PX4 Agricultural Spraying Copter Emergency Procedures

Date : 21.11.2022
Version : 01
Change : 00
Document : Report
Author : Ömer Can VURAL

Contents

1 INTRODUCTION

It is desired to give the Z4 autopilot software, which will be used in the TARDRONE1 agricultural spraying copter developed within Project, the ability to process and make decisions regarding possible malfunctions that may occur during operation. Accordingly, plans have been made and compiled in this report. Within the scope of the studies carried out, the device's mission was observed to be completed in three main stages:

- Takeoff
- Flight
- Landing

and the foreseen problems were examined under these stages.

The aim is to overcome the foreseen situations with the new functions added to the Z4 autopilot software. Since the software is in the development phase, these new functions will be examined in a pseudo-code structure and with the help of algorithms, thus being conceptual.

For this reason, exact numerical values, all possible situations, or commands that need to be processed may not be specified in various sections. Code syntax and algorithmic structures may remain superficial/may not reflect reality. The aim is to create a concept for the general idea.

2 SUMMARY

The TARDRONE1 agricultural spraying copter performs its task of spraying an agricultural area in three stages. These stages can be listed as:

- Takeoff
- Flight
- Landing

The Z4 autopilot software used in the TARDRONE1 copter currently has the features of scanning an area at certain intervals within a virtual geographic boundary created on a map with the ground control station, being able to detect whether it flies outside this boundary during its scanning route, spraying chemicals accordingly, and sending certain values to the ground control station.

However, there are current problems due to the spraying process being initiated before takeoff or before reaching the mission's starting point if the device's takeoff location is within the virtual geographic boundary representing the agricultural area, the geographic boundary data still being processed very superficially, and certain parameters such as chemical tank level and battery level not being processed.

Therefore, to eliminate these problems, ensure the device performs its mission as desired, and meet customer expectations, the mission stages in the Z4 Autopilot software must be clearly defined and necessary additional parameters must be taken into account.

In this study, the agricultural module functions within the Z4 Autopilot software, which gives the device its "agricultural spraying copter" feature, were examined. New features to be added to the module and the necessary inter-module communication infrastructure were analyzed with the help of pseudo-code and algorithms, as if they were added to the "tarim_modulu_main" function at './src/modules/navigator/navigator_main.cpp' within the autopilot software.

Among the proposed solutions designed at the concept stage, those decided for implementation will first be tested on the QGroundControl ground control station program after being added to the Z4 autopilot software. Upon successful completion of these tests, real-world tests will commence.

3 METODOLOJİ

The foreseen problems resulting from planning studies are aimed to be solved by introducing new features to the Z4 Autopilot software, specifically through new functionalities added to the existing "tarim_modulu_main" function within the autopilot software's navigator module.

The coding work will be carried out using the Visual Studio Code editor software.

Simulation studies will be performed via the Gazebo simulation and QGroundControl ground control station software.

The aforementioned studies will be conducted on a virtual machine with Ubuntu 20.04 LTS operating system installed in VirtualBox software.

uORB will be used for module communication within the autopilot software, and the MAVLink communication protocol will be used for sending messages from the autopilot to the ground control station.

Sending messages from the ground control station to the autopilot will be simulated with the MAVLink-Messaging software developed by Project

The work is conceptual and will be planned with the help of pseudo-code and algorithms, followed by simulation studies. To create a conceptual work plan,

3.1 Addressing Possible Situations During Application

Situations that need to be controlled by the device during mission execution, according to the plans, can be categorized under two headings:

- Situations related to new features planned for addition
- Emergency situations

Situations related to new features are defined with the aim of increasing customer satisfaction, improving spraying efficiency, and ensuring safety.

Emergency situations, on the other hand, cover scenarios where the device moves unexpectedly during spraying, experiences a problem with its physical hardware, suffers a loss of connection, or involves certain parameters like chemical/battery fill levels. Additionally, existing software bugs that need correction will also be resolved according to a specific plan.

3.1.1 New Features Planned for Addition

Within the scope of Project's work, the new features planned to be added to the agricultural module of the existing autopilot software in the TARDRONE1 agricultural spraying copter are:

- If the device deviates **±5m from the virtual geographic boundary**, it should enter **HOLD-MODE** (which keeps the device stationary at its last position) for 5 seconds to await a command from the pilot. If no command is received within 5 seconds, the device should automatically return to the ground control point.
- If the mission returns to the ground control point for any reason (emergency, command, etc.), the device should **save its last position** and be able to resume from that last point when it is ready for use again.
- The **chemical spraying speed should dynamically change** based on ground speed and altitude parameters.

3.1.2 Emergency Situations to Address

Within the scope of Project's work, the potential malfunctions that may occur during the TARDRONE1 agricultural spraying copter's mission execution are listed as follows:

- **Monitoring of GPS Jamming value** This emergency situation varies according to the amount of noise in receiving GPS data. A high GPS Jamming value can lead to the device's position information not being obtained, consequently causing the device to crash, get lost, etc. Security criteria have been evaluated as follows:
    - **0 – 40**: GPS Jamming value is within the safe range.
    - **40 – 60**: GPS Jamming value poses a risk. A warning will be sent to the ground control station.
    - **> 60**: GPS Jamming value poses a danger. A warning will be sent to the ground control station, and a command will be awaited from the pilot.
- **Battery level** Failure to monitor the battery level can cause the device to crash when its battery runs out, lead to hardware malfunctions, etc. Security criteria have been evaluated as follows (percentage not specified):
    - **Low**: A "Return to ground control point" warning will be issued to the ground control station.
    - **Critical**: Enter HOLD-MODE for 5 seconds and await a command from the pilot. If no command is received, return to the ground control point.
    - **Danger**: Land at your current location. Send a warning to the ground control station.
- **Chemical tank level** If the chemical tank becomes empty (chemical might be finished, problem with tank connection, etc.), the device cannot perform its mission. Security criteria for refilling the chemical tank have been evaluated as follows:
    - **Full**: Chemical is in the tank.
    - **Empty**: Chemical tank is empty, send a warning to the ground control station, return to the ground control point.
- **Link Loss** This means the connection that allows the pilot to control the device has been cut. Security criteria have been evaluated as follows:
    - **GPS Available**: Send a message to the ground control station and return to the ground control point.
    - **GPS Unavailable**: Make an emergency landing at your current location.

The foreseen emergency situations listed above represent emergencies that may occur while the device is in flight. Possible emergency situations that may occur in takeoff and landing modes are classified as errors.

3.1.3 Error Conditions

Within the scope of Project's work, for situations causing errors when the flight and landing modes are active in the TARDRONE1 agricultural spraying copter:

If the **chemical spraying is active** while the device is taking off, going to the mission start point, returning to the designated point after completing the mission, or landing, it indicates an error where the performed functions are not happening according to the desired mission sequence, and it originates from the autopilot software, ground control station configurations, or a hardware malfunction.

3.2 Application Concept for Planned New Features

The Z4 Autopilot, which is the autopilot software for the TARDRONE1 agricultural spraying copter, had a **geofence parameter** configured from the QGroundControl ground control station. In PX4, the underlying structure of the Z4 Autopilot, this feature was used to prevent the device from going outside the determined geographic boundary. In the Z4 autopilot, this feature, implemented into the agricultural module, returns a value that helps determine whether the device is inside or outside the given geographic boundary.

The new features to be added will be built upon the agricultural module mentioned here.

3.2.1 Adjusting the Device's Geographic Boundary Tolerance

The first of the new features planned for addition under the heading "Addressing Possible Situations During Application" was for the device to enter **HOLD-MODE** for 5 seconds to await a command from the pilot if it moves **±5m away from the virtual geographic boundary**, and then automatically return to the ground control point if no command is received within 5 seconds.

Currently, the geofence control in the agricultural module uses the "**Point Inclusion in Polygon Test – PNPOLY**" algorithm within the 'isInsidePolygonOrCircle' function, located in the 'geofence' block of the 'navigator' module, based on the device's current position.

And before the PX4 autopilot software was configured as Z4, it had another algorithm that detected when the device approached the geofence boundary, issued a warning, and stopped the device (this was canceled when the agricultural module infrastructure was being prepared as it was not used!). This algorithm is implemented through the 'geofence_breach_checker' and 'getFenceViolationTestPoint' functions located in the 'GeofenceBreachAvoidance' collective block within the 'navigator_main.cpp' block of the autopilot software's 'navigator' module, where the agricultural module is also located. The algorithm simply calculates the shortest distance from the device's current location to the geofence boundary (Distance of point from line) and determines a buffer distance. If the distance between the two points is less than the buffer distance, the device slows down and changes mode to avoid breaching the boundary.

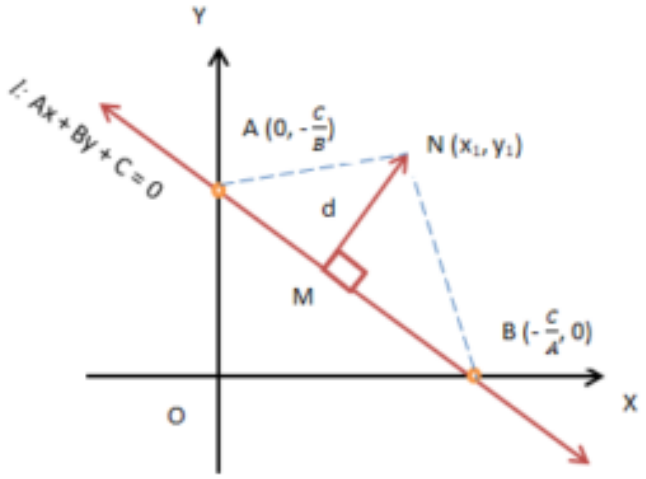

Figure 1 : https://d1whtlypfis84e.cloudfront.net/guides/wp-content/uploads/2018/05/23153857/distance1-300x194.png

To add the geographic boundary tolerance feature, both of these algorithms should be utilized. Since the area inside the polygon will already be the agricultural land, and the device will be following its mission route within this section, there's no need to provide a tolerance inwards towards the geographic boundary. Instead, a 5-meter buffer zone should only be created outwards from the polygon. Therefore, the condition depends on the device being outside the polygon and within 5 meters of it.

In this scenario;

PSEUDO CODE:

```
1.  //          ./src/modules/navigator/navigator_main.cpp
2.  void Navigator::tarim_modul_main(bool &have_geofence_position_data, double lat, double
    lon, double alt)
3.  {
4.    #######
5.    if(!(PoligonunIcinde()))
6.    {
7.              İf(Cografi_Sinir_Uzakligi >= 5m)
8.              {
9.                        ilac_puskurtme_kapali();
10.                       HOLD_MODE();
11.                       Yer_Kontrol_istasyonuna_Mesaj_Gonder();
12.                       if(Komut_Geldi() && (Mesaj_Gonderme_Suresi < 5000ms))
13.                       {
14.                                 Komutu_isle();
15.                       }
16.                       else
17.                       {
18.                                 Yer_Kontrol_Noktasina_Don();
19.                       }
20.             }
21.             else
22.             {
23.             İlac_puskurtme_kapali();
24.             }
25.   }
26.   else
27.   {
28.             İlac_puskurtme_aktif();
29.   }
30.   #######
31. }
32.
```

This defines the general conditioning for the feature we're trying to implement in the autopilot.

This conditioning ensures:

- If the device is **inside the geographic boundary**, it continues the chemical spraying operation.
- If the device is **outside the geographic boundary AND its distance to the geographic boundary is 5 meters or more**, the chemical spraying operation is terminated, and the device enters **HOLD_MODE**. A warning message is sent to the ground control station to await a command from the pilot. The time the message was sent is stored in memory. If **less than 5 seconds have passed** since the message was sent AND a command has been received from the pilot, the command is processed. If the message sending duration has exceeded 5 seconds, the device returns to the ground control point.

3.2.2 Device Resuming from the Point it Left Off

The device might encounter an emergency during operation (e.g., battery/chemical depletion, GPS noise). Decisions the device should make in such situations are outlined under the "Emergency Situations to Address" section. However, if these decisions cause a change in the device's position, we don't want the chemical spraying process for the agricultural land to restart from the beginning. Therefore, the device must record its current position in a memory structure during each cycle.

There can be multiple methods for saving position information to memory (e.g., creating arrays/matrices, saving to external storage devices). Furthermore, since it's also desired to log the operations performed (like a logbook), storing position information in memory for each status and decision made is crucial.

Each operation performed while the device is running is saved to an SD card connected to the device. Therefore, each of the foreseen situations should be placed into an array, and each situation should update its recorded array element, with this array being continuously transferred to the SD card.

So, for this:

- A matrix should be created to store position information (latitude, longitude, altitude) in case of an emergency, and ensure it's stored at the correct index. (e.g., double memory_location [3][emergency_case_size]).
- Each column of the created matrix, which contains 3 x (number of emergency cases) elements, represents an emergency case. Each emergency case can be defined as a struct and stored vectorially in memory.

$$\text{double last\_location } [\boldsymbol{lat\ \ lon\ \ alt}][\boldsymbol{acil\_durum\_s\ size}]$$

$$\text{actuaclly} = \begin{vmatrix} batarya.lat & ilactank.lat & GPS.lat \\ batarya.lon & ilactank.lon & GPS.lon \dots \\ batarya.alt & ilactank.alt & GPS.alt \end{vmatrix}$$

In this case, the pseudo-code of the mechanism is;

```
1.  void Navigator::tarim_modul_main(bool &have_geofence_position_data, double lat, double
    lon, double alt)
2.  {
3.    double son_konum[3][acil durum sayisi];
4.    struct konum_parametreleri_s konum;
5.    struct acil_durum_s acilDurum;
6.    #######
7.    if(!(PoligonunİçindeMi()))
8.    {
9.              if(acil_durum())
10.             {
11.                       ilac_puskurtme_kapali();
12.                       acil_durum_işlevleri();
13.                       konum_kaydet();
14.             }
15.             else
16.             {
17.                       ilac_puskurtme_kapali();
18.             }
19.   else
20.   {
21.             ilaç_puskurtme_aktif();
22.   }
23.   #######
24. }
25.
26. void Navigator::konum_kaydet()
27. {
28.   double
29.   for(i=0;i<konum_boyutu;i++)
30.   {
31.             for(j=0;j<acilDurum_boyutu;j++)
32.             {
33.                       son_konum[i][j] = son_konum[i.konum_parametresi][j.acil_durum];
34.             }
35.   }
36.   kaydet(son_konum[i][j]);
37. }
38.
```

The save_position function defined in the pseudo-code essentially ensures the following:

- If an emergency occurs while the device is outside the polygon, it iterates through the emergency_case_size structure for the specific emergency. It then assigns the latitude, longitude, and altitude information to the elements of the vector at that corresponding index.

3.2.3 Dynamic Adjustment of Spraying Speed

To increase the device's application efficiency, the amplitude of the sprayed area should be maintained at a certain level, and a specific amount of chemical should be sprayed. The feature of dynamically adjusting the spraying speed reduces chemical waste, prevents spraying outside the target area, and increases spraying efficiency by adapting to different crop types.

For the spraying operation to be variable, the following factors are considered:

- **Ground speed**: The device's speed affects spraying efficiency; as speed increases, spraying speed should also increase.
- **Altitude**: The device's altitude affects spraying amplitude; as altitude increases, the sprayed area increases up to a certain point.
- **Terrain shape (slope, narrow areas, etc.)**: The agricultural land might narrow or widen in certain areas. In narrower areas, altitude and spraying speed should be proportionally reduced.
- **Crop type**: Depending on the crop cultivated, spraying might need to be performed at different speeds and from different altitudes.

These are important parameters that need attention. However, the speed and altitude parameters based on crop type will be determined by the pilot via the ground control station and are not currently relevant for the autopilot software. The terrain shape will be calculated based on the distance between the device's nearest point to the polygon, as discussed under "Adjusting the Device's Geographic Boundary Tolerance." Therefore, while changes dependent on ground speed and altitude parameters are proportional to each other, the change in spraying speed based on terrain shape is independent of other parameters.

Since the work will be carried out within a conceptual framework, and ground speed and altitude parameters influence each other, the PWM signal value range supplied to the chemical pump will initially be adjusted based on these two parameters and ratioed 5:5. This means that if a PWM value with a duty cycle between 0-1000 exists, the PID algorithm, adjusted with the help of these parameters, can cause a duty cycle change of up to ±250 (500).

The variation in spraying speed according to terrain shape applies the condition of moving 5m away from the outside of the polygon, as described under "Adjusting the Geographic Boundary Tolerance of the Device," to the inside of the polygon. Differently from performing a PID adjustment, it modifies the PWM value range via a MAP function based on its proximity to the polygon.

Accordingly, the PID algorithm that adjusts the PWM signal given to the pump is constructed as follows:

```
1.  void Navigator::tarim_modul_main(bool &have_geofence_position_data, double lat, double
    lon, double alt)
2.  {
3.    #######
4.    if(!(PoligonunİçindeO))
5.    {
6.              #######
7.    else
8.    {
9.              ilaç_puskurtme_aktif();       // ilaç püskürtme aktif
10.             PWM = yerVPID() + altPID();
11.             ilac_pompasi_PWM = map_shape(PWM, minPWM, maxPWM, 0, 5m);
12.   }
13.   #######
14. }
15. void yerVPID()
16. {
17.   son_zaman = anlik_zaman;
18.   anlik_zaman = zamantut();
19.   gecen_zaman = anlik_zaman - son_zaman;
20.
21.   hata = anlik_puskurtme_orani - hedef_puskurtme_orani;
22.   hiz_PID_oran = oran_hiz * hata;
23.   if(hata < x)
24.   {
25.             hiz_PID_integral = hiz_PID_integral + (integral_hiz * hata);
26.   }
27.   hiz_PID_turev = turev_hiz * ((hata - eski_hata) / gecen_zaman);
28.   hiz_PID = hiz_PID_oran + hiz_PID_integral + hiz_PID_turev;
29.   eski_hata = hata;
30. }
31. void altPID()
32. {
33.   son_zaman = anlik_zaman;
34.   anlik_zaman = zamantut();
35.   gecen_zaman = anlik_zaman - son_zaman;
36.
37.   hata = anlik_puskurtme_orani - hedef_puskurtme_orani;
38.   alt_PID_oran = oran_alt * hata;
39.   if(hata < x)
40.   {
41.             alt_PID_integral = alt_PID_integral + (integral_alt * hata);
42.   }
43.   alt_PID_turev = turev_alt * ((hata - eski_hata) / gecen_zaman);
44.   alt_PID = alt_PID_oran + alt_PID_integral + alt_PID_turev;
45.   eski_hata = hata;
46. }
47. uint64_t map_shape(int pwmIn, int pwmMin, int pwmMax, double distMin, double distMax)
48. {
49.   return (pwmIn - pwmMin) * (distMax - distMin) / (pwmMax - pwmMin) + distMin;
50. }
51.
```

This outlines the conceptual idea for the planned coding work aimed at implementing the variable spraying feature.

3.3 Emergency Situations to Address

Beyond the features to be added to the Z4 Autopilot software, for full customer satisfaction and increased efficiency, the device needs to be able to evaluate certain situations and return correct decisions. As a result of planning, these situations have been termed 'emergency situations' and involve work aimed at resolving errors stemming from the device's software and hardware requirements.

The sources of existing errors are:

- The current agricultural module performs the chemical spraying operation based only on whether it is within the geographic boundary, without considering the flight mode.
- Since battery level and chemical tank level are not monitored, the device will try to continue its mission as if spraying chemicals even if the chemical runs out, or it will crash in the field without notifying the pilot when the battery dies.
- Since GPS noise is not monitored, in case of position information loss, the device may drift to other fields due to reasons such as wind or incorrect mapping.

In emergency situations, the situations addressed are located in different modules, but the agricultural module is part of the "navigator" module. Therefore, inter-module interaction is required to implement the same mechanism for all emergencies into the agricultural module. For this reason, a **uORB network** must be established between the modules from which we will receive emergency information and the 'navigator' module, which hosts the agricultural module. The goal is to build a comprehensive set of conditions that account for each emergency. The obtained data can then be transmitted from the 'navigator' module to the ground control station via the **MAVLink protocol**.

3.3.1 GPS Noise Monitoring

Decisions to be made regarding situations caused by GPS noise are handled in the "gps_failure" block within the "navigator" module. To connect these mechanisms to the agricultural module within the "navigator_main" block, a value assignment should be made as a result of GPS loss situations and published on the uORB channel under the name "**TARIM_MODULU**." Let's call this value gps_fail.

```
1.  void Navigator::tarim_modul_main(bool &have_geofence_position_data, double lat, double
    lon, double alt)
2.  {
3.    #######
4.    struct tarim_modulu_nav_s tarimNav;
5.    uORB::Subscription _tarim_modulu_sub
6.              {
7.              ORB_ID(tarim_modulu), 0
8.              };
9.    #######
10.   if(!(PoligonunİçindeMi()))
11.   {
12.             if(acil_durum())
13.             {
14.                       ilac_puskurtme_kapali();
15.                       switch acil_durum
16.                       case acil_durum == gps_fail
17.                                 if(40 < gpsjam < 60)
18.                                 {
19.                                           Yer_Kontrol_istasyonuna_Mesaj_Gonder();
20.                                 }
21.                                 if(gpsjam > 60)
22.                                 {
23.                                           HOLD_MODE();
24.                                           Yer_Kontrol_istasyonuna_Mesaj_Gonder();
25.                                           if(Komut_Geldi() && (Mesaj_Gonderme_Suresi <
    5000ms))
26.                                           {
27.                                                     Komutu_isle();
28.                                           }
29.                                           else
30.                                           {
31.                                                     Yer_Kontrol_Noktasina_Don();
32.                                           }
33.                                 }
34.                                 else
35.                                 {
36.                                           break;
37.                                 }
38.                       break;
39.                       #######
40.             }
41.             else
42.             {
43.                       ilac_puskurtme_kapali();
44.             }
45.   }
46.   else
47.   {
48.             ilaç_puskurtme_aktif();
49.   }
50.   #######
51. }
52.
53.
```

3.3.2 Battery Level Monitoring

The current software for the device doesn't monitor battery levels. However, the underlying PX4 framework includes a decision-making mechanism for the autopilot based on battery levels. According to this mechanism, if the battery level is low (default: less than 50%), a landing warning is issued to the user. At a critical level, a return-to-control-point warning is issued, and the drone automatically returns to the control point. At a dangerous level, an emergency landing is initiated.

```
1.    void Navigator::tarim_modul_main(bool &have_geofence_position_data, double lat, double
    lon, double alt)
2.  {
3.    #######
4.    struct tarim_modulu_nav_s tarimNav;
5.    uORB::Subscription _tarim_modulu_sub
6.              {
7.              ORB_ID(tarim_modulu), 0
8.              };
9.    #######
10.   if(!(PoligonunİçindeMi()))
11.   {
12.             if(acil_durum())
13.             {
14.                       ilac_puskurtme_kapali();
15.                       switch acil_durum
16.                       #######
17.                       case acil_durum == batarya
18.                                 if(30 < batarya_durum < 50)
19.                                 {
20.                                           Yer_Kontrol_istasyonuna_Mesaj_Gonder();
21.                                 }
22.                                 if(10 < batarya_durum < 30)
23.                                 {
24.                                           HOLD_MODE();
25.                                           Yer_Kontrol_istasyonuna_Mesaj_Gonder();
26.                                           if(Komut_Geldi() && (Mesaj_Gonderme_Suresi <
    5000ms))
27.                                           {
28.                                                     Komutu_isle();
29.                                           }
30.                                           else
31.                                           {
32.                                                     Yer_Kontrol_Noktasina_Don();
33.                                           }
34.                                 }
35.                                 if(batarya_durum < 10)
36.                                 {
37.                                           LAND_MODE();
38.                                           Yer_Kontrol_istasyonuna_Mesaj_Gonder();
39.                                 }
40.                                 else
41.                                 {
42.                                           break;
43.                                 }
44.                       break;
45.                       #######
46.             }
47.             else
48.             {
49.                       ilac_puskurtme_kapali();
50.             }
51.   }
52.   else
53.   {
54.             ilaç_puskurtme_aktif();
55.   }
56. }
```

3.3.3 Monitoring Chemical Tank Level

We are only monitoring for the presence or absence of chemicals in the tank, without regard for the exact fill level. Therefore, only an "**ilac_yok**" (chemical_empty) emergency will be added.

```
1.  void Navigator::tarim_modul_main(bool &have_geofence_position_data, double lat, double
    lon, double alt)
2.  {
3.    #######
4.    struct tarim_modulu_nav_s tarimNav;
5.    uORB::Subscription _tarim_modulu_sub
6.              {
7.              ORB_ID(tarim_modulu), 0
8.              };
9.    #######
10.   if(!(PoligonunİçindeMi()))
11.   {
12.             if(acil_durum())
13.             {
14.                       ilac_puskurtme_kapali();
15.                       switch acil_durum
16.                       #######
17.                       case acil_durum == ilac_yok
18.                                 Yer_Kontrol_istasyonuna_Mesaj_Gonder();
19.                                 Yer_Kontrol_Noktasina_Don();
20.                       break;
21.                       #######
22.             }
23.             else
24.             {
25.                       ilac_puskurtme_kapali();
26.             }
27.   }
28.   else
29.   {
30.             ilaç_puskurtme_aktif();
31.   }
32.   #######
33. }
34.
```

3.3.4 Decision Mechanism in Case of Link Loss

There's a continuous "HEARTBEAT" signal exchange between the device and the ground control station. This signal is sent repeatedly at regular intervals. If the signal takes too long to arrive or doesn't arrive at all, it signifies a **link loss**. Therefore, the final conditioning must account for the "HEARTBEAT" message.

```
1.  void Navigator::tarim_modul_main(bool &have_geofence_position_data, double lat, double
    lon, double alt)
2.  {
3.    #######
4.    struct tarim_modulu_nav_s tarimNav;
5.    uORB::Subscription _tarim_modulu_sub
6.              {
7.              ORB_ID(tarim_modulu), 0
8.              };
9.    #######
10.   if(!(PoligonunİçindeO))
11.   {
12.             if(acil_durum())
13.             {
14.                       ilac_puskurtme_kapali();
15.                       switch acil_durum
16.                       #######
17.                       case acil_durum == HEARTBEAT_YOK
18.                                 if(gps_fail)
19.                                 {
20.                                           LAND_MODE();
21.                                 }
22.                                 else
23.                                 {
24.                                           Yer_Kontrol_Noktasina_Don();
25.                                 }
26.                       break;
27.                       default;
28.             }
29.             else
30.             {
31.                       ilac_puskurtme_kapali();
32.             }
33.   }
34.   else
35.   {
36.             ilaç_puskurtme_aktif();
37.   }
38.   #######
39. }
40.
```

In case of link loss emergency, if there is GPS loss, it lands at the same location. If there is no GPS loss, it returns to the ground control point.