Abstract—The present dissertation reports the development of an application to control an unmanned ground surveillance robot, the RovimViewer. We have undertaken a study concerning the needs of information, by the ground forces, as well as on the physical and security characteristics that must be presented in such systems. There is also the concern to develop a low-cost system, using for this purpose open-source software. Python was used as a language and graphical development environment, QTDesigner, all developed in Debian - Linux environment. The application is composed of five modules: observation and monitoring, configuration and control; Geographic Information System; Diary and display of asynchronous messages; Server Protocol and calculation; Early prediction model. The intention to develop a server capable of protocol translation commands is included in the application, allowing the control of new robot in the network. It requires an abstraction of the application, making it independent from the robotic device. It is included in the system a model for predicting the route through linear coding LPC. In the final phase of the project it will be shown the limitations of the application, as well as an evaluation and testing about the initial purposes. It was then raised the question of image quality versus transfer rate. To conclude, it is possible to confirm the capacity to develop a low cost, sufficiently robust and capable to be used application.

Index Terms—Unmanned vehicles, Unmanned vehicles application control, Open Source, LPC Coding

I. INTRODUCTION

Information is seen as crucial to the development of decision-making. Considering the costs involved due to the entry of forces into hostile terrain. It is requested the development of some equipment that make data acquisition without the involvement of forces in a conflict. Thus, there is the intention to develop this equipment, which is realized in the development of the project ROVIM. The ROVIM project currently has three development issues. The theme for the construction of the steering and braking traction, the theme of autonomous navigation, and the issue addressed in this dissertation which was the development of a monitoring and control application to ROVIM, the RovimViewer. Then comes the proposal of the global system that together allow the acquisition of data and information. Getting this data should be in Real-Time in order to allow a timely decision-making. The system developed under the implementation of a military ground surveillance robot (ROVIM) has as main objective that is the development of a military tool of data acquisition for various ranks of the military hierarchy. This parallel distribution is intended to avoid latency in data distribution. Figure 1 is represented the military hierarchy that needs to be planned (Figure 2) to avoid these latencies. For this purpose several requirements must be taken during the development step.

II. PROJECT DEVELOPMENT

A. Real-Time Systems

In all the autonomous systems there must be a Real-Time. The Real-Time term refers that a task have time restriction to be concluded. This time restriction may have different levels of flexibility. The levels depends on the system that are implemented and they are[2]:

- Soft Real-Time – Time restriction where the system response’s outcome has some utility associated to the application, even after the predetermined limit. There is, after this limit, a degradation of service;
- Firm Real-Time - time restriction where the result of the associated system response loose any useful for the application, after the predetermined limit;
B. Search Requirements

When a project is in development, there is several requirements that must be identified. This requirements could create lines guidance for implementing the project. When dealing with an application to control a military ground surveillance robotic system, there are security and robustness requirements that are strictly necessary. After a brief survey it was found a list of primary goals.

- *RovimViewer* must have a console interface that allows a use by the command and intelligence services of staff, such as, for elements of small units of platoon level;
- Ability to provide *In-Time* data to the command of forces in theater;
- Ability to control equipment, *ROVIMs*, moving through any way, like air or water (including sub-aquatic), transparently to the application and the operator;
- Monitoring the dynamic state of the *ROVIM*, which includes speed, direction, battery status in order to pass security and control feeling to the operator;
- Sending periodic signals to *ROVIM* application that always allows a smooth, linear and fluid control;
- The application must have an easy and intuitive control of multiple *ROVIMs*;
- The control interface should be intuitive, easy to use and have capability to update;
- Specification of the route and / or destination to follow autonomously by the *ROVIM*;
- Route prediction algorithms;
- The whole system is built to provide easy maintenance and reduced cost;
- Low cost production enabling a better cost/benefit relation.

C. Development of Low-level application

After the development of the requirements is necessary to establish priorities for *RovimViewer* development. Considering the following core functionality.

- Get and display the video image;
- Get GPS signal and display position on a *raster*\(^1\) or *vectorial*\(^2\) map previously acquired\(^3\);
- Control and monitor the *ROVIM*;
- Prediction algorithms.

There are threads to deal with each one of the cores. To make a good manage it is necessary implement mechanisms to achieve coherence and stability. To run any multi-task program always exist the problem of resource sharing. Due to this problem it is easy to understand that sometimes the task can not execute. A set of states are implicit in the management of tasks. Figure 3 shows that states and the transition motivation. When a task try to access a busy resource there is the need to wait for resource release signal. If the task that locks the resource is blocked by himself (example: blocked socket) the program goes into a dead-lock mode. To avoid this problem must exist a task management to deal with that issue. However, in RT - Real-Time systems can not exist this kind of blocked implementation because exist time-restrictions to deliver results.

---

\(^1\) Aerial Geo-referenced image

\(^2\) Geo-referenced image consisting of points which constitute lines or polygons that represent an area of land

\(^3\) Provided by the Army Geographic Institute
be carefully chosen to provide a Real-Time System [CAN Network for example]. In short, must be a carefully hardware and software chosen.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Network</th>
<th>Processor</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video Image</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Control ROVIM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Prediction Algorithm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE I**
**RESOURCES USED BY RovimViewer MODULES**

### III. GETTING SYSTEM REQUIREMENTS

From the application (RovimViewer) and robotic device (ROVIM) requirements viewpoint’s, due to the military purpose of them, there are a few items to be considered:

- Have a allowed interface console to be used by the command and intelligence services, as well as by elements of small units of platoon level;
- The control interface should be intuitive, easy to use and capable to be updated;
- The application must have an easy and intuitive control of multiple ROVIMs;
- Should be available in the application a local for writing reports that allow its diffusion through the network;
- Smoothly and smooth command and control ROVIM;
- Ability to provide In-Time data to the command of forces in theater;
- Ability to perform some autonomous tasks as the displacement of a point A to point B, including overcoming obstacles. In case of failure to transpose, should be transmitted a warning message to the operator to take ROVIM control;
- Provide a periodical report sensor read in order to understand clearly the state of ROVIM;
- Provide levels of security and immobilization depending on the mission / task that is played by ROVIM, preventing damage and added value loss such as for the ROVIM, as for the environment;
- Ability to follow a ground force, facing various obstacles on the ground, which can either be natural or artificial;
- Resistance of equipment to adverse weather conditions;
- An autonomy that allows missions / tasks safely and successfully;
- Due to intelligence opponent in the theater, there is the necessity to maintain a short effective range of communications because the electromagnetic spectrum created by wireless links;
- Provide a weapon that allows recognition in force, or even an extra defense in case of an offensive by the opposing force;
- Capacity of interaction between different ROVIMs;
- Have different types of sensors (thermal, tilt, compass, GPS, tension);
- Prediction algorithms route;
- Sending asynchronous messages immediately after the sense that perhaps there will be a failure of the system composed by: location information, time, battery status, speed and direction;
- The whole system baseline is built to provide easy maintenance and reduced cost;
- Low cost production enabling a cost / benefit.

### A. Development and explanation of the RovimViewer modules

The developed project has six modules that are integrated and connected to build the RovimViewer application: Are the application modules:

- Observation and monitoring the environment around the ROVIM;
- Configuration and control commands;
- GIS - Geographic Information System to simulate an aerial ROVIM observation;
- Logbook and presentation of asynchronous messages;
- Server assignment protocol and calculation of additional tasks;
- Anticipation data and information system related to positioning.

It was used a mutex to ensure mutual exclusion in the access to critical areas as well as a non-bloked socket to disable some dead-locks situations associated to a crash of the thread which are locking the resource. In that case the unlock resource is never achieved and the second thread will be waiting indefinitely causing a dead-lock issue. The high level of application was developed using the development software QTDesigner. In this software was built a “draft” of the application RovimViewer. After the graphic development is necessary convert the graphics to a written language. This step is possible using the command pyuic4 -x rovim.ui -o rovim_add.py. After that is achieved a rudimentary application with no functionality. It needs to be completed through insertion of code. One function complete and moves to another one. The graphic aspect of RovimViewer application was developed as needed to address the issues presented and the desire to see implemented certain functionality. The diagram of iteratively implementation steps between Gui - developing and text coding is represented in Figure 4

Offset differences moves some attention to this issue. When moving the ROVIM to go front, it usually drift. To solve the issue was implemented a personalized direction adjustment. This adjustment can be seen in Figure 5. If the ROVIM insists drift left the adjustment is made to the right side. Then, the ROVIM when goes straight it does not drift anymore. Till now the problem was solved. However it still keeps with an associated problem. This time when trying to describe curves. Due to the offset, the response to drift right is more sensitive than the response to drift left. At the software level it is difficult to implement this solution. It should be deactivated the offset when the ROVIM is describing curves. There is also the objective to have an alternative control to joystick. It can be seen in Figure 6.
About thinking of observation the environment around the ROVIM (Figure 7), there is implicit the need about configuration the quality of image. It can be seen in Figure 8. This settings are going to be explored due to the limitations about image transfer rating versus quality and size image.

As in the begin was told, about the application requirements definition, the RovimViewer should control multiple ROVIMs, in Figure 9 is shown a selection table of IP/Port as well as the SIG selection Layers. The selection table is limited to twenty ROVIMs. The roof is only depending of the connection. This twenty entries are only a reasonable number. However, the developed application does not control simultaneous ROVIMs. One test that can be made to verify if the connection support

two connected links is run to different consoles in the same machine, connect them to two different ROVIMs and see if the network supports the data transfer rate. In this experience, the joystick is going to send equal commands to both ROVIMs. Than it is possible to verify the mechanic differences between them.

For the same reason presented previously it is showed a Logging Book in Figure 10. There is represented a track made by a ROVIM. The time of the mission is also memorized.

At the end point of the actual developed RovimViewer, it was initiated the implementation of a server which translate commands from the application to a newer “ROVIM”. The interest of this server is achieve an abstraction that improve the RovimViewer capability. This server expand significantly the application value. After the development it could control.
any type of robots since the command protocols are known and the robot have a connection that could be recognized by RovimViewer. Unfortunately, this server envisioned to be extremely complex and not so simple has was thought in the beginning. It requires a series of changes to the commands protocol already present in RovimViewer due to the diversity of information to be treated. However is described an approach to this adaptation in the full dissertation as well as limitations of the RovimViewer.

\[ \min E \{ e(n)^2 \} \] (2)

After Computing this Equations we find normal equations in the following:
\[ \begin{bmatrix} r(0) & r(1) & \cdots & r(P-1) \\ r(1) & r(2) & \cdots & r(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(P-1) & r(P-2) & \cdots & r(0) \end{bmatrix} \cdot \begin{bmatrix} a(1) \\ a(2) \\ \vdots \\ a(P) \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(P) \end{bmatrix} \] (3)

One way to find the predicted point is compute the following algorithm:

R[i] - Auto-correlation - Equation 4
A[i] - Filter Coefficients
An[i] - New Coefficients
K - Reflection Coefficients
Alpha - Prevision Gain

Compute the Algorithm:

\[ A[0] = 1 \]
\[ K = -R[1]/R[0] \]
\[ A[1] = K \]
\[ Alpha = R[0] \cdot (1-K^2) \] (1)

For \( i = 2 \) To \( M \)

\[ S = \sum_{j=1}^{i-1} (R[j] \cdot A[i-j]) + R[i] \] (2)
\[ K = -S/Alpha \]
\[ An[j] = \sum_{j=1}^{i-1} A[j] + K \cdot A[i-j] \] (3)
\[ An[i] = K \]
\[ Alpha = Alpha \cdot (1-K^2) \]

End

\[ R_{xx}(k) = \sum_{n=n_{0}+N+k}^{n_{0}+N} x_n x_{n-k} \] (4)

V. EXPERIENCE RESULTS

After RovimViewer implementation exist the requirement to test it in order to know how the development goes. This tests should be executed during the development at a predetermined date. With the purpose of testing, the RovimViewer was submitted at two test experiments. The first one consist in estimate the time that was needed to achieve adaptation to RovimViewer. There was the exposition the application to a new unknown pilot. The pilot execute the same track in three attempts and was counted the time he spent. It is expected that time of attempts gradually decreases From the first attempt to the last one. This test check the difficult to adapt a new pilot.

The table presents, in the full dissertation, shows that the time decreases on average to half from the first attempt to the last one. However, a trained pilot can do better times than three attempts pilot.

Some pilots gave their feedback about the presence of delays while controlling the ROVIM. This lead to the construction of
a image quality *versus* frames per second table, present in the full dissertation.

Another test that was made was write the military report: TUTELA; for the acquisition data to make-decision process trough *RovimViewer* application. It was verified that is possible build thouse reports to deliver to the superiors.

**VI. CONCLUSION**

After the tests we realize that is possible implement the application with open-source development software. However should take place a rigorous design software building. There were found some problems in the application due to the use of none ideal communication protocols: TCP/IP. There is not exists security in the communications. The first goal was the application development that achieve the predetermined results: Fluid control the *ROVIN*; Display position in Map; Start a new connection with other *ROVIN*.

**REFERENCES**


