# **Ներկառուցված ծրագրի մասնահատկությունները**

Սնուցումը միացնելուց հետո կատարվում են հետևյալ նախնական կարգաբերման քայլերը․

MX\_GPIO\_Init();

MX\_TIM3\_Init();

MX\_TIM4\_Init();

MX\_UART5\_Init();

MX\_I2C1\_Init();

MX\_DAC\_Init();

LCD\_Init();

Այս քայլերից հետո ծրագիրը հատուկ ռեգիստրից (սնուցումից անկախ) կարդում է (Read\_Coord()) սանրի փաստացի կոորդինատը(այսինքն թղթի կտրվելիք չափսը) և արտացոլում է (LCD\_Write()) ցուցասարքի վրա։ Ցուցասարքը արտացոլում է նաև նախորդ ցանկալի կոորդինատը։ Քանի դեռ օգտագործողը չի փոխել սանրի կոորդինատը, ապա ցանկալի արժեքը կլինի նույնը, ինչ փաստացի արժեքն է։

Օրինակ․

Real 0200.00 mm – փաստացի կոորդինատ

Set 0200.00 mm – ցանկալի կոորդինատ

Եթե օգտագործողին բավարարում է թղթի կտրվելիք չափսը, կարող է անցնել կտրելու փուլ, հակառակ դեպքում ստեղնաշարի օգնությամբ կարող է մուտքագրել նոր կոորդինատը և հրամայել սանրին գնալ նոր կոորդինատ։

Սանրին հրամայելու համար․

1. Ծրագիրը սպասում է ստեղնաշարից թվի մուտքագրմանը(Read\_Keypad())
2. Եթե մուտքագրվածը թիվ է, ապա ծրագիրը զրոյացնում է ցանկալի կոորդինատի արժեքը և պատկերում է մուտքագրված թիվը ցուցասարքի համապատասխան դիրքում(LCD\_Write()) (աջից ձախ տեղաշարժելով մուտքագրված թվերը)։
3. Ծրագիրը պահում է մուտքագրված թիվը վերջնական թիվը որոշելու համար
4. Ծրագիրը կրկնում է 1-3 կետերը, քանի դեռ ստեղնաշարից ստանում է 0-9 թվերը
5. Եթե ծրագիրը ստացել է 0-9-ից կամ #-ից տարբեր նշան, արհամարհում է այն և շարունակում է 1 կետից
6. Եթե ստացել է # նշան, ապա ծրագիրը սպասում է թվի հաստատմանը կրկին #-ի միջոցով կամ թվի ջնջմանը՝ \* -ի միջոցով
7. Եթե ստացել է \*, ապա ծրագիրը ջնջում է վերջին թիվը կամ եթե վերջում # է մուտքագրվել, ապա ջնջում է #-ը՝ տեղաշարժելով թվերը ձախից աջ
8. Եթե ծրագիրը 2-րդ անգամ ստացել է # նշան, ապա 3 կետում պահված թվանշաններից կազմում է թիվ
9. Բաղդատում է կազմված թիվը ընթացիկ դիրքի հետ
10. Որոշում է շարժման ուղղությունը, մեծությունը

Մեծությունը փաստացի և ցանկալի կոորդինատների տարբերությունն է։

Այսքանով որոշվեց կտրվելիք թղթի չափսը։ Ցուցասարքը կարտացոլի նոր կոորդինատը(ցանկալի կոորդինատը) և փաստացի կոորդինատը։

Օրինակ․

Real 0200.00 mm – փաստացի արժեք

Set 0400.00 mm – ցանկալի արժեք

Այնուհետև ծրագիրը տեղափոխում է սանրը անհրաժեշտ դիրք՝ կատարելով հետևյալը.

1. Անջատում է սանրի արգելակը(Brush\_Unlock()) և ստուգում է արգելակի վիճակը: Brush\_Unlock() ֆունկցիան վերադարձնում է հետևյալ վիճակները․

0 – unlock, երբ հոսանք չկա և անջատել ենք սանրի արգելակը, կամ երբ հոսանք կա և անջատել ենք սանրի արգելակը

1 – lock երբ հոսանք կա և միացրել ենք սանրի արգելակը

1. Եթե հոսանք կա և անջատվել է սանրի արգելակը, ապա ծրագիրը շարժման մեծությունից կախված որոշում է շարժման արագությունը և միացնում է (Set\_Inverter()) hաճախային վերափոխիչի առաջ(Brush\_Forward) կամ հետ(Brush\_Back) ելքը:
2. Ծրագիրը պարբերաբար ստուգում է էնկոդերի արժեքը(էնկոդերը միացված է սանրին և գրանցում է սանրի տեղաշարժը) և ցանկալի կոորդինատին մոտենալիս դանդաղեցնում է շարժիչի արագությունը(Change\_Speed(ramp-down) (Change\_Speed(ramp-up) շարժումը սկսելիս)) և հասնելուն պես՝ անջատում հաճախային վերափոխիչը (Set\_Inverter()) ու միացնում է սանրի արգելակը(Brush\_Lock()), որպեսզի շարժիչը իներցիայով առաջ չգնա։

Նախապես հայտարարված ramp\_up, ramp\_down արժեքներով են որոշվում թե որքանով դանդաղի շարժիչի արագությունը։

Ծրագիրը backup register-ում գրանցում է սանրի փաստացի կոորդինատը(Save\_Coord())։ Եթե համակարգի հոսանքը անջատվել է, այդ դեպքում նույնպես ծրագիրը պահպանում է սանրի փաստացի կոորդինատը։

Ծրագրում դրված են 2 soft limit-ներ, որոնցից մեկը վերևից է սահմանափակում սանրի տեղաշարժը, մյուսը՝ ներքևից։ Եվ եթե սանրը հանդիպում է այդ soft limit-ներից որևէ մեկին, ապա ծրագիրը անջատում է շարժիչը։

Ցուցասարքը կարտացոլի նոր(փաստացի) կոորդինատը, և օգտագործողը կարող է անցնել կտրման փուլ։

Օրինակ․

Real 0400.00 mm – փաստացի արժեք

Set 0400.00 mm – ցանկալի արժեք

Կտրելու Փուլ։

1. Կտրելու փուլ անցնելու համար օգտագործողը պետք է սեղմած պահի ոտնակը(պեդալը)։ Ծրագիրը ստուգում է, եթե սեղմված է պեդալը(Read\_Pedal()), սկիզբ է դնում է 5 վայրկյանանոց հապաղմանը, որի ավարտից հետո ծրագիրը ակտիվացնում է կտրելու կոճակները (Cutting\_Buttons): Հապաղումից հետո, եթե ոտնակը դեռ սեղմված է, ապա ծրագիրը միացնում է (Solid\_On()) Solid\_N1-ը (Pedal\_Out), որի կոնտակտները սնում են Solid\_N3 -ին(Cutting ) և Solid\_N4-ին(Press\_Again), որոնք նախատեսված են թուղթը ավելի ուժգին սեղմելու և կտրելու համար։
2. Միաժամանակ սեղմած պահել կտրելու համար նախատեսված երկու կոճակները(Cutting\_Buttons)։ Ծրագիրը ակտիվացնում է (Pressing\_On()) հիդրավլիկի փականը(Press\_Again), որը սկիզբ է դնում 3 վայրկյանանոց հապաղմանը։ Մամլիչը էլ ավելի ուժգին է սեղմում թուղթը։ Հապաղումից հետո ծրագիրը ակտիվացնում է (Cutting\_On()) դանակի կցորդիչը(Cutting): Դանակը իջնում, կտրում է թուղթը և վերադառնում է ելման դիրք։ Դանակի թափանիվի վրա տեղակայված երկու տվիչների արժեքները կարդալով(Read\_Knife\_Sensors()) ծրագիրը որոշում է երբ ակտիվացնի և երբ անջատի դանակի կցորդիչը։ Դանակի կցորդիչը անջատելուց (Cutting\_Off()) հետո, ծրագիրը անջատում է հիդրավլիկայի փականը(Pressing\_Off()): Որից հետո օգտագործողը բաց է թողնում ոտնակը և մամլիչը ազատում է կտրված թուղթը։
3. Ակտիվանում է սանրի տեղաշարժելու մեխանիզմը և կտրված թուղթը հրում է առաջ, որպեսզի օպերատորը հեշտ աշխատի։

Թուղթը նորից կտրելու համար պետք է սկսել 1-ին կետից։

Ոտնակը բաց թողնելուց հետո օգտագործողը ցանկության դեպքում կարող է փոխել սանրի կոորդինատը։

Եթե սանրի գտնվելու կոորդինատը և ցուցասարքին տպված փաստացի կոորդինատը չեն համընկնում, օգտագործողը կարող է փոխել այդ թիվը։

Օգտագործողը կարող է նաև ձեռքով տեղաշարժել սանրը՝ բռնակը սեղմած պահելով։ Ծրագիրը ստուգում է (Read\_Hand\_Catch\_Input()) համապատասխան մուտքը(Hand\_Catch) և եթե նկատվել է բռնակի սեղմում, ապա ծրագիրը անջատում է սանրի արգելակը(Brush\_Unlock())։ Այնուհետև օգտագործողը կարող է տեղաշարժել սանրը։ Ցանկալի դիրք տանելուց հետո օգտագործողը թողնում է բռնակը և ծրագիրը միացնում է արգելակը(Brush\_Lock())։ Ընթացիկ կոորդինատը արտացոլվում է ցուցասարքին(LCD\_Write()) և ծրագիրը պահում է սանրի կոորդինատը պահեստային ռեգիստրի մեջ(Save\_Coord())։ Եթե համակարգը անջատված է, ապա օգտագործողը չի կարող տեղաշարժել սանրը, քանի որ այն բլոկավորված է բլոկավորող համակարգի կողմից։