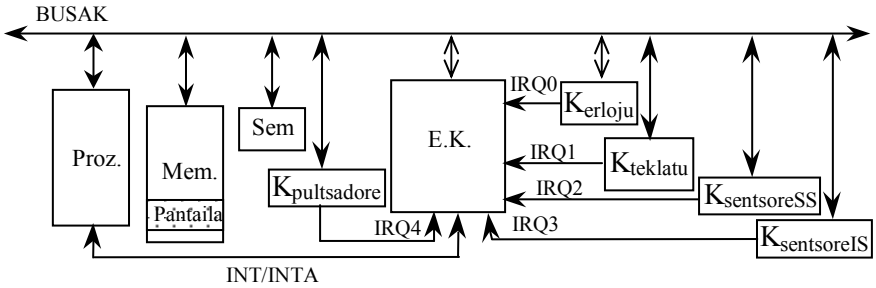
Euskal Herriko Unibertsitatea
Informatika Fakultatea

Konputagailuen Arkitektura eta Teknologia Saila

Konputagailuen Arkitektura I

Sarrera/irteerako azpisistema 1 (ebazpena): Aireportuko hegaztiak

Aireportu bateko hegazkinen aireratze eta lurreratze gunean **hegaztiak** dauden kontrolatzen duen sistema daukagu. Sistemaren hardware egitura klasikoa da, irudian ikus dezakezunez, baina honako gailu berezi hauek ditu: bi sentsore (SS eta IS) eta pultsadore bat. Osagai hauen funtzionamendua geroxeago azalduko da. Horiez gain, semaforo bat kontrolatu behar da; honen kontroladoreak kontrol-erregistro bat baino ez dauka, memorian mapeatuta dagoena, SEM helbidean, hain zuzen.



Mikroprozesadoreak kontrolatu behar dituen periferikoak klasean ikusitakoak dira, bi sentsoreak, pultsadorea eta semaforoa izan ezik. Hauen ezaugarriak hauek dira:

*** SS sentsorea**: Hegaztien Sarrera Sentsorea. Kontrolpeko zonara hegaztiren bat edo gehiago sartzen den detektatzen du eta ondorioz eten-eskaera sortzen du. Bere datu-erregistroan une horretan sartu diren hegaztien kopurua irakur daiteke. Datu-erregistroa irakurri ondoren, `strobe` segida bat egin behar da bere kontrol-erregistroan.

*** IS sentsorea**: Hegaztien Irteera Sentsorea. Kontrolpeko zonatik hegaztiren bat edo gehiago ateratzen den detektatzen du eta ondorioz eten-eskaera sortzen du. Bere datu-erregistroan une horretan atera diren hegaztien kopurua irakur daiteke. Datu-erregistroa irakurri ondoren, `strobe` segida bat egin behar da bere kontrol-erregistroan.

*** SEM semaforoa**: Hegazkinei abisatzeko semaforoa. **Berde** egongo da kontrolpeko zonan hegaztirik ez dagoenean, beraz hegazkinak maniobratu dezake arriskurik ez dagoelako. Alderantzizko kasuan, **Gorri** egongo da. Lehentxeago aipatu denez, memorian mapeatuta dagoen kontrol-erregistroa baino ez dauka.

*** Pultsadorea**: Eten-eskaera bat sortzen du kontrolpeko zonatik hegaztiak uxatu behar dituzten langileek sakatzen dutenean. Pultsadorea sakatuko dute hegazti bat harrapatzen duten bakoitzean.

*** Erlojua**: Segundo batean 18 aldiz eteten du.

*** Teklatua**: Tekla bat sakatzen denean, bi eten-eskaera sortzen ditu: lehenengoa, tekla sakatu (MAKE) dela adierazteko, eta bigarrena, askatu (BREAK) dela adierazteko. Datu-erregistroan, azkenekoz sakatutako teklari dagokion posizio-kodea ("*scan code*") irakur daiteke, eta kontrol-erregistroan `strobe` sekuentzia bat idatzi behar da eten bat onartzen denean. Ez dauka egoera-erregistrorik.

***Etenen kontroladorea**: Erregistro hauek ditu: IMR maskara-erregistroa, IRR eten-eskaeren erregistroa, eta ISR zerbitzuan dauden etenen erregistroa.

Sistemaren **funtzionamendua** hauxe da. **Egoera arrunta edo normala** kontrolpeko zonan hegaztirik ez dagoenean ematen da, eta horrexegatik semaforoa **berde** egongo da. Beste edozein kasutan, semaforoa gorri egongo da.

Hegaztien sarrera detektatzean, sistema **aurre-alarma egoera**ra igaroko da; egoera honetan mantenduko da kontrolpeko zonan hegaztiak dauden bitartean (garbi dago hegaztiak sartu diren bezala atera daitezkeela, hots "de motu propio") eta sistemaz arduratzen den langileak hegaztiak uxatzeko premia adierazten ez duen bitartean. Premiazkoa izango da hegaztiak uxatzea hegazkin batek maniobraren bat egin behar duenean (lurreratu zein aireratu). Kasu horretan, langileak **A tekla** sakatuko du; horren ondorioz, sistema **alarma egoera**ra igaroko da.

Alarma egoeran, semaforoa gorri mantentzeaz gain, soinu-alarma bat sortzen da (suposatu `sortu_alarma()` errutina jadanik idatzita daukagula); soinuak, hegaztiak uxatzeaz aparte, berauek harrapatu behar dituen langile-koadrilari abisatzen dio. "Garbiketa brigada" honek kontrolpeko zonatik alde egiten ez duten hegaztiak sare batez harrapatu behar ditu. Hegazti bana harrapatzean pultsadorea sakatu behar dute sistemari hegazti bat gutxiago dagoela adierazteko.

Brigada honek kontrolpeko zona hegaztiz azkenekoz garbitzen duenean edo hegazti guztiek beren kabuz alde egiten dutenean, sistema **atze-alarma egoera**ra igaroko da, eta soinu-alarma desaktibatuko da (suposatu `desaktibatu_alarma()` errutina jadanik idatzita daukagula). Egoera honetan 2 minutu mantenduko da sistema. Denbora-tarte horretan berriro hegaztiak sartzen badira, alarma egoerara itzuliko da. Bestela, denboraldi hori igarotzen bada hegaztirik gabe, egoera arruntera itzultzen da sistema.

Idatzi lengoaia algoritmikoan programa nagusia eta zerbitzu-errutina hauek: SS sentsorearena, IS sentsorearena, pultsadorearena, teklatuarena, erlojuarena. Lana errazteko, irudika ezazu sistemaren funtzionamendua egoera-automata baten bitartez.

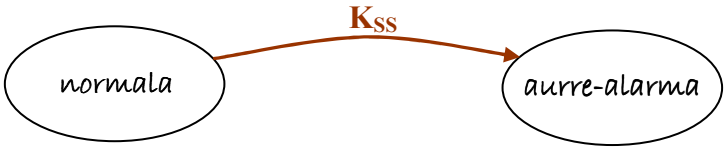
Ebazpena

Lehenik, sistemaren funtzionamendua islatzen duen automata bilatzen saiatuko gara. Horretarako, biziki garrantzitsua da sistemaren funtzionamenduaren egoera edo etapa posible guztiak ondo bereiztea, eta baita ere egoera batetik beste batera pasatzeko egin beharreko urratsak identifikatzea ere. Oro har, egoeren arteko trantsizioak kanpoko gertaeren ondorioak izango dira, eta horiek periferikoen ekintzen bidez adieraziak izango dira, sinkronizazio-modua edozein izanik: inkesta bidezkoa ala etenen bidezkoa.
Ariketa honetan, lehenengoa izanik, enuntziatuan bertan oso argi ageri dira egoerak. Izan ere, honako egoera hauek aipatzen ditu enuntziatuak: egoera *normala*, *aurre-alarma* egoera, *alarma* egoera, eta *atze-alarma* egoera.
Goazen orain egoeren arteko trantsizio posible guztiak bilatzera, egoeraz egoera. Hau da, azter ditzagun sistema honetan gerta daitezkeen gertaera posible guztiak eta ikus dezagun zein diren gertaera horien ondorioak, sistemaren egoeraren arabera. Horretarako, arreta handiz berrirakurriko dugu enuntziatua.

1. Egoera *normala*:

- Egoeraren ezaugarriak:
  Enuntziatuan irakur dezakegu: "Kontrolpeko zonan hegaztirik ez dagoenean ematen da eta semaforoa berde egongo da".
- Trantsizio posibleak:

a) Enuntziatuan irakur dezakegu: "Hegaztien sarrera detektatzean, sistema *aurre-alarma* egoerara igaroko da".
Eta nola detektatzen du sistemak hegaztien sarrera? Bada, horretarako dago, hain zuzen, SS izeneko sentsorea: "Hegaztien Sarrera Sentsorea". Honen kontroladoreak eten eskaera sortuko du sentsoreak hegazti bat edo gehiago kontrolpeko zonan sartu direla detektatzen duenean. Eten eskaera onartzean, prozesadoreak sentsore horri dagokion zerbitzu-errutina exekutatuko du, eta bertan adierazi beharko du programatzaileak egoera aldaketa gertatuko dela.
Guk, automatan, honelaxe adieraziko dugu: bi egoeren arteko geziak trantsizioaren nondik norakoa adierazten du –*normala* egoeratik *aurre-alarma* egoerara–, eta geziaren gaineko etiketak, berriz, zein den trantsizio hori eragin duen periferikoa edo gertaera.



↳ SS sentsorearen zerbitzu-errutinaren eginkizunak:
Aurreko egoera-trantsizioaz gain, SS sentsoreari dagokion zerbitzu-errutinak beste eginkizun batzuk ere baditu:

SS.1) SS sentsorearen kontroladoreko datu-erregistroa (`R_DAT_KSS` etiketaren bidez adieraziko dugu haren helbidea sarrera/irteerako mapan) irakurri beharko du, zenbat hegazti sartu diren jakiteko. Horretarako, `InPort(erreg-helb)` funtzioa erabiliko dugu, eta horrek itzultzen duen balioa aldagai lokal batean gordeko dugu, honelaxe:

```
hegazti_kop_sartu = InPort(R_DAT_KSS);
```

SS.2) Garbi dago *normala* egoeran ez dagoela hegaztirik (hori baita abiapuntua), baina hortik aurrerako beste egoera guztietan oso garrantzitsua izango da kontrolpeko zonaren barruan dauden hegaztien kopuru zehatza ezagutzea, hegaztiak sartzen diren bezalaxe posible delako kontrolpeko zonatik ateratzea, eta, ondorioz, posible litzatekeelako *aurre-alarma* egoeratik *normala* egoerara itzultzea hegazti guztiak aterako balira (geroxeago aztertuko dugu zehazki aukera hau). Hortaz, une oro jakin behar da zenbat hegazti dauden kontrolpeko zonaren barruan. Horretarako, aldagai global bat erabiliko dugu: `hegazti_kop` (beharrezkoa da globala izatea, toki batean baino gehiagotan eguneratu ahal izango delako horren balioa, ikusiko dugun bezalaxe). Hala, aldagai global horren hasierako balioa 0 izango da (eskuarki, programa nagusian egingo ditugu aldagaien beharrezko hasieraketak), baina SS sentsoreak hegaztiak sartu direla detektatzean, modu egokian eguneratu beharko dugu aldagai horren balioa, sartu diren hegaztien kopurua gehituz. Honelaxe:

```
hegazti_kop = hegazti_kop + hegazti_kop_sartu;
```

SS.3) Enuntziatuan esandakoari jarraiki, SS sentsorearen kontroladoreko kontrol-erregistroaren gainean (`R_KON_KSS`) "strobe" sekuentzia bat egin behar da. Horretarako, suposatuko dugu jadanik idatzita dagoen funtzioa daukagula, honako hau, hain zuzen:

```
strobe(R_KON_KSS);
```

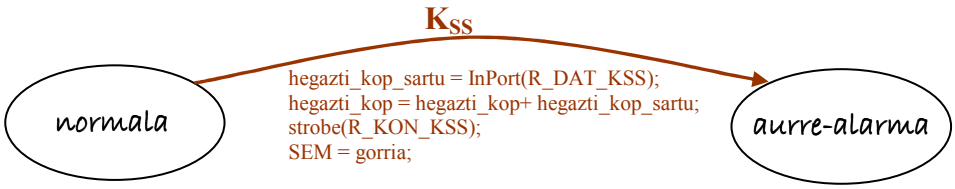
SS.4) Azkenik, semaforoaren kolorea aldatu behar da, gorrira, eta horrela mantenduko da kontrolpeko zonan hegaztiak dauden bitartean. Semaforoaren kontroladoreak memorian mapeatutako kontrol-erregistro bat baino ez duenez, `SEM` memoria helbidean, hain zuzen, `SEM` etiketa aldagai globala balitz bezala maneiatu dezakegu, memoria posizio bat adierazten duelako. Hortaz, honelaxe adieraziko dugu kolore-aldaketa:

```
SEM = gorria;
```

Dena den, aldagai global gisa erabili beharrean, helbide gisa ere erabil dezakegu `SEM` etiketa, eta `idatz_mem(helbidea,datua)` funtzioa erabil dezakegu memorian idazteko, modu honetan:

```
idatz_mem(SEM,gorria);
```

Gauza edo ekintza horiek automatan bertan argitzea komeni da, hori lagungarria izango baita zerbitzu errutinaren kodea idaztean. Horretarako, trantsizioaren geziaren azpian idatziko ditugu ekintza horiek guztiak, honelaxe:

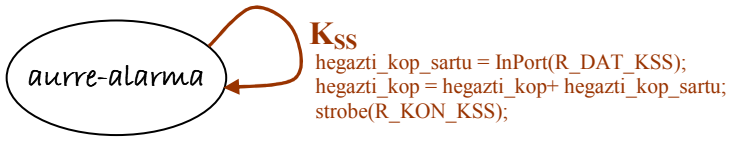


b) Printzipioz, *normala* egoeratik ateratzeko ez dago trantsizio posible gehiagorik.

2. *Aurre-alarma* **egoera:**

- Egoeraren ezaugarriak:
  Enuntziatuan irakur dezakegu: “Egoera honetan mantenduko da sistema kontrolpeko zonan hegaztiak dauden bitartean (garbi dago hegaztiak sartu diren bezala atera daitezkeela, hots, “de motu propio”) eta sistemaz arduratzen den langileak hegaztiak uxatzeko premia adierazten ez duen bitartean”.
  Beraz, hortik aukera batzuk aurreikusten dira:

  ▪ Alde batetik, sistema *aurre-alarma* egoeran dagoen bitartean, posible da hegazti gehiago sartzea kontrolpeko zonan, eta horrek ez du eraginik izango sistemaren egoeran, hots, sistema egoera horretan mantenduko da hegazti gehiago sartu arren. Hori bai, kontrolpeko zonaren barruan dauden hegaztien kopurua modu egokian eguneratu beharko da. Eta hori guztia, berriz ere, SS sentsorearen zerbitzu-errutinean egin behar da: aurreko ekintzak (SS.1, SS.2 eta SS.3) errepikatu behar dira, baina garbi izan behar dugu SS.4 pixka bat aldatu behar dugula, kasu honetan, *aurre-alarma* egoeran, ez delako semaforoaren kolorea aldatu behar (jadanik gorria baitago) eta gainera ez da egoera-trantsiziorik gertatuko, sistema *aurre-alarma* egoeran geratuko baita. Hortaz, automatan honelaxe adieraziko dugu aukera hau:



Eta SS.4 eginkizuna honela aldatuko dugu, automataren egoera kontuan hartzeko:

SS.4) Egoera-trantsizioa eta semaforoaren kolorearen aldaketa egingo da soilik beharrezkoa denean, hau da, automataren egoera *normala* baldin bada, bestela ez. Horretarako, aldagai global bat erabiliko dugu automataren egoera kontuan hartzeko: `egoera_automata` aldagaia, hain zuzen:

```
if (egoera_automata == normala)
   {
       SEM = gorria;
       egoera_automata = aurre-alarma;
   }
```

▪ Beste aldetik, sistema *aurre-alarma* egoeran dagoen bitartean, posible da hegazti batzuk kontrolpeko zonatik ateratzea, baina horrek ez du eraginik izango sistemaren egoeran, baldin eta kontrolpeko zonaren barruan oraindik hegazti batzuk geratzen badira. Hori bai, kasu honetan ere, kontrolpeko zonaren barruan dauden hegaztien kopurua modu egokian eguneratu beharko da. Baina: nola daki sistemak hegazti batzuk kontrolpeko zonatik atera direla? Bada, horretarako dago, hain zuzen, IS izeneko sentsorea: “Hegaztien Irteera Sentsorea”. Honen kontroladoreak eten eskaera sortuko du sentsoreak hegazti bat edo gehiago kontrolpeko zonatik atera direla detektatzen duenean. Eten eskaera onartzean, prozesadoreak sentsore horri dagokion zerbitzu-errutina exekutatuko du, eta bertan adierazi beharko ditu programatzaileak exekutatu beharreko ekintzak. Hona hemen:

↳ IS sentsorearen zerbitzu-errutinaren eginkizunak:

IS.1) IS sentsorearen kontroladoreko datu-erregistroa (`R_DAT_KIS`) irakurri beharko du, zenbat hegazti atera diren jakiteko. Hala, `InPort` funtzioak itzultzen duen balioa aldagai lokal batean gordeko dugu, honelaxe:

```
hegazti_kop_atera = InPort(R_DAT_KIS);
```

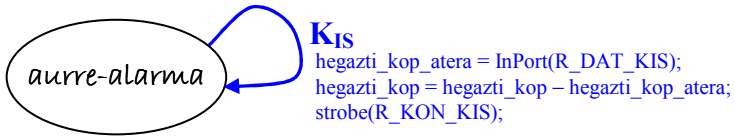
IS.2) `hegazti-kop` aldagai globala eguneratu beharko dugu, atera diren hegaztien kopurua kenduz, honelaxe:

```
hegazti_kop = hegazti_kop - hegazti_kop_atera;
```

IS.3) Sentsore honen kasuan ere, kontroladoreko kontrol-erregistroaren gainean (`R_KONT_KIS`) "strobe" sekuentzia bat egin behar da. Honelaxe:

```
strobe(R_KON_KIS);
```

Honelaxe adieraziko dugu aukera hau automatan:



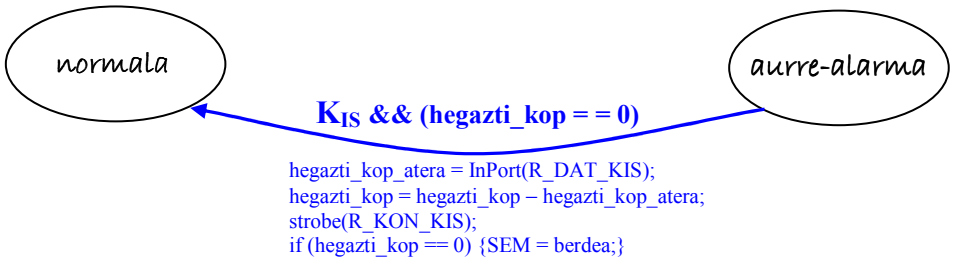
- Baina, zer gertatzen da `hegazti_kop` aldagaia eguneratu ondoren, haren balioa 0 bada? Hau da, zer gertatzen da hegazti guztiak kontrolpeko zonatik ateratzen badira? Enuntziatuan ez digute garbi adierazten, baina hemen logikak agintzen du: logikoena da sistema *normala* egoerara itzultzea, kontrolpeko zonan jadanik ez baita hegaztirik geratzen. Hori egoera-trantsizio bat denez gero, hurrengo atalean azalduko dugu.
- Azkenik, enuntziatua berriro irakurriz gero, konturatuko gara "egoera honetan mantenduko dela sistema […] sistemaz arduratzen den langileak hegaztiak uxatzeko premia adierazten ez duen bitartean". Garbi dago kasu honetan ere egoera-trantsizio bat gertatuko dela, eta horrexegatik hurrengo atalean azalduko dugu.

- Trantsizio posibleak:

a) Beraz, IS sentsoreak eteten duenean, `hegazti_kop` aldagaia eguneratu ondoren, haren balioa 0 baldin bada, orduan sistema *normala* egoerara itzuliko da, kontrolpeko zonan ez baita hegazti gehiagorik geratzen. Eta, horrekin batera, semaforoa berriz ere berde jarri beharko dugu. Hori guztia, automatan, gezi baten bidez adieraziko dugu, eta IS sentsorearen zerbitzu-errutinean, honelaxe:

IS.4) Egoera-trantsizioa eta semaforoaren kolorearen aldaketa, beharrezkoa baldin bada, hau da, sistema *aurre-alarma* egoeran baldin badago eta kontrolpeko zonan ez bada hegazti gehiagorik geratzen:

```
if ((hegazti_kop == 0)&&(egoera_automata == aurre-alarma))
      {
          SEM = berdea;
          egoera_automata = normala;
      }
```



b) Gogora dezagun sistema *aurre-alarma* egoeratik aterako dela baita ere "sistemaz arduratzen den langileak hegaztiak uxatzeko premia adierazten" duenean. Eta "premiazkoa izango da hegaztiak uxatzea hegazkin batek maniobraren bat egin behar duenean (lurreratu zein aireratu). Kasu honetan, langileak A tekla sakatuko du; horren ondorioz, sistema *alarma* egoerara igaroko da". Garbi dago, beraz, teklatuaren kontroladorearen eten-eskaera izango dela gertaera horren adierazlea, baina A tekla bakarrik hartu beharko dugula kontuan, ez beste edozein tekla, eta bakarrik tekla sakatzean eta ez askatzean –gogoratu teklatuaren kontroladoreak bi eten-eskaera egiten dituela tekla bakoitzeko: bata tekla sakatzean (MAKE), eta bestea tekla askatzean (BREAK)–. Hortaz, egoeren arteko trantsizioa baldintzapekoa izango da: A tekla sakatzen denean, soilik. Hauek dira egin beharreko eginkizunak:

↳ Teklatuaren zerbitzu-errutinaren eginkizunak:

Tek.1) Teklatuaren kontroladoreko datu-erregistroa (`R_DAT_Ktek`) irakurri beharko du, langileak zein tekla sakatu duen jakiteko. Kontuan izan teklatuaren kontroladoreko datu-erregistroan irakurtzen den balioa teklaren posizio-kodea edo "*scan code*" bat dela, inoiz ez teklari esleitutako karaktere zehatza. Horregatik, eskuarki, posizio-kode hori itzuli beharko dugu, zein tekla sakatu den jakiteko.

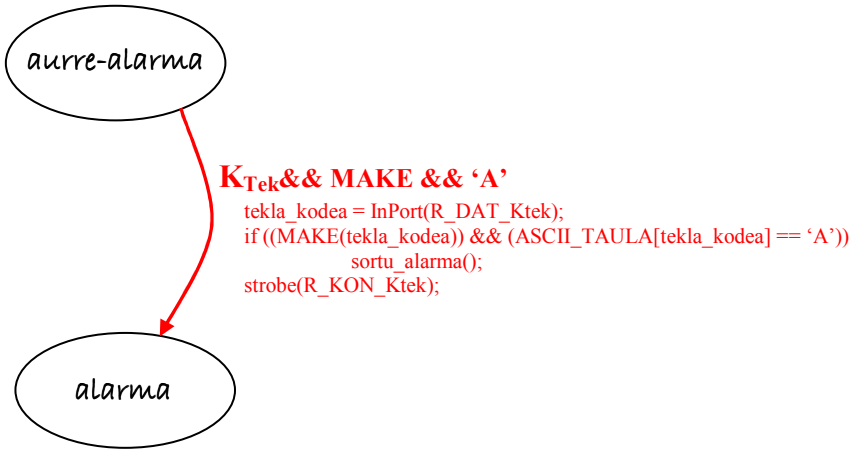
```
tekla_kodea = InPort(R_DAT_Ktek);
```

Tek.2) Orain, tekla sakatu duen (MAKE) ala askatu duen (BREAK) begiratu behar dugu –suposatuko dugu badugula funtzio bat jadanik idatzita hori egiteko: `MAKE(tekla_kodea)`–, eta gainera begiratu behar dugu ea A tekla den –horretarako, posizio-kodea ASCII karaktere bihurtuko dugu memorian gordeta dagoen taula baten bitartez, teklaren posizio-kodea erakusle gisa erabiliz: `ASCII_TAULA[tekla_kodea]`, eta irakurketa horrek posizio horretan dagoen ASCII karakterea itzultzen du –, eta hori guztia *aurre-alarma* egoeran soilik, beste edozein egoeratan ez baitzaio kasurik egin behar teklatuari. Eta baldintza horiek betetzen direnean, egoera-aldaketa eragiteaz gain, soinu-alarma bat sortu behar da, eta horretarako badago `sortu_alarma()` izeneko errutina berezi bat, jadanik idatzita dagoena. Hortaz, honelaxe geratuko da kodea:

```
if ((MAKE(tekla_kodea))&&(ASCII_TAULA[tekla_kodea]== 'A')
     &&(egoera_automata == aurre-alarma))
                {
                    sortu_alarma();
                    egoera_automata = alarma;
                }
```

Tek.3) Teklatuaren kasuan ere, kontroladoreko kontrol-erregistroaren gainean (`R_KON_Ktek`) "strobe" sekuentzia bat egin behar da. Honelaxe:

```
strobe(R_KON_Ktek);
```

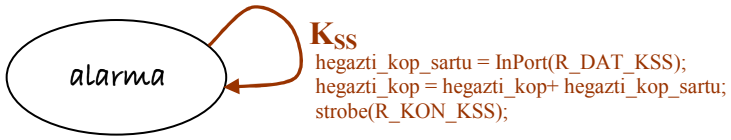


3. *Alarma* **egoera:**

- Egoeraren ezaugarriak:

  Enuntziatuan irakur dezakegu: "*Alarma* egoeran, semaforoa gorri mantentzeaz gain, soinu-alarma bat sortzen da […]; soinuak, hegaztiak uxatzeaz aparte, berauek harrapatu behar dituen langile-koadrilari abisatzen dio. "Garbiketa brigada" honek kontrolpeko zonatik alde egiten ez duten hegaztiak sare batez harrapatu behar ditu. Hegazti bana harrapatzean, pultsadorea sakatu behar dute, sistemari hegazti bat gutxiago dagoela adierazteko".
  Aurreko kasuan bezalaxe, aukera batzuk aurreikusten dira kasu honetan ere:

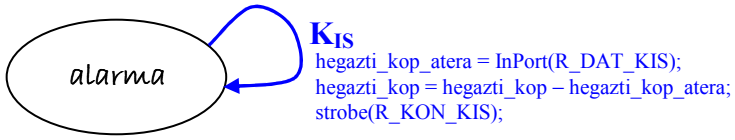
  - Alde batetik, *alarma* egoeran ere, posible da hegazti gehiago sartzea kontrolpeko zonan, eta horrek ez du eraginik izango sistemaren egoeran, hots, sistema egoera horretan mantenduko da hegazti gehiago sartu arren. Baina kontrolpeko zonaren barruan dauden hegaztien kopurua modu egokian eguneratu behar da. Beraz, lehenago aipatutako ekintzak errepikatu behar dira eta kasu honetan ere ez da egoera-trantsiziorik gertatuko ez eta semaforoaren kolorea aldatuko (jadanik gorria baitago). Hortaz, automatan honelaxe adieraziko dugu aukera hau:



  Eta SS sentsorearen zerbitzu-errutinean, SS.4 eginkizuna ez dugu aldatu behar izango, jadanik hartu baitugu kontuan semaforoaren kolore-aldaketa eta egoera-aldaketa *normala* egoeran bakarrik gertatzen direla.

  - Beste aldetik, *alarma* egoeran ere, posible da hegazti batzuk kontrolpeko zonatik ateratzea beren kabuz, alarmaren soinuak ikaratuta-edo, eta horrek ez du eraginik izango sistemaren egoeran, baldin eta kontrolpeko zonaren barruan oraindik hegazti batzuk geratzen badira.

Honelaxe adieraziko dugu aukera hau automatan:



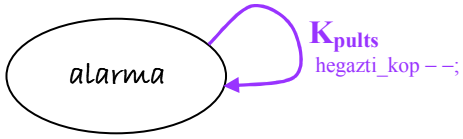
- Garbiketa brigadako langileek, soinuak ikaratuta alde egiten ez duten hegaztiak harrapatu behar dituzte, banan-banan, eta bat harrapatzen duten bakoitzean, pultsadorea sakatu behar dute. Hala, pultsadorearen kontroladorearen etena gertatuko da egoera honetan, eta eten bakoitzean hegazti-kopurua batean txikiagotu beharko da (`hegazti_kop --`), hegazti bat gutxiago geratzen delako kontrolpeko zonan. Baina ez da egoera-trantsiziorik gertatuko kontrolpeko zonan hegaztirik dagoen bitartean. Hori bai, kasu honetan ere, teklatuaren kasuan egin dugun bezala, pultsadorearen etenak *alarma* egoeran soilik hartu behar dira kontuan, ez beste edozein egoeratan, gerta baitzitekeen pultsadorea istripuz sakatzea sistema beste egoera batean dagoenean, eta orduan ez zaio kasurik egin behar, funtzionamendu desegokia saihesteko.

↳ Pultsadorearen zerbitzu-errutinaren eginkizunak:

Pults.1) Sistema *alarma* egoeran baldin badago, `hegazti-kop` aldagai globala eguneratu beharko dugu, hegazti bat bakarra harrapatu dutela kontuan hartuta, honelaxe:

```
if (egoera_automata == alarma)
   {
      hegazti_kop --;
   }
```

Honelaxe adieraziko dugu aukera hau automatan:



- Eta azkenik, berriz ere, zer gertatzen da aurreko bi kasuetan (K_{IS} eta K_{pults} kontroladoreen zerbitzu-errutinetan) `hegazti_kop` aldagaia eguneratu ondoren, haren balioa 0 bada? Hau da, zer gertatzen da kontrolpeko zonan hegaztirik geratzen ez denean? Enuntziatuan garbi adierazten digute: *atze-alarma* egoerara pasatuko da sistema. Hori egoera-trantsizio bat denez gero, hurrengo atalean azalduko dugu.

- Trantsizio posibleak:

a) Gogora dezagun enuntziatuan irakur dezakeguna *alarma* egoerari buruz: "Brigada honek kontrolpeko zona hegaztiz azkenekoz garbitzen duenean, sistema *atze-alarma* egoerara igaroko da". Beraz, kontrolpeko zonan jadanik hegaztirik geratzen ez denean, egoera-trantsizioa gertatuko da, eta baldintza hori gerta daiteke pultsadoreak eteten duenean edo IS sentsoreak eteten duenean, bietan `hegazti_kop` aldagaia eguneratu ondoren, haren balioa 0 izan daitekeelako, eta orduan sistema *atze-alarma*

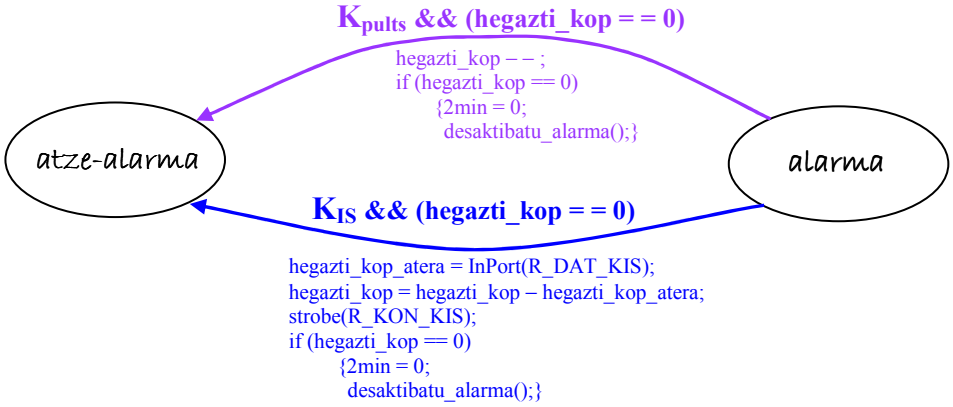
egoerara pasatuko da, kontrolpeko zonan ez baita hegazti gehiagorik geratzen. Halaber, alarma desaktibatu beharko da, eta horretarako badaukagu jadanik idatzita dagoen errutina: `desaktibatu_alarma()`. Horrez gain, sistemak egoera berrian 2 minutu egon beharko duenez hegaztirik gabe, egoeren arteko trantsizioa da unerik egokiena 2 minutuak kontatuko dituen aldagai globala hasieratzeko (`2min` izena jarriko diogu), hortik aurrera kontatzeko.

Hala, honelaxe geratuko zaizkigu bi periferikoen zerbitzu errutinak egoera-trantsizio honi dagokionez:

Pults.2) eta **IS.5)** Egoera-trantsizioa, beharrezkoa baldin bada:

```
if ((hegazti_kop == 0)&&(egoera_automata == alarma))
   {
        2min = 0;
        desaktibatu_alarma();
        egoera_automata = atze-alarma
   }
```

Eta automatan:



4. *atze-alarma* **egoera:**

- Egoeraren ezaugarriak:
  Enuntziatuan irakur dezakegunaren arabera, sistema *atze-alarma* egoeran "2 minutu mantenduko da. Denbora-tarte horretan berriro hegaztiak sartzen badira, *alarma* egoerara itzuliko da. Bestela, denboraldi hori igarotzen bada hegaztirik gabe, egoera arruntera itzuliko da eta hegazkinak maniobra egin ahal izango du arazorik gabe."
  Bertan ditugu adieraziak gerta daitezkeen bi trantsizioak, eta horiek geroko utziko ditugu. Baina, trantsizioez gain, kontuan hartu behar dugu sistema egoera horretan mantentzen den bitartean, 2 minutuko denbora-tartea noiz agortuko zain dagoen bitartean, alegia, sistemaren erlojuari kasu egin behar zaiola, hain zuzen ere, jakin ahal izateko noiz pasatuko diren 2 minutuak. Baina, noski, egoera aldatu gabe. Horrek esan nahi du erlojuaren kontroladorearen zerbitzu-errutinan `2min` aldagaia modu egokian eguneratu behar dela, eta horretarako aukera bat baino gehiago dago. Ikus ditzagun 2 aukera desberdin.
  Jakin badakigu erlojuaren kontroladoreak segundo batean 18 eten eskaera sortzen dituela. Beraz, erlojuaren eten bakoitzean `2min` aldagaia zuzenean eguneratzen badugu, tarteko aldagairik erabili behar izan gabe, hau da, `2min` aldagaiak erlojuaren

etenen kopurua kontatzen badu, orduan 2 × 60 × 18 balioa hartzen duenean, jakingo dugu 2 minutu pasatu direla (2 minutu × 60 segundo/minutu × 18 eten/segundo).
Hori bai, `2min` aldagaia *atze-alarma* egoeran soilik eguneratu behar da, ez beste edozein egoeratan.
Hala, lehenengo kasu honetan, honelaxe geratuko zaigu erlojuaren zerbitzu-errutinan egin beharreko eginkizunak adierazten dituen kodea:

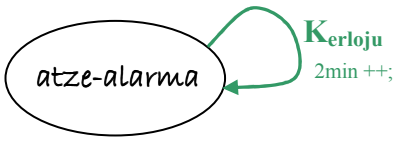
↳ Erlojuaren zerbitzu-errutinaren eginkizunak:

Erloju.1) `2min` aldagaiaren eguneraketa erlojuaren eten bakoitzean (1. aukera):

```
if (egoera_automata == atze-alarma)
   2min++;
```

Modu honetan jakingo dugu 2 minutu pasatu direla `2min` aldagaiak 2 × 60 × 18 balioa hartzen duenean.

Honelaxe adieraziko dugu aukera hau automatan:



Baina sistema batzuetan beharrezkoa izan daiteke segundoak independenteki kontrolatu behar izatea, segundoan behin zerbait berezia egin behar delako. Kasu horretan, erlojuaren eten eskaeren kopurua kontatu beharko dugu, eta, horretarako, aldagai lokal bat erabiliko dugu: `tik` izena emango diogu. Aldagai hori behin bakarrik hasieratuko dugu 0 balioarekin, gure programa exekutatzen hasten denean, erlojuak lehenengo aldiz etetean (hori egiteko, erlojuaren zerbitzu-errutinan honela idatziko dugu: `static int tik = 0`). Tarteko aldagai horri esker, 18 `tik` pasatu ondoren, jakingo dugu noiz pasatu den segundo bat, eta hortik abiatuta segundoak eta minutuak kontatu ahal izango ditugu, eta horrela detektatu noiz pasatzen diren 2 minutuak. Baina horrek luzatzen du idatzi beharreko kodea; horregatik, tarteko aldagai gehiagorik erabili behar izan gabe, zuzenean eguneratuko dugu `2min` aldagaia segundoan behin. Horrela eginez gero, 2 minutuak pasatu direla jakingo dugu `2min` aldagaiak 2 × 60 balioa hartzen duenean, segundotan neurtzen ari baikara (2 minutu × 60 segundo/minutu).
Hala, bigarren kasu honetan, honelaxe geratuko zaigu erlojuaren zerbitzu-errutinan egin beharreko eginkizunak adierazten dituen kodea:

↳ Erlojuaren zerbitzu-errutinaren eginkizunak:

Erloju.1) `2min` aldagaiaren eguneraketa segundoan behin (2. aukera):

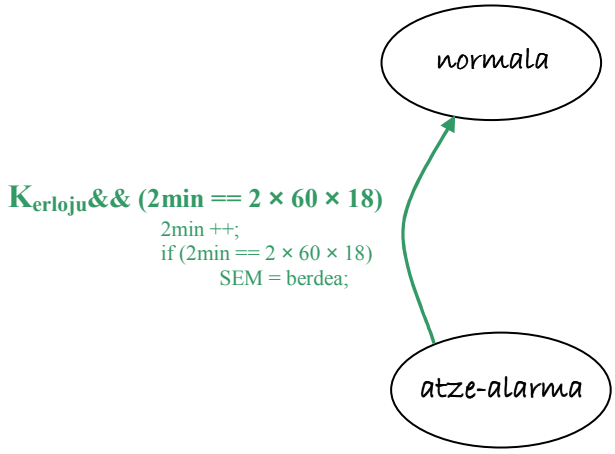
```
static int tik = 0;

tik++;
if (tik == 18)
 {
   tik = 0;
   if (egoera_automata == atze-alarma)
        2min++;
 }
```

Baina bigarren aukera honek arazo bat dakar: esan bezala, `tik` aldagaia 0 balioarekin hasieratuko dugu exekuzioaren hasieran, eta gero segundo bakoitzean berriro (`tik` = 18 denean, `tik` = 0 egiten dugu berriro, etenen kopurua beste behin kontatzen hasteko), baina ez edozein unetan, bestela segundoen kontaketa jarraitua ondoratuko baikenuke (hau da, `tik` aldagaia, exekuzioan zehar, nahi dugunean hasieratuz gero, segundo batek 10 `tik` bakarrik iraun lezake, ez 18, edo 5 `tik`, edo…). Horregatik, denbora-tarte jakin bat kontrolatu nahi dugunean -2 minutu, esate baterako $-$`,` orduan ez dugu `tik` hasieratuko gure denbora-tartea kontatzen hasi behar denean, gure aldagai berezia baizik $-$ `2min` aldagaia, esaterako $-$. Zer gerta daiteke orduan? Bada, agian `tik` aldagaia 17 balioarekin harrapatu dugula 2min aldagaia hasieratu dugunean. Horrela, erlojuaren zerbitzu-errutina berriro exekutatzen denean, `tik` = 18 izango da, eta gure baldintza betetzen bada, `2min` aldagaia inkrementatuko da batean, segundo oso bat pasatu dela adieraziz, baina ez da segundo oso bat pasatu `2min` aldagaia hasieratu dugunetik, `tik` bat bakarrik pasatu da! Horrek esan nahi du gure denbora-tartea neurtzean ia-ia segundo bateko errorea gerta daitekeela, ausaz, kontrolik gabe, bigarren aukera hau erabiliz gero (lehenengo aukeran ez da horrelakorik gertatzen, gure aldagaia ez delako `tik` aldagaiaren menpekoa). Horregatik, denbora-tarte laburrak kontrolatu behar direnean, bigarren aukera hau ez da gomendagarria, errorea handiegia izan baitaiteke.

- Trantsizio posibleak:

a) Beraz, 2 minutuko tartea pasatzen bada kontrolpeko zonara hegaztirik sartu gabe, sistema *normala* egoerara igaroko da (esan bezala, erlojuaren zerbitzu-errutinarako, 1. aukera hartuko dugu kontuan soilik automatan). Semaforoa berde jarri beharko dugu, hegazkinari abisatzeko maniobra egin dezakeela.



Eta zerbitzu-errutinan honelaxe adieraziko dugu egoera-trantsizioa:

Erloju.2) Egoera-trantsizioa, beharrezkoa baldin bada, hau da 2 minutuak pasatu baldin badira (1. aukera):

```
if (2min == 2 × 60 × 18)
   {
        SEM = berdea;
        egoera_automata = normala;
   }
```

Gogoan izan erlojuaren zerbitzu-errutinaren lehenengo eginkizunean baldintza bat jarri dugula, hau da, `2min` aldagaia bakarrik eguneratu behar da sistema *atze-alarma* egoeran baldin badago. Horregatik, bigarren eginkizun hau ere, aurrekoaren baldintzaren barruan egongo da, hau da `2min` aldagaiaren muga-balioa egoera horretan bakarrik begiratuko dugu, ez bestetan.

b) Baina 2 minutuko tartea pasatu baino lehen hegaztiak berriro sartzen badira kontrolpeko zonara, orduan sistema *alarma* egoerara pasatuko da berriz ere, hegazti guztiak berriro uxatu arte.

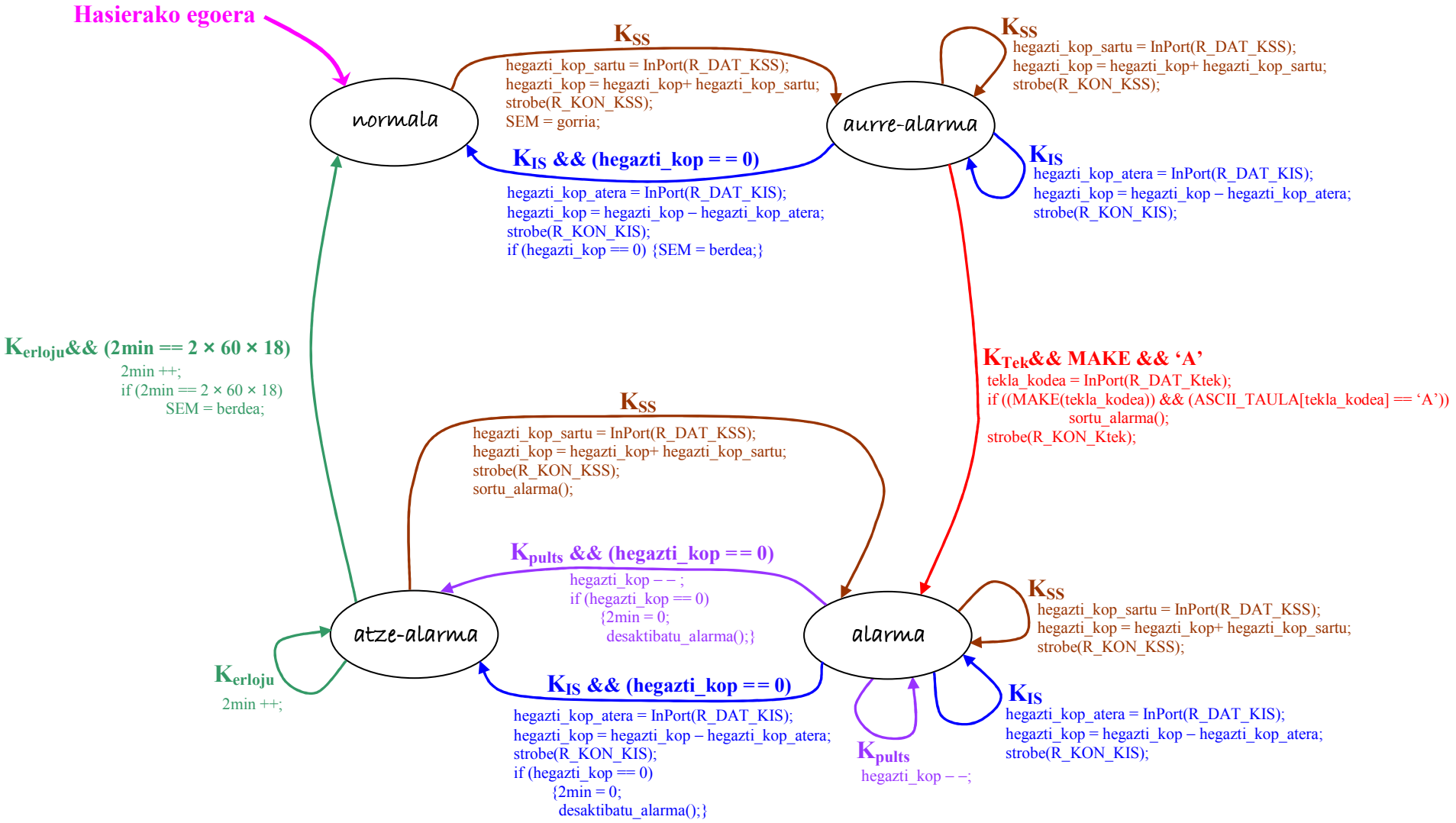
Hala, honelaxe geratuko zaigu SS sentsorearen zerbitzu errutina egoera-trantsizio honi dagokionez:

SS.5) Egoera-trantsizioa, beharrezkoa baldin bada:

```
if (egoera_automata == atze-alarma)
     {
          sortu_alarma();
          egoera_automata = alarma;
     }
```

Dagoeneko, xehetasun handiz aztertu dugu sistema, eta automata eta periferikoen zerbitzu-errutinen eginkizunak zehaztu ditugu. Orain, automataren irudi partzialak bakar batean biltzea, eta programa nagusiaren eta zerbitzu-errutinen kodea idaztea besterik ez zaigu falta.

Kasu posible guztiak denak batera bilduz, honelaxe geratuko da automata:



Eta, orain, kodea idaztea besterik ez da falta. Honako hauek idatzi behar ditugu: programa nagusia, non begizta etengabea exekutatuko baita, eta periferikoen etenen zerbitzu-errutinak: K_{SS}, K_{IS}, $K_{teklatu}$, K_{pults} eta K_{erloju}.

Programa nagusia.

Programa nagusian, hainbat ekintza egin behar dira, hala nola: aldagaiak hasieratu (behar izanez gero), eten-bektorearen hainbat osagai aldatu (programaren hasieran) eta berreskuratu (bukaeran) –gure zerbitzu-errutinak exekutatuko direla ziurtatzeko, eta ez balizko sistema eragilearenak–, eta begizta bat ezarri gure programa behin eta berriro exekuta dadin, amaitu behar ez den bitartean.

- Hasieratu beharreko aldagaiak:

  Hasteko, suposatuko dugu sistema martxan jartzen denean, kontrolpeko zonan ez dela hegaztirik izango, eta sistema *normala* egoeran egongo dela. Garbi izan behar dugu, beraz, sistemaren funtzionamendu egokia ziurtatzeko beharrezkoa dela uneoro jakitea zein egoeratan dagoen sistema, eta horretarako aldagai global bat behar da, `egoera_automata` izendatuko duguna, zeinaren hasierako balioa *normala* izango baita. Horrekin batera, `hegazti_kop` aldagaia 0 balioarekin hasieratu beharko dugu, eta semaforoa berde jarri.

  Beste aldetik, erlojuaren kontrolerako 2. aukera aintzat hartuko bagenu (erlojuaren kontroladorearen etenak kontatzen joatea, alegia), `tik` aldagaia ere hasieratu beharko genuke, gure programan ongi kontrolatzeko "gure denbora". Baina, ikusi dugun bezala, `tik` aldagaia erlojuaren zerbitzu-errutinan bertan hasieratu daiteke `static int tik = 0` eginez, edo, bestela, programa nagusiaren hasieran.

  Hala, honako hauek dira beharrezko hasieraketak:

  - ✓ `egoera_automata = normala`
  - ✓ `hegazti_kop = 0`
  - ✓ `SEM = berdea`
  - ✓ `tik = 0` (bigarren aukeran, soilik, eta zerbitzu-errutinan ez badugu egiten `static int tik = 0`)

  Azpimarratu behar dugu ez dela beharrezkoa beste aldagaiak hasieratzea, erabili behar diren unean bertan balio egokiarekin hasieratzen direlako. Esate baterako, `2min` aldagaia *alarma* egoeratik *atze-alarma* egoerara pasatzean hasieratuko da 0 balioarekin.

- Aldatu eta berreskuratu beharreko eten-bektorearen osagaiak:

  Sistema hau kontrolatzeko PC arrunt bat erabiltzen badugu, orduan guztiz beharrezkoa da eten-bektorearen hainbat osagai aldatzea, gure periferikoen etenei dagozkienak hain zuzen, kontrola guk nahi dugun moduan egin dezan PCak, hau da, gure periferikoek etena eskatzen dutenean, guk idatzitako zerbitzu-errutinak exekuta daitezen, eta ez sistema eragileak aurre programatuak dituenak.

Hala, kasu honetan honako hauek aldatu behar ditugu: erlojuarena (eten-bektorearen 0x1C sarreran dagoena), teklatuarena (0x09 sarrerakoa), SS sentsorearena (0x0A sarrerakoa), IS sentsorearena (0x0B sarreran dagoena) eta pultsadorearena (0x0C sarreran dagoena). Horretarako, suposatuko dugu jadanik idatzita dagoen funtzioa badugula, eta nahikoa dela aldatu beharreko sarrera parametro gisa pasatzea. Honelaxe:

✓ `Aldatu_EB(0x1C, 0x09, 0x0A, 0x0B, 0x0C)`

Era berean, gure programa amaitzen denean, orduan eten-bektorearen osagai horiek berreskuratu beharko ditugu, sistema eragileak PCaren kontrola har dezan berriz ere. Honelaxe:

✓ `Berreskuratu_EB(0x1C, 0x09, 0x0A, 0x0B, 0x0C)`

- Begiztaren kontrola:

  Ziurtatu behar dugu gure programa etengabe exekutatzen dela, guk amaitzeko esaten ez diogun bitartean. Horretarako, begizta bat jarriko dugu, zeina behin eta berriro errepikatuko baita, amaitu artean. Kasu honetan, enuntziatuari erreparatuz, ez dago aurreikusita inolako aukerarik gure programaren exekuzioari amaiera emateko. Hori dela eta honako hau erabil dezakegu:

  ✓ `while (true)` edo `while (1)`

  Baina, oro har, programa amaitu beharko balitz, aldagai global bat erabil genezake hori adierazteko:

  ✓ `while (!amaitu)`

  Hala, `amaitu` aldagaiaren hasierako balioa 0 izango litzateke, eta uneren batean `amaitu = 1` egingo genuke, programaren exekuzioa amaitzeko.

  Azken hau berdin egin zitekeen aldagai berezi bat erabili beharrean, egoera jakin bat erabiliz:

  ✓ `while (egoera_automata != azkena)`

- Begiztaren gorputza:

  Aurreko `while` horren barruan, gure aplikazioaren gorputza idatzi beharko dugu, hau da, gure programak egin behar duena.

  Oro har, gorputza horretan inkesta bidez sinkronizatzen diren periferikoen inkestak egiten dira, non periferikoaren egoera-erregistroa aztertzen den, detektatu ahal izateko noiz dagoen prest periferikoa datu bat transferitzeko, edo egin behar duena egiteko. Inkesta begiztaren barruan egonik, horrek esan nahi du dagozkien ekintzak behin eta berriro errepikatuko direla programa amaitzen ez den bitartean.

  Ariketa honetan, periferiko guztiak etenen bidez sinkronizatzen direnez, begiztaren gorputza hutsik egongo da, programa nagusiak ez baitu ezer egin behar, ekintza guztiak periferikoen zerbitzu-errutinetan egiten baitira. Hori dela eta, honela idatziko dugu begizta:

  ```
  while (true);
  ```

  Puntu eta koma (`;`) horrek adierazten du begiztaren baldintza betetzen den bitartean ez dela ezer egiten, ez baitago sententzia bat bera ere.

Laburbilduz, honelaxe geratuko da programa nagusiaren kodea:

```
void main()
{
    //aldagaien hasieraketa
 egoera_automata = normala;
 hegazti_kop = 0;
 SEM = berdea;
 tik = 0;  // (erlojuaren bigarren aukeran, soilik, static ez bada erabiltzen)
    //eten-bektoreko osagaien aldaketa
 Aldatu_EB(0x1C, 0x09, 0x0A, 0x0B, 0x0C);
    //begizta nagusia
    while (true);  //begiztaren barruan ez da ezer egiten
    //eten-bektoreko osagaien berreskurapena programaren amaieran
 Berreskuratu_EB(0x1C, 0x09, 0x0A, 0x0B, 0x0C);
 }
```

- K_{SS} kontroladorearen zerbitzu-errutina.

Jadanik ikusi dugu zein diren zerbitzu-errutina honetan exekutatu beharreko eginkizunak. Orain, horiek ordenatzea besterik ez zaigu falta. Hona hemen:

```
void interrupt KSS_ZE()
{
    //etena sortu den unean sartutako hegaztien kopurua irakurri
 hegazti_kop_sartu = InPort(R_DAT_KSS);
    //kontrolpeko zonaren barruan dauden hegaztien kopurua eguneratu
 hegazti_kop = hegazti_kop + hegazti_kop_sartu;
    //"strobe" sekuentzia kontrol-erregistroaren gainean
 strobe(R_KON_KSS);
    //egoera-trantsizio egokiak bideratu
 if (egoera_automata == normala)
  {
   SEM = gorria;
   egoera_automata = aurre-alarma;
  }
 if (egoera_automata == atze-alarma)
  {
   sortu_alarma();
   egoera_automata = alarma;
  }
 Eoi();
 IRET;
}
```

Azpimarratu nahi dugu K_{SS} kontroladoreak edozein egoeratan sor dezakeela eten eskaera (ikus automata); horregatik, lehenengo ekintzak (`InPort`, `Strobe`) ez dira egoera baten menpekoak: etena sortzen den guztietan exekutatu behar dira.

➢ K_{IS} kontroladorearen zerbitzu-errutina.

```
void interrupt K_IS_ZE()
{
        //etena sortu den unean ateratako hegaztien kopurua irakurri
  hegazti_kop_atera = InPort(R_DAT_KIS);
        //kontrolpeko zonaren barruan dauden hegaztien kopurua eguneratu
  hegazti_kop = hegazti_kop - hegazti_kop_atera;
        //"strobe" sekuentzia kontrol-erregistroaren gainean
  strobe(R_KON_KIS);
        //egoera-trantsizio egokiak bideratu
  if (hegazti_kop == 0)
    {
         if(egoera_automata == aurre-alarma)
              {
                 SEM = berdea;
                 egoera_automata = normala;
              }
         else if (egoera_automata == alarma)
              {
                 2min = 0;
                 desaktibatu_alarma();
                 egoera_automata = atze-alarma;
              }
     }
  Eoi();
  IRET;
}
```

K_{IS} kontroladorearen kasuan ez da gertatzen K_{SS} kontroladorearen kasuan gertatzen zena; hau da, K_{IS}-k ezin du egoera guztietan sortu etena (ikus automata). Izan ere, sistemaren egoera *normala* denean, edo *atze-alarma* denean, orduan kontrolpeko zonan ez dago hegaztirik, eta, ondorioz, ezinezkoa da K_{IS}-k etena sortzea, ez baitu inoiz detektatuko hegaztiak ateratzen direla kontrolpeko zonatik. Baina pentsatzen badugu sentsorearen funtzionamendua ez dela guztiz fidagarria, orduan zerbitzu-errutina honen hasieran baldintza bat jar dezakegu, hasierako ekintzak (`InPort`, `Strobe`) bakar bakarrik exekuta daitezen egoera jakin batzuetan eta ez beste batzuetan (hori da teklatuaren eta pultsadorearen kasuan egin duguna, bi kasu horietan litekeena delako erabiltzaileak tekla bat edo pultsadorea nahi gabe sakatzea egoera desegoki batean, eta baldintza jarri ezean sistemak funtzionamendu desegokia erakutsiko luke). Funtzionamendu egokia ziurtatu nahiko bagenu, baldintza hau jarriko genuke zerbitzu-errutina honen hasieran:

```
if ((egoera_automata == aurre-alarma)||( egoera_automata == alarma))
  {  hegazti_kop_atera = InPort(R_DAT_KIS);
     ...
```

Hau da, baldintza gehituz gero, zerbitzu-errutina honen eginkizunak sistema *aurre-alarma* edo *alarma* egoeretan dagoenean bakarrik exekutatuko dira, beste egoeretan zerbitzu-errutina exekutatuko den arren, `Eoi()` eta `IRET` bakarrik exekutatuko dira.

- $K_{teklatu}$ teklatuaren zerbitzu-errutina.

```
void interrupt Ktek_ZE()
{
        //sakatutako teklari dagokion posizio-kodea irakurri
  tekla_kodea = InPort(R_DAT_Ktek);
        //"strobe" sekuentzia kontrol-erregistroaren gainean
  strobe(R_KON_Ktek);
        //egoera-trantsizio egokiak bideratu
 if ((MAKE(tekla_kodea))&&(ASCII_TAULA[tekla_kodea]=='A')
    &&(egoera_automata == aurre-alarma))
    {
       sortu_alarma();
       egoera_automata = alarma;
    }
  Eoi();
  IRET;
}
```

Kasu honetan, datu-erregistroa egoera guztietan irakurtzen da, eta `strobe` ere, egoera guztietan egiten da, baina sakatu dena A tekla izanez gero, bakarrik egingo zaio kasu sistema *aurre-alarma* egoeran dagoenean, bestela ez zaio kasurik egingo.

- K_{pults} pultsadorearen zerbitzu-errutina.

```
void interrupt Kpults_ZE()
{
        //kontrolpeko zonaren barruan dauden hegaztien kopurua eguneratu
  if (egoera_automata == alarma)
     {
        hegazti_kop --;
       //egoera-trantsizio egokiak bideratu
        if (hegazti_kop == 0)
       {
          2min = 0;
          desaktibatu_alarma();
          egoera_automata = atze-alarma;
       }
     }
  Eoi();
  IRET;
}
```

Kasu honetan ere, zerbitzu-errutina honi dagozkion ekintzak sistema *alarma* egoeran dagoenean bakarrik exekutatuko dira, bestela pultsadoreari ez zaio kasurik egingo, eta `Eoi()` eta `IRET` bakarrik exekutatuko dira.

- K_{erloju} erlojuaren zerbitzu-errutina (1. aukera).

```
void interrupt K_erloju_ZE()
{
        //2min aldagaia eguneratu erlojuaren eten bakoitzean
  if (egoera_automata == atze-alarma)
   {
     2min++;
 //egoera-trantsizio egokiak bideratu
     if (2min == 2 × 60 × 18)
     {
       SEM = berdea;
       egoera_automata = normala;
     }
   }
  IRET;
 }
```

Kasu honetan ere, zerbitzu-errutina honi dagozkion ekintzak sistema *atze-alarma* egoeran dagoenean bakarrik exekutatuko dira, bestela erlojuari ez zaio kasurik egingo eta errutina honen bukaeran dagoen IRET exekutatuko da soilik.

- K_{erloju} erlojuaren zerbitzu-errutina (2. aukera).

```
void interrupt K_erloju_ZE()
{
 static int tik = 0;
       //segundoak kontrolatu
 tik++;
 if (tik == 18)
      {
       tik = 0;
             //2min aldagaia eguneratu segundoan behin
        if (egoera_automata == atze-alarma)
        {
           2min++;
             //egoera-trantsizio egokiak bideratu
           if (2min == 2 × 60)
         {
              SEM = berdea;
              egoera_automata = normala;
         }
        }
      }
 IRET;
}
```

Jadanik komentatu dugu zein izan daitekeen bigarren aukera honen arazoa: `2min` aldagaia 0 balioarekin hasieratzen dugunean `tik` 17 balioarekin harrapatzen badugu, orduan `2min` aldagaia batean inkrementatuko da segundo bat pasatu baino askoz lehenago. Horregatik baztertuko dugu aukera hau.