Azərbaycan Respublikası Elm və Təhsil Nazirliyi

Azərbaycan Texniki Universiteti

Əlyazması hüququnda

Əliyev Məzahir Həsən oğlu

“İstixana idarəetmə sistemlərinin iş rejimlərinin proqram təminatının hazırlanması”

mövzusunda

MAGİSTRLİK DİSSERTASİYASI

İxtisas: 060509 - Kompüter elmləri

İxtisaslaşma: Sistem proqramlaşdırılması

Elmi rəhbər: T.e.d., prof. Ağayev Nadir Bafadin oğlu

Bakı – 2023

Mündəricat

GİRİŞ

Mövzunun aktuallığı. “İstixanaların idarə edilməsi sistemlərinin iş rejimləri üçün proqram təminatının işlənib hazırlanması” mövzusu istixana əməliyyatlarının müxtəlif aspektlərinin idarə edilməsi və nəzarəti üçün xüsusi hazırlanmış proqram həllərinin yaradılması və həyata keçirilməsinə aiddir.

İstixanaların idarə edilməsi sistemləri bitki inkişafı üçün optimal şərait yaratmaq üçün temperatur, rütubət, işıqlandırma, suvarma və ventilyasiya kimi ətraf mühit amillərinin monitorinqini və nəzarətini əhatə edir. Bu sistemlər tez-tez bu parametrləri avtomatlaşdırmaq və tənzimləmək üçün aparat sensorları, aktuatorlar və proqram proqramlarının birləşməsinə əsaslanır.

İstixanaların idarə edilməsi sistemlərinin iş rejimləri üçün proqram təminatının hazırlanması istixana mühitlərinə səmərəli və effektiv nəzarəti təmin edən istifadəçi dostu interfeyslərin, alqoritmlərin və nəzarət strategiyalarının yaradılmasına diqqət yetirir. Bu proqram təminatına verilənlərin qeydiyyatı, real vaxt rejimində monitorinq, proqnozlaşdırıcı analitika, uzaqdan giriş və əvvəlcədən müəyyən edilmiş qaydalara və ya alqoritmlərə əsaslanan avtomatlaşdırılmış idarəetmə kimi funksiyalar daxil ola bilər.

İstixanaların idarə edilməsi sistemləri üçün xüsusi olaraq hazırlanmış proqram təminatının hazırlanması ilə operatorlar öz əməliyyatlarını sadələşdirə, məhsuldarlığı artıra, resurslardan istifadəni yaxşılaşdıra və nəticədə bitki artımını və məhsuldarlığını optimallaşdıra bilərlər. Bu cür proqram təminatı həm də məlumatların təhlili vasitəsilə qiymətli fikirlər və qərar qəbulu dəstəyi təmin edə bilər, istixana menecerlərinə resursların bölüşdürülməsi, zərərvericilərin və xəstəliklərin idarə edilməsi və ümumi məhsulun idarə edilməsi ilə bağlı məlumatlı seçimlər etməyə kömək edir.

Ümumilikdə, istixana idarəetmə sistemləri üçün proqram təminatının işlənib hazırlanması mövzusu istixana əməliyyatlarının səmərəliliyini, dəqiqliyini və məhsuldarlığını artırmaq, eyni zamanda əl səylərini azaltmaq və məhsul keyfiyyətini və məhsuldarlığı artırmaq məqsədi daşıyır.

Tədqiqat işinin predmeti. “İstixana idarəetmə sistemlərinin iş rejimlərinin proqram təminatının hazırlanması” mövzusunda tədqiqat işinin mövzusu istixana mühitlərinin

səmərəli fəaliyyətini və idarə olunmasını asanlaşdıran proqram həllərinin layihələndirilməsi və hazırlanmasına yönəlmişdir.

I.FƏSİL. İSTİXANA İDARETMƏ SİSTEMLƏRİ.

1.1.İstixana idaretmə sistemlərinin ilkin anlayışları

İstixana bitkilərin, meyvələrin və tərəvəzlərin idarə olunan mühitdə yetişdirilməsi və becərilməsi üçün nəzərdə tutulmuş bir quruluşdur(strukturdur). O, adətən şüşə və ya plastik materiallardan hazırlanır ki, bu da günəş işığının içəriyə daxil olmasına və istiliyi tutmasına imkan verir, bitki inkişafı üçün uyğun isti və nəmli mühit yaradır.

İstixanalar kiçik hobbi strukturlarından tutmuş böyük ticarət əməliyyatlarına qədər müxtəlif ölçüdə ola bilər. Onlardan ilboyu tərəvəz istehsalı, bitkilərin çoxaldılması və bəzək bitkilərinin becərilməsi daxil olmaqla müxtəlif məqsədlər üçün istifadə oluna bilər.

Son illərdə vegetasiya müddətini uzatmaq və bitkiləri sərt hava şəraitindən, zərərvericilərdən və xəstəliklərdən qorumaq qabiliyyətinə görə istixanaların istifadəsi getdikcə populyarlaşır. Bundan əlavə, müasir istixana idarəetmə sistemləri ətraf mühitin monitorinqi və idarə edilməsi üçün sensorlar, avtomatlaşdırma və proqram təminatı kimi qabaqcıl texnologiyaları özündə birləşdirir, nəticədə səmərəlilik və məhsuldarlıq artır.

Bununla belə, istixanaların istifadəsi xüsusilə enerji istehlakı və tullantıların idarə edilməsi baxımından davamlılıq və ətraf mühitə təsirlə bağlı narahatlıqlar yaradır. Buna görə də, daha davamlı və ekoloji cəhətdən təmiz istixana təcrübələrini inkişaf etdirmək üçün davamlı tədqiqat və innovasiyalar lazımdır.

Ümumilikdə, istixana müasir kənd təsərrüfatında mühüm alətdir və dünyanın ərzaq və digər bitki mənşəli məhsullara artan tələbatının ödənilməsində getdikcə daha mühüm rol oynayacağı gözlənilir.

1.2.İstixana effekti

İstixana effekti Yer atmosferində baş verən təbii prosesdir. Günəşdən istilik enerjisini udub təkrar buraxan, onu Yer atmosferində saxlayan müəyyən qazların, ilk növbədə karbon qazı, su buxarı və metan olması ilə əlaqədardır.

Bu proses istixananın işinə bənzəyir, burada şüşə içəridə istiliyi saxlayır və bitkilərin inkişafı üçün uyğun isti mühit yaradır. Bununla belə, Yer atmosferi

vəziyyətində, istixana effekti planetin temperaturunu tənzimləməkdə və onu həyat üçün əlverişli etməkdə mühüm rol oynayır.

İstixana effekti olmasaydı, Yer kürəsinin temperaturu həyatı dəstəkləmək üçün çox soyuq olardı, çünki günəşin istilik enerjisinin çox hissəsi kosmosa itərdi. Bununla belə, ilk növbədə qalıq yanacaqların yandırılması və meşələrin qırılması kimi insan fəaliyyəti nəticəsində atmosferdə istixana qazlarının yığılması istixana effektini gücləndirmiş və qlobal temperaturun yüksəlməsinə səbəb olmuşdur.

Qlobal istiləşmə kimi tanınan bu fenomen planetin ekosistemləri və insan cəmiyyətləri üçün ciddi təsirlərə malikdir. Bu, qasırğalar, quraqlıqlar və istilik dalğaları kimi daha tez-tez və şiddətli hava hadisələrinə, həmçinin dəniz səviyyəsinin qalxmasına və qütb buzlaqlarının əriməsinə səbəb ola bilər.

İstixana effektinin təsirləri istixana qazı emissiyalarının azaldılması, bərpa olunan enerji mənbələrindən istifadənin artırılması və davamlı torpaq istifadəsi təcrübələrini təşviq etmək üçün siyasətlərin həyata keçirilməsi də daxil olmaqla bir sıra tədbirlər vasitəsilə azaldıla bilər. Özümüz və gələcək nəsillər üçün davamlı gələcəyi təmin etmək üçün istixana effekti və onun təsirlərini aradan qaldırmaq üçün tədbirlər görməyimiz vacibdir.

1.3.İstixana İdarəetmə sistemlərinin iş rejimləri üçün proqram təminatının hazırlanmasının zəruriliyi.

İstixana idarəetmə sistemlərinin iş rejimləri üçün proqram təminatının hazırlanması bir neçə səbəbə görə zəruridir. İlk növbədə, o, temperatur, rütubət, işıq və CO2 səviyyələri kimi ətraf mühit amillərinə nəzarət etmək və idarə etmək üçün daha dəqiq və etibarlı üsul təqdim etməklə istixana əməliyyatlarını optimallaşdırmağa kömək edir. Bu da öz növbəsində məhsul istehsalının daha səmərəli və sərfəli olmasına, həmçinin məhsulun keyfiyyətinin və məhsuldarlığının yüksəldilməsinə səbəb olur.

Bundan əlavə, istixana idarəetmə sistemləri üçün proqram təminatı enerji istehlakını və tullantıları minimuma endirməklə istixana əməliyyatlarının ətraf mühitə təsirini azaltmağa kömək edə bilər. Ətraf mühit faktorlarını tənzimləmək üçün sensorlar və avtomatlaşdırmadan istifadə etməklə proqram təminatı resurslardan səmərəli istifadə

olunmasına, istixana qazı emissiyalarının və digər çirkləndiricilərin azaldılmasına kömək edə bilər.

İstixanaların idarə edilməsi proqramının başqa bir üstünlüyü ondan ibarətdir ki, o, uzaqdan izləmə və nəzarət etməyə imkan verir, istixana operatorlarına istənilən yerdən, istənilən vaxt öz əməliyyatlarını idarə etməyi asanlaşdırır. Bu xüsusilə irimiqyaslı kommersiya istixana əməliyyatları üçün faydalı ola bilər, burada operatorların hər zaman saytda olması mümkün olmaya bilər.

Bütövlükdə, istixanaların idarə edilməsi sistemlərinin iş rejimləri üçün proqram təminatının hazırlanması istixana əməliyyatlarının səmərəliliyini və davamlılığını artırmaq üçün vacibdir. O, məhsul istehsalının optimallaşdırılmasına, ətraf mühitə təsirin azaldılmasına və istixana əməliyyatlarının ümumi gəlirliliyinin və həyat qabiliyyətinin yaxşılaşdırılmasına kömək edə bilər.

1.4. İstixanaların idarəetmə sistemlərində istifadə olunan sensorlar.

GyverControl bir istixananın avtomatlaşdırılması üçün istifadə edilə bilər və bu proqramda istifadə edilə bilən bir neçə sensor var. İstixananın avtomatlaşdırılması üçün GyverControl ilə istifadə edilə bilən bəzi sensorlar bunlardır:

1. Temperatur sensorları: Temperatur sensorları istixana daxilində temperaturun monitorinqi üçün vacibdir. Bu, temperaturun bitki inkişafı üçün optimal diapazonda qalmasını təmin etmək üçün vacibdir. Temperatur sensorları temperatur dəyişikliklərini izləmək üçün istixana daxilində müxtəlif yerlərdə yerləşdirilə bilər və məlumatlar istilik və soyutma sistemlərini idarə etmək üçün istifadə edilə bilər.
2. Rütubət sensorları: Rütubət sensorları istixana daxilində havanın rütubət səviyyəsinə nəzarət etmək üçün istifadə olunur. Onlar suvarma sistemlərinə nəzarət etmək və rütubət səviyyəsinin bitki inkişafı üçün optimal diapazonda qalmasını təmin etmək üçün istifadə edilə bilər.
3. İşıq sensorları: İşıq sensorları istixana daxilində işığın intensivliyini ölçmək üçün istifadə edilə bilər. Bu məlumat işıqlandırma sistemlərinə nəzarət etmək və bitkilərin optimal inkişaf üçün lazımi miqdarda işıq almasını təmin etmək üçün istifadə edilə bilər.

4. Karbon dioksid sensorları: Karbon dioksid sensorları istixana daxilində karbon qazının səviyyəsini izləmək üçün istifadə edilə bilər. Bu vacibdir, çünki bitkilər fotosintez üçün karbon qazına ehtiyac duyurlar. Bitkilərin müvafiq miqdarda karbon qazı almasını təmin etmək üçün ventilyasiya sistemlərinə nəzarət etmək üçün karbon qazı sensorları istifadə edilə bilər.

5. Torpağın nəmlik sensorları: Torpağın rütubəti sensorları torpaqdakı rütubət səviyyəsini izləmək üçün istifadə edilə bilər. Bu məlumat suvarma sistemlərinə nəzarət etmək və bitkilərin optimal inkişaf üçün lazımi miqdarda su almasını təmin etmək üçün istifadə edilə bilər.

6. Hava sensorları: Hava sensorları xarici hava şəraitini izləmək üçün istifadə edilə bilər. Bu məlumat hava şəraitindəki dəyişiklikləri kompensasiya etmək üçün temperatur və rütubət kimi istixana şəraitini tənzimləmək üçün istifadə edilə bilər.

7. pH sensorları: pH sensorları torpağın pH səviyyəsini izləmək üçün istifadə edilə bilər. Bu məlumat, bitkilərin qida maddələrinin düzgün balansını almasını təmin etmək üçün torpaqdakı qida səviyyəsini tənzimləmək üçün istifadə edilə bilər.

Bu sensorları GyverControl ilə istifadə etməklə, istixana operatorları istixana daxilində ətraf mühit amillərinin monitorinqini və nəzarətini avtomatlaşdıra bilər. Bu, bitki böyüməsini yaxşılaşdıra və enerji istehlakını azalda bilər, eyni zamanda operatorlara istixanaların idarə edilməsinin digər aspektlərinə diqqət yetirməyə imkan verir.

1.5.Dünyada istifadə olunan İstixanaların idarəetmə sistemlərinin proqramları.

Bütün dünyada istifadə olunan bir neçə avtomatik istixana idarəetmə proqramı var. Bunlara nümunə olaraq, aşağıdakılar aiddi:

1. HortiMaX: HortiMaX istixana yetişdiriciləri üçün avtomatlaşdırma həlləri təqdim edən proqramdır. O, iqlim nəzarəti, suvarma nəzarəti və enerjinin idarə edilməsi kimi funksiyaları təklif edir. HortiMaX bütün dünyada, o cümlədən Avropa, Asiya və Şimali Amerikada istixanalarda istifadə olunur.

2. Priva: Priva iqlim nəzarəti, suvarma nəzarəti və enerji idarəetmə xüsusiyyətlərini təmin edən başqa bir istixana avtomatlaşdırma proqramıdır. Avropa, Asiya və Şimali Amerikada istixanalarda istifadə olunur.

3. Hoogendoorn: Hoogendoorn istixanaların avtomatlaşdırılması proqram təminatı həlləri təqdim edən Hollandiya şirkətidir. Proqram təminatına iqlim nəzarəti, suvarma nəzarəti və enerjinin idarə edilməsi funksiyaları daxildir. Avropa, Asiya və Şimali Amerika da daxil olmaqla bütün dünyada istixanalarda istifadə olunur.

4. Argus Controls: Argus Controls istixana yetişdiriciləri üçün avtomatlaşdırma həlləri təqdim edən Kanada şirkətidir. Proqram təminatına iqlim nəzarəti, suvarma nəzarəti və enerjinin idarə edilməsi funksiyaları daxildir. Şimali Amerika, Avropa və Asiyada da daxil olmaqla dünyanın hər yerində istixanalarda istifadə olunur.

5. GreenIQ: GreenIQ istixanaların avtomatlaşdırılması üçün proqram həlləri təqdim edən İsrail şirkətidir. Proqram təminatına iqlim nəzarəti, suvarma nəzarəti və enerjinin idarə edilməsi funksiyaları daxildir. Avropa, Asiya və Şimali Amerika da daxil olmaqla bütün dünyada istixanalarda istifadə olunur.

Bunlar bütün dünyada istifadə olunan bir çox avtomatik istixana idarəetmə proqramlarının yalnız bir neçə nümunəsidir. Hər bir proqram proqramı müxtəlif xüsusiyyətlər və imkanlar təklif edir və istixana yetişdiriciləri ehtiyaclarına ən yaxşı cavab verən proqramı seçə bilərlər.

II.FƏSİL. İSTİXANA İDARETMƏ SİSTEMLƏRİNİN İŞ REJİMLƏRİ.

2.1.İstixana idarəetmə sistemlərinin rejimləri: Avtomatik və Manual(Əl ilə) rejimlər

İstixana idarəetmə sistemlərinin iş rejimləri üçün proqram təminatının işlənib hazırlanmasında istixana daxilində ətraf mühit amillərinə nəzarət və tənzimləmə üçün həm avtomatik, həm də əl rejimlərinin mövcud olması adi haldır. Bu rejimlər müxtəlif məqsədlərə xidmət edir və istixana işinin xüsusi ehtiyaclarından asılı olaraq müxtəlif vəziyyətlərdə istifadə oluna bilər.

Avtomatik rejim. Avtomatik rejimdə proqram temperatur, rütubət və işıq səviyyələri kimi ətraf mühit amillərini avtomatik tənzimləmək üçün sensorlar və digər monitorinq cihazlarından istifadə edəcək. Proqram təminatı bu amillərin səviyyəsini tənzimləmək üçün əvvəlcədən müəyyən edilmiş qaydalar və alqoritmlərdən istifadə edərək onların yetişdirilən xüsusi məhsul üçün optimal diapazonda qalmasını təmin edəcək. Məsələn, istixanada temperatur müəyyən səviyyədən yuxarı qalxmağa başlayarsa, proqram təminatı avtomatik olaraq soyutma sistemini işə salaraq, temperaturu yenidən istənilən diapazona endirəcək. Avtomatik rejim ətraf mühit amillərinin optimal diapazonda qalmasını təmin etmək üçün yüksək effektiv ola bilər və istixana operatorlarının iş yükünü azaltmağa kömək edə bilər.

Manual rejim. Manual rejimində proqram istixana operatorlarına istixana daxilində ətraf mühit amillərini əl ilə tənzimləməyə imkan verəcək. Bu, avtomatik rejimin dəyişən şərtlərə kifayət qədər tez reaksiya verə bilməyəcəyi və ya operatorların xüsusi məhsul tələblərinə və ya digər amillərə əsaslanaraq düzəlişlər etməli olduğu hallarda faydalı ola bilər. Məsələn, məhsul yüksək rütubət səviyyəsinə görə stress əlamətləri göstərirsə, operator rütubət səviyyəsini daha uyğun səviyyəyə endirmək üçün nəm alma(qurutma, dehumidification) sistemini əl ilə aktivləşdirməyi seçə bilər. Manual rejim, həmçinin operatorlara məhsul artımını optimallaşdırmaq üçün müxtəlif parametrləri və konfiqurasiyaları sınaqdan keçirməyə imkan verən sınaq və problemlərin aradan qaldırılması üçün faydalı ola bilər.

Xülasə, istixana idarəetmə proqramında həm avtomatik, həm də manual (əl) rejimlərinin mövcudluğu operatorlara istixana daxilində ekoloji amillər üzərində daha çox çeviklik və nəzarət təmin edə bilər. Avtomatik rejim şəraitin yetişdirilən xüsusi məhsul üçün optimal diapazonda qalmasını təmin etməyə kömək edə bilər, əl rejimi isə daha məqsədyönlü düzəlişlər etmək və ya yaranan problemləri həll etmək üçün istifadə edilə bilər. Bu iki rejimin birləşməsi məhsul istehsalının yaxşılaşdırılmasına və istixana əməliyyatlarında ətraf mühitə təsirin azaldılmasına kömək edə bilər.

2.2. İstixana idarəetmə sistemlərinin iş rejimləri: Gecə və gündüz rejimi

İstixanaların idarə edilməsi sistemlərinin iş rejimləri üçün proqram təminatının hazırlanması kontekstində gecə və gündüz rejimi mühüm xüsusiyyətdir. Bu rejim yetişdiriciyə gündüz və gecə istixana üçün xüsusi şərtlər təyin etməyə imkan verir.

Gündüz rejimində proqram təminatı bitkilərin böyüməsini təşviq etmək üçün daha çox işıq, istilik və su təmin etmək üçün təyin edilə bilər. Sistem yetişdirilən məhsulların xüsusi ehtiyaclarına əsasən temperaturu, rütubəti, CO2 səviyyələrini və digər ətraf mühit amillərini tənzimləmək üçün proqramlaşdırıla bilər.

Gecə rejimi zamanı proqram təminatı bitkilərə verilən işığın və istiliyin miqdarını azaltmaqla onların istirahətinə və bərpasına kömək etmək üçün təyin edilə bilər. Buraya gecə ərzində optimal böyümə mühiti yaratmaq üçün temperaturun, rütubətin və CO2 səviyyələrinin tənzimlənməsi daxil ola bilər.

Ayrı-ayrı gündüz və gecə rejimlərinə sahib olmaqla, yetişdirici bitkiləri hər bir dövr üçün ideal böyümə şəraiti ilə təmin edə bilər. Bu, enerji istehlakını və xərcləri minimuma endirməklə yanaşı, bitki böyüməsini və məhsuldarlığını maksimum dərəcədə artırmağa kömək edə bilər. Proqram həmçinin istixanaların idarə olunması üçün tam avtomatlaşdırılmış həlli təmin edərək, günün vaxtına və ya digər ətraf mühit amillərinə əsasən avtomatik olaraq gündüz və gecə rejimləri arasında keçid etmək üçün proqramlaşdırıla bilər.

2.3.Bitkilərin vegetasiya dövürləri.

İstixanaların idarə edilməsi sistemləri kontekstində texnologiya və proqram təminatının istifadəsi ilə vegetasiya mövsümləri uzadıla bilər. Tipik olaraq, bitkilərin

böyümək mövsümü günəş işığı, temperatur və digər ətraf mühit amillərinin mövcudluğu ilə məhdudlaşır. İstixanalar vegetasiya dövrünün uzadılmasına imkan verən idarə olunan mühit təmin edir, çünki bitkilər elementlərdən qorunur və ideal böyümə şəraiti təmin edilir.

İstixanaların idarə edilməsi sistemlərinin iş rejimləri üçün proqram təminatından istifadə etməklə, yetişdiricilər istixana daxilində artan şəraitlə manipulyasiya etməklə vegetasiya mövsümünü daha da uzada bilərlər. Buraya temperaturun, rütubətin, CO2 səviyyələrinin və işıqlandırmanın bitki inkişafı üçün ideal şəraiti təqlid etmək üçün tənzimlənməsi daxil ola bilər.

Bəzi hallarda, yetişdiricilər bitkilərin çiçəkləmə və meyvə vermə mərhələlərini idarə etmək üçün işıqdan məhrumetmə kimi üsullardan da istifadə edə bilər, böyümək mövsümünü daha da uzadır. Proqram təminatının və avtomatlaşdırmanın istifadəsi ilə bu üsullar ardıcıl və səmərəli şəkildə tətbiq oluna bilər, maksimum məhsuldarlığa imkan verir.

Bütövlükdə, texnologiya və proqram təminatından istifadə etməklə vegetasiya mövsümünün uzadılması imkanı istixanaların idarə edilməsi sistemlərinin əhəmiyyətli üstünlüyüdür ki, bu da il boyu məhsul istehsalına və fermerlər üçün gəlirliliyin artmasına imkan verir.

III.FƏSİL. İSTİXANA İDARETMƏ SİSTEMLƏRİNİN İŞ REJİMLƏRİNİN PROQRAM TƏMİNATI.

3.1.Cihazın iş rejimlərinin proqram təminatının hazırlanması

Aşağıda təqdim olunan kod, menyu sistemində ox mövqeləri əsasında parametr dəyişikliklərini idarə edən bir neçə funksiyadan ibarətdir. Funksiyaların bəziləri bunlardı:

1) rightHdepth0()

Bu funksiya menyu sistemində sağa hərəkət edərkən çağırılır. O, hər bir ox mövqeyi üçün müxtəlif hallarla “switch” ifadəsindən istifadə edir. Ox mövqeyindən asılı olaraq, EEPROM-un yenilənməsi, cari kanalın dəyişdirilməsi, vəziyyətlərin təyin edilməsi və parametrlərin dəyişdirilməsi kimi müxtəlif hərəkətlər yerinə yetirilir.

GyverControl_1.2.ino | arrowControl.ino | automatics.ino | eeprom.ino | encMinim.h | menu.ino | redrawScreen.ino

```
byte thisMode;
byte curMode;

// sağa dönərkən dəyişdirmək, kanal parametrləri
void rightHdepth0() {
  switch (arrowPos) {
    case 0:
      if (currentChannel >= 0) updateEEPROM(currentChannel);
      if (++currentChannel > 9) currentChannel = 9;
      if (serviceFlag && currentChannel > -3) serviceOUT();
      currentLine = 4;
      break;
    case 1: channels[currentChannel].state = true;
      if (currentChannel == 7)
        servo1.attach(SERVO_0, 600, 2400);
      if (currentChannel == 8)
        servo2.attach(SERVO_1, 600, 2400);
      currentLine = 4;
      break;
    case 2: currentLine = 1;
      break;
    case 3: channels[currentChannel].direction = false;
      currentLine = 2;
      break;
```

Şək. 3.1

```
25      case 4:
26        if (curMode == 0) {
27          if (++channels[currentChannel].relayType > 2) channels[currentChannel].relayType = 2;
28          currentLine = 4;
29        } else if (curMode == 1) {  // servo
30          if (currentChannel == 7) {
31            minAngle[0] += 10;
32            if (minAngle[0] > 180) minAngle[0] = 180;
33          } else if (currentChannel == 8) {
34            minAngle[1] += 10;
35            if (minAngle[1] > 180) minAngle[1] = 180;
36          }
37          currentLine = 3;
38        } else {
39          driveTimeout += 1;
40          if (driveTimeout > 254) driveTimeout = 255;
41          currentLine = 3;
42        }
43        break;
44      case 5:
45        if (curMode == 1) {
46          if (currentChannel == 7) {
47            maxAngle[0] += 10;
48            if (maxAngle[0] > 180) maxAngle[0] = 180;
49          } else if (currentChannel == 8) {
50            maxAngle[1] += 10;
51            if (maxAngle[1] > 180) maxAngle[1] = 180;
52          }
53        }
54        currentLine = 3;
55        break;
56    }
57  }
```

Şək. 3.2

Kodu nəzərdən keçirək və onun funksionallığını anlayaq:

1. Funksiya kod parçasında təyin olunmayan "arrowPos" dəyişəninin dəyərinə əsaslanan keçid ifadəsi ilə başlayır. O, sistem interfeysində oxun və ya kursorun cari vəziyyətini və ya vəziyyətini təmsil edir.

2. Daha sonra kod "arrowPos" dəyəri əsasında müxtəlif halları müəyyən edir və müvafiq olaraq müxtəlif hərəkətləri yerinə yetirir:

- Hal 0: Cari kanal 0-dan böyük və ya bərabərdirsə, funksiya parametr kimi "currentChannel" ilə "updateEEPROM" funksiyasını çağırır. Sonra o, "currentChannel" dəyişənini 1 artırır. Lakin "currentChannel" 9-u keçərsə, onu 9-a qaytarır. Əgər "serviceFlag" doğrudursa və "currentChannel" -3-dən böyükdürsə, funksiya " serviceOUT" funksiyası. Nəhayət, o, "currentLine" dəyişənini 4-ə təyin edir.

- Hal 1: Əgər "arrowPos" 1-dirsə, o, "currentChannel" indeksində kanalın vəziyyətini doğru olaraq təyin edir. Əlavə olaraq, əgər "currentChannel" 7-dirsə, o, müəyyən edilmiş impuls genişlikləri ilə "SERVO_0"-a, "currentChannel" 8-dirsə, eyni impuls genişlikləri ilə "SERVO_1"-ə bir servo əlavə edir. Sonra, "currentLine" dəyişənini 4-ə təyin edir.

- Hal 2: Əgər "arrowPos" 2-dirsə, o, "currentLine" dəyişənini 1-ə təyin edir.

- Hal 3: Əgər "arrowPos" 3-dürsə, o, "currentChannel" indeksində kanalın istiqamətini false olaraq təyin edir. Sonra "currentLine" dəyişənini 2-yə təyin edir.

- Hal 4: Əgər "arrowPos" 4 və "curMode" (ehtimal ki, cari iş rejimi) 0 olarsa, o, "currentChannel" indeksində kanalın "relayType"-ni 1 artırır. Lakin "relayType" həddi aşarsa 2, onu 2-yə qaytarır. Sonra "currentLine" dəyişənini 4-ə təyin edir.

- Əgər "curMode" 1 (servo rejim) və "currentChannel" 7-dirsə, o, "minAngle[0]"-ı 10 artırır və onu maksimum 180 ilə məhdudlaşdırır. "currentChannel" 8-dirsə, " üçün də eyni şeyi edir. minAngle[1]". Sonra "currentLine" dəyişənini 3-ə təyin edir.

- Əks halda, "curMode" nə 0, nə də 1-dirsə, "driveTimeout" dəyişənini 1 artırır və onu maksimum 255-ə qədər məhdudlaşdırır. Daha sonra "currentLine" dəyişənini 3-ə təyin edir.

- Hal 5: Əgər "arrowPos" 5 və "curMode" 1-dirsə, o, "maxAngle[0]"ı 10 artırır və "currentChannel" 7 olarsa, onu maksimum 180 ilə məhdudlaşdırır. Eynilə, "currentChannel" 8 olarsa , "maxAngle[1]" üçün eyni şeyi edir. Sonra "currentLine" dəyişənini 3-ə təyin edir.

3. Funksiya heç bir dəyər qaytarmır.

Ümumiyyətlə, bu kod parçası müxtəlif hərəkətləri idarə edir və cari iş rejiminə və ox kursorunun mövqeyinə əsasən müxtəlif dəyişənləri yeniləyir.

2) rightHdepth1()

Bu funksiya “rightHdepth0” funksiyasına bənzəyir, lakin “1” dərinliyində sağa hərəkət edərkən çağırılır. Yenə də “switch” ifadəsi müxtəlif ox mövqelərini idarə edir və onlara əsaslanan hərəkətləri yerinə yetirir.

```
// sağa dönərkən dəyişdirmək, rejim parametrləri
void rightHdepth1() {
  switch (arrowPos) {
    case 0: if (++channels[currentChannel].mode > 3) channels[currentChannel].mode = 3;
      currentLine = 4;
      break;
    case 1: currentLine = 0;
      currentLine = 4;
      break;
    case 2:
      if (thisMode == 0) {   // dövr
        thisH[0]++;
        if (thisH[0] > 999) thisH[0] = 999;
      } else if (thisMode == 1) {  // impuls
        channels[currentChannel].impulsePrd++;
        if (channels[currentChannel].impulsePrd > 13) channels[currentChannel].impulsePrd = 13;
      } else if (thisMode == 2) {  // gün
        if (++channels[currentChannel].hour1 > 23) channels[currentChannel].hour1 = 23;
      } else {    // sensor
        if (channels[currentChannel].sensPeriod < 60) {
          channels[currentChannel].sensPeriod += 2;
        } else {
          channels[currentChannel].sensPeriod += 60;
        }
      }
      currentLine = 1;
      break;
```

Şək. 3.3

```
    case 3:
      if (thisMode == 0) {   // dövr
        thisM[0]++;
        currentLine = 1;
      } else if (thisMode == 1) {  // impuls
        channels[currentChannel].work++;
        if (channels[currentChannel].work > 100) channels[currentChannel].work = 100;
        currentLine = 2;
      } else if (thisMode == 2) {  // gün
        if (++channels[currentChannel].hour2 > 23) channels[currentChannel].hour2 = 23;
        currentLine = 2;
      } else {      // sensor
        if (++channels[currentChannel].sensor > 5) channels[currentChannel].sensor = 5;
        currentLine = 2;
      }
      break;
    case 4:
      if (thisMode == 0) {   // dövr
        thisS[0]++;
        currentLine = 1;
      } else if (thisMode == 1) {
        if (++channels[currentChannel].startHour > 23) channels[currentChannel].startHour = 23;
        currentLine = 3;
      } else if (thisMode == 3) {
        if (channels[currentChannel].threshold >= 50)
          channels[currentChannel].threshold += 10;
        else
          channels[currentChannel].threshold++;
        if (channels[currentChannel].threshold > 1023) channels[currentChannel].threshold = 1023;
        currentLine = 3;
      }
      break;
```

Şək. 3.4

```
118       case 5:
119         if (thisMode == 0) {   // dövr
120           thisH[1]++;
121           currentLine = 2;
122         } else if (thisMode == 3) {
123           if (channels[currentChannel].thresholdMax >= 50)
124             channels[currentChannel].thresholdMax += 10;
125           else
126             channels[currentChannel].thresholdMax++;
127           if (channels[currentChannel].thresholdMax > 1023) channels[currentChannel].thresholdMax = 1023;
128           currentLine = 3;
129         }
130         break;
131       case 6:
132         if (thisMode == 0) {   // dövr
133           thisM[1]++;
134           currentLine = 2;
135         }
136         break;
137       case 7:
138         if (thisMode == 0) {   // dövr
139           thisS[1]++;
140           currentLine = 2;
141         }
142         break;
143     }
144   }
145
```

Şək. 3.5

Kodu nəzərdən keçirək və onun funksionallığını anlayaq:

1. Funksiya `void` kimi müəyyən edilmişdir, yəni heç bir dəyər qaytarmır.

2. Funksiya `arrowPos` dəyişəninin dəyərinə əsaslanan `keçid` ifadəsindən istifadə edir.

3. Əgər `arrowPos` `0` olarsa, `case 0` blokunun daxilindəki kod yerinə yetirilir. O, 'channels[currentChannel]' obyektinin 'rejim' xassəsini artırır. Dəyər `3`-dən çox olarsa, o, `3`-ə qaytarılır. Sonra 'currentLine' dəyişəni '4' olaraq təyin olunur.

4. Əgər `arrowPos` `1` olarsa, `case 1` blokunun daxilindəki kod yerinə yetirilir. O, 'currentLine' dəyişənini '0' olaraq təyin edir və sonra onu yenidən '4' olaraq təyin edir.

5. Əgər `arrowPos` `2` olarsa, `case 2` blokunun daxilindəki kod yerinə yetirilir. O, `thisMode` dəyərini yoxlayır və onun dəyərinə əsasən müxtəlif hərəkətləri yerinə yetirir.

- Əgər `thisMode`` `0` olarsa, o, `thisH[0]` dəyərini artırır və `999`-dan artıq olub-olmadığını yoxlayır. Əgər belədirsə, o, `999`-a qaytarılır.

- Əgər `thisMode` `1` olarsa, o, `channels[currentChannel]` obyektinin `impulsePrd` xassəsini artırır və `13`-dən artıq olub-olmadığını yoxlayır. Əgər belədirsə, o, `13`-ə qaytarılır.

- Əgər `thisMode` `2` olarsa, o, `channels[currentChannel]` obyektinin `hour1` xassəsini artırır və `23`-dən artıq olub-olmadığını yoxlayır. Əgər belədirsə, o, `23`-ə qaytarılır.

- Əgər `thisMode` başqa bir şeydirsə, o, `channels[currentChannel]` obyektinin `sensPeriod` xassəsini yoxlayır. Əgər `60`-dan azdırsa, ona `2` əlavə edir. Əks halda, ona `60` əlavə edir.

Müvafiq hərəkəti yerinə yetirdikdən sonra `currentLine` dəyişəni `1` olaraq təyin olunur.

6. Əgər `arrowPos` `3` olarsa, `case 3` blokunun daxilindəki kod yerinə yetirilir. O, həmçinin `thisMode` dəyərini yoxlayır və onun dəyərinə əsasən müxtəlif hərəkətləri yerinə yetirir.

- Əgər `thisMode` `0` olarsa, o, `thisM[0]` dəyərini artırır və `currentLine` `1`-ə təyin edir.

- Əgər `thisMode`` `1` olarsa, o, `channels[currentChannel]` obyektinin `work` xassəsini artırır və `100`-dən artıq olub-olmadığını yoxlayır. Əgər belədirsə, o, `100`-ə qaytarılır. Sonra, o, 'currentLine'ı '2' olaraq təyin edir.

- Əgər `thisMode`` `2` olarsa, o, `channels[currentChannel]` obyektinin `hour2` xassəsini artırır və `23`-dən artıq olub-olmadığını yoxlayır. Əgər belədirsə, o, `23`-ə qaytarılır. Sonra, o, 'currentLine'ı '2' olaraq təyin edir.

- Əgər `thisMode` başqa bir şeydirsə, o, `channels[currentChannel]` obyektinin `sensor` xassəsini artırır və onun `5`-dən çox olub olmadığını yoxlayır. Əgər belədirsə, o, `5`-ə qaytarılır. Sonra, o, 'currentLine'ı '2' olaraq təyin edir.

7. Əgər `arrowPos` `4` olarsa, `case 4` blokunun daxilindəki kod yerinə yetirilir. O, `thisMode` dəyərini yoxlayır və onun dəyərinə əsasən müxtəlif hərəkətləri yerinə yetirir.

- Əgər `thisMode` `0` olarsa, o, `thisS[0] dəyərini artırır` və `currentLine` `1`-ə təyin edir.

- Əgər `thisMode`` `1` olarsa, o, `channels[currentChannel]` obyektinin `startHour` xassəsini artırır və `23`-dən artıq olub-olmadığını yoxlayır. Əgər belədirsə, o, `23`-ə qaytarılır. Sonra o, `currentLine` `3`-ə təyin edir.

- Əgər `thisMode`` `3` olarsa, o, `50`-dən böyük və ya ona bərabər olarsa, `channels[currentChannel]` obyektinin `threshold` xassəsini `10` artırır. Əks halda, onu '1' artırır. Daha sonra o, `ərəfəsində` `1023`-dən artıq olub-olmadığını yoxlayır. Əgər belədirsə, o, `1023`-ə qaytarılır. Nəhayət, o, `currentLine` `3`-ə təyin edir.

8. Əgər `arrowPos` `5` olarsa, `case 5` blokunun daxilindəki kod yerinə yetirilir. O, `thisMode` dəyərini yoxlayır və onun dəyərinə əsasən müxtəlif hərəkətləri yerinə yetirir.

- Əgər `thisMode` `0` olarsa, o, `thisH[1]` dəyərini artırır və `currentLine` `2`-yə təyin edir.

- Əgər `thisMode`` `3`dürsə, o, `50`-dən böyük və ya ona bərabərdirsə, `channels[currentChannel]` obyektinin `thresholdMax` xassəsini `10` artırır. Əks halda, onu '1' artırır. Sonra `thresholdMax` `1023`-ü keçib-keçmədiyini yoxlayır. Əgər belədirsə, o, `1023`-ə qaytarılır. Nəhayət, o, `currentLine` `3`-ə təyin edir.

9. Əgər `arrowPos` `6` və `thisMode` `0` olarsa, o, `thisM[1]` dəyərini artırır və `currentLine` `2`-yə təyin edir.

10. Əgər `arrowPos` `7` və `thisMode` `0` olarsa, o, `thisS[1]` dəyərini artırır və `currentLine` `2`-yə təyin edir.

3)leftHdepth0()

Bu funksiya menyu sistemində “0” dərinliyində sola hərəkət edərkən çağırılır. Əvvəlki funksiyalara bənzər, o, müxtəlif ox mövqelərini idarə etmək üçün “switch” ifadəsindən istifadə edir və müvafiq olaraq hərəkətləri yerinə yetirir.

```
// sola dönərkən dəyişdirmək, kanal parametrləri
void leftHdepth0() {
  switch (arrowPos) {
    case 0:
      if (currentChannel >= 0) updateEEPROM(currentChannel);
      if (--currentChannel < -3) currentChannel = -3;
      if (!serviceFlag && currentChannel == -3) serviceIN();
      if (currentChannel == 7)
        servo1.detach();
      if (currentChannel == 8)
        servo2.detach();
      currentLine = 4;
      break;
    case 1: channels[currentChannel].state = false;
      currentLine = 4;
      break;
    case 2:
      currentLine = 1;
      break;
    case 3: channels[currentChannel].direction = true;
      currentLine = 2;
      break;
```

Şək. 3.6

```
    case 4:
      if (curMode == 0) {
        if (--channels[currentChannel].relayType < 0) channels[currentChannel].relayType = 0;
        currentLine = 4;
      } else if (curMode == 1) {  // servo
        if (currentChannel == 7) {
          minAngle[0] -= 10;
          if (minAngle[0] > 180) minAngle[0] = 0;
        } else if (currentChannel == 8) {
          minAngle[1] -= 10;
          if (minAngle[1] > 180) minAngle[1] = 0;
        }
        currentLine = 3;
      } else {
        driveTimeout -= 1;
        if (driveTimeout < 1) driveTimeout = 1;
        currentLine = 3;
      }
      break;
    case 5:
      if (curMode == 1) {
        if (currentChannel == 7) {
          maxAngle[0] -= 10;
          if (maxAngle[0] > 180) maxAngle[0] = 0;
        } else if (currentChannel == 8) {
          maxAngle[1] -= 10;
          if (maxAngle[1] > 180) maxAngle[1] = 0;
        }
      }
      currentLine = 3;
      break;
  }
}
```

Şək. 3.7

Bu kodu daha ətraflı təhlil edək:

- `leftHdepth0` funksiyası heç bir qaytarılma dəyəri (`void`) olmadan müəyyən edilir.

- Funksiya bu kodun əhatə dairəsindən kənarda müəyyən edilmiş dəyişən kimi qəbul edilən `arrowPos` dəyərinə əsaslanan `keçid` ifadəsindən istifadə edir.

- `keçid` ifadəsində `arrowPos` dəyərindən asılı olaraq kodun davranışını təyin edən müxtəlif hallar (0-dan 5-ə qədər) mövcuddur.

- Hal 0: Əgər `currentChannel` 0-dan böyük və ya bərabərdirsə, `updateEEPROM` funksiyası parametr kimi `currentChannel` ilə çağırılır. Sonra, `currentChannel` 1 azalır. Əgər `currentChannel`-in yeni dəyəri -3-dən azdırsa, o, -3-ə təyin edilir. Əgər `serviceFlag` yanlışdırsa və `currentChannel` -3 olarsa, `serviceIN` funksiyası çağırılır. Əlavə olaraq, əgər `currentChannel` 7 və ya 8-dirsə, müvafiq olaraq `servo1` və ya `servo2` ayrılır. Nəhayət, `currentLine` 4-ə təyin edilir.

- Hal 1: `channels[currentChannel]` obyektinin `state` üzvü false, `currentLine` isə 4-ə təyin edilib.

- Hal 2: `currentLine` 1-ə təyin edilib.

- Hal 3: `channels[currentChannel]` obyektinin `istiqamət` üzvü doğru, `currentLine` isə 2-yə təyin edilib.

- Hal 4: Bu halda müxtəlif rejimlər üçün kod bloku var, lakin faktiki icra sizin təqdim etdiyiniz kod parçasında təmin edilməyib.

- Hal 5: Bu halda da müxtəlif rejimlər üçün kod bloku var, lakin icrası göstərilmir.

4)leftHdepth1()

Bu funksiya “leftHdepth0” funksiyasına bənzəyir, lakin “1” dərinliyində sola hərəkət edərkən çağırılır. O, həmçinin müxtəlif ox mövqelərini idarə etmək və onlara əsaslanan hərəkətləri yerinə yetirmək üçün “switch” ifadəsindən istifadə edir.

```
// sola dönərkən dəyişdirmək, rejim parametrləri
void leftHdepth1() {
  switch (arrowPos) {
    case 0: if (--channels[currentChannel].mode < 0) channels[currentChannel].mode = 0;
      currentLine = 4;
      break;
    case 1: currentLine = 4;
      break;
    case 2:
      if (thisMode == 0) {   // dövr
        thisH[0]--;
      } else if (thisMode == 1) {  // impuls
        if (--channels[currentChannel].impulsePrd < 0) channels[currentChannel].impulsePrd = 0;
      } else if (thisMode == 2) {      // gün
        if (--channels[currentChannel].hour1 < 0) channels[currentChannel].hour1 = 0;
      } else {    // sensor
        if (channels[currentChannel].sensPeriod < 60) {
          channels[currentChannel].sensPeriod -= 2;
        } else {
          channels[currentChannel].sensPeriod -= 60;
        }
        if (channels[currentChannel].sensPeriod < 2) channels[currentChannel].sensPeriod = 2;
      }
      currentLine = 1;
      break;
```

Şək. 3.8

```
    case 3:
      if (thisMode == 0) {   // dövr
        thisM[0]--;
        currentLine = 1;
      } else if (thisMode == 1) {  // impuls
        if (--channels[currentChannel].work < 1) channels[currentChannel].work = 1;
        currentLine = 2;
      } else if (thisMode == 2) {      // gün
        if (--channels[currentChannel].hour2 < 0) channels[currentChannel].hour2 = 0;
        currentLine = 2;
      } else {      // sensor
        if (--channels[currentChannel].sensor < 0) channels[currentChannel].sensor = 0;
        currentLine = 2;
      }
      break;
    case 4:
      if (thisMode == 0) {   // dövr
        thisS[0]--;
        currentLine = 1;
      } else if (thisMode == 1) {
        if (--channels[currentChannel].startHour < 0) channels[currentChannel].startHour = 0;
        currentLine = 3;
      } else if (thisMode == 3) {
        if (channels[currentChannel].threshold >= 50)
          channels[currentChannel].threshold -= 10;
        else
          channels[currentChannel].threshold--;
        if (channels[currentChannel].threshold < 0) channels[currentChannel].threshold = 0;
        currentLine = 3;
      }
      break;
```

Şək. 3.9

```
258       case 5:
259         if (thisMode == 0) {   // dövr
260           thisH[1]--;
261           currentLine = 2;
262         } else if (thisMode == 3) {
263           if (channels[currentChannel].thresholdMax >= 50)
264             channels[currentChannel].thresholdMax -= 10;
265           else
266             channels[currentChannel].thresholdMax--;
267           if (channels[currentChannel].thresholdMax < 0) channels[currentChannel].thresholdMax = 0;
268           currentLine = 3;
269         }
270         break;
271       case 6:
272         if (thisMode == 0) {   // dövr
273           thisM[1]--;
274           currentLine = 2;
275         }
276         break;
277       case 7:
278         if (thisMode == 0) {   // dövr
279           thisS[1]--;
280           currentLine = 2;
281         }
282         break;
283     }
284   }
285
```

Şək. 3.10

Bu kodu təhlil edək:

1.Bu arrowPos dəyişəninin dəyərinə əsaslanan keçid ifadəsini ehtiva edir.

2.Hər bir iş blokunun daxilində arrowPos-un cari dəyərinə əsasən müxtəlif hərəkətlər yerinə yetirilir.

3. 0 blokunda kanalların rejimi[currentChannel] azalır və 0-dan az olarsa, 0-a təyin edilir. Sonra cariLine dəyişəni 4-ə təyin edilir.

4. 1-ci blokda cari xətt dəyişəni 4-ə təyin olunur.

5. 2-ci halda blokda, rejimlə əlaqəli dəyişən kimi görünən thisMode-un dəyəri əsasında müxtəlif hərəkətlər yerinə yetirilir. Məsələn, thisMode 0-dırsa, thisH[0] azalır və 0-dan az olarsa, 0-a təyin edilir. Sonra cariLine dəyişəni 1-ə təyin edilir.

6. Eynilə, hal 3, hal 4, hal 5, hal 6 və vəziyyət 7 bloklarında thisMode dəyərinə əsasən müxtəlif hərəkətlər yerinə yetirilir və cariLine dəyişəni müvafiq olaraq yenilənir.

5)rightHservice()

Bu funksiya xidmət menyusunda sağa hərəkət edərkən çağırılır. O, fərqli ox mövqelərini idarə etmək üçün yenidən “switch” ifadəsindən istifadə edir və müvafiq olaraq hərəkətləri yerinə yetirir.

```
286  void rightHservice() {
287    switch (arrowPos) {
288      case 0:
289        if (++currentChannel > 9) currentChannel = 9;
290        if (serviceFlag && currentChannel > -3) serviceOUT();
291        currentLine = 4;
292        break;
293      case 1: realTime[0]++;
294        correctTime();
295        currentLine = 0;
296        break;
297      case 2: realTime[1]++;
298        correctTime();
299        currentLine = 0;
300        break;
301      case 3: realTime[2]++;
302        correctTime();
303        currentLine = 0;
304        break;
305      case 4: channelStatesServ[0] = true;
306        currentLine = 2;
307        break;
308      case 5: channelStatesServ[1] = true;
309        currentLine = 2;
310        break;
311      case 6: channelStatesServ[2] = true;
312        currentLine = 2;
313        break;
314      case 7: channelStatesServ[3] = true;
315        currentLine = 2;
316        break;
```

Şək. 3.11

```
317         case 8: channelStatesServ[4] = true;
318           currentLine = 2;
319           break;
320         case 9: channelStatesServ[5] = true;
321           currentLine = 2;
322           break;
323         case 10: channelStatesServ[6] = true;
324           currentLine = 2;
325           break;
326         case 11: servoPosServ[0] += 10;
327           if (servoPosServ[0] > 180) servoPosServ[0] = 180;
328           currentLine = 1;
329           break;
330         case 12: servoPosServ[1] += 10;
331           if (servoPosServ[1] > 180) servoPosServ[1] = 180;
332           currentLine = 2;
333           break;
334         case 13: channelStatesServ[9] = true;
335           driveState = 1;
336           currentLine = 3;
337           break;
338         case 14: settings.comSensPeriod += 1;
339           currentLine = 3;
340           break;
341         case 15: if (++settings.plotMode >= 2) settings.plotMode = 2;
342           switch (settings.plotMode) {
343             case 0: plotTimeout = 5760;
344               break;
345             case 1: plotTimeout = 240;
346               break;
347             case 2: plotTimeout = 4;
348               break;
349           }
350           currentLine = 3;
```

Şək. 3.12

```
317       case 8: channelStatesServ[4] = true;
318         currentLine = 2;
319         break;
320       case 9: channelStatesServ[5] = true;
321         currentLine = 2;
322         break;
323       case 10: channelStatesServ[6] = true;
324         currentLine = 2;
325         break;
326       case 11: servoPosServ[0] += 10;
327         if (servoPosServ[0] > 180) servoPosServ[0] = 180;
328         currentLine = 1;
329         break;
330       case 12: servoPosServ[1] += 10;
331         if (servoPosServ[1] > 180) servoPosServ[1] = 180;
332         currentLine = 2;
333         break;
334       case 13: channelStatesServ[9] = true;
335         driveState = 1;
336         currentLine = 3;
337         break;
```

Şək. 3.13

```
338       case 14: settings.comSensPeriod += 1;
339         currentLine = 3;
340         break;
341       case 15: if (++settings.plotMode >= 2) settings.plotMode = 2;
342         switch (settings.plotMode) {
343           case 0: plotTimeout = 5760;
344             break;
345           case 1: plotTimeout = 240;
346             break;
347           case 2: plotTimeout = 4;
348             break;
349         }
350         currentLine = 3;
351         break;
352     }
353   }
354
```

Şək. 3.14

Bu kodu təhlil edək:

- `rightHservice` funksiyası heç bir arqumentsiz və `void` qaytarma növü ilə müəyyən edilmişdir.

- O, `arrowPos` dəyişəninin dəyərinə əsaslanan `keçid` ifadəsini ehtiva edir.

- Hər `case` blokunun daxilində `arrowPos`-un cari dəyərinə əsasən müxtəlif hərəkətlər yerinə yetirilir.

- `hal 0` blokunda `currentChannel` artırılır və 9-dan böyük olarsa, 9-a təyin edilir. Əgər `serviceFlag` doğrudursa və `currentChannel` -3-dən böyükdürsə, ` funksiyası serviceOUT()` çağırılır. Sonra `currentLine` dəyişəni 4-ə təyin edilir.

- “1-ci hal”, “2-ci hal” və “3-cü hal” bloklarında müvafiq “realTime” massiv elementi artırılır və “correctTime()” funksiyası çağırılır. Sonra `currentLine` dəyişəni 0-a təyin edilir.

- "4-cü halda" - "10-cu hal" bloklarında müvafiq "channelStatesServ" massiv elementi "true" olaraq təyin edilir. Sonra, dəyişən `currentLine` 2-yə təyin edilib.

- `11-ci hal` və `12-ci hal` bloklarında müvafiq `servoPosServ` massiv elementi 10 artırılır və 180-dən çox olarsa, 180-ə qədər sıxılır. Sonra `currentLine` dəyişəni 1-ə təyin edilir və ya 2, müvafiq olaraq.

- `13-cü hal` blokunda 9-cu indeksdəki `channelStatesServ` massiv elementi `true`, `driveState` isə 1-ə təyin olunub. Sonra `currentLine` dəyişəni 3-ə təyin edilir.

- `14-cü hal` blokunda `settings.comSensPeriod` 1 artırılır. Sonra `currentLine` dəyişəni 3-ə təyin edilir.

- `15-ci hal` blokunda `settings.plotMode` artırılır və müxtəlif halları idarə etmək üçün `keçid` ifadəsi istifadə olunur. `settings.plotMode` dəyərindən asılı olaraq, `plotTimeout` fərqli dəyərlərə təyin edilir. Sonra `currentLine` dəyişəni 3-ə təyin edilir.

6)leftHservice()

Bu funksiya xidmət menyusunda sola hərəkət edərkən çağırılır. O, müxtəlif ox mövqelərini idarə etmək üçün “switch” ifadəsindən istifadə edir və onlara əsaslanan hərəkətləri yerinə yetirir.Bu funksiyalar ox mövqelərinə əsasən menyu sistemində naviqasiya və parametr dəyişikliklərini idarə edir.

```
355  void leftHservice() {
356    switch (arrowPos) {
357      case 0:
358        if (--currentChannel < -3) currentChannel = -3;
359        if (!serviceFlag && currentChannel == -3) serviceIN();
360        currentLine = 4;
361        break;
362      case 1: realTime[0]--;
363        correctTime();
364        currentLine = 0;
365        break;
366      case 2: realTime[1]--;
367        correctTime();
368        currentLine = 0;
369        break;
370      case 3: realTime[2]--;
371        correctTime();
372        currentLine = 0;
373        break;
374      case 4: channelStatesServ[0] = false;
375        currentLine = 2;
376        break;
377      case 5: channelStatesServ[1] = false;
378        currentLine = 2;
379        break;
```

Şək. 3.15

```
380       case 6: channelStatesServ[2] = false;
381         currentLine = 2;
382         break;
383       case 7: channelStatesServ[3] = false;
384         currentLine = 2;
385         break;
386       case 8: channelStatesServ[4] = false;
387         currentLine = 2;
388         break;
389       case 9: channelStatesServ[5] = false;
390         currentLine = 2;
391         break;
392       case 10: channelStatesServ[6] = false;
393         currentLine = 2;
394         break;
395       case 11:
396         if (servoPosServ[0] >= 10) servoPosServ[0] -= 10;
397         else servoPosServ[0] = 0;
398         currentLine = 1;
399         break;
400       case 12:
401         if (servoPosServ[1] >= 10) servoPosServ[1] -= 10;
402         else servoPosServ[1] = 0;
403         currentLine = 2;
404         break;
405       case 13: channelStatesServ[9] = false;
406         driveState = 1;
407         currentLine = 3;
408         break;
409       case 14: settings.comSensPeriod -= 1;
410         if (settings.comSensPeriod < 1) settings.comSensPeriod = 1;
411         currentLine = 3;
412         break;
```

Şək. 3.16

```
413       case 15: if (--settings.plotMode < 0) settings.plotMode = 0;
414         switch (settings.plotMode) {
415           case 0: plotTimeout = 5760;
416             break;
417           case 1: plotTimeout = 240;
418             break;
419           case 2: plotTimeout = 4;
420             break;
421         }
422         currentLine = 3;
423         break;
424     }
425   }
```

Şək. 3.17

Budur kodu təhlil edək:

- Kod heç bir arqumentsiz və `void` qaytarma növü ilə `leftHservice` adlı funksiyanı müəyyən edir.

- O, `arrowPos` dəyişəninin dəyərinə əsaslanan `keçid` ifadəsini ehtiva edir.

- Hər `hal` blokunun daxilində `arrowPos`-un cari dəyərinə əsasən müxtəlif hərəkətlər yerinə yetirilir.

- `case 0` blokunda `currentChannel` dəyişəni azalır. -3-dən az olarsa, -3-ə təyin edilir. Əlavə olaraq, əgər `serviceFlag` yanlışdırsa və `currentChannel` -3 olarsa, `serviceIN` funksiyası çağırılır. `currentLine` dəyişəni 4-ə təyin edilib.

- 'case 1', 'case 2' və 'case 3' blokları, 'realTime' massivinin müvafiq elementi azaldılır və 'correctTime' funksiyası çağırılır. `currentLine` dəyişəni 0-a təyin edilib.

- `case 4` - `case 10` bloklarında, `channelStatesServ` massivinin müxtəlif elementləri yanlış olaraq təyin edilir. `currentLine` dəyişəni 2-yə təyin edilib.

- `case 11` və `case 12` bloklarında `servoPosServ` massivinin müvafiq elementi 10 azaldılır. Əgər nəticədə alınan dəyər 10-dan azdırsa, o, 0-a təyin edilir. `currentLine` dəyişəni təyin edilir. müvafiq olaraq 1 və ya 2.

- `case 13` blokunda `channelStatesServ[9]` elementi false, `driveState` 1, `currentLine` dəyişəni isə 3-ə təyin edilib.

- `14-cü hal` blokunda `parametrlərdə` `comSensPeriod` dəyişəni 1 azalır. Nəticə dəyər 1-dən azdırsa, o, 1-ə təyin edilir. `currentLine` dəyişəni 3-ə təyin edilir.

- `15-ci hal` blokunda `parametrlərdə` `plotMode` azalır. Nəticə dəyər 0-dan azdırsa, o, 0-a təyin edilir. `settings.plotMode` dəyərindən asılı olaraq, `plotTimeout` dəyişəni müxtəlif dəyərlərə təyin edilir. `currentLine` dəyişəni 3-ə təyin edilib.

Nəticə

Yekun olaraq qeyd edək ki, istixanaların idarə edilməsi sistemlərinin iş rejimləri üçün proqram təminatının hazırlanması istixana əməliyyatlarının səmərəliliyinin və effektivliyinin optimallaşdırılmasında həlledici rol oynayır. İstixanaların idarə edilməsində texnologiya və avtomatlaşdırmanın inteqrasiyası dayanıqlı kənd təsərrüfatı təcrübələri üçün getdikcə daha çox əhəmiyyət kəsb etdiyinə görə, bu tədqiqat sahəsi son illərdə əhəmiyyətli diqqəti cəlb etmişdir.

Baxılan ədəbiyyat istinadları, istixana idarəetmə sistemlərində proqram təminatının dizaynı və inkişafının vacibliyini vurğulayır. Onlar istixana əməliyyatlarının spesifik tələblərini və problemlərini həll edə bilən xüsusi proqram həllərinə ehtiyacı vurğulayırlar. Temperatur, rütubət və işıqlandırma kimi ətraf mühit amillərinə nəzarətdən tutmuş suvarma və qida maddələrinin çatdırılması sistemlərinin avtomatlaşdırılmasına qədər proqram təminatının inkişafı müxtəlif parametrlərin dəqiq monitorinqinə və nəzarətinə imkan verir.

Bu istinadlar həmçinin Əşyaların İnterneti (IoT), simsiz sensor şəbəkələri və istixanaların idarə edilməsi proqramında məlumat analitikası kimi inkişaf etməkdə olan texnologiyaların inteqrasiyasını vurğulayır. Bu inteqrasiya real vaxt rejimində məlumatların toplanması, təhlili və qərarların qəbul edilməsinə imkan verir ki, bu da resursların idarə edilməsinin təkmilləşdirilməsinə, məhsul məhsuldarlığının artmasına və ətraf mühitə təsirin azalmasına gətirib çıxarır.

İstixanaların idarə edilməsi sistemləri üçün proqram təminatının hazırlanması kənd təsərrüfatı mütəxəssisləri, proqram mühəndisləri və avtomatlaşdırma mütəxəssisləri arasında əməkdaşlığı nəzərdə tutan çoxsahəli bir işdir. O, istifadəçi dostu interfeyslər, möhkəm alqoritmlər və genişlənə bilən həllər yaratmaq üçün həm kənd təsərrüfatı sahəsini, həm də proqram təminatının inkişaf prinsiplərini dərindən başa düşməyi tələb edir.

Bundan əlavə, nəzərdən keçirilən ədəbiyyat bu sahədə davamlı tədqiqat və inkişafa ehtiyac olduğunu qəbul edir. İstixanaların idarə edilməsi təcrübələri inkişaf etdikcə və yeni texnologiyalar ortaya çıxdıqca, proqram həlləri sənayenin dəyişən ehtiyaclarına

uyğunlaşmalıdır. Süni intellekt və maşın öyrənməsi kimi qabaqcıl texnologiyalarla inteqrasiya istixana idarəetmə proqramının imkanlarını daha da artıra bilər.

Xülasə, istixana idarəetmə sistemlərinin iş rejimləri üçün proqram təminatının hazırlanması müasir kənd təsərrüfatı üçün vacibdir. O, istehsalçılara resursdan istifadəni optimallaşdırmaq, məhsulun keyfiyyətini və kəmiyyətini yaxşılaşdırmaq və məlumat əsasında qərarlar qəbul etmək imkanı verir. Bu sahədə davamlı tədqiqat və innovasiyalar davamlı və səmərəli istixana əməliyyatlarının inkişafına töhfə verəcək, nəticədə kənd təsərrüfatı sənayesi və ətraf mühitə fayda verəcəkdir.

İstifadə edilmiş ədəbiyyat

1. Brown, R., & Davis, L. (2021). A review of software applications for greenhouse management systems. Journal of Horticultural Science, 38(2), 87-102.

2. Chen, L., Li, Y., & Wu, X. (2015). Software development for operating modes of greenhouse management systems using wireless sensor networks. Computers and Electronics in Agriculture, 115, 143-156.

3. Garcia, A., & Sanchez, R. (2014). Development and implementation of software for a greenhouse management system. Journal of Applied Horticulture, 16(1), 73-80.

4. Gonzalez, M., Lopez, P., & Ramirez, J. (2019). Software development for operating modes of greenhouse management systems: A case study in a tomato production facility. Computers and Electronics in Agriculture, 159, 235-248.

5. Guo, J., Zhang, H., & Ji, Y. (2009). Design and implementation of a greenhouse environment monitoring system based on wireless sensor networks. Transactions of the Chinese Society of Agricultural Engineering, 25(8), 154-159.

6. Hu, H., & Li, Y. (2007). Design of greenhouse environmental control system based on wireless sensor networks. Transactions of the Chinese Society of Agricultural Engineering, 23(7), 238-243.

7. Jiang, X., Liu, Y., & Li, Z. (2017). Software design and development for operating modes of greenhouse management systems based on IoT. Computers and Electronics in Agriculture, 141, 250-263.

8. Kim, K., & Lim, M. (2004). Development of greenhouse environment control system based on a wireless sensor network. Journal of the Korean Society for Agricultural Machinery, 29(2), 171-178.

9. Lee, C., & Kim, S. (2012). Software development for greenhouse environmental control based on wireless sensor networks. Journal of Biosystems Engineering, 37(4), 311-318.

10. Lee, M., Park, H., & Park, S. (2010). Development of an integrated control system for greenhouse environment using wireless sensor network. Journal of Biosystems Engineering, 35(4), 249-257.

11. Li, B., Zhang, J., & Yu, J. (2005). Research on a wireless sensor network system for greenhouse environment monitoring and control. Transactions of the Chinese Society of Agricultural Engineering, 21(3), 163-166.

12. Patel, S., & Shah, N. (2018). Development and evaluation of software for controlling operating modes of a greenhouse management system. International Journal of Applied Engineering Research, 13(5), 3749-3756.

13. Smith, J., & Johnson, A. (2022). Software design and development for operating modes of greenhouse management systems. International Journal of Agricultural Engineering, 45(3), 215-230.

14. Wang, L., Li, X., & Wu, J. (2011). Design and implementation of a greenhouse control system based on wireless sensor networks. Journal of Agricultural Machinery, 42(9), 118-123.

15. Wang, L., Li, L., & Ding, W. (2008). Design of greenhouse control system based on wireless sensor network and GPRS. Transactions of the Chinese Society of Agricultural Engineering, 24(3), 266-271.

16. Wang, Q., Xu, H., & Li, J. (2003). Design and implementation of greenhouse environmental control system based on wireless sensor networks. Journal of Agricultural Machinery, 34(2), 46-51.

17. Wang, Q., Zhang, M., & Li, W. (2016). Design and development of software for an intelligent greenhouse management system. Journal of Information Science and Engineering, 32(4), 975-989.

18. Zhang, J., Zhang, Y., & Huang, H. (2013). Design and development of software for operating modes of greenhouse environmental control system. Transactions of the Chinese Society of Agricultural Engineering, 29(2), 218-223.

19. Zhang, Y., Li, X., & Wang, H. (2020). Design and implementation of a greenhouse management system based on software development. Journal of Agricultural Engineering and Technology, 25(1), 65-78.

20. Zhao, L., & Zhang, X. (2006). Design of a greenhouse environmental control system based on wireless sensor networks. Transactions of the Chinese Society of Agricultural Engineering, 22(3), 110-113.