Wide-Area Data Acquisition System with a Precise Time Synchronization

Wide-Area Data Acquisition System with a Precise Time Synchronization

M A Popov[1], V P Kapitonov[1], M Y Keyno[2]

[1]Department "Information Technologies and Systems", Far Eastern State Transport University Khabarovsk, Russia
[2]Department "Locomotives", Far Eastern State Transport University Khabarovsk, Russia

E-mail: pm@festu.khv.ru

Abstract. The paper describes a data acquisition system developed in the Far Eastern State Transport University for the state monitoring of the geographically distributed systems. Currently this type of tasks often arises in the various sectors of industry and on the railroads. Using the modern hardware and software platform the prototype of the distributed monitoring system was designed. Various technologies were combined in one solution: sensor networks, microprocessors, data transmitters, wide-area networks, data warehouses, algorithms for data processing and visualization. When data arrive from remote nodes and most of the data contains the fast changing signal arrays, the efficiency of signal processing highly depends on the data synchronization. The main feature of the described system is full automation of the measurements with precision time stamps. It is based on light microprocessor platform operated by Python scripts. The distributed database allows building the united database nodes, which are located on the remote regional servers. The Python-based frameworks were used in order to operate the data in the mid-level and for client interface. The advanced math can be used for the data processing thanks to the precise data synchronization. The proposed approach seems to be very promising for the design of the monitoring systems for the power grids and for the railway infrastructure and rolling stock.

1. Introduction

There is a wide spectrum of tasks for data acquisition from vehicles and other remote objects, including those, located far from the existing communication lines. Sometimes it is very hard to acquire a comprehensive picture of a system behavior and estimate its current state.

In order to identify and analyze the various processes occurring at the objects, it is necessary to synchronize the data. The data should be holistic, the errors and gaps in the aggregation process must be minimized. The main problem for the quality of data aggregation from various remote systems is their synchronization.

Since the satellite global positioning system has been introduced, the optical, radio and wired sync techniques were replaced by this the global available source of precision time. Researchers and developers from many countries around the world use this reliable source of precision time in their works. For example, K. Behrendt and K. Fodero from Schweitzer Engineering Laboratories researched the various aspects of the tolerance and precision of time synchronization in the power grids [1]. R. Kim and his co-researchers from Japan and the United States found that modern low-cost GNSS

receivers are able to provide secure PPS for sync purposes in the measurement systems for low fundamental frequencies [2]. Scientists from Poland and the United States (P. Włodarczyk at al.) developed the low cost multi-channel data acquisition unit with embedded time sync by both UART and PPS ports [3]. The time sync scheme for the mesh type sensor networks was proposed by R. Khosla in his Master's thesis for Norwegian University of Science and Technology [4].

By the common approach, data must be collected and stored in a single information center (Fig.1) for further processing. The data storage system should provide fast access to the collected information, as well as be scalable and distributed to allow future expansion of the system if needed.

The collected and pre-processed data should be displayed using a unified web interface. The protection of equipment, software and data from an unauthorized access must be carried out at the physical and software level.

2. The Structure of measurement system

The developed data collection and analysis system (Fig.1) consists of four main components:

- The object units for data collection;
- The communication subsystem;
- The data warehouse/datacenter for data processing;
- The web-based interface for client's access.

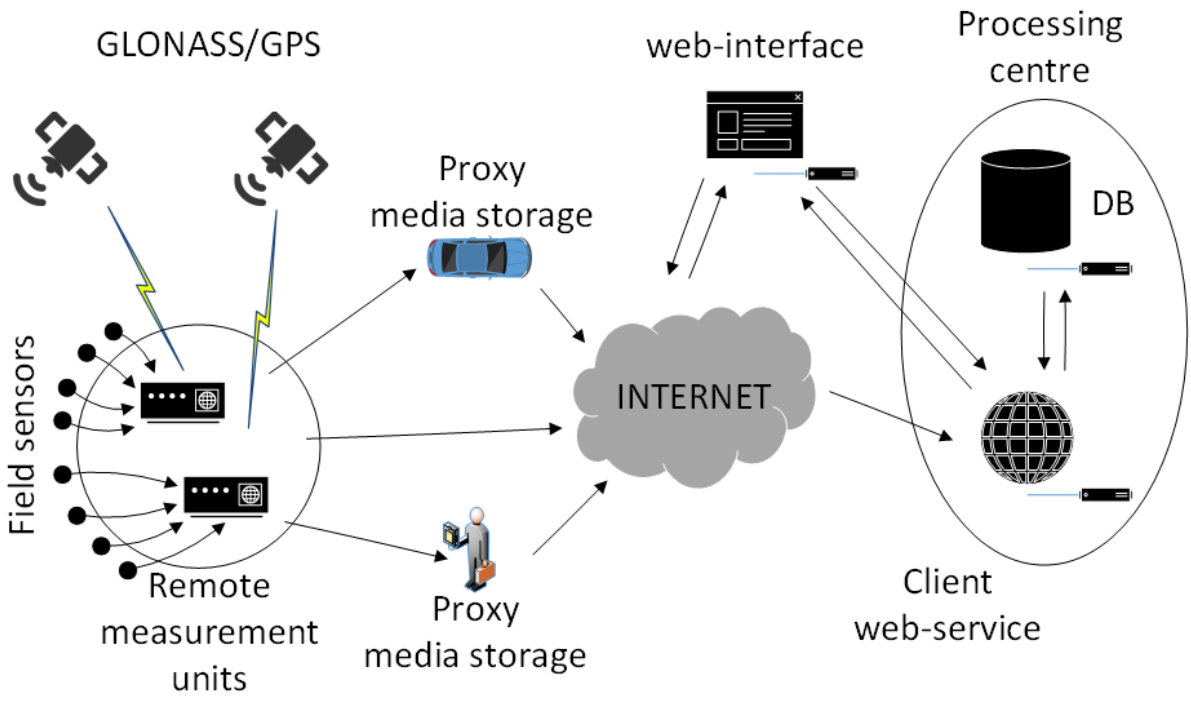

Figure 1. The structure of the proposed measurement system.

An object's data collection units (remote data sources) represent the lowest level of the system. Typically they are based on a MCU or common purpose CPU. Both have onboard timers and can acquire timed sequences of data from physical sensors or an ADC.

The main issue with these timed sequences is their synchronism. Usually, each unit has its own time settings and it means that we can't use the data from different sources in order to compose the whole scene. Nowadays the best way to receive precise time on Earth surface is to acquire the signal from GPS/GLONASS satellites. Thanks to sync by atomic clock, the common use of GNSS as positioning system may be added by using GNSS as a source of precise time.

The communication subsystem is implemented with a hardware platform and software, which are necessary to provide data transfer from a source to the warehouse. The objects, which are connected by copper or optical cables, can transmit data using wide bandwidth channels. The second group of objects can be connected to global networks using wireless technologies. The most sophisticated is transferring data from isolated objects, which operate in the remote areas without wire or wireless coverage. In this case physical media can be used to transfer data to the point, which is connected to the global network. The data can be collected manually using SD cards or automatically using temporary wireless connections to moving locomotives or other vehicles, which are equipped with an onboard proxy media storage, while passing nearby the data source.

The data warehouses are maintained for collection and processing of the large volumes of data, received from various data sources. Additionally they can provide a client's access to the data or results of data processing. Also the data warehouse can operate as a proxy for analytical servers of upper levels.

The client's access subsystem consists of a server- and client-side modules. The server modules interact with databases, process the client's requests, generate the text and the images for the web page which will be sent to the end user and presented by the standard web-browsers.

3. The choice of software for the development of the information system

On the first stage the software development tools and database for implementation of the information system were selected.

The Python language was selected as the main software development language [5]. The reasons for choosing the Python were:

- The simplicity and the speed of the development. When developing small programs with a minimal graphical interface, Python requires less time in comparison with such languages as C, C ++, C #, Java.
- The openness. Python is an open project and it is developed by a large number of programmers. The interpreter of the language is distributed free of charge, and a significant number of programs were written without the use of the specialized technologies and are also freely available.
- The cross-platforming. Programs, which are written in Python, can be compiled under the most of the modern operating systems. So, the same program code, written in Python, can be executed in Windows, OS X and Linux without any problems.

The PyCharm (Fig.2) was chosen as an environment for the development of the project [6]. PyCharm is an integrated development environment for the Python programming language. It provides tools for code analysis, a graphical debugger, a tool for running unit tests and supports web development on Django. PyCharm is developed by JetBrains company and it is based on IntelliJ IDEA.

The capabilities of the IDE PyCharm:

- Static code analysis, syntax and errors highlighting.
- Navigating on the project and on the source code: displaying the file structure of the project, a quick transition between files, classes, and methods.
- Refactoring: renaming, extracting a method, declaration of variables and constants, raising and lowering a method, and so on.
- Tools for web development using the Django framework
- Built-in debugger for Python
- Built-in tools for unit testing
- Developed using Google App Engine
- Supported by version control systems: common user interface for Mercurial, Git, Subversion, Perforce and CVS with support for change and merge lists.

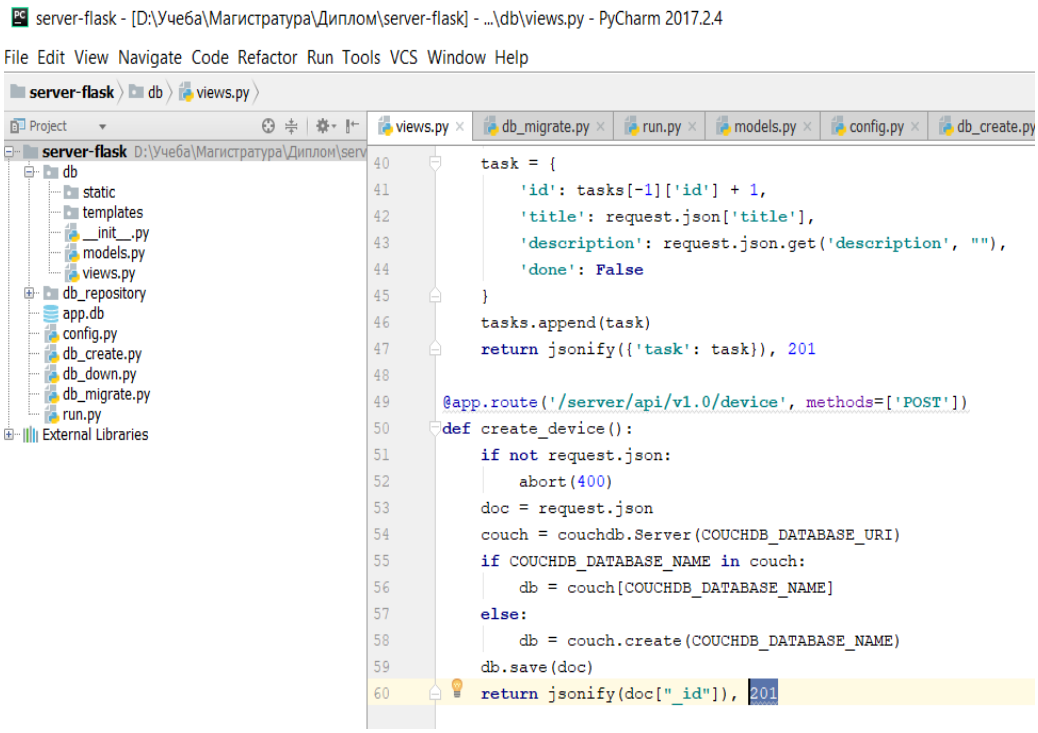

Figure 2. The code view in the PyCharm IDE.

4. Software implementation of data processing system

The data processing system consists of four main subsystems: the software for remote objects, the server software, the database and the client-side software.

The version 3.6 of Python interpreter is necessary for the computer system adjustments at the object's measurement unit. The whole software module consists of 3 files: two contain the source code in Python language and the last keeps the settings. The software modules are automatically run on the restart of the microcomputer.

The first script provides data acquisition and time reading. It is simple by nature – contains an infinite loop, in which uniquely named empty local files in JSON format are created at each iteration. The time stamp from GNSS receiver, which arrives by the UART port, is used to generate the unique file name. Then the script permanently acquires data from local sensors and puts data sets in the memory array. After the given cycle count the script flushes the memory array to the file in JSON format to the local storage device. Simultaneously, together with an ADC polling, the PPS signal from GNSS receiver is read at each iteration and put into the array.

The second script is much more sophisticated – its main task is transferring JSON files to main server and cleaning the local storage. The selected for implementation database CouchDB [7] has a port for direct reception of JSON-objects, but by security reasons the proxy was configured. It gives more flexibility and hides the database server from an open environment.

Like the first script, it has an infinite loop for continuous work. When a new file is found in the data storage directory, the coping procedure will be initialized. Every JSON-file with non-zero size will be transferred to the server by IP-connection. The files, which are created more than 24 hours age, are deleted automatically. When JSON object is received correctly, the server generates an acknowledgement code "201" for the remote system. After receiving an acknowledgement and performing all actions the script activates the sleeping period for the given time period. In case of any error or timeout the data are stored at the remote node and will be transferred to the server in the next session.

The pilot system was deployed for testing purposes. The nodes were located in the Far Eastern region of Russia. The statistical processing of the service information about received packets shows that typical delivery time is about 2 sec. with a range from 1.1 to 2.98 sec. It is suitable for operative monitoring systems, but not for control (Fig.3).

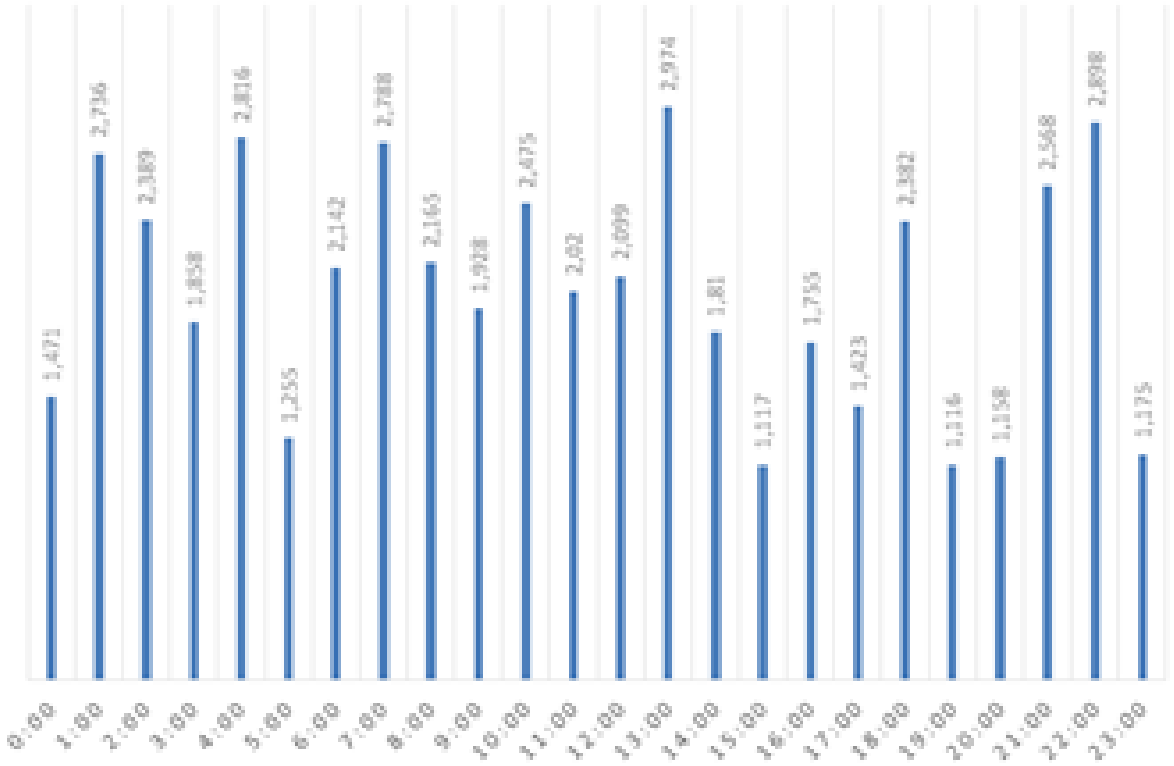

Figure 3. The delivery time fluctuation over day.

The server-side software was created using microframework Flask [8] for Python. The web-API was built using the REST (REpresentational State Transfer) architecture [9]. The server module provides data integrity and the REST requests processing. All JSON objects are stored in the main database and remote access is provided using the web-client software.

The client software is based on the Django framework [10]. All the data are presented in tables and graphical modes. Graphical view provides customizable scroll lines for adjusting the time axis and editing the set of signals. A user authentication is used to grant an appropriate access to data.

5. Conclusions

The performed work proves that with the use of inexpensive modern components it is possible to build a distributed synchronized monitoring system for data collection, storage, and processing.

The use of available navigation receiver modules in bundle with single-board microcomputers and modular ADCs allows creating the efficient and cheap data collection systems that required significant investments previously.

6. References

[1] Behrendt K 2006 The Perfect Time: An Examination of Time-Synchronization Techniques *Schweitzer Engineering Laboratories,Inc* 19

[2] Kim R 2012 Preliminary study of low-cost GPS receivers for time synchronization of wireless sensors *Proc. SPIE 8345, Sensors and Smart Structures Technologies for Civil, Mechanical,*

and Aerospace Systems 83451A (3 April 2012) 9 doi: 10.1117/12.915394
[3] Włodarczyk P 2014 Multi-Channel Data Acquisition System with Absolute Time Synchronization *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 150-154
[4] Khosla R 2017 Time synchronization across multiple devices in network nodes *Magister Thesis. Norwegian University of Science and Technology September* 191
[5] Prokhoryonok N A Python 3 The most necessary things *S.-Peterburg «BHV-Piter»* 461
[6] PyCharm – intellectual IDE for Python URL: https://jetbrains.ru/products/pycharm/ (access date 20.03.2018)
[7] Apache CouchDB 2.1 Documentation URL: http://docs.couchdb.org/en/2.1.1/ (access date 20.03.2018)
[8] Flask Documentation URL: http://flask.pocoo.org/docs/0.12/ (access date 20.03.2018)
[9] Richardson L 2007 RESTful Web Services *O'Reilly Media* 454
[10] Django Documentation URL: https://docs.djangoproject.com/en/2.0/ (access date 20.03.2018)