Fraunhofer FKIE

FRAUNHOFER INSTITUTE FOR COMMUNICATION, INFORMATION PROCESSING AND ERGONOMICS, FKIE

FIELD TEST TOOL OPERATION MANUAL

Prepared by: Carlos Tampier Cotoras

Table of contents

1 Introduction

The Field Test Tool (FTT) is composed of four software modules: a database, a ROS interface, a user interface and an automatic report generator. A diagram of the system's architecture is shown in Fig. 1.
This document focuses on explaining the usage of the interface components. That is, the user interface and the ROS interface. For an overview of the complete system, installation and execution instructions, please read the README.md file in the project repository.

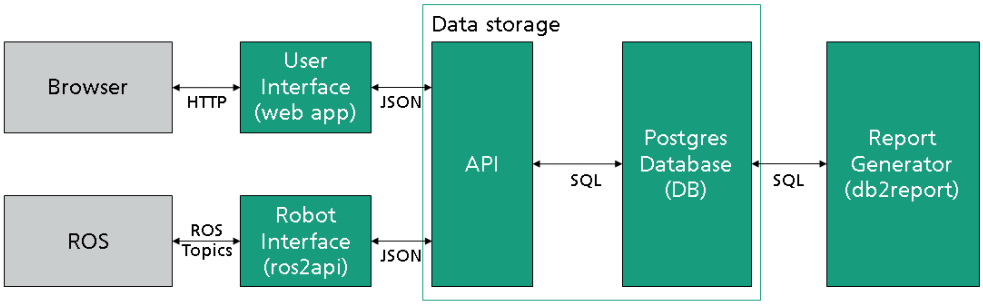

Fig. 1 System's architecture.

This system is designed to store the robot operation mode and its position, along with other context data that provides insight of the working conditions. It should be noted, that the operational data is stored in the database under a tree-layer logging structure, namely:

- Level 1: Test Event
- Level 2: Shift
- Level 3: Leg

The role of each one will become clearer with the explanation of the user interface further down in this document. Furthermore, each time the autonomous system's operation mode changes, a new entry, called "Segment", is created in the database. Segments define the portions of the robot path. Therefore, the complete robot trajectory of a system's test run will be divided into these segments. One for every mode change.

2
ROS Interface

The ROS interface is a component that needs to be set up and integrated with the autonomous system to be tested. To properly carry out this integration, the following should be noted.
The "ros2api.py" python ROS node subscribes to topics and listens to the robot TF in order to send HTTP POST requests to the database API. The gathered data includes robot's operation mode (manual/autonomous), its position, and optionally, images from an onboard camera.
The FTT can work with either GNSS data or locally referenced position data. If working with the latter, a map of the environment is also required.
Tab. 1 summarizes the data acquired from the robot environment, along with some relevant comments. The parameters of the launcher file of this node (remap and parameter arguments) are described in the README.md file.

Resource	Default source	POST request	Notes
Subscriber: industrial_msgs/ RobotMode	Topic: /robot_mode	segment	Mandatory. A new segment creation request is issued every time the message's "val" variable switches between 1 (manual) and 2 (auto).
Subscriber: sensor_msgs/ NavSatFix	Topic: /robot_position	pose	Mandatory if working with global positions. POST requests sent on a timed-based schedule.
Subscriber: sensor_msgs/ Image	Topic: /image_raw	image	Optional. The last received image is sent after the creation of a new segment.
Subscriber: sensor_msgs/ CompressedImage	Topic: /image_compressed	image	Optional. The last received compressed image is sent after the creation of a new segment. If both the /image_raw and /image_compressed topics are remapped, both will post their image with the new segment. To only send one image, do not remap both image subscribers.
Subscriber: nav_msgs/ OccupancyGrid	Topic: /map	map_image	Mandatory if working with local positions. POST requests sent on a timed-based schedule (only if a new message arrives to the subscriber).
TF listener: "map frame" → "robot frame"	Frames: "map" → "base_link"	local_pose	Mandatory if working with local positions. POST requests sent on a timed-based schedule.

Tab. 1 Data acquired from the ROS environment.

3
Using the web GUI

The web GUI allows the human operator to interact and manipulate the data stored in the database. This helps with the inclusion of relevant context information about the autonomous system's condition during testing. The GUI also controls the starting and finishing of new records. The initial view of the web application is shown in Fig. 2. Three log levels are used to store the robot' data: a "Test Event", a "Shift", and a "Leg".

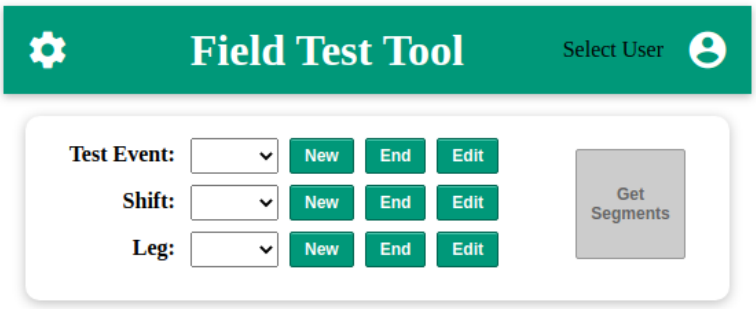

Fig. 2 Starting view of the FTT web GUI.

The first time using the tool, some additional data will need to be forwarded to the database through the web GUI. Clicking on the top left cog will open a new tab in your browser (Fig. 3), where data for the tables "performer", "personnel", "pose_source", and "vehicles" can be entered.

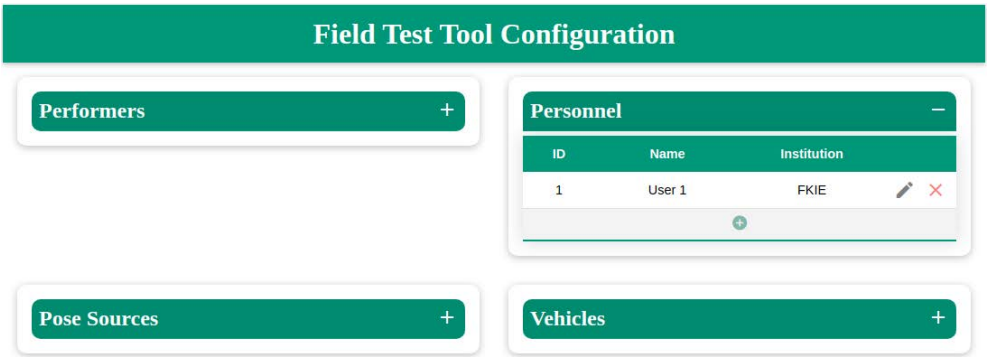

Fig. 3 Starting view of the FTT web GUI.

Once the configuration data has been saved, you can go back to the main page to use the tool. In order to use the application, a user must first be selected. Clicking on the top right user icon will deploy a selection list, as shown in Fig. 4. Then a new entry for each logging level can be created by pressing on the "New" button of each one. A dialog will open for each newly created log level, as can be seen in Fig. 5, Fig. 6 and Fig. 7. The same dialog will show when the "edit" button is pressed. Each dialog allows the user to input relevant context information about the system's test.

A “Test Event” entry has fields for the test location, version and time zone, as well as a field to input any additional comment.

A “Shift” entry allows the identification of the test performer, as well as the involved personnel. The test intent, a workspace description and the used vehicle can also be added. A field for any additional comments is also provided here.

A “Leg” entry is the last log identifier for the recording instance. The current weather, configured position data source, and additional comments are entered here.

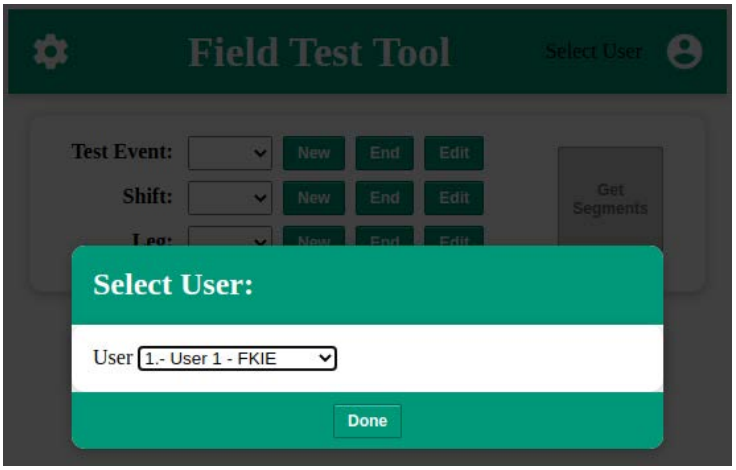

Fig. 4 User selection.

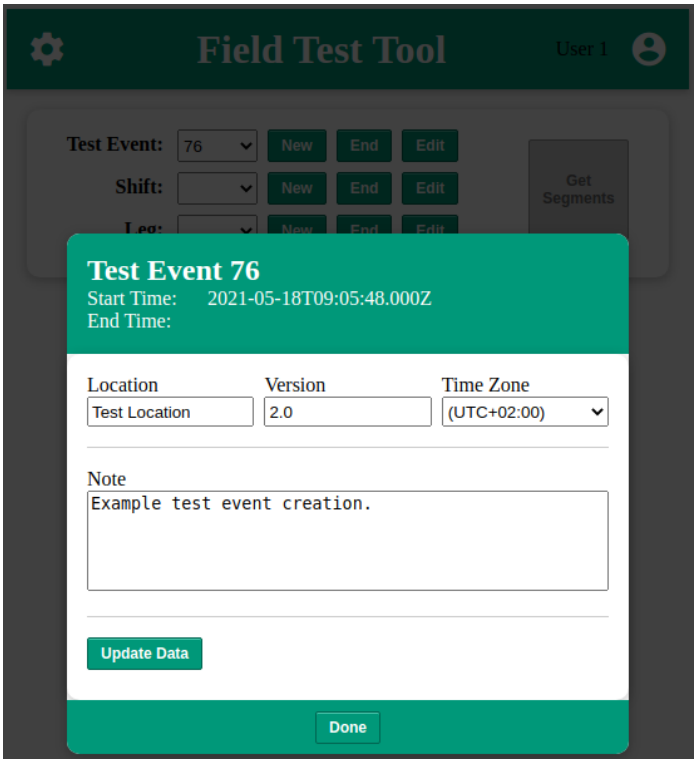

Fig. 5 Test event creation/editing dialog.

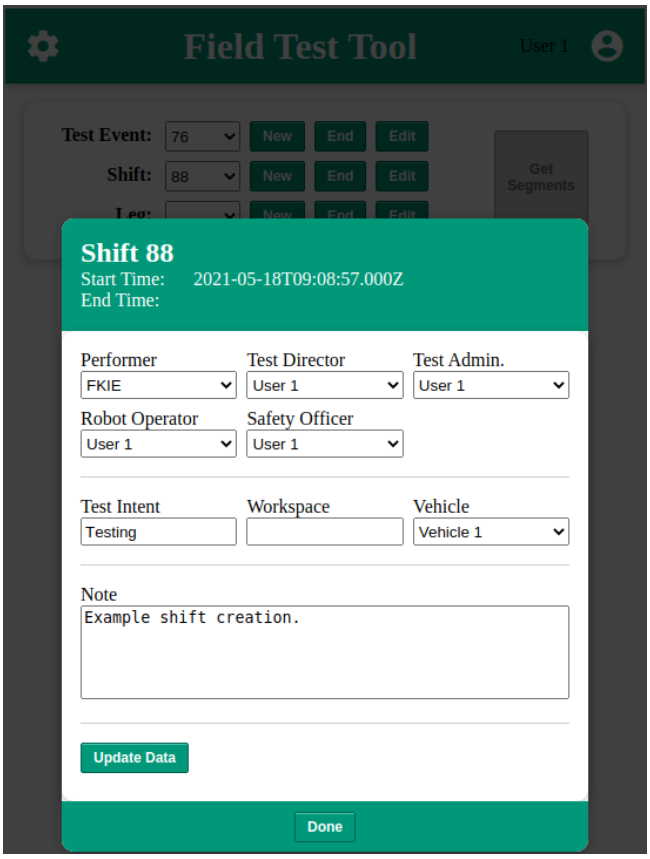

Fig. 6 Shift creation/editing dialog.

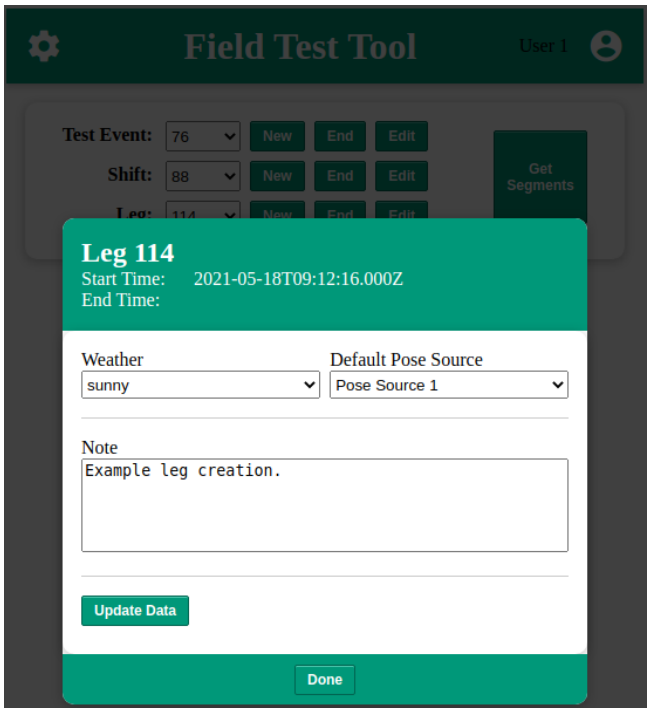

Fig. 7 Leg creation/editing dialog.

Once all log level have been created, the "Get Segments" button to the right of the log level selection can be pressed. The GUI will then appear as shown in Fig. 8.

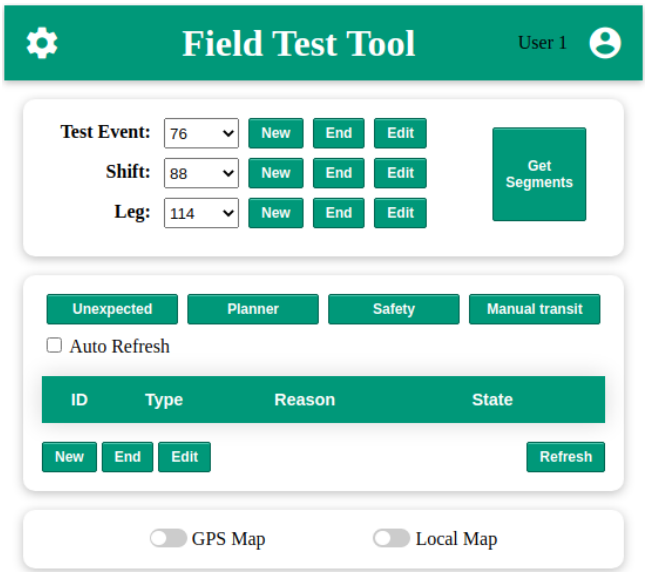

Fig. 8 View of the GUI ready to start a recording.

Before starting the autonomous system's test, it is recommended to check the "Auto Refresh" box and activate the map display (either global or local), as shown in Fig. 9.

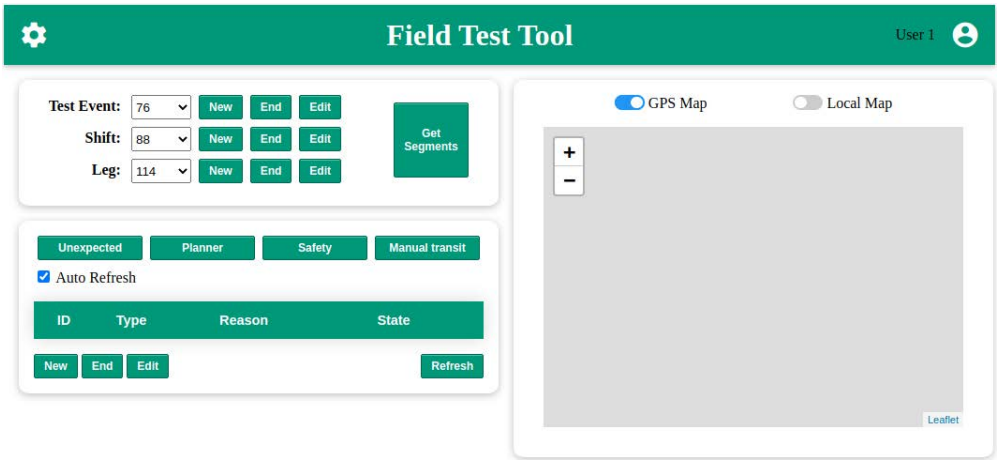

Fig. 9 View of the GUI with auto refresh and active map.

The ROS interface module should start creating data entries in the database as the robot performs the intended tests. With the "Auto Refresh" box checked, the GUI will update every 2 seconds, showing the robot position in the map, and a list of all manual (ITO) segments. An example is shown in Fig. 10 (with global data display) and Fig. 11 (with local data display). In these examples, Clearpath's Husky ROS stack was used in simulation.

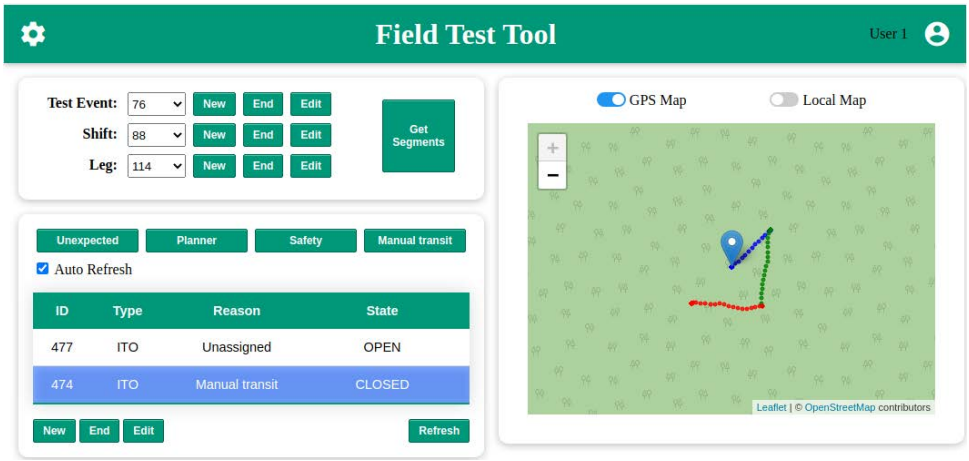

Fig. 10 View of the GUI during the autonomous system's test with global position data.

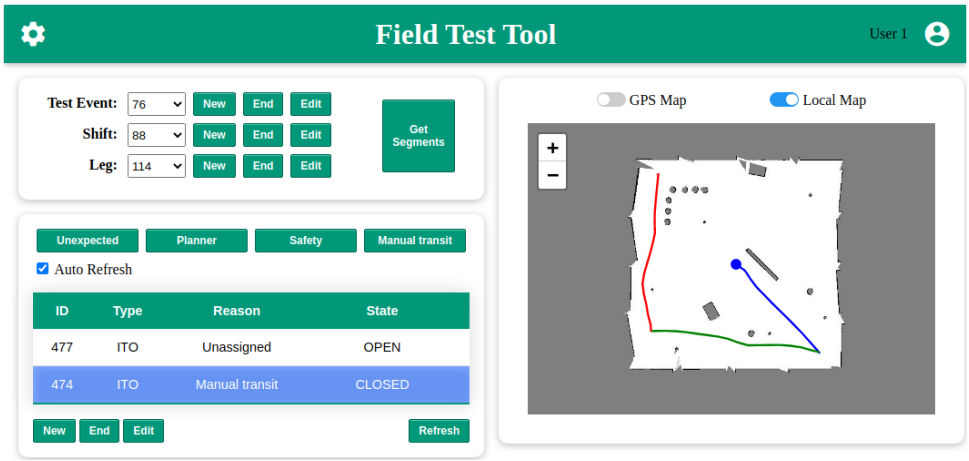

Fig. 11 View of the GUI during the autonomous system's test with local position data.

Manual segments can also be added through the GUI by the user. It is recommended, though, that segment creation is mainly done by the ROS interface node. Nevertheless, once the robot is running in manual mode, the creation of a second manual segment might be desired, in order to register a new event which forces the robot to stay in manual mode for longer than expected for the original transition cause.

Transition causes (ITO Reason) can be quickly assigned to manual segments by selecting one from the list and then clicking on one of the four buttons over it: "Unexpected", "Planner", "Safety" and "Manual Transit". An extended list of autonomous to manual transition reasons can be found when clicking on the "Edit" button at the bottom of the segment list. Doing this will display the editing dialog for the selected manual segment. Fig. 12 shows the mentioned dialog. Further information, such as annotation of obstacles, lighting conditions or terrain slope can be added here. Several independent notes can also be added to the selected segment and pictures can also be uploaded from the user's device.

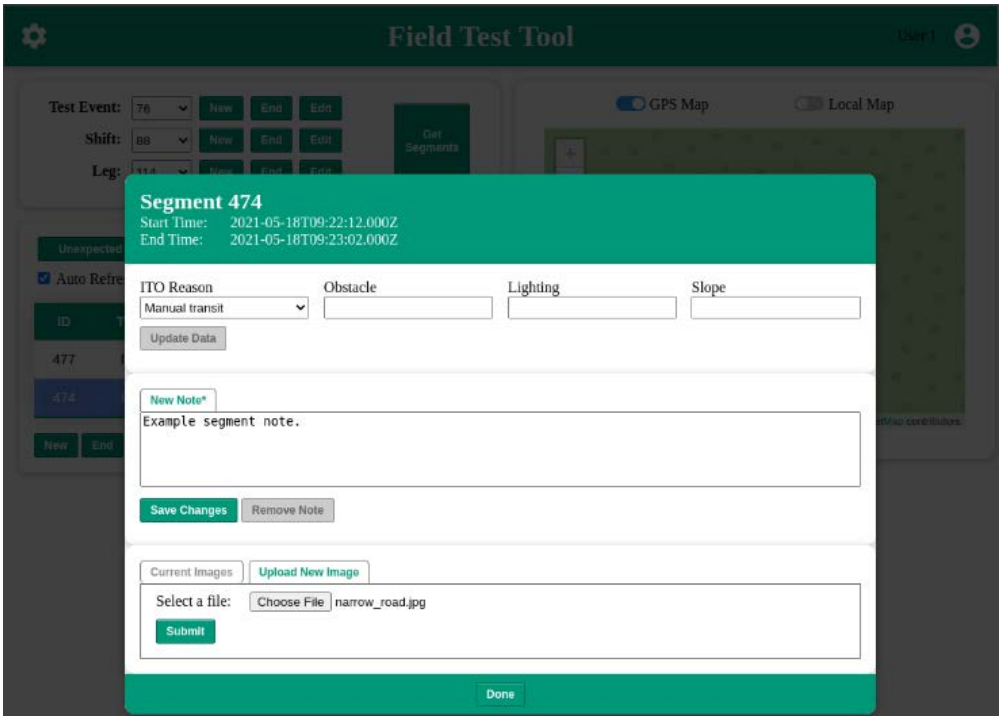

Fig. 12 Segment creation/editing dialog.

Once the test is finished, the recording instance should be closed by the user. "Closing" a log level or a segment means assigning an "end time" to the corresponding entry in the database. This information is then used in the report generator. Closing any log level will automatically close any entries below it (e.g. closing the "Test Event" will close any open shift, leg and segments). Once closed, no new data coming from the robot will be added. Nonetheless, the user can still modify the additional context information.

Note that segments can also be closed manually from the GUI. This will only close the portion of the segment storing the "ITO Reason", the database will still keep the "main" manual segment open until a new autonomous segment is created or the leg is closed. Internally, this is implemented with manual segments having "child" manual segments. The parent segment stores the position data and images, while the children store the transition reason.

In order for the automatic report generator to work properly, the "Test Event" must be closed before invoking the report generator's script.