Internet Protocol Based Digital Counters for Legacy Fatigue Testing Machines

Juraj PANČÍK
Department of Informatics and Quantitative Methods
Bankovní institut vysoká škola, a.s.
Prague, Czech Republic
jpancik@bivs.cz

Marek KUKUČKA
Institute of Automotive Mechatronics
Faculty of Electrical Engineering and Information Technology, Slovak University of Technology
Bratislava, Slovak Republic
marek.kukucka@stuba.sk

Abstract – **In presented paper we describe a design and implementation of an information system based on base of embedded client and Linux application server architecture. Presented information system serves for remote collection, storing and processing information on the number of carried out mechanical cycles for group of older existing fatigue test machines in an automotive components production. This upgrade of existing infrastructure we built on the electronic counter module with Ethernet interface and on the known Internet Protocol (SNMP). Eight electronic counter modules replaced existing electromechanical digital totalizing counters and they count of outputs from 8 pieces of mechanical impulse test machines (each of them with 2Hz output frequency). The web and application servers are built on the Linux operating system. The MVC architecture (Model-View-Controller) serves as the base for web application and it was written together with backend application server around the Ruby on Rails web framework. The database is built on the MySQL server. We carried out three ways to verify of accuracy of this manner counting and collecting of mechanical cycles and we reached results comparable with manual collecting of data. Thanks to automatic detection of ends of fatigue tests we reached more effective utilization of test machines (more than 25%). Our solution of upgrading of the existing industrial infrastructure for fatigue tests is very reasonably priced and is suitable for older industrial facilities.**

Keywords - fatigue tests; mechanical cycles counting; SNMP; automotive testing

I. INTRODUCTION

A fatigue testing machine may be classified from different viewpoints such as purpose of the test, type of stressing, means of producing the load, operation characteristics, type of load etc. The purpose of the investigation is the most important item for the research and of course, this is known when starting the investigation. Hence, it makes sense to base the classification on the purpose of the test [1]. Base on the purpose of the test, fatigue testing machine can be divided into the following [2]: 1. General purpose fatigue testing machine, 2. Special purpose fatigue testing machine, 3. Equipment for testing parts and assemblies (it was our case)

A. Cycle counting

Cycle counting is used to summarize (often lengthy) irregular load-versus-time histories by providing the number of times cycles of various sizes occur. The definition of a cycle varies with the method of cycle counting. These practices cover the procedures used to obtain cycle counts by various methods, including level-crossing counting, peak counting, simple-range counting, range-pair counting, and rain flow counting. Cycle counts can be made for time histories of force, stress, strain, torque, acceleration, deflection, or other loading parameters of interest [3].

B. Specifications of Mechanical cycle

According [3] we applied simple-range counting method. For this method, a range is defined as the difference between two successive reversals, the range being positive when a valley is followed by a peak and negative when a peak is followed by a valley. Positive ranges, negative ranges, or both, may be counted with this method. If only positive or only negative ranges are counted, then each is counted as one cycle. In this text we mean term mechanical cycle as synonymous for one cycle.

C. Electromechanical digital counter

The electromechanical counters are digital counters that serve as a part device of test machines. They give the equivalent value of number of mechanical cycles. In this case the electromechanical counter is synonym of the totalizing counter. Totalizing counter displays a total of events, parts, products, strokes, revolutions, etc. Electronic and electromechanical counters accept a variety of inputs, from sensors, switches, encoders and relays that increment the counter each time a pulse is received. In our case original digital totalizing counter accept inputs 24 V pulses from test machine at end of each mechanical cycle.

II. SYSTEM DESIGN OBJECTIVES

Our goal was creating the information system for collecting, storing, processing and visualization of data from a set of test machines with digital electromechanical totalizing counters. The design of this information system has to take into account existing information system with electromechanical counter of mechanical cycles and with manual data collection. The new information system must enable remote data collection and must take advantage of existing plant network infrastructure at plant (LAN network). As secondary objectives are: price of solutions using modern technology based on the IoT

ISBN 978-80-261-0602-9,

(Internet of Things), ease of implementation, scalability, Web-based user interface and use of modern tools for data processing and their analyzing. Manager's objective is to obtain an overview about of utilization of the testing machines, reducing of downtime of test machines and the developing of visualization management dashboard based on web technologies.

III. REQUIREMENTS

A. System requirements

1) The need to maintain existing electromechanical totalizing counters

2) Increasing of the utilization efficiency of testing machines by reducing downtime (note : tested parts will be suddenly destroyed and test engineers should be informed about it in the shortest time)

3) Automated data collection will replace the manual collection of data from electromechanic counters whose is in connection with manual reading and writing data into the paper sheets

4) As legacy there is the need of ensuring reading data from electromechanical counters (manual reading) and simultaneously from electronic modules (automatic reading)

5) The utilization of existing network infrastructure so that existing counter of mechanical cycles will be resolved with IP-based electronic counter module (IP - Internet Protocol) which will have LAN network connectivity

6) The need to centralize data on accumulated mechanical cycles to the database

7) The need for data evaluation and visualization in one place using the application server

8) Take advantage of client - server architecture, where the client is IP counter module and server will consist of web server, application server and database

9) To ensure long-term reliability and accuracy of mechanical cycles counting

10) To build application servers on the basis of free available open source tools/components

11) Good price of a programming and hardware solutions

B. Electronic Counter IP modules Requirements

1) The counting inputs are electrical pulses from test machine with positive amplitude 0 - 24V (this is also parallel input to the existing electromechanical counter (totalizing counter)) with 2 Hz input frequency (therefore 2 mechanical cycles per second)

2) In the COUNT mode the electronic counter divides of input pulses from testing machine in any ratio (the grouping of mechanical cycles). It will generate output information for application server based on defined number of input pulses

3) Electronic counter IP module has possibility of operating in a simulated input pulses counting mode (PHANTOM mode). In this mode it automatically generates output information the application server

4) IP module supports common internet protocol

C. Application server requirements

1) Possibility of parallel information process entry from at least 8 IP modules (at frequency of 2 Hz input pulses for each IP module, such way the overall frequency can reach 16 Hz)

2) Free of charge LINUX operating system

3) Graphical visualizing web pages (document) generation with application server databases as sources of data

4) Common software framework will serve as base for application server building

IV. DESIGN AND IMPLEMENTATION

A. Design scope

Proposal of our information system is based on the future use of technology IoT (Internet of Things). We decided at this moment for SNMP – it is well-known Internet protocol. Next we decided to take on the market available IP hardware module with SNMP and its known hardware and software solution. Later we can change this decision and we can select more modern IoT protocol as MQTT, DDS etc. Another basis for the system design is using of cheaply operating system and database server. We are using Linux operating system and the MySQL database server. Very important for subsequent processing of the collected data is the ability to export data (CSV file extension) and the ability to direct connection of appropriate analytical server to the MySQL database.

B. Architecture of the system

The system architecture is described at Fig.1. We consider that this figure clearly illustrates the components of the information system. The explanation is needed for the embedded web server, which serves to collect data on temperature and humidity from the sensor. Like IP module for electronic counter this is another type of manufactured IP module [4]. This IP module communicates with its surrounding with HTTP protocol. Our application server respects this and managed provided data as the client. The client parses receive messages via HTTP.

C. Electronic Counter IP Module

The core of electronic counter is prescaler with Atmel ATtiny85 and IP module TCW112-CM [5] (Fig.2 and Fig.3). Optically isolated input signals ensure fault tolerance. Prescaler mode switch ensures COUNT and PHANTOM mode mentioned in requirement part. Prescaler board is powered from the IP module. The prescaler divides the frequency of the input pulses with split ratio 1: 100. Output of the prescaler ensures a signal for the digital input of the module TCW112-CM. On this base the IP module creates one SNMP event per each 100 input pulses (mechanical cycles) which is processed by application server. To avoid of generating SNMP events that belongs to analog input we joined the analog input with analog power (Fig.2). We placed all 8 electronic counter IP modules at one wall mounted panel for presentation purposes for our projects sponsors (Fig.4).

D. SNMP protocol

Simple Network Management Protocol (SNMP) is an Internet-standard protocol for collecting and organizing information about managed devices on IP networks and for modifying that information to change device behavior [6]. Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks and more. SNMP traps enable an agent (therefore in our case the electronic counter IP module) to notify the management station (application server) of significant events by way of an unsolicited SNMP message. Trap includes current `sysUpTime` value, an `OID` numbers identifying the type of trap and optional variable bindings (Fig.7). Destination addressing for traps is determined in an application-specific manner typically through trap configuration variables in the MIB (Management Information Base).

E. Application server side description

We developed on the server side the application with name `'Monitoring Dashboard'`. This web application was written on the framework Ruby on Rails, or simply Rails. The Rails is a Model–View–Controller (MVC) architecture framework, providing default structures for a database, a web service, and web pages. It encourages and facilitates the use of web standards such as JSON or XML for data transfer, and HTML, CSS and JavaScript for display and user interfacing [7]. The application server can be divided into three parts:

1. Web server (front-end). It consists from a web-based user interface, which provides the user output (collected data from database), as well as providing user input (system configuration or data export). The main advantage of web applications is that the user requires no installation and they are independent of platforms (OS) and device (PC, tablet, smart phone) which users access the system. The only requirement is a modern web browser. The web server ensures the user interface for front-end. The application `'Monitoring Dashboard'` uses the web server Puma.

2. Database systems. Relational database management system MySQL records mechanical cycles and also stores system configuration. It consists from several tables with one-to-many relations. Non-relational database system Redis is used by task queues system with name Net-SNMP. The Net-SNMP is packet of tools for work with SNMP protocol. It ensures the receiving of SNMP traps, the notification coming from the electronic counter IP module, processing ones and it ensures its subsequent forwarding to the application server (backend).

3. Application server (Backend). It consists from a set of components that are invisible to ordinary users. Backend application is designed for the Linux platform (ideal for Ubuntu LTS). Backend ensures the following data stream designed for processing each received SNMP messages (each of them represents 100 input mechanical cycles):

1. Electronic counter IP module: sends SNMP trap notification

2. Net-SNMP : SNMP trap notification acceptance (`snmpd` Linux process), processing notifications (`snmptt` Linux process), transmission of relevant data to the web application

3. Backend : receiving data, building job, insertion of the job to the Task Queue (task-queue system is used to prevent the system bottlenecks)

4. Backend : job worker processes tasks in the task queue and writes data to the relational database

F. SNMP traps processing in the application server

To show of incoming SNMP messages we can use LINUX console command: `tcpdump -i eth0 port 162`. Example of the SNMP message dump is at Fig.7. Number of displayed messages should be equal to the number of messages recorded in the database MySQL. Description of template for filtering SNMP traps is at Fig.7. The template works with OID string`'.1.3.6.1.4.1.38783.0.1'`. The processing of the SNMP trap notification begins its assignment to the template and launching the system command that is defined in section EXEC. For each assigned trap notification start to run the following code in the web application (`Resque.enqueue (CounterlabsWorker," $aA", "$x $X")`). This code creates a new task in the task queue 'counterlabs' where the `$aA` is IP address of the sender trap notification and `$x $X` is the time of receipt of the notification. Task queue is processed by an queue worker process, which in an infinite loop checks the status of the task queue `'counterlabs'` and if it found new task of carrying out the method of `'perform'` with parameters IP address and time of receipt, that were recorded in the creation of this task. The method `'perform'` moved these parameters down to the method' `ClabsRecord.receive`'. Method 'receive' moves parameters to method 'commit_record', which creates a new database record and, if check of integrity is successful, record will be stored in the database.

V. RESULTS

Counting of mechanical cycles is a long process. During this time, the number of mechanical cycles lies in orders of hundreds of thousands or millions. It is very important to be sure that electronic counter IP module counts mechanical cycles correctly and without errors.

A. Accuracy verification of counting of mechanical cycles

We checked three ways to verify the accuracy of counting of the mechanical cycles:

1) The verification of the operation of IP module electronics and verification of prescaler solution on the ATMEL AVR base. We used Nanoline PLC (Programmable Logic Controller) from Phoenix Contact, USA, which generates the exact number of 24V pulses as input for electronic counter IP module. The number of pulses generated by the PLC Nanoline is compared with the number of received SNMP traps. Here, we have gained knowledge of the

electrical interferences at unconnected inputs (they generate false SNMP traps) that led to the grounding of the analog input of TCW112CM (see Fig.2, the shortcut between AIN and +VA inputs.).

2) The verification of the application server data throughput. The aim was to verify the application server data throughput so that we have verified the ability to capture SNMP traps in real time. Real-time verification is done by eight IP modules worked in PHANTOM mode. Each prescaler generated in this mode one pulse per minute, which means that an application server received eight SNMP traps per minute. We compared the number of application servers recorded SNMP traps with the calculated number of pulses generated by a calculation of time x input pulses frequency. The last method uses comparison of recorded values of mechanical cycles in the application server database with the values recorded by totallizing counter.

B. Tools for data visualization a data processing

The basic display of measured mechanical cycles is through the website. Fig.5 shows the hourly number of reported SNMP traps for one IP module (1 SNMP trap = 100 mechanical cycles). Example of graphical representation of measured temperature and humidity in the test laboratory is at the Fig.6. The web interface enables the data export to the CSV files and subsequent processing in various tools. Processing capabilities and data analysis, we extended with the R language server based on free Microsoft (previous Revolution) R Server.

VI. CONCLUSION

In described information system we have created the conditions for long-term testing and for the accurate and stable counting of mechanical cycles in the automotive testing. Thanks to automatic detection of ends of fatigue tests we reached more effective utilization of test machines (more than 25%). Our possible contribution is in low cost upgrade older test machines with embedded platform, free of charge LINUX server and web application framework without need of buying expensive industrial grade communication systems. This approach can be our possible contribution to the education of young people which are usually enthusiastic for modern commercial web development but they are not finding a web development in the industry.

REFERENCES

[1] WALODDI, W. Fatigue Testing And Analysis Of Results. Pergamon Press, 1961. ISBN 1483154165.

[2] GBASOUZOR, A.I., OKEKE,O.C., CHIMA,L.O.: Design and Characterization of a Fatigue Testing Machine. San Francisco, USA , 2013. Proceedings of the World Congress on Engineering and Computer Science 2013 Vol I.

[3] ASTM International. Standard Practices for Cycle Counting in Fatigue Analysis, Designation: E 1049 – 85 (Reapproved 2005). ASTM International, 100 Barr Harbor Drive, PO Box C700, West Conshohocken, PA 19428-2959, United States., 2005.

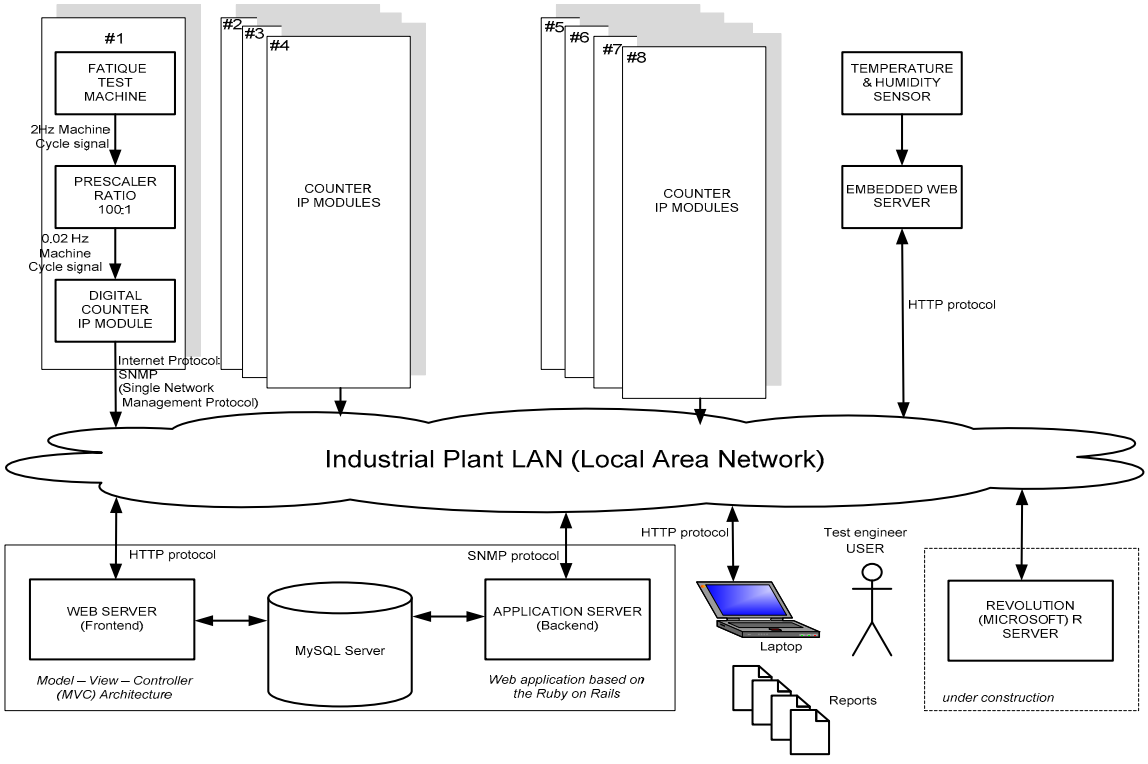
[4] Teracom Ltd. TCW122B-CM - Remote environmental monitoring. [Online] Teracom Ltd. [27.3.2016] http://www.teracom.cc/products/tcw122b-cm-remote-environmental-monitoring/.

[5] —. TCW112-CM - Environmental IP monitoring board. [Online][27.3.2016.] http://www.teracom.cc/products/tcw112-cm-environmental-ip-monitoring-board/.

[6] MAURO, D.R., SCHMIDT, K.J. Essential SNMP. Sebastobol : O'Reilly Media, Inc., 2005. ISBN: 0-596-00840-6.

[7] HARTL, M. Ruby on Rails Tutorial: Learn Web Development with Rails.Addison-Wesley, 2015. ISBN: 0134077709.

Figure 1. Architecture of the system for collection data from fatigue test machines in automotive industry plant



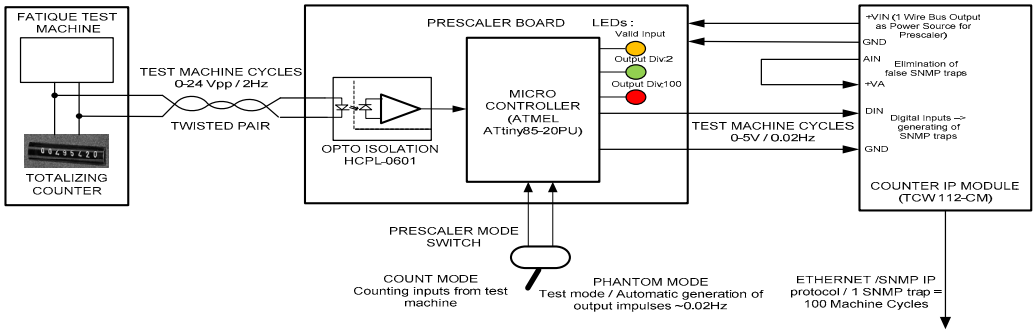


Figure 2. Electronic counter IP module – signal processing diagram

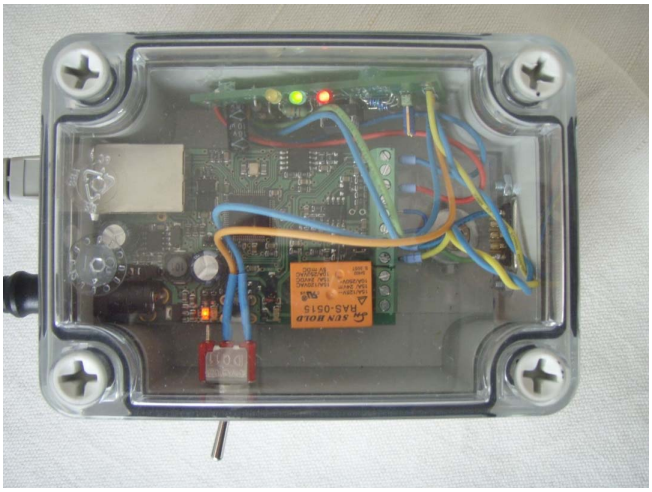


Figure 3. Photography of the electronic counter IP module

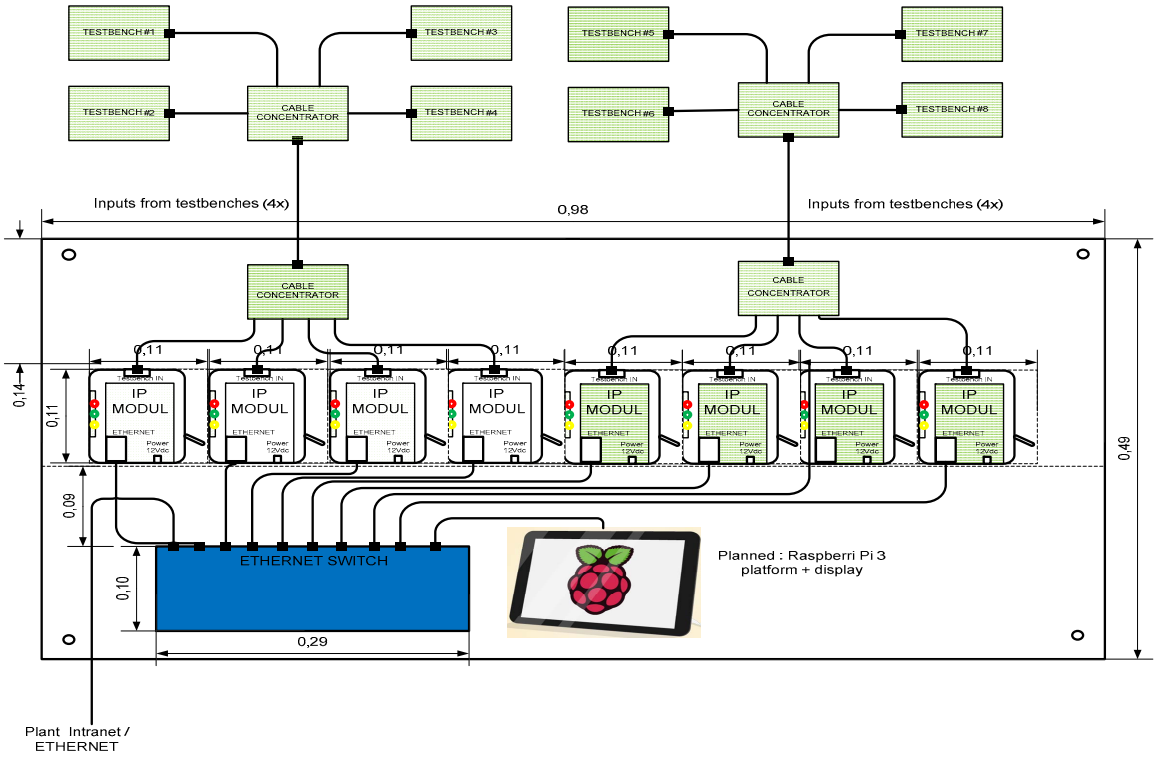


Figure 4. Wall mounted panel with 8 electronic counter IP modules – construction view

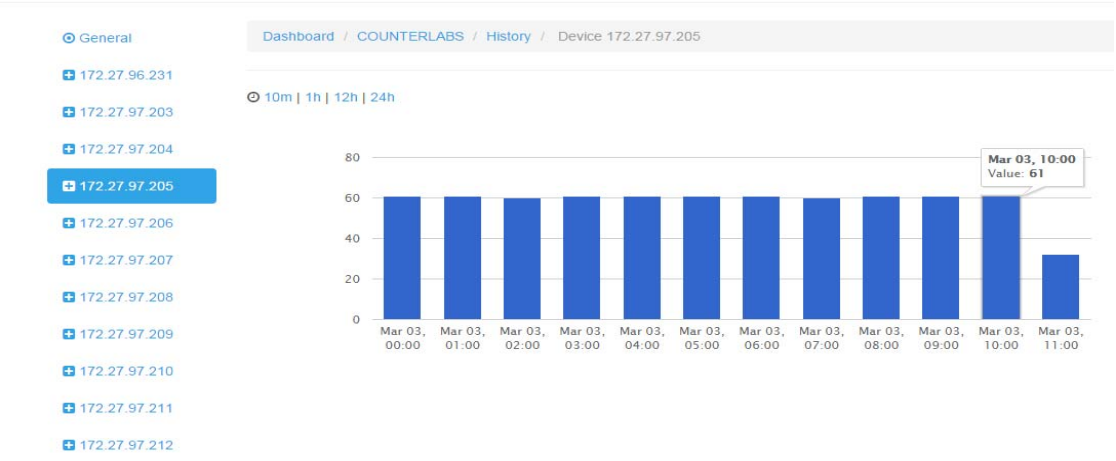


Figure 5. Web graph visualization of mechanical cycles (x100) for 12 hour - recorded from one electronic counter IP module

Figure 6. Web graph visualization of measured temperature and humidity in the test laboratory (Google Chart API tools)

SNMP message output :

```
192.168.1.2.65534 > 192.168.1.1.snmp-trap:  { SNMPv2c { V2Trap(80) R=1
system.sysUpTime.0=647815 S:1.1.4.1.0=E:38783.0.1 E:38783.3.1.0=1 } }
15:31:47.164107 IP (tos 0x0, ttl 100, id 1142, offset 0, flags [none], proto
UDP (17), length 127)
```

Template for SNMP filter trap :

```
moni_dashboard / config / snmptt.conf
#
EVENT clabs .1.3.6.1.4.1.38783.0.1 "Status Events" Normal
FORMAT counterlabs
EXEC cd /var/www/moni_dashboard && bin/rails runner -e production
        'Resque.enqueue(CounterlabsWorker, "$aA", "$x $X")'
SDESC
        counterlabs
EDESC
```

Figure 7. LINUX console command response as one SNMP message and the template for SNMP filter trap