 ATmega328 միկրոկոնտրոլերի ճարտարապետություն [[7](#table_of_content)]

Մանրամասն քննարկենք ճարտարապետության յուրաքանչյուր բաղադրիչի առանձնահատկությունները և դերը ճարտարապետության մեջ. **I/O Module 1, 2, ..., n:** Մուտք/ելք - ի մոդուլներ, որոնց քանակը կախված է միկրոկոնտրոլերի տեսակից և պահանջներից: Հանդիսանում է տվյալների փոխանցման «կամուրջ» Arduino Nano հարթակի և ATmega328 միկրոկոնտրոլերի միջև:

**Analog comparator:** Անալոգային կոմպարատորը համեմատող սարք է, որն իր երկու մուտքերում ստանում է երկու անալոգային ազդանշան և ելքում ստանում է բարձր մակարդակի ազդանշան, եթե ոչ շրջադարձային (+) մուտքի ազդանշանը ավելի մեծ է, քան հակադարձ (-) մուտքինը, հակառակ դեպքում` ցածր մակարդակի ազդանշան:

---

Կոմպարատորի ելքային ազդանշանի արժեքը ընդհանուր դեպքում չի սահմանվում, երբ մուտքային լարումներն իրար հավասար են: Սովորաբար, տրամաբանական սխեմաներում բարձր մակարդակի ազդանշանին տրվում է տրամաբանական 1 արժեք, իսկ ցածր մակարդակի ազդանշանին` 0:

**Watchdog Timer:** Հատուկ հսկիչ ժամաչափ է, որը նախատեսված է միկրոկոնտրոլերի աշխատանքի կտրուկ «սառեցման» պայմաններում համակարգի վերբեռնմանը: Ժամաչափի արժեքը պարբերաբար զրոյացվում է ղեկավարվող համակարգի կողմից: Երբ որոշակի ժամանակահատվածում զրոյացում տեղի չի ունենում իրականացվում է համակարգի վերեռնման գործընթաց: Որոշ դեպքերում ժամաչափը կարող է վերբեռնման ազդանշան ուղարկել ղեկավարող համակարգին, մնացած դեպքերում համակարգի վերբեռնումը իրականացվում է ապարատային եղանակով:

**SPI Unit:** Միկրոկոնտրոլերներում տվյալների փոխանցման համար կիրառվող հատուկ շինա, որն ապահովում է տվյալների փոխանցման բարձր արագություն և շահավետություն:

**Interrupt Unit:** Ընդհատման մոդուլ, որը նախատեսված է միկրոկոնտրոլերին շտապ ընդհատման վերաբերյալ տեղեկատվություն փոխանցելու համար: Միկրոկոնտրոլերը իր հերթին պահպանում է իր տվյալ պահի վիճակը և «ընդհատման» ազդանշանի բացակայության պարագայում շարունակում աշխատանքը:

**Data Bus 8-bit:** Տվյալների փոխանցման 8 բիթանոց շինա:

**I/O Lines:** Տվյալների մուտք/ելք - ի գծեր:

**EEPROM:** Ոչ էներգակախյալ հիշողություն, որում տվյալների պահպանումը տեղի է ունենում էլեկտաէներգիայի առկայությունից անկախ:

**Data SRAM:** Տվյալների պահպանման էներգակախյալ օպերատիվ հիշողություն:

**ALU:** Թվաբանական - տրամաբանական սարքավորում (ԹՏՍ), որը նախատեսված է իրեն տրված օպերանդների հետ կատարել թվաբանական կամ տրամաբանական գործողություններ: **32x8 General Purpose Registers:** Օպերանդների և արդյունքի պահպանման համար նախատեսված ռեգիստրներ (32 ռեգիստր, յուրաքանչյուրը 8 բիթ): Ընդհանուր ծավալը կազմում է 32 բայթ:

**Status and Control:** Հանդիսանում է Arduino Nano հարթակից ստացված հրամանների պահպանման ռեգիստր:

**Program Counter:** Միկրոկոնտրոլերում հրամանների կատարման հաջորդականությունը սահմանող ռեգիստր՝ հրամանի հաշվիչ:

**Flash Program Memory:** Ֆլեշ հիշողություն:

**Instruction Register:** Պահպանում է ներկայումս կատարված կամ վերծանված հրամանը:

**Instruction Decoder:** Իրականացնում է հրամանի ապակոդավորում և ուղղում վերջինս համապատասխան հասցեով:

ATmega328 -ը ընտրված է ներկայիս բազմաթիվ Arduino տպասալերի համար հիմնական միկրոկոնտրոլեր։ Գործնականում սա այն առավելությունն է, որը հիմք է տալիս ներկա աշխատանքում օգտագործել Atmega328 միկրոկոնտրոլերը, քանի որ Arduino համակարգերը գերազանցապես հարմարեցված են պարզ, արագ աշխատանքների համար, իրենց հետ տրամադրվում է անվճար IDE, բազմաթիվ սենսորներ և գրադարաններ, որոնք իրենց հերթին ունեն օգտագործման ուղեցույց։

## 1.5 Թեզի կատարման հիմնավորումը

IoT համակարգերի զարգացման հետ կապված, դրա արդիականությունը հաշվի առնելով, միկրոկոնտրոլերի տվյալների մշակման ընթացքում առաջացող մի շարք խնդիրներից տարընթերցումը հախթահարելու համար ուղղված լուծումներն են այս թեզի շրջանակներում ընդգրկված։ Այն աշխատանքներում, որտեղ տվյալների ամբողջականությունը հանդիսանում է կարևորագույն խնդիր, անհնար է դառնում առանց կողմնակի միջամտության ապահովել դա։ Այս թեզի հիմքում ընկած է այդպիսի միջամտություն, որը որպես հավելում IoT համակարգին դարձնում է այն տվյալների կորստի տեսանկյունից առավել հուսալի։

## Առաջին գլխի եզրահանգում

Ուսումնասիրվել են միկրոկոնտրոլերի առանձնահատկությունները, տեսակները, ֆունկցիոնալ բլոկերը։ Մասնավորապես նկարագրվել և այնուհետև համեմատվել են PIC, ARM, AVR, RENESANS, 8051 միկրոկոնտրոլերները, ուսումնասիրվել է բուֆերիզացման իտեգրացիայի արդյունքում որոշ ճարտարապետություններում օպտիմիզացիայի օրինակներ (կոնվերային լուծման հնարավորության ստեղծման, ժամանակակից պրոցեսորների և օպերացիոն համակարգերի հետ առնչվող ընդհատումների գաղափարի ինտեգրացման օրինակներ), ներկայացվել է ARM ATmega328 միկրոկոնտրոլերի ընտրման հիմնավորումը և ուսումնասիրվել է այդ միկրոկոնտրոլերի ճարտարապետությունը, կառուցվածքը, տեխնիկական բնութագրերը։

# ԳԼՈՒԽ 2․ ԱՆՀՐԱԺԵՇՏ ՍԱՐՔԱՎՈՐՈՒՄՆԵՐԻ ԸՆՏՐՈՒԹՅՈՒՆԸ, ԿՈՆՎԵՐԱՅԻՆ ՃԱՐՏԱՐԱՊԵՏՈՒԹՅԱՆ ԿԱԶՄԱԿԵՐՊՄԱՆ ՍԿԶԲՈՒՆՔՆԵՐԸ

Ժամանակակից տպասալերի արտադրության մեջ մեծ տարածում ունեն Arduino տպասալերը, որոնք շատ հարմար են հետազոտական աշխատանքների համար: Կարելի է գտնել բազմաթիվ Arduino տպասալերի տեսակներ, որոնց վրա տեղադրված է AVR ընտանիքին պատկանող միկրոկոնտրոլերներ։ Սովորաբար տպասալի հետ նաև տրամադրվում է Arduino IDE ծրագրավորման միջավայրը անվճար իր հիմնական օգտագործվող գրադարաններով։

Այդպիսի տպասալերից է UNO R3 -ը, որը օգտագործում է Atmega328p միկրոկոնտրոլերը և որի հետ միարժամանակ փաթեթի մեջ ընգրկված են անհրաժեշտ մի շարք սարքավորումներ՝ սենսորներ, լուսադիոդներ, գրադարաններ։ Այժմ ուսումնասիրենք թե ինչ առանձնահատկություններ և ինչ հնարավորություններ է տրամադրում այս տպասալը ։

## 2.1 Միկրոկոնտրոլերի տպասալի ֆիզիկական կոմպոնենտները

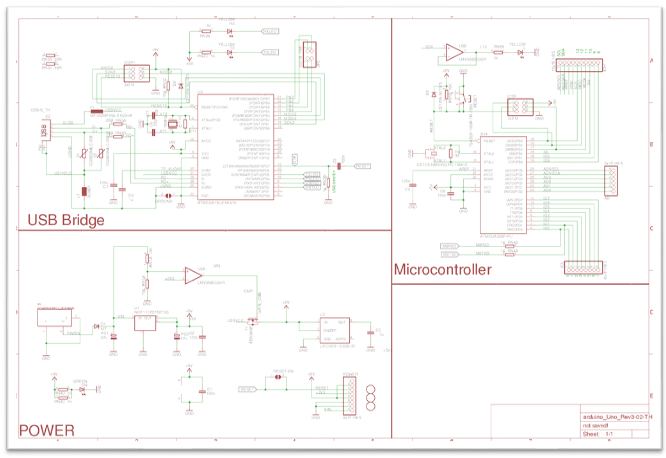
Նախքան Arduino UNO համակարգի hardware -ը հասկանալը մենք պետք է ծանոթանանք ընդհանուր համակարգի նկարագրության հետ։

Arduino IDE միջավայրով կոմպիլացված ծրագրային կոդը այնուհետև ներբեռնվում է միկրոկոնտրոլեր օգտագործելով USB միացումը։ Քանի որ հիմնական միկրոկոնտրոլերը չունի USB ընդունիչ, անհրաժեշտ է որ Arduino UNO համակարգը իր մեջ ապահովի հաղորդակցություն միկրոկոնտրոլերի (UART interface) և հոսթի (USB) միջև։

Այս հաղորդակցությունը Arduino համակարգերի վերջին թողարկումներում ապահովում է ATmega16U2 միկրոկոնտրոլերը, այն հարմարեցված է այնպես որ կատարի փոխարկիչի (bridge) գործառույթներ շնորհիվ ներկառուցված USB ընդունիչի և UART interface -ի։

Arduino UNO համակարգին Էլեկտրական սնուցմամբ ապահովելու համար օգտագործվում է USB հաղորդալարը։ Մեկ այլ եղանակ էլեկտրական սնուցման կարող է լինել DC jack -ը։ Այս երկու տարբերակների համատեղելիությանը կանդրադառնանք <<Էլեկտրական սնուցումը>> ենթաբաժնում։

Ստորև ներկայացված է սխեմատիկ նկարագրությունը Arduino UNO համակարգի։



Նկ. 2.1 Arduino UNO R3 տպասալի ֆիզիկական մասերը

**Միկրոկոնտրոլերը**

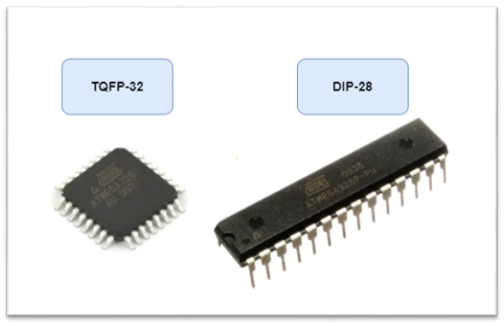
Arduino UNO R3 տպասալի համար որպես հիմնական միկրոկոնտրոլեր օգտագործվում է AVR ընտանիքի ATmega328 միկրոկոնտրոլերը։ Այն 8 -բիթանի է, ինչը նշանակում է, որ նրա տվյալների շինան և ներքին ռեգիստերները հարմարեցված են ընդունել միարժամանակ 8 զուգահեռ տվյալների ազանշաններ։

ATmega328 ունի հետևյալ 3 հիշողության տեսակները։

1. **Flash հիշողություն։** 32կբ ոչ էներգոկախյալ հիշողություն, որը օգտագործվում է ծրագրերի պահպանման համար։ Սա է բացատրում նաև այն, որ համակարգը ամեն անգամ աշխատանքի համար կարիք չունի բեռնելու ծրագիրը երբ որ վերջինս անջատում ենք հոսանքի սնուցման աղբյուրից։
2. **SRAM հիշողություն։** 2կբ էներգոկախյալ հիշողություն։ Այն օգտագործվում է ծրագրի աշխատանքի ժամանակ ՝ ծրագրի կողմից փոփոխականների պահպանման համար։
3. **EEPROM հիշողություն։** 1կբ ոչ էներգոկախյալ հիշողություն։ Այն օգտագործվում է այնպիսի փոփոխականների համար, որոնց գոյությունը պետք է անկախ ծրագրի կատարման ժամանակահատվածից։ Նրանցում պահպանված արժեքները հասանելի կլինեն համակարգը հոսանքի սնուցումից անջատելուց և միացնելուց հետո։

**Կոմպոնենտները**

ATmega328 միկրոկոնտրոլերը հանդես է գալիս տարբեր կոմպոնենտներով , Arduino UNO R3 ում DIP-28 կոմպոնենտն է, ինչը նշանակում է որ առկա են 28 միացումներ (pins)։ Այս միացումները նախատեսված են հոսանքի սնուցման և մուտք/ելքերի (I/O) համար։ Միացումներից շատերը կարող են օգտագործվել տարբեր ռեժիմներով ՝ իրականացնելով տարբեր գործողություններ, շնորհիվ ներքին ֆունկցյոնալ բլոկերի կառուցվածքի։ Մեկ այլ կոմպոնենտ TQFP-32 -ն է, նկարում պատկերված են վերոնշյալ երկու կոմպոնենտները։



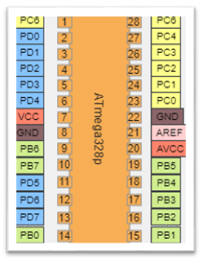
Նկ. 2.2 TQFP-32 և DIP-28 կոմպոնենտները

**Սնուցման աղբյուր**

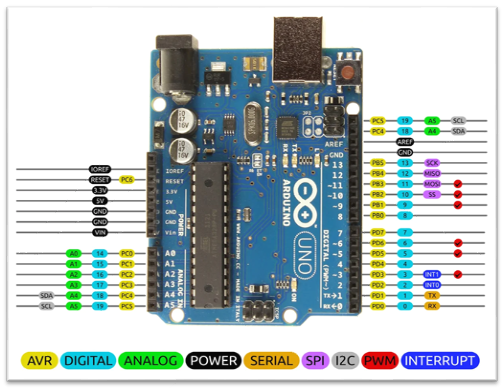
ATmega328p -ն ընդունում է 1․8 վ․ ից 5․5 վ․ հոսանք, որպես սնուցման աղբյուր։ Սակայն պետք է նկատի ունենալ, որ գոյություն ունեն աշխատանքի հաճախականության սահմանափակումներ, որոնք կախված են սնուցման վոլտի չափսից։ Լավագույն աշխատանքի հաճախության կարելի է հասնել 4․5 վ․ հոսանքի սնուցման դեպքում։

**Մուտք/ելքի միացումները**

ATmega328p -ն ունի երեք պորտեր՝ PORTC, PORTB և PORTD։ Բոլոր միացումները այս երեք պորտերի կարող են օգտագործվել որպես ընդհանուր նշանակության թվային մուտք/ելք միացումներ (general-purpose digital I/O) ։ Որոշ միացումներ (pins) կարող են օգտագործվել որպես PWM\* ելքի միացումներ, որոնք ներքոնշյալ միացումների դիագրամում պատկերված են <<~>> տարբերանշանով։



Նկ. 2.3 Atmega328p միկրոկոնտրոլերի միացումները



Նկ. 2.4 Arduino UNO R3 տպասալի միացումները

**ADC մուտքերը**

Միկրոկոնտրոլերը ունի 6 -պորտեր ՝ PORTC0-ից PORTC5 և 10 -բիթանոց A/D փոխարկիչ։   
Այս պորտերը միացված են անալոգային միացման համար առանձնացված մասում ՝ Arduino UNO -տպասալի վրա։ Սակայն այս հատվածից կարող է միկրոկոնտրոլեր փոծանցվել նաև թվային ազդանշան (digital I/O)։ Բերված նկարում պատկերված է բլոկային դիագրամը ATmega328 միկրոկոնտրոլերի որտեղ կարմիր գծերով առանձնացված են A/D -փոխարկիչին առնչվող միացումները, դրանք են ՝

1. **AVCC։** Սնուցման միացում
2. **AREF:**  Ընտրովի տարբերակ, այն դեպքում երբ անհրաժեշտություն է առաջանում ADC -ն հղել արտաքին հոսանքի այլ ոչ թե ներքին Vref -ի։ Այն կարգավորվում է օգտագործելով ներքին ռեգիստրները։

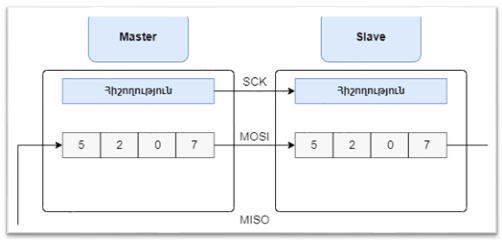
**Ունիվերսալ ասինխրոն հաղորդիչ (UART)**

UART -ը սերյական ինտերֆեյս է և ATmega328 -ն ունի դրանից միայն մեկ մոդուլ։  
RX և TX միացումները կապված են USB-to-UART փոխարկիչին, որը գտնվում է տպասալի վրա և PD0, PD1 թվային մուտքերին։ Պետք է նկատի ունենալ այն հանգամանքը որ չպետք է օգտագործել UART մոդուլը, երբ PD0 և PD1 թվային մուտքերը օգտագործվում են տվյալների փոխանցման կամ ստացման համար։

**Հաջորդական պերիֆերիկ ինտերֆեյս (SPI)**

ATmega328 -ունի մեկ SPI մոդուլ, որը կատարում է Master – Slave փոխհամագործակցություն։ Շնորհիվ այս մոդուլի կարելի է Arduino UNO տպասալը միացնել մի շարք այլ սարքերի և կատարել միարժամանակյա աշխատանք, փոխանցելով անհրաժեշտ տվյալներ սարքերի միջև։ Ասյ միացումները ապահովելու համար օգտագործվում են Arduino UNO տպասալի հետևյալ պորտերը։

1. 11 <-> MOSI (Master – out, Slave - in)
2. 12 <-> MISO (Master – in, Slave - out)
3. 13 <-> SCK (Serial clock)



Նկ. 2.5 Master-Slave փոխհարաբերակցությունը

**I2C կամ TWI**

Այս ինտերֆեյսը բաղկացած է երկու պորտերից, սերիալական տվյալների (serial data) և սերիական հաշվիչի (serial clock) այն է SDA, SCL։ Դրանց կարող ենք հասանելիություն ստանալ երկու տարբերակով։ Թվային մուտքերի վերջին երկու պորտերով և անալոգային մուտքերի առաջին երու պորտերով։

**Այլ ֆունկցիոնալություններ**

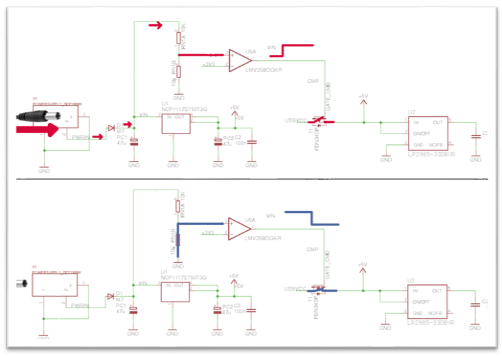
Նկար 1 ում, պատկերված է Arduino UNO R3 տպասալի բաղկացուցիչ մասերը։ Դրանցից մենք արդեն ուսումնասիրել ենք միկրոկոնտրոլերը, նրա ֆունցիոնալությունը և կիրառությունը։ Այժմ ուսումնասիրենք մյուս երկու մասերի ֆունկցիոնալ նկարագրությունները։

**USB-to-UART Bridge**

Այս կոմպոնենտի գործառությունն է ՝փոխակերպել USB ինտերֆեյսից եկած ազդանշանները UART ազդանշանների, նպատակը՝ ազդանշանները դարձնել միկրոկոնտրոլերի համար հասկանալի։ Ինպես արդեն նշվել էր, այս փոխակերպման համար Arduino UNO R3 տպասալում օգտագործվում է մեկ այլ նախկինում AVR ընկերության կողմից ստեղծված միկրոկոնտրոլեր՝ Atmega16U2։ Շնորհիվ նրա մեջ ներբեռնած հատուկ ծրագրային ապահովման (firmware) այն հարմարեցված է կատարելու USB ից UART փոխակերպումը։

**Սնուցման աղբյուր**

Որպես սնուցման աղբյուր կարող է ծառայել USB ինտերֆեյսը կամ DC jack -ը: Այս բլոկը կառուցված է այնպես, որ ցանկացած պահի սնուցման աղբյուր ընտրելիս Arduino UNO R3 տպասալը գտնում է հնարավոր սնուցման աղբյուրները և ընտրում դրանցից մեկը ավտոմատ կերպով։



Նկ. 2.6 Arduino UNO R3 տպասալի սնուցման երկու եղանակները

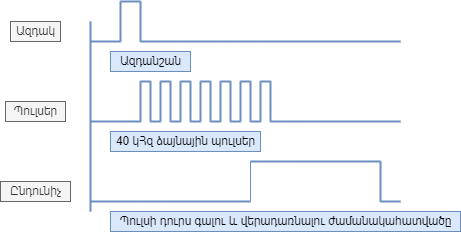
Այս նկարում պատկերված է հոսանքի սնուման աղբյուրի ընտրման, փոխարինման մեխանիզմը։ DC jack ով միացումը ծածկում է USB ով միացմանը։ Դրանց միջև ընտրությունը պետք է կատարել կախված այն հանգամանքից թե ինչ ծրագիր ենք աշխատացնում տպասալով, քանի որ դրանք օժտված են տարբեր միլիամպերներով։

## **2.2 Սենսորների և շիֆթ ռեգիստրների ճարտարապետության ընտրությունը**

Ներկայացված աշխատանքում կիրառվում են երեք սենսորներ ։

1. HC-SR04 ուլտրաձայնային սենսոր
2. MQ-2 գազի սենսոր
3. DHT11 ջերմաստիճանի և խոնավության սենսոր։

**HC-SR04 ուլտրաձայնային սենսոր**։ Այս սենսոր հեռավորության չափման համար է նախատեսված։ Այն արձակում է 40 kHz հաճախականությամբ ութ ուլտրաձայն, որը տարածության մեջ հանդիպելով պինդ մարմինների անդրադարձվում է հետ դեպի սենսոր։



Նկ. 2.7 HC-SR04 սենսորի ազդանշանների գծապատկերը

Այդ պուլսերը հատուկ ազդակի դեպքում արձակվում են իրար հետևից ութ անգամ իսկ հաջորդիվ կատարվում է ժամանակի և ազդանշանի վերադարձի գրանցումները, որից հետո արդեն հաշվարկվում է տարածությունը սենսորի և օբյեկտի միջև։



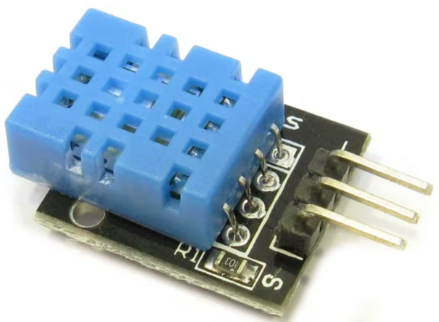
Նկ. 2.8 HC-SR04 սենսոր

**MQ-2 գազի սենսոր**։ Գազի այս սենսորը ունի պոտենցիոմետր որը հնարավորություն է տալիս փոփոխելու սենսորի զգայունությունը։ Այն նախատեսված է մի քանի այրվող գազերի հայտնաբերման համար որոնցից են բութան, պրոպան, մեթան, հիդրոգեն ինչպես նաև ալկոհոլ։ Բարձր ելքային լարմամբ ազդանշան է արձակում գազի առկայության դեպքում, ցածր ազդանշան է արձակում գազի բացակայության դեպքում։



Նկ. 2.9 MQ-2 գազի սենսոր

**DHT11 ջերմաստիճանի և խոնավության սենսոր** ։ Այս սենսորը հայտնաբերում է ջրի գոլորշիներ օդում ՝չափելով դիմադրությունը երկու էլեկտրոդների միջև։ Այն ներկառուցված ունի իր մեջ ջերմաստիճանի սենսոր, որը կատարում է ջերմաչափի գործառույթներ։



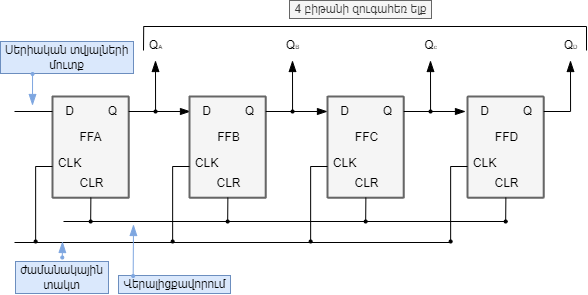
Նկ. 2.10 DHT11 ջերմաստիճանի և խոնավության սենսոր

**Շիֆթ ռեգիստրներների ընտրությունը**

Շիֆթ ռեգիստրները հանդիսանում տրամաբանական էլեկտրական շղթաներ, որոնք կարողեն օգտագործվել որպես հիշողություններ կամ որպես բինար տվյալների փոխանցման համար սարքավորումներ։ Այդպիսի սերիական սարքերը բեռնում են իրենց մուտքերում առկա տվյալները, այնուհետև դրանք տեղափոխում են իրենց ելքերի վրա մեկ ժամանակային տակտում, որի համար էլ անվնանվում են <<Shift Registers>>: Շիֆթ ռեգիստրը հիմնականում բաղկացած է մի քանի մեկ բիթանոց D - տիպի ֆլիպ-ֆլոպ ռեգիստրներից, որոնցից յուրաքանչյուրը պահպանում է կամ տրամաբանական <<0>> կամ <<1>>, որոնք միացված են իրար սերիական միացումով այնպես, որ մեկ ռեգիստրի ելքը դառնում է հաջորդ ռեգիստրի մուտքը և այլն: Շիֆթ ռեգիստրները հիմնականում նախատեսում են <<SET>>, <<RESET>> հնարավորություններով, որոնք հնարավորություն են ստեղծում անհրաժեշտ պահին վերալիցքավորելու կամ դատարկելու ռեգիստրները իրենց պարունակություններից։ Ընդհանուր առմամբ շիֆթ ռեգիստրները օգտագործվում են տվյալների տեղափոխման համար ՝չորս հիմնական ռեժիմներով։

1. Սերիական մուտք, սերիական ելք (SISO):
2. Սերիական մուտք, զուգահեռ ելք (SIPO)։
3. Զուգահեռ մուտք, սերիական ելք (PISO)։
4. Զուգահեռ մուտք, զուգահեռ ելք (PIPO)։

SISO – շիֆթ ռեգիստրները, ստանում են մեկ հոսքով հաջորդական տվյալներ, փոխանցում են հաջորդաբար միացված ռեգիստրների միջով այդ տվյալները ՝սինխրոնիզացված ժամանակային տակտերով։ SIPO – ռեգիստրները, կատարում են տվյալների փոխանցում նույն եղանակով ՝ սերիական տվյալների ժամանակային տակտերով փոխանցումներ, սակայն դրան յուրաքանչյուր ռեգիստրի ելքերից դուրս են գալիս միացումներ։

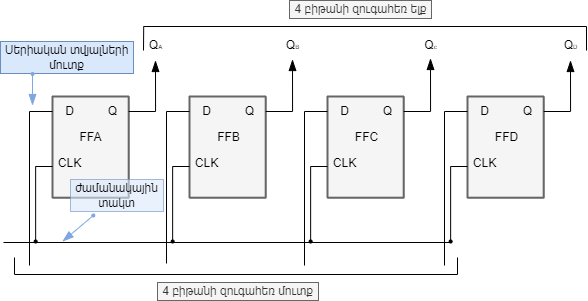


Նկ. 2.11 4 – բիթանի շիֆթ ռեգիստր։ SIPO ճարտարապետություն

Ժամանակային տակտը միացված է չորս ֆլիպ – ֆլոպների CLK (clock) – մուտքերին, իսկ վերալիցքավորման միացումը կապված է CLR (clear) – մուտքերին։ QA , QB , QC և QD ն զուգահեռ ելքերն են, որոնք յուրաքանչյուր ժամանակային տակտում (clock) տալիս են տվյալներ՝ <<0>> կամ <<1>> ։

PISO – շիֆթ ռեգիստրները կատարում են հակառակ գործողությունը, դրանց մուտքերը նորմալ ռեժիմով, յուրաքանչյուր ռեգիստրին տրամադրում են տվյալներ զուգահեռ ռեժիմով։ Այնուհետև, ռեժիմի փոփոխմամբ կատարվում է հաջորդական տվյալների ռեգիստրներից փոխանցում և տվյալները սինխրոնիզացված դուրս են գալիս վերջին ռեգիստրի ելքից։

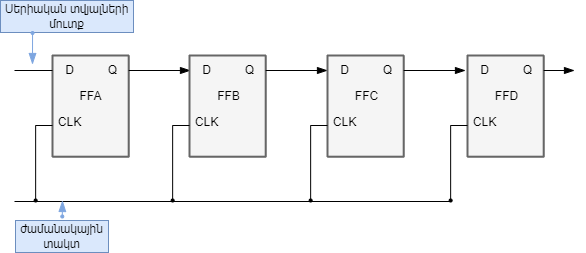
PIPO – շիֆթ ռեգիստրները չունեն սերիական տվյալների փոխանցման ռեժիմ, դրանք նախատեսված են միմիայն տվյալների զուգահեռ ռեժիմով աշխատանքի համար։



Նկ. 2.12 4 – բիթանի շիֆթ ռեգիստր։ PIPO ճարտարապետություն

Դրանք ունեն միայն երեք միացություններ ՝ զուգահեռ մուտք (PI), զուգահեռ ելք (PO) և ժամանակային տակտ (Clk)։ Կարելի է նմանեցնել այս շիֆթ ռեգիստրի աշխատանքը SISO շիֆթ ռեգիստրի աշխատանքի հետ այն առումով, դրանք պահպանում են տվյալներ կախված ժամանակային տակտերի հաճախականությունից։ Կարելի է նկատել որ դրանք չունեն ռեգիստրների հաղորդակցություններ, քանի որ սերիական տվյալների փոխանակություն այստեղ տեղի չի ունենում։

Շիֆթ ռեգիստրների ճարտարապետությունները նախատեսված են հատուկ նշանակության խնդիրներ լուծելու համար։ Ներկա աշխատանքում կոնվերային եղանակով տվյալների փոխանցումը իրականացնելու համար մեզ անհրաժեշտ է շիֆթ ռեգիստրի ընտրությունը կատարենք ոչ միայն դրա տվյալների հաջորդաբար կամ զուգահեռ փոխանցման հատկանիշով պայմանավորված այլ նաև ընտրենք այնպիսի ճարտարապետույթուն, որը հնարավորություն կընձեռնի ժամանակավորապես պահպահենլու տվյալները և մինչ դրանք վերջնակետ փոխանցելը միարժամանակ ընթերցի հաջորդ տվյալները։ Այսինքն այս աշխատանքում անհրաժեշտ են շիֆթ ռեգիստրներ, որոնք ունակ կլինեն նաև որպես հիշողություն ծառայել։ Նկատի ունենալով այս հանգամանքը մեզ մոտ ընտրության երկու հնարավորություն է մնում , սակայն եթե հաշվի առնենք, որ սերիական ձևով տվյալների փոխանցումը ամբողջությամբ բավարարում է խնդրի պահանջներին ապա նախապատվությունը կտանք SISO շիֆթ ռեգիստրներին։



Նկ. 2.13 4 – բիթանի շիֆթ ռեգիստր։ SISO ճարտարապետություն

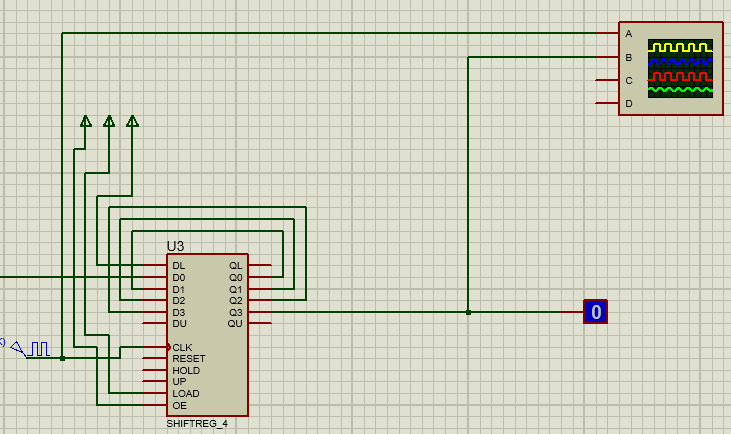
SISO – շիֆթ ռեգիստրները ունեն երեք կոնֆիգուրացյա սերյական մուտք (SI), սերյական ելք (SO) և ժամանակային տակտ (Clk):

## 2.3 Կոնվերային ճարտարապետության սկզբունքները, իրականացումը IoT համակարգում

Անթիվ IoT սարքավորումներ միացված են իրար գլոբալ թվային ենթակառուցվածքներում, որոնք իրար հետ կապ են հաստատում ցանցերի միջոցով ինչպես նաև ամպային տեխնոլոգյաներով, կիսում են տվյալներ, ստանում են և գրանցում են դրանք՝ կատարում հաշվարկներ և տարբեր գործառույթներ միարժամանակ։ Շրջակա միջավայրի պարամետրիկ ճշգրիտ գնահատումը՝ իրական ժամանակում, հնարավորություն է տալիս կատարել ճիշտ որոշումներ, որոնք երբեմն կարող են լինել աննախադեպ կարևոր։ Ահա այս պարամետրիկ գնահատումը այժմ բախվում է մի շարք խնդիրների հետ, կապված տվյալների սինխրոնիզացիայի և ցանցերում գերբեռնվածության հետ, որոնք առաջանում են սենսորների ակտիվ աշխատանքի հետ։ Վերոնշյալ խնդիրները արդի այս համակարգերում առաջացնում են տվյալների տարընթերցման թե ստացողի համար և թե այդ տվյալները ստեղծող կողմի համար։ Այս աշխատանքում ներկայացված է IoT համակարգ, որի վրա իրականացված է կոնվերային ճարտարապետությամբ, սինխրոնիզացված տվյալների ընթերցում անմիջապես միկրոկոնտրոլերից և փոխանցում դեպի վերջնակետ որը կարող է հանդիսանալ IoT համակարգի մաս կամ այդ ցանցին միացված կառավարող որևիցե կենտրոն։ Այս մոտեցմամբ տվալները որոնք ձեռք են բերվում շրջակա միջավայրի պարամետրիկ գնահատման արդյունքում ընթերցվում և փոխանցվում են սինխրոնիզացված և կոնվերայի եղանակով, նպատակ ունենալով հախթահարելու հնարավոր տարընթերցման հետ կապված խնդիրները։

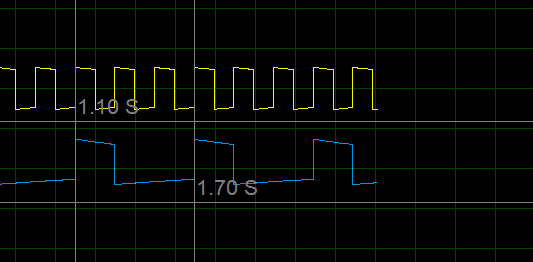
Բոլորիս տանը այսօր կարող ենք գտնել սենսորներ էլեկտրականության խափանման, գազի արտահոսքի հայտնաբերման համար։ Սա շրջակա միջավայրի պարամետրական գնահատման պարզագույն օրինակներից մեկն է, որը առկա գրեթե բոլորի տանը։ Դրանց պարամետրիկ ճշտգրիտ և ժամանակին գնահատումը իհարկե լուծում է անվտանգային կարևորագույն խնդիրներ։ Սակայն այն խնդիրը, որը կարող է առաջանալ տարընթերցման և ոչ սինխրոն աշխատանքի հետ մենք բարեբախտաբար չենք կարող այս համակարգի մեջ տեսնել, քանի որ յուրաքանչյուր սենսոր ստանում է և փոխանցում է տվյալները մեկ մշակող միավորի, որը կարող է լինել ինչ-որ արտադրողի կողմից ստեղված միկրոկոնտրոլեր։ Բայց ինչ կարող է լինել եթե տվյալ միջավայրում սենսորները լինեն ոչ թե երկու կամ երեք հատ այլ լինեն հազարավոր։ Բնականաբար բոլոր սենսորների տվյալները մշակելու և որոշակի գործողություններ կատարելու համար նպատակահարմար չէ օգտագործել յուրաքանչյուրի համար առանձին միկրոկոնտրոլեր, ինչը նաև գումարային տեսանկյունից տրամաբանական չէ։ Ահա սա այն դեպքն է երբ մենք կտեսնենք տվյալների տարընթերցման և սինխրոնիզացման հետ կապված խնդրիներ։ Սենսորների իրական ժամանակում աշխատանքը ուղեկցվում է նրանց կողմից անդադար տվյալների փոխանցմամբ և ընդունող կողմի անդադար տվյալների ընթերցման։ Քանի որ տարբեր արտադրողների կողմից տրամադրվող սենսորներ, նրանց աշխատանքի բնույթից, նրանց և միկրոկոնտրոլերի հեռավորությունից կախված, ինչպես նաև աշխատանքի հաճախականությունից լինում են տարբեր։ Այնպես որ կանխատեսել դրանց հաջորդական աշխատանքը և կանոնավոր, սինխրոն տվյալների փոխանցում և/կամ ընդունումը հնարավոր չէ։ Այս աշխատանքում կներկայացվի կոնվերային աշխատանքի սկզբունքով պայմանավորված լուծում, որը միտված կլինի հախթահարելու այս պահին առկա խնդիրը IoT համակարգերում։

Նախքան կոնվերային ճարտարապետության առավելություններից օգտվելը նախ և առաջ անհրաժեշտ է իրական պատկերի վրա ուսումնասիրել թե ինչպես է IoT համակարգը դրսևորում իրեն առանց որևիցե լուծման ինտեգրացիայի։ Եթե որպես օրինակ վերցնենք մեկ սենսորից ստացած տվյալի փոխանցումը ապա կնկատենք, որ յուրաքանչյուր ժամանակային տակտում, միկրոկոնտրոլերից փոխանցվում է մեկ տվյալ։



Նկ. 2.14 4 – բիթանի շիֆթ ռեգիստր և թվային օսցիլոսկոպ

Նկարում պատկերված շիֆթ ռեգիստրի <<D0>> - մուտքին միացված է միկրոկոնտրոլերի տպասալից ելնող թվային ազդանշանը, որը փաստում է սենսորներից մեկի աշխատանքի մասին։ Շիֆթ ռեգիստրը չորս բիթանի է, նկարում պատկերված ձևով միացությունների կազմակերպմամբ այն համապատասխանեցվում է չորս ՝ մեկ բիթանի ռեգիստրների հաջորդական դասավորված աշխատանքին։ Այն կարող է պահպանել իր մեջ չորս բիթ հիշողությամբ տվյալ, այն սինխրոնիզացված է ժամանակային տակտով և հետազոտման նպատակով <<Q3>> - ելքը միացված է թվային օսցիլոսկոպի <<B>> - մուտքին։ Թվային օսցիլոսկոպի <<A>> - մուտքին է նաև միացված <<Clk>> ազդանշանը:

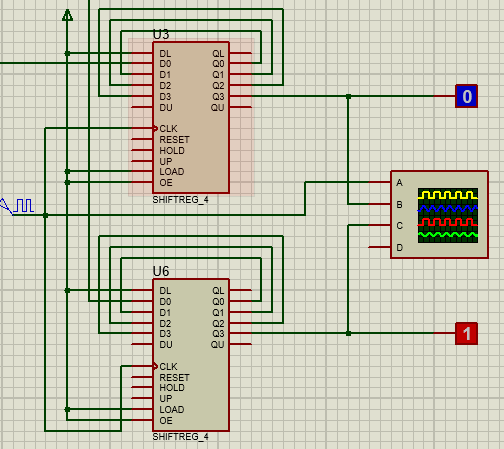


Նկ. 2.15 Clk և առաջին սենսորի ազդանշանները

Աշխատանքում բոլոր պարամետրերը փոփոխելի են, համեմատությունները կատարվել են միկրոկոնտրոլերից ազդանշանի փոխանցման տևողության 100, ազդանշանների միջև 400 – միլիվարկյան, clock ազդանշանի 4 – հերց հաճախականության պայմաներում։ Clk ազդանշանի դրական ճակատով տեղի է ունենում տվյալի փոխանցումը, հաջորդ դրական ճակատի դեպքում ավարտվում է տվյալի փոխանցումը և այդ վիճակում գտնվում մեկ ամբողջական ժամանակային տակտ։

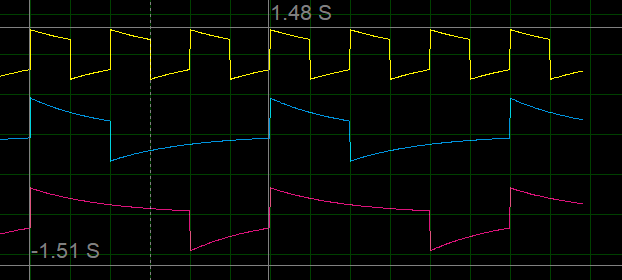
Նման կոնֆիգուրացյայով կարելի է փաստել որ ընդունող կողմը կարող է ստանալ տվյալները յուրաքանչյուր 3 տակտ տևողությամբ ժամանակահատվածներով։ Հարկ է նշել, որ կոնֆիգուրացյայի փոփոխման դեպքում տվյալների փոխանցումը ցանկալի է կատարել 1 տակտի ընթացքում, քանի որ դրան խոչնդոտող որևիցե աարգելք չկա։ Իհարկե նախորոք նկատի ունենալով միմիանց հետ աշխատանքի մեջ գտնվող սարքերի թույլատրելի թողունակության չափերը և վերցնել այն հաճախականությունը ամբողջ համակարգի համար, որը կլինի նույն համակարգի մեջ գտնվող սարքերից ամենաքիչ թողունակություն ունեցողը։

Այս օրինակը պատկերում է մեկ սենսորի աշխատանքի հետ կապակցված ազդանշանի փոխանցում։ Այժմ պատկերենք այնպիսի IoT համակարգ, որը ունի մեկից ավել սենսորներ, և փոխանցումը կատարվում է երկու զուգահեռ միացումներով։



Նկ. 2.16 4 – երկու շիֆթ ռեգիստրներ և թվային օսցիլոսկոպ

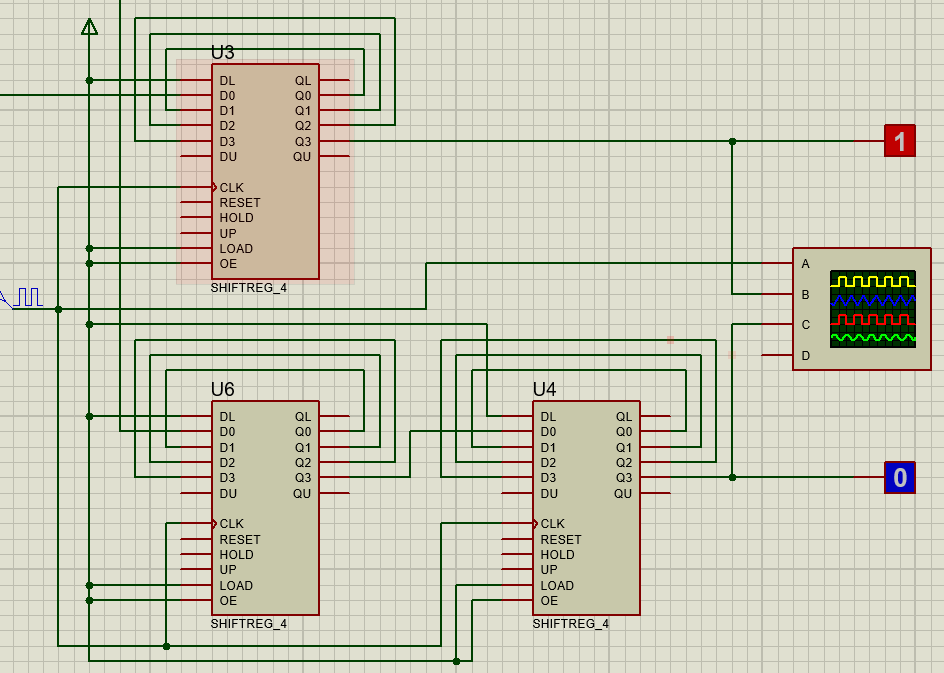
Տեղադրված են երկու 4 բիթ հիշողությամբ շիֆթ ռեգիստրներ, որոնք կապակցված են յուրաքանչյուր սենսորի աշխատանքին։ Այժմ միկրոկոնտրոլերը հաջորդական կերպով պետք է ընթերցի յուրաքանչյուր սենսոր՝ իր տեխնիկական հնարավորությունների չափով, ինպես նաև սենսորներից յուրաքանչյուրի աշխատանքի բնույթից նրա աշխատանքի ժամանակը նույպես կախված կլինի։ Որպես օրինակ, եթե վերցնենք ուլտրաձայնային, տարածության մեջ հեռավորություններ որոշող HC-SR04 սենսորի աշխատանքը , ապա այն միանշանակ ավելի երկար ժամանակ է պահանջելու համեմատած <<շոկ>> սենսորի, որը նախատեսված է բախումներ հայտնաբերելու համար։



Նկ. 2.17 երկրորդ սենսորի աշխատանքը պահանջում է ավելի երկար   
Clk տակտեր մեկ անգամ տվյալ ընթերցելու համար։

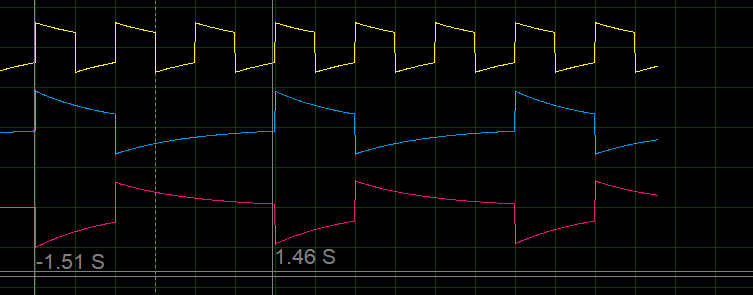
Ինչպես երևում է ժամանակային այս տարածությունը բավարար չէ անզեն աչքով տեսնելու այն փոքր միջակայքը, որը կփաստեր հաջորդական միկրոկոնտրոլերի կողմից զդանշանների փոխանցումը։ Եվ ինչպես երևում է, երկրորդ սենսորի աշխատանքի պարբերությունը ավելի մեծ է քան առաջին սենսորի աշխատանքինը, սակայն կարելի է ասել, որ երեք ժամանակային տակտը ամբողջովին բավարարում է երկու սենսորի աշխատանքի համար։ Այս եղանակով մենք տեսնում ենք, որ տվյալները գրեթե միարժամանակ են փոխանցվում առանձին միացումներով և ընդունող կողմը պետք է մշակի և կիրառի հատուկ ռազմավարություն, որպեսզի ապահովի առանձին սենսորների տվյալների ստացումը։ Կարելի է դիտարկել յուրաքանչյուր տակտ մեկ սենսորի տվյալներ ստանալու համար ինչպես նաև կարելի է ապահովել յուրաքանչյուր սենսորին մեկ միացում ՝ ընդունող կողմում, որը բնականաբար շահավետ լուծում չի կարող հանդիսանալ, եթե դիտարկենք միացվող սենսորների քանակի բազմապատկում։

Այստեղ կարելի է կիրառել կոնվերային լուծում, որը ընդունող կողմի համար կլինի պարզ և չի պահանջի միացումների քանակի աճ և հետևաբաև ծախսեր։ Շիֆթ ռեգիստրների այս դասավորվածության, միացումների տրամաանության մեջ որևիցե բան չի փոխվում, կոնվերային ճարտարապետության իրականացումը հնարավորինս պարզ է և պահանջում է մեկ հավելյալ 4 բիթային հիշողությամբ ռեգիստր։



Նկ. 2.18 4 – բիթանի 3 շիֆթ ռեգիստրներ կոնվերային միացմամբ

U4 – անվամբ ռեգիստրը տեղակայվեց թվով երկրորդ սենսորի աշխատանքի հետ կապված U5 – ռեգիստրի կողքին, ստանալով դրա <<Q3>> - ելքից փոխանցվող թվային ազդանշանը իր <<D0>> - մուտքին։ U4 – ռեգիստրը իր մուտքին առաջին անգամ տվյալ կարող է ստանալ 4 ժամանակային տակտ անց, երբ իրենից նախորդ U6 – ռեգիստրը ավարտի առաջին տվյալի փոխանցումը։



Նկ. 2.19 Կոնվերային կերպով ազդանշանների ընթերցում

Ինչպես պատկերված է նկարում, այժմ միևնույն 3 տակտ ժամանակահատվածում ընթերցվում է երկու ազդանշան, սակայն այժմ դրանք կատարվում են Clk – ազդանշանի դրական ճակատով տարբեր տակտերում։ Այս կերպ, ընդունող կողմը, սենսորների կատարած աշխատանքի մասին կարող է իմանալ մեկ միացումով՝ կիրառելով մուլտիպլեքսոր կամ PISO - շիֆթ ռեգիստր, գրանցելու համար այն սենսորի հասցեն, որը տվյալ պահին, անկասկած արդեն կատարել է իր գործողությունները և այժմ կարող է տրամադրել իր գործողությունների արդյունքում ստացած տվյալները։ Հետազոտության գործիք հանդիսացող թվային օսցիլոսկոպի հնարավորություններից ելնելով (սպասարկում է չորս մուտքային միացումներ) այս աշխատանքում կավելացնենք ևս մեկ սենսորի աշխատանքի հետ կապված ազդանշան և այն կտեղավորենք 3 տակտում։ Կարելի է փաստել, որ փոփոխելով ժամանակային տակտի հաճախականությունները կարող ենք միացումներ ապահովել բազմաթիվ սենսորների համար և այնպես կազմակերպել, որ դրանցից յուրաքանչյուրը ժամանակային տվյալ տակտում ընթերցվի առանձին։

## Երկրորդ գլխի եզրահանգում

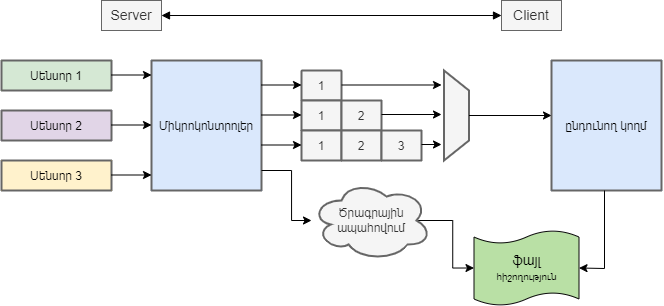
Այս գլխում ներկայացված են աշխատանքում կիրառվող սենսորների աշխատանքային բնութագրերը, տեխնիկական առանձնահատկությունները որոնք անհրաժեշտ պարամետրիկ միջավայրի գնահատման համար և հանդիսանում են տվյալների աղբյուրներ միկրոկոնտրոլերի համար։ Մասնավորապես ընտրվել է երեք սենսոր ՝ HC-SR04 ուլտրաձայնային, MQ-2 գազի հայտնաբերման և DHT11 խոնավության և ջերմաստիճանի սենսորներ։ Ներկայացվել է դրանց աշխատանքի սկզբունքները և անհրաժեշտ պայմանները։ Այնուհետև կատարվել է ընտրություն շիֆթ ռեգիստրների ճարտարապետության՝ ելնելով խնդրի պայմաններից։ Գլխի վերջնամասում ներկայացվել է կոնվերային ճարտարապետության ինտեգրման սկզունքները, դրա համար ահրաժեշտ գործիքակազմը և իրականացվել է կոնվերային կերպով սենսորներից կախված ազդանշանների փոխանցում։

# **ԳԼՈՒԽ 3. ՍԱՐՔԱՎՈՐՈՒՄՆԵՐԻ ՄԻԱՑՈՒՄԸ ԵՎ IOT ՀԱՄԱԿԱՐԳԻ ԻՐԱԳՈՐԾՈՒՄԸ**

Նախքան սարքավորումների աշխատանքի համար անհրաժեշտ միացումները հաստատելը անցնենք ընդհանուր IoT համակարգի ճարտարապետությանը, ընդհանրացված ֆունցկյոնալ բլոկերի նշանակություններին։ Այնուհետև կծանոթանանք համակարգի տարերին ՝ սենսորներին, ռեգիստրներին, միկրոկոնտրոլերի և մոնիտորինգի համար նախատեսված սարքավորումների միացմանը։ Այժմ նկարագրենք IoT համակարգի շղթան։

## 3.1 IoT համակարգի շղթայի նկարագիրը

IoT համակարգը բաղկացած է շրջակա միջավայրի պարամետրիկ գնահատման համար նախատեսված սենսորներից, այդ սենսորների տվյալները ստացող միկրոկոնտրոլերի հենքով տպասալից, շիֆթ ռեգիստրներից և տվյալները ստացող ընդունող կողմից։ Համակարգի ճարտարապետության կառուցվածքը ունի հետևյալ տեսքը։



Նկ. 3․1 IoT համակարգի ընդհանուր կառուցվածքը

<<Client – Server>> մոդելավորմամբ, ընդունող կողմը, որը ստանալու է սենսորների աշխատանքների կատարման վերաբերյալ ազդանշաններ և կարող է դիմել համապատասխան այդ ազդանշանների հիշողությանը ՝ տվյալները ստանալու համար, հանդիսանում է Client: Սենսորների աշխատանքը կառավարող միկրոկոնտրոլերը, որը ստանալու է հնարավոր բոլոր իրեն միացված սենսորներից տվյալները և ուղարկելու է համապատասխան թվային ազդանշաններ դրանց աշխատանքի ընթացքում համապատասխանում է Server - ին։ Server –ի պարտավորությունների մեջ են նաև գտնվում սենսորներից ստացած տվյալների փոխանցումը , պահպանումը հիշողության մեջ, որից ընթերցելու հասանելիություն պետք է ունենա Client - կողմը։ Սենսորների տվյալները կարդալու և գրանցելու համար օգտագործվող միկրոկոնտրոլերը նախատեսվում է միայն այդ աշխատանքի համար, քանի որ ցանկացած այլ նպատակով որևիցէ գործողություն կատարելու համար այն կդաթարեր միևնույն հաճախականությամբ սենսորներից ընթերցել տվյալներ և այդպիսով հիմնական այն առաքելությունը, որը դրվում է Server – միկրոկոնտրոլերի վրա կխափանվի։ Իտարբերություն Server – միկրոկոնտրոլերի, Client կողմում ընդունող կողմը պետք է նշանակված լինի միայն առկա տվյալների մշակման համար։ Որպես օրինակ, եթե Server – միկրոկոնտրոլերը ընթերցել և գրանցել է գազի սենսորի տվյալները, որոնք փաստում են օդում գազի առկայության մասին, ապա դրանք հայտնաբերվում են և համապատասխան գործողությունները իրականացվում են միայն Client – կողմում։ Եթե ընդունող կողմը հանդիսանում է IoT համակարգի մաս՝ միկրոկոնտրոլեր ինպես և Server հատվածում, ապա ընդհանուր տվյալների մշակման համար կունենանք բարձր արագագործություն։ Ընդունող կողմը ստանում է թիվ, որը նույնականացված է հատուկ սենսորի և հիշողության հատուկ հատվածի հետ։ Այդ թիվը տասական համակարգով այն 2n – աստիճան թիվն է, որի n – ը համապատասխանում է սենսորներից մեկի համարի հետ։ Ենթադրենք ժամանակի տվյալ տակտում, հաջորդական ռեգիստրներից դուրս եկող թվային ազդանշանները ունեն հետևյալ տեսքը՝ 010, սա նշանակում է որ երկրորդ համարով սենսորը աշխատանք է կատարել և որոշակի տվյալներ գրանցվել են հիշողության մեջ։ Համակարգի շղթայի մեջ պատկերված <<Ֆայլ, հիշողությունը>> այն ֆիզիկական տիրույթն է, որի մեջ թուլատրված կլինի գրանցելու գործողությունները Server կողմի համար և ընթերցելու հնարավորություններ Client կողմի համար:

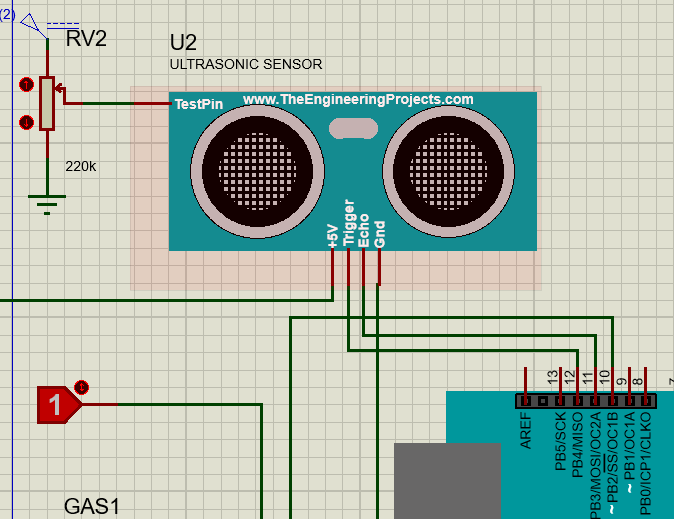
## 3.2. Համակարգի բաղկացուցիչ մասերի կարգավորումները և միակցումը

Յուրաքանչյուր սենսոր միացված է Server միկրոկոնտրոլերի առանձին թվային կամ անալոգային մուտքերին։ Սնուցման համար օգտագործում են միկրոկոնտրոլերի 5 վոլտ ելքը իսկ թեստային արժեքներ ստանալու համար օգտագործում ենք հավելյալ ռեզիստորներ կամ սենսորներին արդեն իսկ ինտեգրված համակարգեր։

**Ուլտրաձայնային սենսորը** պահանջում է հինգ միացումներ։

1. 5 վոլտ սնուցման համար միացում (Vcc pin)
2. Տրիգեր միացում (Trigger pin)
3. Էքո ազդանշանի համար միացում (Echo pin)
4. Հողանցման միացում (Ground pin)
5. Թեստ (Test pin)

Test pin – ը ապահովում է տարբեր արդյունքներ սենսորի համար, այն օգտագործվում է միայն թեստային պայմաններում։ Առանցքային են համարվում տրիգեր և էքո միացումները, դրանք համապատասխանաբար միացված են միկրոկոնտրոլերի տպասալի 12 և 11 թվային ոտքերին։



Նկ. 3․2 HC-SR04 սենսորի միացումը տպասալին

Arduino IDE – խմբագրիչում նշանակում ենք միացումների պորտերի համարները, այնուհետև <<setup>> ֆունկցիայի մեջ, որը կատարվում է մեկ անգամ, ծրագրի աշխատանքի ամենասկզբում, վերագրում ենք ուլտրաձայնային սենսորի աշխատանքի հետ կապված թվային ազդանշանը որպես ելքային ազդանշան միկրոկոնտրոլերի համար։ Այս սենսորի հատուկ թվային ազդանշանի ելքի համար նշանակում ենք թվով 3 –րդ միկրոկոնտրոլերի թվային միացումը։



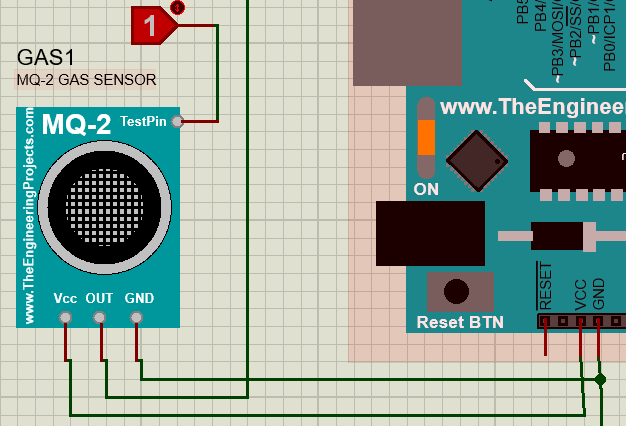
Նկ. 3․3 HC-SR04 սենսորի միացման նկարագրությունը

Ուլտրաձայնային սենսորի ֆունկցիաները սահմանված են <<SR04.h>> գրադարանում և քանի որ այն չի հանդիսանում ստանդարտ գրադարանային ֆայլ Arduino IDE խմբագրիչի համար, այդ իսկ պատճառով անհրաժեշտ է ավելացնել դա սկզբում։ Գրադարանային ֆունկցյաներց օգտվելու համար անհրաժեշտ է ստեղծել այդ գրադարանի օբյեկտ, տրամադրելով էխո և տրիգեր միացումների համարները։

**Գազի սենսորը** պահանջում է երեք միացումներ։

1. 5 վոլտ հոսանքի սնուցման միացում (Vcc pin)
2. Ելքային միացում (Out pin)
3. Հողանցման միացում (Ground pin)

Test pin – ը ապահովում է տարբեր արդյունքներ սենսորի համար, այն օգտագործվում է միայն թեստային պայմաններում։ Առանցքային է համարվում ելքային միացումը որը կապված է միկրոկոնտրոլերի տպասալի 10 – րդ թվային ոտքին։



Նկ. 3․4 MQ-2 գազի սենսորի միացումը տպասալին

Arduino IDE – խմբագրիչում նշանակում ենք միացման պորտի համարը, այնուհետև <<setup>> մեկ անգամ կատարվող ֆունկցյայի մեջ արժեքավորում ենք <<Out pin -ը>> որպես միկրոկոնտրոլերի տպասալի ելքային միացում իսկ սենսորի հատուկ թվային ազդանշանի ելքի համար նշանակում ենք թվով 4 –րդ միկրոկոնտրոլերի թվային միացումը։



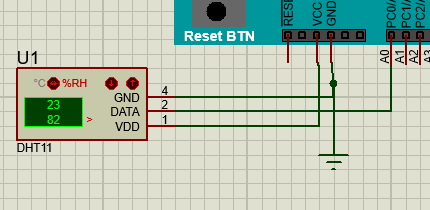
Նկ. 3․5 MQ-2 գազի սենսորի միացման նկարագրությունը

Այս սենսորը միկրոկոնտրոլերի 10 –րդ համարով ոտքին փոխանցում է 0 կամ 1։ Այն դեպքում երբ թունավոր գազի առկայություն է նկատվում օդում այն փոխանցում է 1 ազդանշանը։ Այս պարզ աշխատանքի պատճառով, գազի MQ-2 սենսորը չի պահանջում հավելյալ ֆունկցյաներ, հետևաբար գրադարանային ֆայլ կցելու և օբյեկտ ստեղծելու կարիք չկա։

**Խոնավության և ջերմաստիճանի** սենսորը պահանջում է երեք սենսոր։

1. 5 վոլտ հոսանքի սնուցման միացում (Vcc pin)
2. Տվյալի միացում (Data pin)
3. Հողանցման միացում (Ground pin)

Այս սենսորը իր մեջ ինտեգրված պարունակում է թեստային տվյալներ փոխանցելու ֆունկցյոնալությունը, այդ իսկ պատճառով հավելյալ միացում չի պահանջում։ Առանցքային է համարվում ելքային միացումը որը կապված է միկրոկոնտրոլերի տպասալի անալոգային <<A0>> - ոտքին։



Նկ. 3․5 DHT11 ջերմաստիճանի և խոնավության  
սենսորի միացումը տպասալին

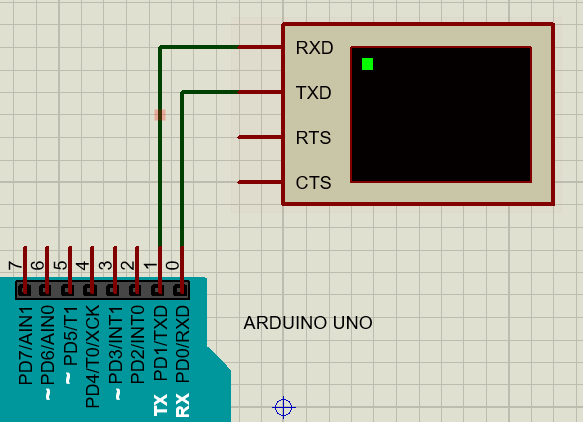
Arduino IDE – խմբագրիչում նշանակում ենք միացման պորտի համարը, այնուհետև <<setup>> մեկ անգամ կատարվող ֆունկցյայի մեջ վերագրում ենք <<Data out -ը>> միկրոկոնտրոլերի տպասալի ելքային ազդանշան, որը սենսորի աշխատանքին կապակցված թվային ազդանշանն է ։ Այն նշանակում ենք թվով 2 –րդ միկրոկոնտրոլերի թվային միացմանը։



Նկ. 3․6 DHT11 ջերմաստիճանի և խոնավության  
սենսորի միացման նկարագրությունը

Ուլտրաձայնային սենսորի ֆունկցիաները սահմանված են <<dht.h>> գրադարանում և քանի որ այն չի հանդիսանում ստանդարտ գրադարանային ֆայլ Arduino IDE խմբագրիչի համար, այդ իսկ պատճառով անհրաժեշտ է ավելացնել դա սկզբում։ Գրադարանային ֆունկցյաներից օգտվելու համար անհրաժեշտ է ստեղծել այդ գրադարանի օբյեկտ։

**Վիրտուալ տերմինալ** այս սարքը անհրաժեշտ է աշխատանքի ընթացքում տեքստային ինֆորմացիայի տպման համար, որը կոնսոլային վիրտուալ միջավայրում արտապատկերվում է իրական ժամանակում։ RXD և TXD վիրտուալ տերմինալի պորտերը միացվում են միկրոկոնտրոլերի տպասալի համապատասխանաբար TX, RX (0, 1) թվային ոտքերին։



Նկ. 3․7 Վիրտուալ տերմինալի միացումը տպասալին

**Թվային օսցիլոսկոպ**։ Այս սարքի պորտերը կարող են միացվել թվային տվյալներ պարունակող հաղորդալարերին։ Այն աշխատում է համակարգի հետ զուգահեռ , իրական ժամանակում, ցուցադրում է հատուկ տերմինալով իր պորտերով անցնող թվային տվյալները։

**4 – բիթ հիշողությամբ ռեգիստր**։ Ունի 4 մուտքային տվյալների և 4 ելքային տվյալների համար նախատեսված պորտեր։ Տվյալների փոխանցումը մուտքային պորտերից դեպի զուգահեռ ելքային պորտեր կատարվում են Clk – ով միացված ժամանակային տակտի դրական ճակատով։ <<Q0>> պորտին միացվում է միկրոկոնտրոլերի տպասալի թվային ելքերը, որոնք փաստում են սենսորների աշխատանքների մասին։ <<Q3>> ելքային պորտով թվային տվյալները փոխանցվում են Client կողմ կամ եթե տվյալ ռեգիստրը չի գտնվում հաջորդաբար տեղադրված ռեգիստրների ամենավերջում, ապա այն փոխանցում է իրենից հետո տեղադրված ռեգիստրի <<Q3>> մուտքին։

## Միկրոկոնտրոլերի հենքով կոնվերային ճարտարապետությամբ IoT համակարգի իրականացումը

Ինչպես արդեն նշվել է Server – կողմում, միկրոկոնտրոլերի գերակա խնդիրը հանդիսանում է սենսորների տվյալների ընթերցումը։ Քանի որ, միկրոկոնտրոլերը մեկ միավոր հաշվողական սարք է այն չի կարող կատարել իրական զուգահեռ աշխատանք, հետևաբար անվերջ աշխատող ռեժիմով այն պետք է գտնվի անվերջ ցիկլի մեջ, հաջորդաբար ընթերցելով առկա, իրեն միացված բոլոր սենսորները։ Յուրաքանչյուր սենսորի աշխատանքի կատարման սկզբում ստեղծվում է սենսորներին կցված թվային ազդանշան, որը փոխանցվում է շիֆթ ռեգիստրներին։ Սենսորի աշխատանքի ավարտից հետո իրեն կցված թվային ազդանշանը ստանում է 0 արժեքը։



Նկ. 3․8 երեք սենսորների աշխատանքի կոդային  
նկարագրությունը

<<digitalWrite()>> - ֆունկցյան ստանդարտ գրադարանային ֆունկցյա է, նախատեսված թվային պորտերին ազդանշաններ հաղորդելու համար։ <<delay()>> ֆունկցյան ընդունում է որպես արգումենտ միկրովարկյան, այն նույնպես ստանդարտ գրադարանային ֆունկցյա է, որը նախատեսված է դադարեցնելու միկրոկոնտրոլերի աշխատանքը տրված ժամանակահատվածով։ Օգտագործված delay – ֆունկցյանները անհրաժեշտ են հետազոտման համար տվյալների պրոցեսները վերահսկելու և գնահատելու համար։ Դրանք պետք է իրական աշխատանքում հեռացվեն։

<<DHT\_11\_sensor>> - DHT11 սենսորի տվյալները ստանալու համար ստեղծված ֆունկցյա է, որը ընթերցում է խոնավության տոկոսը և ջերմասիտճանը։ Ներկա աշխատանքում DHT11 սենսորի համար իրականացված է տվյալների փոխանցման ֆունկցյոնալությունը։ Այս փոխանցումը իրականացվում է I2C արձանագրությամբ Client կողմում միկրոկոնտրոլերի միջև և ունի դեմոնստրատիվ նշանակություն։



Նկ. 3․9 DHT11 սենսորի աշխատանքի նկարագրությունը

Փոխանցման համար հարկավոր է հաշվի առնել արձանագրության հատուկությունները։ Օրինակ, I2C արձանագրությամբ խոնավության և ջերմաստիճան տվյալների փոխանցման համար օգտագործվել է ավելի մեծ արժեք պահպանող ստանդարտ տիպ uint16\_t, քանի որ տվյալի համար այդ արձանագրության մեջ առանձնացված է 8 բիթ տարացություն։ Ընդունող կողմը ստանում է տվյալները և որպես տվյալների մշակման համար նշանակություն ունեցող սարք կատարում է որորշակի ստուգման գործողություն, այս դեպքում, կարմիր լուսային ազդանշանը սկսում է վառվել այն ժամանակ երբ ջերմաստիճանը բարձր է 22օ  C –ն։ Այդ միկրոկոնտրոլերի աշխատանքի նկարագրությունը բերված է հետևյալ նկարում։



Նկ. 3․10 I2C արձանագրությամբ տվյալների փոխանցման  
նախապատրաստությունը

I2C – արձանագրությամբ փոխանցված տվյալները, որոնք բաժանել էինք երկու հատվածների, այստեղ առանձին – առանձին ստանում ենք դրանք և արժեքավորում խոնավության և ջերմաստիճանի փոփոխականները։ I2C – արձանագրությամբ տվյալներ փոխանցելու համար (master – slave միացություններով) անհրաժեշտ է ավելացնել <<Wire.h>> գրադարանային ֆայլը իսկ setup – ֆունկցյայի մեջ թույլատրել արձանագրությամբ տվյալների փոխանակումը այն ժամանակ երբ այդ պրոտոկոլով տվյալներ կհաղորդվեն։



Նկ. 3․11 I2C արձանագրությամբ տվյալների փոխանցման  
նախապատրաստությունը

<<receiveEvent()>> ֆունկցյան աշխատում է ամեն անգամ երբ կատարվում է տվյալների տրանզակցյա I2C – արձանագրությամբ նշանակված պորտերով։ Իսկ անվերջ ցիկլի մեջ տեղադրում ենք սենսորի տվյալները մշակող ֆունկցյա ՝ <<check\_dht\_sensor()>>, որը ինչպես արդեն նշեցինք կատարում է ստուգում, և միացնում է կարմիր լուսադիոդը երբ ինչ որ սահմանից բարձր ջերմաստիճան է հայտնաբերվում։



Նկ. 3․12 Client կողմում սենսորի տվյալների   
մշակումը

<<HC\_SR04\_sensor>> - այս ֆունկցյան ընթերցում է հեռավորությունը, արտածում վիրտուալ տերմինալով ընթերցած տվյալները։



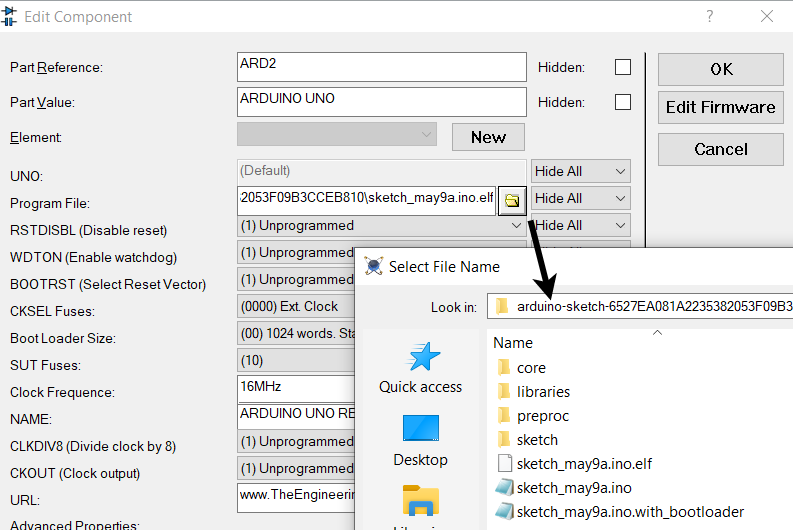
Նկ. 3․13 HC-SR04 սենսորի աշխատանքը

<<MQ\_2\_sensor>> - ֆունկցյան ընթերցում է գազի սենսորի աշխատանքի ընթացքում առաջացող թվային տվյալը։



Նկ. 3․14 MQ-2 սենսորի աշխատանքի նկարագրությունը

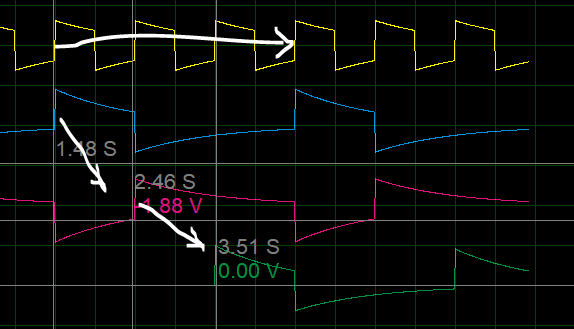
Դիզայնը որը պատրաստվել է <<Proteus 8>> համակարգչային ճարտարապետությունների մոդելավորման ծրագրով աշխատացնելու նպատակով պետք է ներբեռնել միկրոկոնտրոլերների մեջ Arduino IDE խմբագրիչով կազմած ծրագրերի իրականացումները, որոնք ունեն (ino.elf) ընդլայնումները։



Նկ. 3․15 ino.elf ֆայլի տեղակայումը միկրոկոնտրոլերի վրա

Յուրաքանչյուր նոր փոփոխություններ, որոնք կատարվում են կոդում պետք է անցնեն վերիֆիկացիա, որից հետո կստեղծվի նոր ino.elf ֆայլ և միկրոկոնտրոլերների վրա անհրաժեշտություն կառաջանա նորից տեղադրելու այդ նոր ֆայլերը։ Անհրաժեշտ է համոզվել որ միկրոկոնտրոլերները ունեն վերջին թարմացված ծրագրային կոդի կոմպիլացված ino.elf ֆայլերը։ Սա ընդհանուր համակարգի տեսքն է <<Proteus 8>> համակարգչային ճարտարապետությունների մոդելավորման ծրագրով։

Աշխատանքի ընդացքում վիրտուալ տերմինալները ցույց են տալիս իրական ժամանակում սենսորների աշխատանքի ընթացքում ընթերցված տվյալները։ Թվային օսցիլոսկոպը ցույց է տալիս կոնվերային կերպով սենսորների աշխատանքին կցված թվային ազդանշանները։



Նկ. 3․16 Կոնվերային սենսորների ազդանշանների  
փոխանցումը

Ծրագիրը աշխատեցնելուց մի քանի clk ժամանակահատված հետո էկրանին ավտոմատ կբացվեն մոնիտորինգի պատուհանները, վիրտուալ տերմինալները և թվային օսցիլոսկոպը։ Որոշ ժամանակ անց, երբ Client միկրոկոնտրոլերը մշակի սենսորներից DHT11 – ի տվյալները , ապա կմիացնի կարմիր լուսադիոդային լամպը, քանի որ տեղի կունենա նրա իրականացման մեջ նշված պայմանը, այն է ջերմաստիճանը բարձր կլինի 22օ C․ ից (նկարները հավելվածում)։



Նկ. 3․17 Դիզայնի աշխատանքը իրական ժամանակում

## Երրորդ գլխի եզրահանգում

Երրորդ գլուխը բաղկացած է երեք ենթամասերից։ Առաջին հատվածում ներկայացված է IoT համակարգի ընդհանուր կառուցվածքը, Client – Server գործառույթները, դրանց միջև գտնվող հիշողությունը, ռեգիստրային ինտեգրացիան և դրանց նշանակությունները , հնարավոր կիրառումները։ Երկրորդ հատվածում ներկայացված է IoT համակարգի տարրերի աշխատանքին անհրաժեշտ միջվայրերի ստեղծումը, միկրոկոնտրոլերի, սնուցման աղբյուրների և հողանցման միացումների գրաֆիկական, ծրագրային նկարագրությունները։ Երրորդ հատվածում ներկայացված է միկրոկոնտրոլերի հենքով հրամանների կոնվերային ճարտարապետության իրագործումը <<Proteus 8>> մոդելավորման ծրագրի միջոցով, բերված են մոնիտորինգի գործիքներից ստացված արդյունքները։ Արդյունքնում համակարգի աշխատանքով պայմանավորված սինխրոնիզացվել են սենսորներից տվյալների ընթերցման և փոխանցման գործողությունները։ Յուրաքանչյուր 3 Clk տակտում մշակվում է 3 սենսոր, սենսորների թվաքանակի աճման զուգընթաց փոփոխման է ենթակա ընդհանուր համակարգի հաճախականությունը, որի նպատակը կայանում է Client կողմում յուրաքանչյուր պահի մեկ սենսորի ընթերցման ֆունկցիոնալությունը ապահովելը։

**ԵԶՐԱԿԱՑՈՒԹՅՈՒՆ**

Ուսումնասիրվել են միկրոկոնտրոլերների ճարտարապետությունները, ընդհանրությունները և առանձնահատկությունները։

Վերլուծվել են կոնվերային ճարտարապետության, բուֆերիզացման սկզբունքները։

Մշակվել է կոնվերային իրականացմամբ սենսորների ազդանշանների փոխանցման գործառույթները շիֆթ ռեգիստրների միջոցով։

Իրականացվել է համակարգ՝ կիրառելով մշակված ալգորիթմը՝ իրականացնելով IoT համակարգ, որը միկրոկոնտրոլերի հենքով կատարում է հրամանների կոնվերային ճարտարապետությամբ աշխատանք։

**ՏԵՐՄԻՆԱԲԱՆՈՒԹՅՈՒՆ**

IoT – internet of things, իրերի ինտերնետ – ներկառուցված համակարգերի, ֆիզիկական սարքերի ցանց

I2C – արձանագրություն, IoT համակարգերում սարքերի միջև հաղորդակցություն ապահովելու համար նախատեսված

Server – այս աշխատանքում այն հատվածն է որը պատասխանատու է սենսորների տվյալների ստացման փոխանցման համար

Client – այս աշխատանքում այն հատվածն է, որը պատասխանատու է սենսորների տվյալների մշակման, IoT համակարգի սարքերի կառավարման համար

Shift register – ռեգիստրների միակցություն, նախատեսված սինխրոն ժամանակային տակտով տվյալների պահպանման, փոպանցման համար

**ԳՐԱԿԱՆՈՒԹՅԱՆ ՑԱՆԿ**

1. L. Atzori et al., “The internet of things: A survey,” Comput. Netw., vol. 54, no. 15. pp. 2787–2805. 2010.
2. Stankovic, J.A., “Research directions for the internet of things,” IEEE Internet of Things Journal, vol 1, no. 1, pp. 3–9, 2014.
3. Gartner Forecasts 2021 - Worldwide 5G Network Infrastructure Revenue to Grow 39% - https://www.gartner.com/en/newsroom/press-releases/2021-08-04-gartner-forecasts-worldwide-5g-network-infrastrucutre-revenue-to-grow-39pc-in-2021
4. Arduino UNO R3 Product Reference Manual SKU: A000066
5. ATmega168/328P pin mapping ATmega168/328P-Arduino Pin Mapping - https://docs.arduino.cc/hacking/hardware/PinMapping168

Electronic shift registers - <https://www.electronics-tutorials.ws/sequential/seq_5.html>

Gumerov R.I. Workshop on microprocessors. Part one: AVR microcontrollers. Management. - Kazan: KGU, 2009, -37 p.

Fundamentals of Superscalar Processors John Paul Shen and Mikko H. Lipasti 2013 reissued by Waveland Press, Inc

**ՀԱՎԵԼՎԱԾ**

Server – միկրոկոնտրոլերի համար ծրագրային իրականացումը Arduini IDE - ով

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104 | *//Standard libraries*  #include <Wire.h>  #include <dht.h>  #include <SR04.h>  *//define variables*  *//#define DHTTYPE DHT11*  #define DHT\_SENSOR\_PIN A0  #define TRIG\_PIN 12  #define ECHO\_PIN 11  #define SMOKE\_PIN 10  *//define out pins*  #define DHT\_OUT 2  #define ECHO\_OUT 3  #define SMOKE\_OUT 4  *//forward declarations*  dht DHT;  SR04 sr04 = SR04(ECHO\_PIN, TRIG\_PIN);  *//global variables*  int loop\_cycles = 0;  void setup()  {  *//start serial transaction*  Wire.begin();  Serial.begin(9600);  *//define pinmodes*  pinMode(DHT\_OUT, OUTPUT);  pinMode(ECHO\_OUT, OUTPUT);  pinMode(SMOKE\_OUT, OUTPUT);  pinMode(SMOKE\_PIN, INPUT);  delay(100);  }  void loop()  {  *//Serial.print("Loop cycles: ");*  *//Serial.println(loop\_cycles);*    *//humidity measurement*  digitalWrite(DHT\_OUT, HIGH);  DHT\_11\_sensor();  delay(100);  digitalWrite(DHT\_OUT, LOW);    *//distance measurement*  digitalWrite(ECHO\_OUT, HIGH);  HC\_SR04\_sensor();  delay(100);  digitalWrite(ECHO\_OUT, LOW);  *//smoke measurement*  digitalWrite(SMOKE\_OUT, HIGH);  MQ\_2\_sensor();  delay(300);  digitalWrite(SMOKE\_OUT, LOW);  *//loop\_cycles++;*  }  void DHT\_11\_sensor()  {  DHT.read11(DHT\_SENSOR\_PIN);  uint16\_t dht\_data = 0;  dht\_data = DHT.temperature;  uint8\_t tmp\_hum = DHT.humidity;  dht\_data = (dht\_data << 8) | tmp\_hum;  byte dht\_arr[2];  dht\_arr[0] = (dht\_data >> 8) & 0xFF;  dht\_arr[1] = dht\_data & 0xFF;  Wire.beginTransmission(9);  Wire.write(dht\_arr, 2);  Wire.endTransmission();  }  void HC\_SR04\_sensor()  {  long distance = sr04.Distance();    Serial.print("D = ");  Serial.print(distance);  Serial.println(" cm");  }  void MQ\_2\_sensor()  {  bool isSmoke = false;  Serial.print("S = ");  **if**(digitalRead(SMOKE\_PIN) == HIGH) {  isSmoke = true;  } **else** {  isSmoke = false;  }  Serial.println(isSmoke); |

Client – միկրոկոնտրոլերի համար ծրագրային իրականացումը Arduini IDE – ով

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | #include <Wire.h> *//This library is used for I2C communication*  #define temp\_alert\_pin 13  #define dht\_in 2  *// Globals*  byte temp\_a;  byte hum\_a;  uint8\_t temperature = 0;  uint8\_t humidity = 0;  void setup() {  pinMode(temp\_alert\_pin, OUTPUT);  pinMode(dht\_in, INPUT);  Wire.begin(9);  Wire.onReceive(receiveEvent);  Serial.begin(9600);  }  void receiveEvent(int bytes) {  **if**(digitalRead(dht\_in)) {  temp\_a = Wire.read();  hum\_a = Wire.read();  }    }  void loop() {  check\_dht\_sensor();  delay(1000);  }  void check\_dht\_sensor() {  temperature = temp\_a;  humidity = hum\_a;  **if** (temperature > 20) {  digitalWrite(temp\_alert\_pin, HIGH);  } **else** {  digitalWrite(temp\_alert\_pin, LOW);  }  Serial.print("T = ");  Serial.print(temperature, 1);  Serial.print(" deg. C, H = ");  Serial.print(humidity, 1);  Serial.println("%");  temp\_a = 0;  hum\_a = 0;  } |

Դիզայնի ընդհանուր տեսքը աշխատանքի ընթացքում

