European Commission

JRC TECHNICAL REPORTS

RIO
Remote InterOperability platform

An extensible multipurpose platform to build a network of interoperable devices

2020
GALLIANO, D A
MANCON, L
ANNUNZIATO, A

This publication is a Technical report by the Joint Research Centre (JRC), the European Commission's science and knowledge service. It aims to provide evidence-based scientific support to the European policymaking process. The scientific output expressed does not imply a policy position of the European Commission. Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use that might be made of this publication.

Contact information
Name: Daniele A. Galliano
Address: TP 680, JRC Ispra I-20127
Email: daniele.galliano@ec.europa.eu
Tel.: +39 0332 78 3525

EU Science Hub
https://ec.europa.eu/jrc

JRC115046

PDF ISBN 978-92-79-98770-0 doi:10.2760/0188

Luxembourg: Publications Office of the European Union, 2020

Contents

Foreword

The European Crisis Management Laboratory develops new technologies to improve the Civil Protection mechanism, including sensors and alerting devices. The experience of several years and the technical innovations led to develop an Operating System (OS) able to manage all the functionality requested from remotely deployed devices, that must be reliable and resilient. This document describes both the software developed for the devices and the infrastructure required to operate them.

Abstract

The Remote InterOperability platform requires a software to manage the remotely deployed devices and an infrastructure to support it.

The RIO software on the devices acts as a guest Operating System hosted by a native one to manage the features installed in the device. It is therefore able to implement remote sensors as well as alerting devices.

In order to develop such a flexible software, .Net Core has been adopted, since this programming environment is available on many platforms including 32 and 64 bit Windows machines, Linux machines including small microcomputers and macOS machines.

RIO needs an infrastructure to communicate with the devices; its description includes the network servers used to implement it, their configuration, the protocols and the formats adopted.

1 Introduction

In 2009 the JRC patented the Tsunami Alerting Device, a device capable to warn the population about an impending event like a tsunami or a cyclone. This device was extensively tested in Setubal, Portugal, until 2017, when a final setup was designed. The new device was installed and successfully tested in February 2018.

In 2014 a new development produced the Inexpensive Device for Sea Level measurement. Instead of custom components, the initial design of the IDSL was based only on retail components, in order to reduce the cost and to speed up the prototyping. Its final configuration requires only a custom integration board to have a clean set of connections, but the rest is still based on components easily available on the market[1].

With the IDSL the Raspberry Pi (RPi) was introduced to control the main features of the device. It allowed a more rapid development of the software, that also maintained a great flexibility. The benefits from the adoption of the Raspberry Pi and its Linux-like OS include also the remote management thanks to VPN software like Hamachi-LogMeIn.

In 2018 the Last Mile project was launched to prove the feasibility of a network of devices able to identify a threat for the population and provide it with the means to face it. In order to create such a family of devices and organize their interactions, RIO platform was chosen.

After the successful experimentation in 2016, a new GPS based system was developed to measure the sea level in open sea. This system will also be implemented using RIO devices.

[1] http://publications.jrc.ec.europa.eu/repository/bitstream/JRC102851/lb0116832enn.pdf

2 Reference technology

The RIO platform is based on RIO devices and on a set of network services composing the RIO infrastructure.

2.1 RIO Device architecture

So far, RIO devices are based on a central processing unit provided by a Raspberry Pi and a variety of additional devices controlled by it.

The Operating System chosen to run the Raspberry Pi is Raspbian, an evolution of the Debian distribution of Linux developed for this family of devices. This choice is not related to the programming platform, that is available also for other Operating Systems running on these devices (e.g. Windows IoT), but on the existing software and tested configuration developed for the IDSL project.

The programming environment is based on .Net Core, a flavour of .Net addressing a huge variety of devices and Operating Systems. The version used until now is 2.1, but a passage to 2.2 will be evaluated. .Net Core requires anyway that the ARM CPU of the Raspberry Pi is provided with a set of instructions available only from ARM 7 on: this requires to use Raspberry Pi 3 or 2+ V1.1.

Apart from considerations related to the specific platform, the RIO software can run on different devices and remains open to others, which will be available in the future.

The software is developed in C#.

2.2 RIO infrastructure

In order to provide the RIO devices with the means to exchange data including alerts and to be remotely managed, these services are used:

- The WebCritech website will host the information related to the devices and shows the data received from them. For alerting devices, it will offer a management interface to set the alert messages and signals. The website is implemented as an ASP.Net application running on IIS. The website, being stateless, can be served by several servers, in order to balance the load while offering redundancy.
- The SensorGrid service will provide the means to communicate information to/from the devices and between them. The services offered are two:
    - Message Queue: messages can be posted in order to be dispatched to all subscribing clients. There are several channels offering these services:
        - Alert: it allows posting the alerts as computed by the devices depending from their internal logic
        - Telemetry: carries all data sent by the devices, in order to be stored in the databases
        - Heartbeat: all the device confirms the RIO OS is alive by sending a message every minute
        - Management: each device is provided with its private management channel
    - CAP dispatcher: the server accepts and verify CAP messages from clients. If valid, the message is acknowledged and dispatched to all subscribers; otherwise, a diagnosis of the format error is sent back. When subscribing, a client may request to be sent a set of previously dispatched messages.
- A relational database is used for all the information related to the device management and configuration. Part of the data received are also stored there. A SqlServer instance contains a set of tables where to store the configuration and other ancillary information about the RIO devices. A particular attention deserves

the rulesets: these sets carries for each alerting RIO device a set of couples made by a condition and some actions. When the device evaluates the condition and it is verified, the related actions are performed. This mechanism allows the devices to activate their alerting features based on the alert conditions generated by other devices, like sensors.

- A fast time series database stores all the measurements data received from the devices and the alert messages. In order to reduce the response time, a time series database is used to store all information sent by the devices in terms of sensor readings. The CAP dispatcher also relies upon this database.

3 RIO device

A RIO device can be any device able to run the RIO OS and connected to the Internet, including PCs. The first set of devices was implemented by managing with RIO slight modifications of the IDSL. Best part of the setup was kept unchanged, but the CPU was changed to use only the compatible versions of the RPi.

3.1 Reference implementation

A RIO device requires a CPU with its mass storage, a reliable power source and the Internet connection.

The IDSL was developed using RPi as central unit, a rechargeable battery, and a 3G modem-router.

3.1.1 Storage

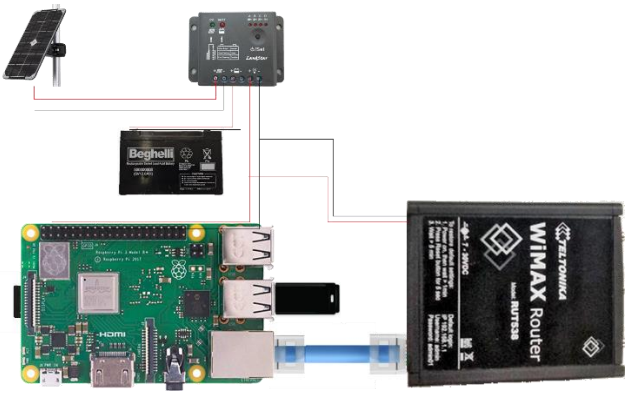
The OS of the RPi and the RIO software are both stored in a microSD card: the RPi uses it as the boot device. In order to reduce its aging, the best part of the file system is used read-only. The best part of dynamic information is stored in memory, since they are volatile and cause no harm if lost due to a power down. An additional mass storage is provided by an USB stick connected to the RPi. In case of aging, it can be easily replaced even by untrained people.

It is suggested to adopt in the future the RPi module, a more compact device relying on a custom board, that has better performance also thanks to its mass storage.

3.1.2 Connections

A 3G modem router implements a network local to the device. The model used so far is the Teltonika RUT-500, that allows up to four Ethernet connections and a WiFi. The RPi is connected to it together with optional devices like Ethernet-controlled relays or alerting panels.

It has the common feature of home devices, including firewalling, port forwarding and the like. The best part of them are of limited use, due to the limitation of the 3G services. It requires a SIM card and usual devices consume around 2,5 Gbyte traffic per month.



1 RIO device schema

3.2 RIO Operating System

The RIO OS revolves around a main component, the Manager, which orchestrates the activities of all other components. The OS requires an implementation, that is a main program, which is launched by the native OS. This program will instantiate the Manager providing it with some basic information, like the Settings. The implementation will then leave the control to the Manager, which will stop only when a shutdown is requested. At that point, the implementation should stop also, thus releasing all the resources to the native OS.

3.2.1 Main components

The Settings loaded by the implementation and provided to the Manager contain the information required to configure a set of features, where the features are specific software components aimed at providing different services, usually the way to interact with external devices. Once configured, a feature instantiates one or more tasks to perform specific actions. All tasks are required to publish a Status and a set of Metrics for remote monitoring.

3.2.1.1 Manager

The Manager is a singleton class, therefore only one Manager can be present, which offer static methods, the access to the single instance and one static event, Notify. The implementation should subscribe to the event to receive all the communications coming from the RIO OS, being them tagged with their specific nature of errors, debug or telemetry.

The Manager offers also methods to submit to it management requests or alert messages: it will answer with the message to be sent as a response, if any.

3.2.1.2 Feature

A feature defines something the RIO is able to do for the user. It is usually a driver for an external device, but can be any type of software, including a signal generator, a message dispatcher and the like.

The feature is loaded dynamically from the libraries deployed together with the RIO software: any dll file whose name begins with RIO, JRC or TAD is searched for the presence of a feature. The features provide the list of parameters they need to be configured, including the type of the parameter and a default value.

The Settings may contain more than one reference to the same feature, since it is possible a feature instantiates more than one task of the same type with different configurations.

3.2.1.3 Task

In case a feature configuration is enabled, it will instantiate a Task. The Task is the single activity performed by the RIO. It can be event-driven and activate an external device when requested (e.g. a siren), it can continuously monitor a connection with a sensor and post the acquired data, and it can poll information from another system.

The task can be started, stopped, enabled and disabled. It must publish its status, an object providing meaningful information about it, its configuration and its running state. A task is also requested to publish its Metrics, an object providing a quantitative description of its running status like the quality of the last hour of sensor readings.

3.2.2 RIO implementation

The implementation is a program of the native OS, that is launched and creates the environment for the RIO OS, linking this guest OS with the native OS of the device.

The implementation has to interact with a repository, usually the file system, to load and store the Settings. Since the Settings notify every change, the implementation must react to this event and save the new configuration.

The Manager provides unsolicited information through the Notify event: the implementation has to communicate with the RIO infrastructure and make this information available to all. The Manager can also be requested by the implementation to perform management activities or to evaluate an Alert message. For these reasons, the implementation sets up the communication channels with the message queue and the other services of the RIO infrastructure. The implementation must also publish every minute a Heartbeat message to make the infrastructure aware that it is working, even if no other

information is published. This requirement will be less demanding once the RIO project will complete: now it is necessary to assess the stability and the reliability of the software.

A reference implementation of such an important component is provided, in order to provide a starting point for other projects.

3.2.3 RIO modules

To add a new feature to the RIO OS, it is sufficient creating a new library based on the RIO base library and containing the software required to implement the new features.

Based on the Microsoft Extensibility Framework (MEF), the plugin system of the RIO requires the Feature classes to implement the `IFeature` interface and to be decorated as exporting it, like in the example below.

```
[Export(typeof(IFeature))]
public class Simulation : IFeature
{
    public string Name => typeof(Simulation).Name;
    public IEnumerable Configuration …
    public IEnumerable Commands …
    public IEnumerable<ITask> Setup(Settings configuration, Feature settings) …
}
```

The decoration is used when loading the library to find all classes to be used.

The Configuration is a list of configuration parameters the feature needs to configure its tasks, like the name of a serial port or its speed, a port number or a complete URL.

The Commands similarly provides a list of commands the task will execute. The commands' implementations require the list of parameters they need to perform the requested action.

In order to send special data types used by the feature, it must supply their converters to serialize and deserialize their contents. For simple and standard data types it is not necessary.

The default implementation of the Name property is the best solution.

The Setup method should always return an `ITask`, even if it was not possible to configure it: it is the only way for the `ITask` to report the error.

Suppose now a Feature describe a humidity sensor and the RIO device is equipped with three of them positioned in different points of a greenhouse. The Settings will contain three different description of the same feature with different parameters. In order to tell apart which sensor the data are coming from, a different name is supplied to each configuration. There cannot be then two different configurations with the same name, even if related to different features. In case Settings loaded by the implementation contains such a duplicate, only the first will be kept and the others deleted.

All messages shared with the RIO infrastructure will then specify where the information comes from. If measurements are sent to the infrastructure, all measures from the same sensor will be grouped together.

4 Existing implementations

The RIO OS was developed with a few use cases in mind, in order to have real scenarios to test the OS since the early stages of the development.

Remote sensing requires a stable software able to run for months without any intervention, while providing a constant level of service.

Early warning requires a system always reachable and swift reacting.

These use cases allowed also to test the software in real-like conditions, proving its reliability.

4.1 Differential GNSS sea level measurement

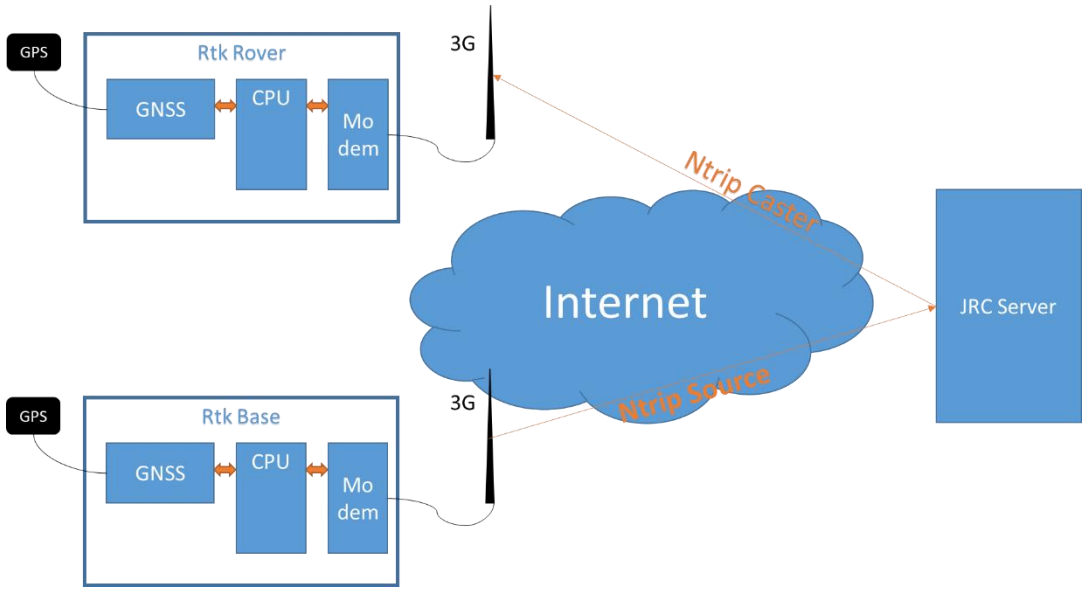
As described in "Design, integration and performance assessment of a GNSS buoy for precise real-time sea-height retrieval", [Annunziato, Galliano, Susi, Basso and Fortuny], a floating device for sea level positioning is developed at the Joint Research Centre. It aims at acquiring off-shore data, not disturbed by the harbour effects, and providing earlier alerting.

The software specifically measures the relative height of one RIO device with respect to another. A device is installed in the buoy and acts as an RTK rover, while the other device is its reference, is installed inland and acts as the RTK base.

The RTK base shares its GNSS observations with the RTK rover, which uses them together with its own to estimate its position relative to the base. The means to share observations with the rover are various: by radio link, by Internet, or more recently they are sent in a data stream from the satellites.

Sharing by Internet uses a protocol called ntrip and loosely based on http. In this setup, a Ntrip source provides data to an accessible Ntrip caster, that shares the information with all Ntrip clients.

An architecture based on two RIO devices and a server hosted by the JRC is shown in 2 RIO based Ntrip setup.



2 RIO based Ntrip setup

The GNSS receiver that has been used is a commercial off the shelf product from uBlox, the evaluation kit C94-M8P, that incorporates the NEO-M8P GNSS chipset, a positioning module. This evaluation kit consists of two separate boards, each integrating an RTK-enabled GPS/GLONASS chipset and a UHF radio modem: one acts as the base, while the other, the rover, is embarked on the buoy.

4.1.1 Ntrip source

The RIO setup for an Ntrip source foresees an additional device, a GNSS receiver providing RTK information to be shared with another device. In order to receive them, the receiver needs to be configured to set which communication channels using to send this information. The receiver is connected with the RIO with an USB cable that is also used to power the device. This means that a serial connection needs to be configured.

The format used to share the RTK observation is RTCM3, but also NMEA messages are requested and the position of the device determined. uBlox uses a proprietary protocol, called UBX, for the complete management of the devices. This is used, for instance, to tell the receiver to send through the USB connection all messages, RTCM3, NMEA and UBX.

Before the RTK corrections can be used by other devices, their quality must reach a certain level. The source can be configured to refer to a specific location or can be requested to determine its location during a survey, the longer the better. The survey is usually configured to last at least 15 minutes and to reach an uncertainty below three metres.

The Ntrip protocol requires the corrections to be published on a server (Ntrip caster): an address and a port number are needed to do so.

Therefore, the default configuration provided by the NtripSource feature is:

```
        "CollectorAddress": "139.191.244.85"
        "CollectorPort": "4003"
        "LogFile": "rtcm-{0:yyyy-MM-dd}.log"
        "Port": "COM1"
```

The Collector Address is the IP address (a DNS name can be used as well) of the Ntrip caster and Collector Port is the port listening for new Ntrip sources.

The Port is the name of the serial port used to communicate with the receiver.

LogFile is the template of the name of the file to be used to store all the information received by the GNSS receiver and can be empty or absent to disable the logging.

In the status report of the Ntrip source, the module will report the status of the serial connection with the receiver and the status of the network connection with the caster.

The telemetry of the source carries the longitude and the latitude of the determined position together with the status of the survey:

- No survey: the receiver is not able to determine its position, because the antenna is not receiving enough good signals
- Active survey: the receiver is reading signals and using it to determine its position with the required quality
- Valid survey: the receiver position is determined with the required quality

When the survey is completed, usually the rover is able to fix its position relative to the source.

4.1.2 GPS receiver

The GPS receiver may be used for normal GNSS devices or for differential systems based on its configuration. This allows using it simply to add GPS capabilities to a RIO device or to implement more complex scenarios like this one.

In case the base and the rover are connected by radio-link, the module is not aware of this: it will receive high quality data, which will be sent as telemetry. When it is configured to use Ntrip protocol, it establishes the connection with the Ntrip caster and feeds the GNSS device with the RTK corrections received. Independently from the source, being it a base RIO device or a sensor network, the GNSS device will use the RTK corrections to elaborate high quality data.

Part of the configuration is similar to the configuration of the Ntrip source:

```
        "Port": "COM1"
        "Speed": "57600"
        "LogFile": "gnss-{0:yyyy-MM-dd}.log"
        "Caster": "http://username:password@139.191.244.85:4004/TAD-00Y"
```

The Port is the name of the serial port used to communicate with the receiver and Speed is the Baud rate chosen.

LogFile is the template of the name of the file to be used to store all the information received by the GNSS receiver and can be empty or absent to disable the logging.

If a valid URL is present in Caster, it is used to connect and receive from the caster the RTCM corrections to be fed into the GNSS device.

When operating in differential mode, this module sends two different telemetry packets depending on the quality of the data. Until the quality does not meet the requested quality, the device posts only its position and the quality of the GNSS solution: "No solution", "Float solution" or "Fixed solution".

When a Fixed solution is achieved, the telemetry includes the relative and absolute height and the distance from the base.

4.2 Tsunami Alerting Device

This project, patented by the JRC in 2009, is born to deliver prompt warning to the population and it is aimed at achieving this goal with different technologies and means. The prototypal setup is an LED matrix panel, that usually provides useful information to the population. When its alert mode is activated, the panel displays directions about an impending emergency and make everybody aware of the alert using sirens, loudspeakers and flashing lights.

The prototype installed in Setubal passed a lengthy test and led to the definitive implementation as a RIO device.

One of the modules developed for the TAD manages the LED display, while the other interacts with various services to exchange information.

4.2.1 External display

This module is specifically developed for the display produced by Tech SRL, an Italian company leading the LED sign market. The LED matrix installed in the TAD provides a few http services to setup the messages to be displayed and to query the panel hardware.

Therefore, the module configuration requires the following parameters:

```
        "DisplayAddress": "http://192.168.1.206/editor.htm"
        "DisplayActuator": "http://192.168.1.206/extoutput.htm"
        "DisplayBinData": "http://192.168.1.206/displaydata.bin"
```

DisplayAddress is used to set the message to be displayed by the panel.

DisplayActuator is used to turn on and off the siren and the loudspeaker, starting the playback of a recorded message.

DisplayBinData allows retrieving the present configuration of the display, including the displayed message.

4.2.2 Data fetch

The NotificationPuller module queries the panel, the modem router and a web site for different purposes.

4.2.2.1 Tad polling

Every 30", the panel is queried to know its status, which includes the date and time as known by the system, its GPS location, the air temperature and pressure, the status of the siren and of the loudspeaker.

Location, temperature and pressure are then sent as Telemetry to the RIO infrastructure.

4.2.2.2 Web polling

Every two seconds, the TAD polls for a CAP message. In case of a new message, the TAD will look for a message type *setPage*, and uses its information to update the display. In order to avoid replaying a message, the TAD must acknowledge it.

The user can generate these messages programmatically or using the user interface provided by the RIO device page in the Web Access point.

Because of the large traffic and power consumption of this method, it will be replaced by a push service, based on the message queues.

4.2.2.3 SMS polling

The RIO queries the modem router of the TAD, fetches the SMS it received and uses them to set the message to be displayed and to turn on and off the siren and the loudspeaker.

4.3 IDSL

In 2014 JRC developed a new type of Sea Level Measurement, named *Inexpensive Device for Sea Level Measurements* (IDSL)[2], in order to support the efforts of the Member States to improve the monitoring network for Tsunami that was being established. Based on the experience of other similar devices and the needs required by a Tsunami analysis, the following requirements have been fixed for the mareographs:

- High quality of the data with an error of 0.5 cm maximum (sensitivity justified by the expected error in the sea level calculations)
- Short acquisition time interval, 15 s maximum (to have a well defined sea level wave description over time)
- Small transmission latency, smaller than 30 s (this is particular important for small basins with low travel time)
- Low overall cost, less than 1.5 kEuro
- Autonomy, at least 3 days without solar irradiation (the autonomy can be increased to 7 days with an over cost and weight on the battery)

The initial experimental campaign organized by JRC in collaboration with ISPRA, showed very positive outcome for the first 6 months of continuous operation. This convinced the UNESCO/IOC to accept the proposal of JRC to test 20 new devices for an extended experimental campaign of at least 1 year. In the first campaign 15 devices were installed; so a second campaign was launched to install the remaining 5 but 16 requests have been received. At the moment we have a network composed of 31 devices on sea plus 1 device on the Seveso river.

In order to replace the software developed for the IDSL with a RIO version, its functionality was ported to a module, which implements both the sensor reading and the analysis, including the alert assessment.

The configuration settings are therefore:

[2] A. Annunziato - THE INEXPENSIVE DEVICE FOR SEA LEVEL MEASUREMENTS –Tsunami Society International - ISSN 8755-6839 http://www.tsunamisociety.org/344Annunziato.pdf

```
        "Port": "COM1"
        "Speed": "9600"

        "SensorMultFac": "-1"
        "SensorAddFac": "0"
        "SonarMinLevel": "0.3"
        "SonarMaxLevel": "5"
        "SonarMaxDifference": "0.5"
        "SaveAllData": "true"

        "ShortWindow": "60"
        "LongWindow": "600"
        "Ratio": "4"
        "Threshold": "0.08"
        "Period": "5"
        "AddRMS": "0.1"
        "BackFactor": "0"
```

Port and Speed are used to configure the serial connection with the distance sensor.

Since the sensor measures the distance from the water surface, SensorMultFac and SensorAddFac are used to estimate the height of the water. SonarMinLevel and SonarMaxLevel define an interval of validity for the measures obtained from the sensor, outside which the readings are discarded. SonarMaxDifference is also used to filter invalid data, since it does not allow more than a certain quantity of variation from readings.

SaveAllData is a flag to require that all readings are stored in a local file with the date in the name.

The analysis is performed comparing the expected behaviours estimated on two moving windows averages, their length is defined by ShortWindow and LongWindow.

Period defines how many seconds of collected data are used to feed the moving windows. Ratio, Threshold and AddRMS are the parameters of the algorithm to estimate the measurements identify an anomalous behaviour.

BackFactor is not used, but kept for compatibility purposes.

5 RIO infrastructure

In order to exchange information and to allow remote management, the RIO devices are provided with a set of services combined in an infrastructure. It relies upon servers located in the JRC, but easily relocatable and where possible redundant.

5.1 Web Access point

WebCritech is a website used both to receive data and to publish them through a rich user interface or as downloadable datasets. Both elaborated and raw data are stored and available for public users. The website allows reading all data received as Telemetry by the devices, provides data about the quality of the transmission and statistics about the availability of the device in the past.

From the page dedicated to the device it is possible accessing to the RIO management interface, which allows configuring the device modules, enabling and disabling them, shutting down the RIO OS, requesting commands to be executed by the modules.

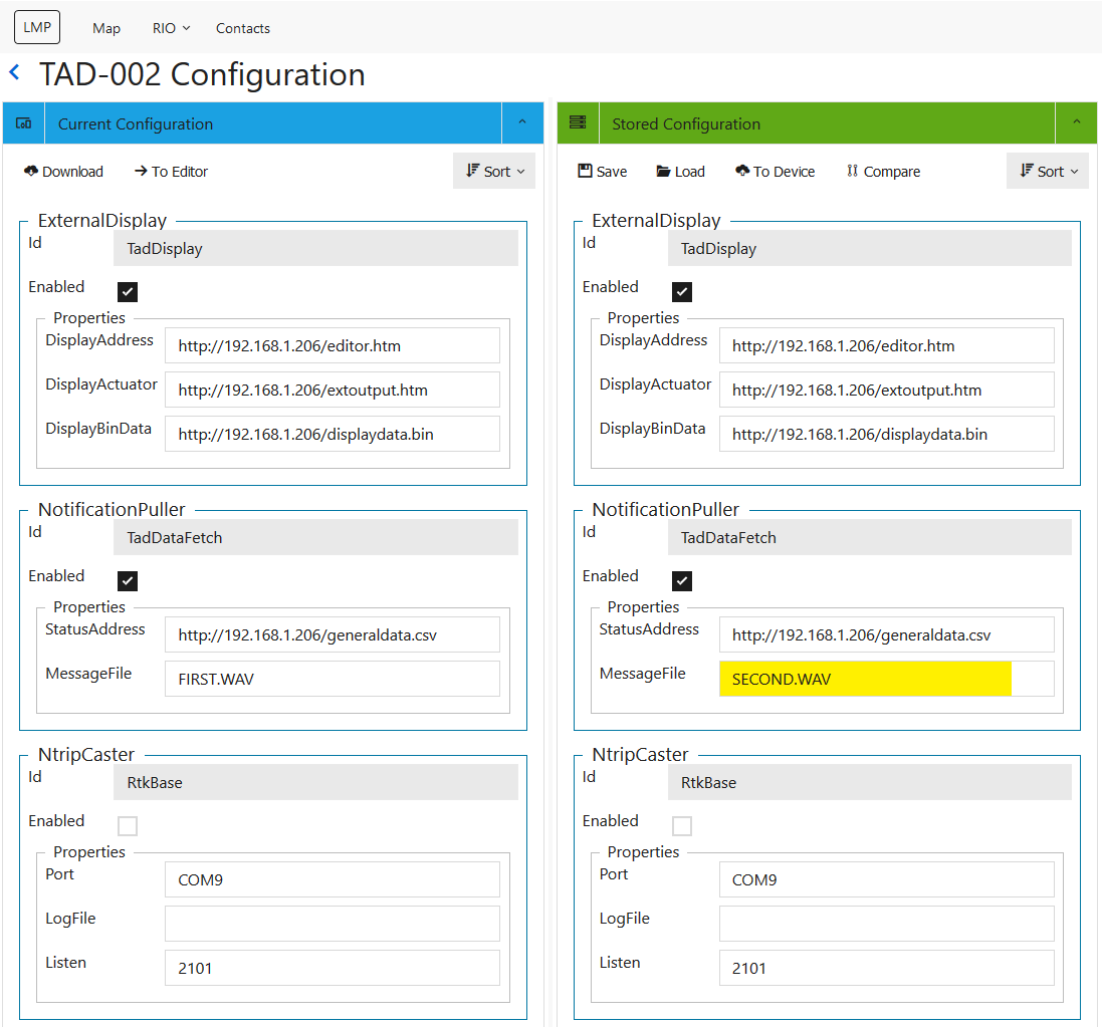


Figure 3 RIO configuration - web interface

It allows also defining the set of rules the device uses to react to the status of the network, e.g. to warn the population on the basis of the alert status of other devices.

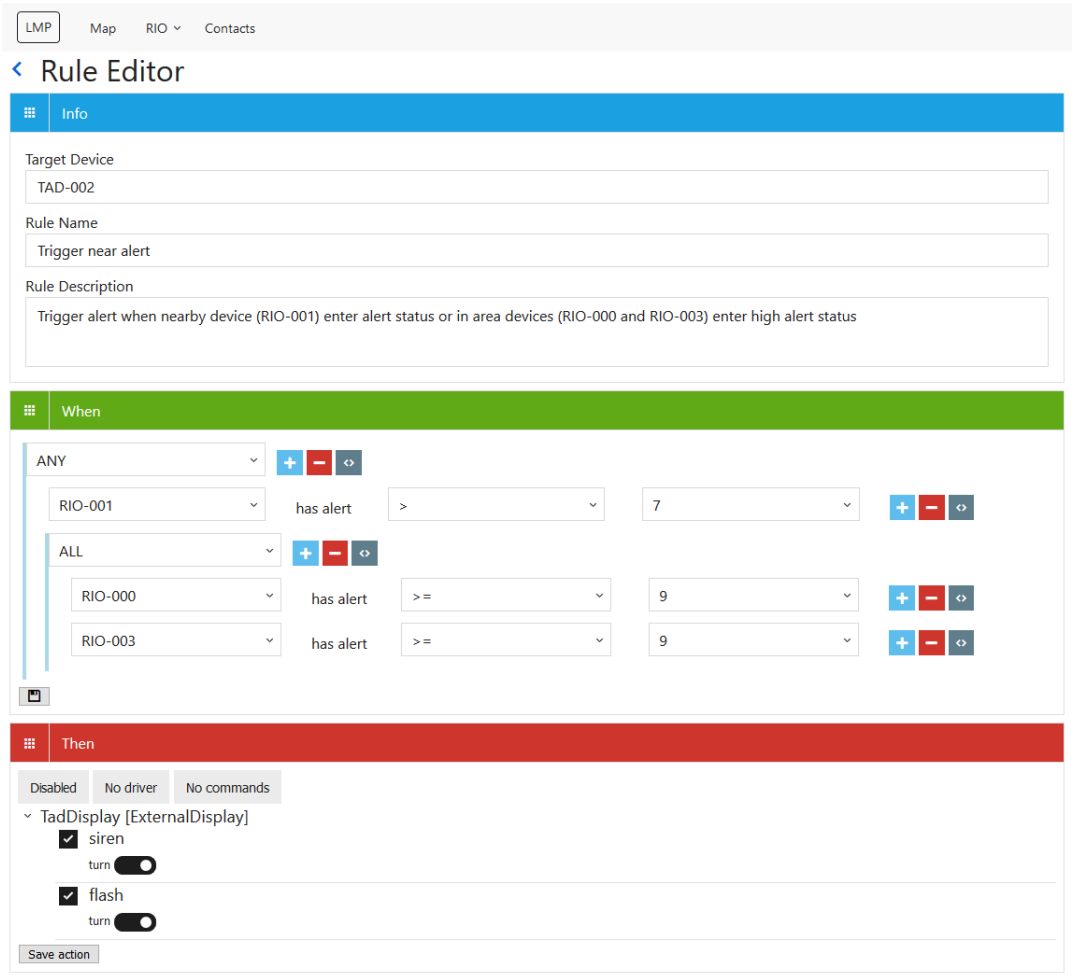


Figure 4 RIO Ruleset definition

The website allows also to post the Telemetry information as a backup of the default method.

Sending alerts to a single device or a set of devices can also be done manually by selecting a point or area in map and then define alert type, level and message.

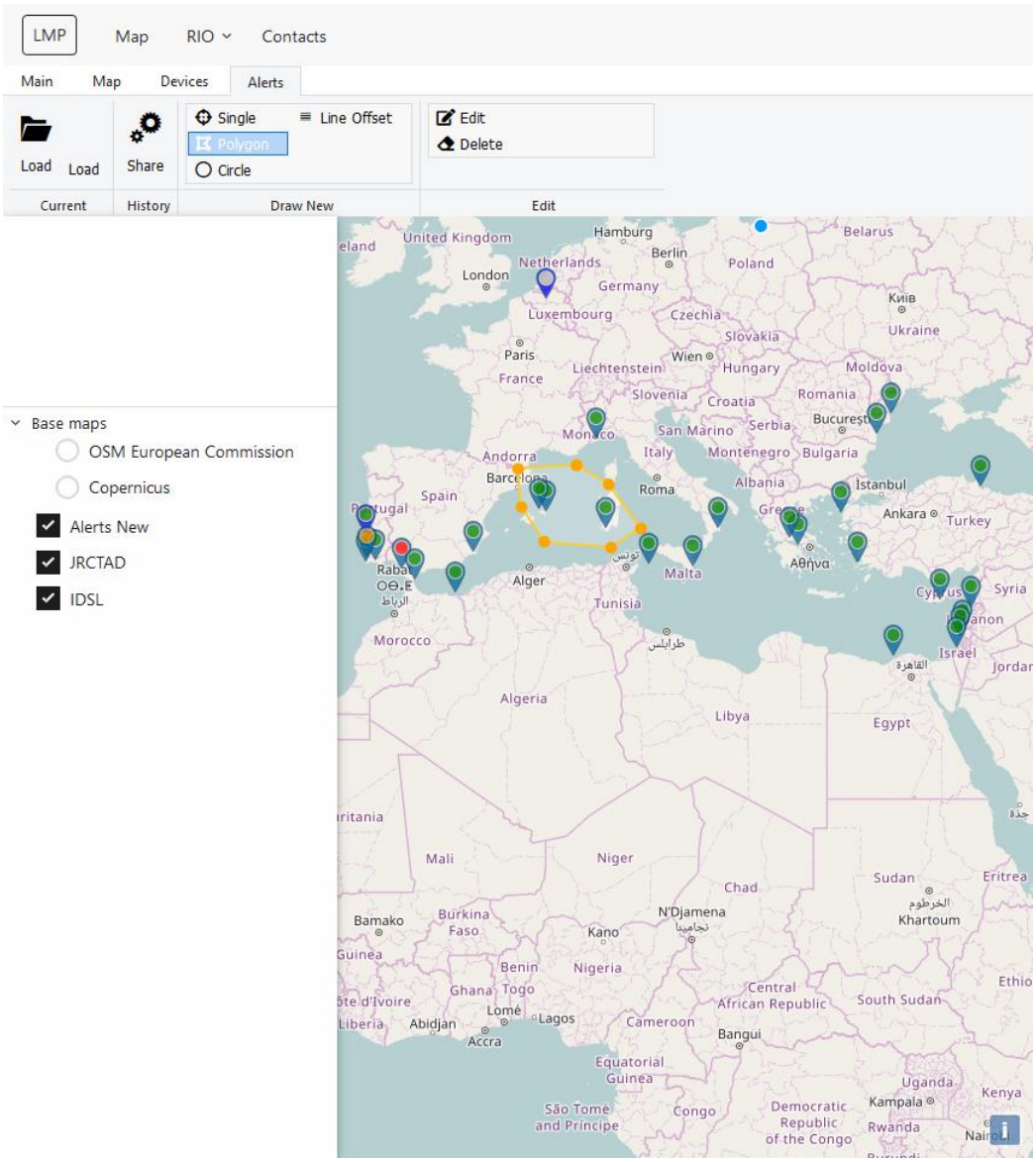


Figure 5 RIO alerting map (work in progress)

The Website implements also a push notification system for alert, telemetry and heartbeat for live monitoring devices status. This is achieved by developing the full website in Microsoft ASP.Net Core 2.2 the latest release of the .Net platform integrating SignalR and allowing website deployment on different platforms (Windows, Linux).

5.2 Message Queue

All messages to and from the RIO devices are sent through a queued messaging system based on REDIS.

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. The messages are piped through channels and the users can publish on the channels or subscribing to receive messages.

This is the only features of Redis used in the RIO infrastructure. Redis has proved effective and it also offers, still to be tested, the capability to be redundant by configuring more than one server in a cluster.

The present implementation offers its services on a connection that is not encrypted and is protected only by a password. It is sufficient for the prototype stage of the RIO infrastructure. The plan is to test the SSL connections as well.

RIO devices use the following channels:

- Telemetry: all telemetry messages are sent over this channel. Usually, only the storage services subscribe to it; but management software can be interested as well.
- Alert: alert messages are sent on this channel by all the sensors detecting an alert or management software which wants to raise an alert
- Heartbeat: the RIO devices post periodically onto this channel their Id and a timestamp, in order to notify they are up and running
- RIO-*Id*-Mgmt: there is a channel like this for every device and it is used to exchange management messages, such as configuration changes and ruleset definition.

5.3 CAP dispatcher

The Common Alerting Protocol (CAP) is a simple but general format for exchanging all-hazard emergency alerts and public warnings over all kinds of networks. CAP allows a consistent warning message to be disseminated simultaneously over many different warning systems, thus increasing warning effectiveness while simplifying the warning task. CAP also facilitates the detection of emerging patterns in local warnings of various kinds, such as might indicate an undetected hazard or hostile act. And CAP provides a template for effective warning messages based on best practices identified in academic research and real-world experience.

JRC adopted CAP since 2008 and endorses its use since then. In this activity JRC found a partner in the Italian Fire Brigade, (Corpo Nazionale Vigili del Fuoco - CNVVF), who has the merit of lobbying the adoption of CAP through the Italian Parliament, so that Italy is the first Member State enforcing by law the use of CAP to exchange alert related information.

The CAP dispatcher used by the RIO infrastructure is internally implemented as a Windows service, which allows client connecting on a well-known port. The client must announce to the server and declare if it is also a reader or just a writer and if it requires past messages before receiving the new ones.

This is an example of session configuration:

```
<SessionConfiguration IsReader="false" Previous="00:20:00" />
```

In this case the client states that it is not interested in receiving the CAPs circulating in the system, but it wants to read those circulated in the last 20 minutes: probably they will be used to avoid duplications.

Previous can be either an absolute date with time, or a time interval.

In any case, all client can post CAPs into the system, one at a time. After posting a CAP, the client must wait for the server acknowledgement, either positive or negative, before sending another one.

In case the CAP is valid, it is broadcasted to all readers before being stored in the database. After these two actions are finished, the client receives its acknowledgement:

```
<message id="CAP identifier" stored="OK"/>
```

In case errors are found, their list is sent back to the client. It will include parsing errors or CAP Missing Data warnings.

5.4 Databases

The RIO infrastructure exploits two different databases: a relational database and a timeseries database.

In order to store the devices information like their configuration, the infrastructure uses the same database developed for all the devices used by CRITECH and published by the Web Access point. This is a SqlServer instance, but any RDBMS can fit.

The timeseries database is used to store all the Telemetry data and the CAP messages. RIAK TS has been chosen at the moment, because it offers a limited Sql programming interface. Not only it is estimated having better performances for a huge amount of data, but by design it also offers redundancy and load balancing: when using several RIAK servers, they share the information in order to distribute the load evenly with some overlapping, in order to allow the system degrading without stopping working in case of a node failure.

6 Conclusions

The base of the RIO OS is already in use in Setubal since February. It is working fine, is very reliable and stable.

Presently, the ongoing tests on the modules and functionality are also up to the needs: a new panel is working continuously since October to assess the reliability of RIO OS. In December 2018 an extended test will prepare the GNSS system to be put in place in the Mediterranean Sea. In the meantime, an IDSL version running RIO OS will also undergo intensive tests.

The infrastructure is not very secure: best part of the protocols used send unencrypted data across the network and only a few restrictions are adopted for incoming connections. It is fine for the present experimentation, but hardening measures are already foreseen:

- SSL channels will be used for sensitive information;
- A PKI will guarantee both ends of the connection about the peer identity.

The problem is that encrypted transmissions burden the RIO device with additional computation requirements; therefore, connection-oriented protocols must be preferred to stateless communications like https.

List of abbreviations and definitions

RIO	Remote InterOperability
IDSL	Inexpensive Device for Sea Level measurement
RPi	Raspberry Pi
TAD	Tsunami Alerting Device
CAP	Common Alerting Protocol

Annexes

Annex 1. Present configuration

Externally available access points of the RIO infrastructure are Web Access point (webcritech.jrc.ec.europa.eu) and the server providing both the Message Queue and the CAP dispatcher (CritechServices.jrc.ec.europa.eu).

Italic means the server is not accessible from outside the infrastructure.

Service	**Server**	**Port**
Web access point	WebCritech.jrc.ec.europa.eu	80/443
Redis	CritechServices.jrc.ec.europa.eu	4005
Riak	*s-jrcipsccrt021.jrc.it*	*8087*
SqlServer	*V-JRCIPSCCRT004*	*1433*
CAP dispatcher	CritechServices.jrc.ec.europa.eu	4002
Ntrip caster	CritechServices.jrc.ec.europa.eu	4004
Ntrip subscription	CritechServices.jrc.ec.europa.eu	4003

GETTING IN TOUCH WITH THE EU

In person

All over the European Union there are hundreds of Europe Direct information centres. You can find the address of the centre nearest you at: https://europa.eu/european-union/contact_en

On the phone or by email

Europe Direct is a service that answers your questions about the European Union. You can contact this service:

- by freephone: 00 800 6 7 8 9 10 11 (certain operators may charge for these calls),

- at the following standard number: +32 22999696, or

- by electronic mail via: https://europa.eu/european-union/contact_en

FINDING INFORMATION ABOUT THE EU

Online

Information about the European Union in all the official languages of the EU is available on the Europa website at: https://europa.eu/european-union/index_en

EU publications
You can download or order free and priced EU publications from EU Bookshop at: https://publications.europa.eu/en/publications. Multiple copies of free publications may be obtained by contacting Europe Direct or your local information centre (see https://europa.eu/european-union/contact_en).

KJ-02-19-010-EN-N

The European Commission's science and knowledge service

Joint Research Centre

JRC Mission

As the science and knowledge service of the European Commission, the Joint Research Centre's mission is to support EU policies with independent evidence throughout the whole policy cycle.

EU Science Hub

ec.europa.eu/jrc

@EU_ScienceHub

EU Science Hub - Joint Research Centre

Joint Research Centre

EU Science Hub

doi:10.2760/0188

ISBN 978-92-79-98770-0

Publications Office of the European Union