Streaming Pulse Data to the Cloud with Bluetooth LE or NODEMCU ESP8266

Andrej Škraba, Andrej Kološvari, Davorin Kofjač
University of Maribor
Cybernetics & Decision Support System Laboratory
Faculty of Organizational Sciences
Kidričeva cesta 55a, Kranj, Slovenia

Radovan Stojanović
University of Montenegro
Faculty of Electrical Engineering
Džordža Vašingtona bb., Podgorica, Montenegro

Vladimir Stanovov, Eugene Semenkin
Siberian State Aerospace University
Computer Science and Telecommunications Institute
31, Krasnoyarsky Rabochy Ave., Krasnoyarsk,
Russian Federation

Abstract— **The paper describes the development of the three prototypes which enable monitoring of heart pulse sensor data on the cloud. The prototypes are based on a) Bluetooth module, b) Low Energy Bluetooth module and c) ESP8266 Wi-Fi module. The client side implementation is developed by the application of JavaScript. Node.js was used on the server side with application of node-serialport library with Bluetooth modules. In the case where ESP8266 Wi-Fi module was used, the data was transmitted directly to the cloud. The three different prototypes were compared according to the power consumption and complexity of design. The code example is provided illustrating the approach to develop cloud based graphing interface for biomedical mobile monitoring.**

Keywords- pulse sensor; bluetooth; node.js; socket.io; cloud

I. INTRODUCTION

Provision of the mobile, cloud based monitoring of crucial biomedical data from the sensors is needed for online monitoring of patients and elderly people [1, 2, 3]. There are several technologies available in this area however, reliable open source solutions are still in development. New, promising technology in this demanding area is JavaScript / ECMA Script with node.js which enables easy access to hardware protocols and provides efficient interface to the mobile devices and Internet as found in our previous research [4, 5, 6, 7]. Node.js with JavaScript / ECMA Script has been applied in several cases for providing web / cloud interface to the hardware sensors [8]. By the node.js technology the hardware could easily be exposed to the Internet / cloud, simplifying development of both, client side part of the application as well as the server side of application. The possibility to use one language for the both sides also adds to consistency and possibility for rapid development. In [9] the complete IoT system was considered emphasizing the following important subsystems: Driver, Discovery, Management and Repository. Present paper deals mainly on the field of Driver subsystem, touching also the discovery part. There are also possibilities to use mobile network access such as GPRS or 4G LTE [10]. JavaScript / ECMA script with node.js has been indicated as possible technology stack for seamlessly integrating sensor hardware with the cloud [11]. Regarding the difficulties at Bluetooth web sensor applications the latency should be mentioned [11], however for many medical applications the frequency of 50 samples/s might prove fully functional [11]. As the test of the concept, the implementation of putting the sensor data to the cloud with three different technologies will be described: a) Bluetooth, b) Bluetooth Low Energy and c) NODEMCU ESP8266 [12].

II. SYSTEM ARCHITECTURE

Fig. 1 shows setup schematics for the case where the microcontroller Arduino UNO is used to pre-process the data from the sensor (in our case heart rate sensor SEN-11574).

The data is sent by the UART to the Bluetooth module HC-06. On the other side, the Linux Ubuntu 14.04 x86 Intel Compute Stick receives serial data over the integrated Bluetooth adapter and by the node.js server side technology exposes the data from the sensor to the cloud/Internet. The data can be observed in real time by users using smart devices such as phones, tablets, TVs and PCs. The sensor data can be observed by larger number of users with web/Internet access.

The C preprocessing code is put on the Arduino in order to provide proper TX-RX data stream which is then transferred to the Bluetooth module, in our case HC-06. Data is wirelessly conveyed to the integrated Bluetooth adapter on the server side, in our case on Intel Compute Stick Mini PC x386. Upon the Linux the node.js [13] is installed providing the possibility to run JavaScript / ECMAScript code on the server side. Important library is node-serialport [14] providing the functionality of serial protocol over the USB. For the development of GUI, the socket.io library was used. On the top of the software stack is HTML5 / JavaScript / ECMAScript which enables us to develop an appealing GUI which runs on all modern devices with installed Internet browser. This could be accessed from phones [15, 16], Smart TVs, tablets and other devices in order to monitor the sensor data in near-real time, depending on the speed of available network.

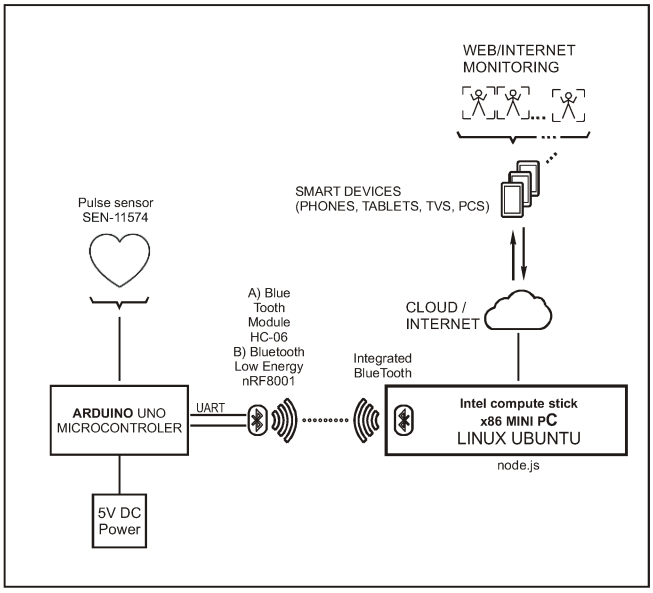

Figure 1: Schematics of putting the analog input from the sensor in the cloud. Here the Arduino UNO is used for preparing the data for serial communication and transmitted with HC-06 or nRF8001 BT LE.

Another possible implementation of putting pulse data to the cloud is shown in the Fig. 2. In this case, we have used NODEMCU ESP8266 module. Comparing the system configurations, the usage of NODEMCU needs less components and is directly connected to the Wi-Fi.

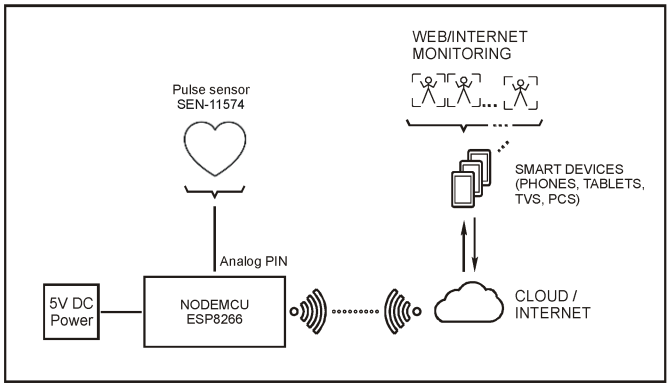

Figure 2: Schematics of putting the analog input from the sensor in the cloud. Here the NODEMCU ESP8266 is used sending data from the pulse sensor to the cloud over the Wi-Fi.

In this case, all the sensor data processing is performed on the NODEMCU ESP8266 and transmitted over Wi-Fi WebSocket to the cloud. Additional data processing can be then performed on the client side with JavaScript.

III. SOFTWARE CONSIDERATIONS

Entire software development, except preprocessing code on Arduino and NODEMCU ESP8266, was done in JavaScript / ECMA Script, both, on the server side as well as on the client side. Client part of the code covers entire GUI which was done in html5. Fig. 3 shows a part of the code with the "sensorRead" function. Function is called by "Value" argument which values are displayed on the interactive chart. The chart is coded in JavaScript / ECMA Script with canvas 2d html5 object providing effective display of the sensor data. Interactive graph only takes a few lines of code and the real time charting is readily available.

```
socket.on('sensorRead', function(value) {
  log(value);
  sensValue = value;
  ctx.clearRect(0,0,canvas.width, canvas.height);
  ctx.beginPath();
  y.splice(0,1);
  y[299] = sensValue;
  for (i=0; i<300; i++) { // Graphing loop
    ctx.lineTo(x[i], (300 - (y[i] / 1023) * 300 ));
  }
  ctx.stroke();
  ctx.fillText(sensValue, 1, 10);
  ctx.fillText("300", 273, 10);
  ctx.fillText("0", 273, 297);
});
```

Figure 3: Part of the client side code where "sensorRead" custom function is engaged, which draws the interactive graph of values gained from the sensor.

As mentioned client and server side of the code is written in JavaScript / ECMA script which eases the burden of different languages usage such as e.g. c / PHP / JavaScript.

IV. HARDWARE REALIZATION WITH RESULTS IN THE BROWSER

We have implemented and compared three hardware realizations that enable us to stream data to the cloud. The configurations are: a) HC-06 Bluetooth module based configuration, b) Low Energy Bluetooth module nRF8001 configuration and c) NODEMCU ESP8266 based configuration. In the case where Bluetooth module were used, the Arduino UNO microcontroller was used for pre-processing data from the sensor. Fig. 4a. shows hardware prototype with (5) HC-06 Bluetooth module, (1) is SparkFun Pulse Sensor, (4) Arduino UNO with TX and RX connection, (2) smartphone and (3) 4AA Battery Pack. In this case the Arduino UNO is used only as the pre-processor for the RS232 protocol providing the proper form of the data from the sensor for the transmission over the Serial. It is certainly costly to have an entire microcontroller board in the design therefore more convenient would be, for example, that the sensor would have embedded RS232 protocol in order to reduce the costs of the components. Such a realization is more convenient for the wearable applications which are found in biomedicine [4, 5, 6]. By developed software stack and hardware realization the results could be monitored in the web browser. In our case we have used Google's Chrome browser which is preferred browser for described application. It is also convenient to use the Chrome browser on the mobile device, such as phone or tablet, as well as, for example, Smart TV. Other browsers can be used but they should have support for socket interactivity.

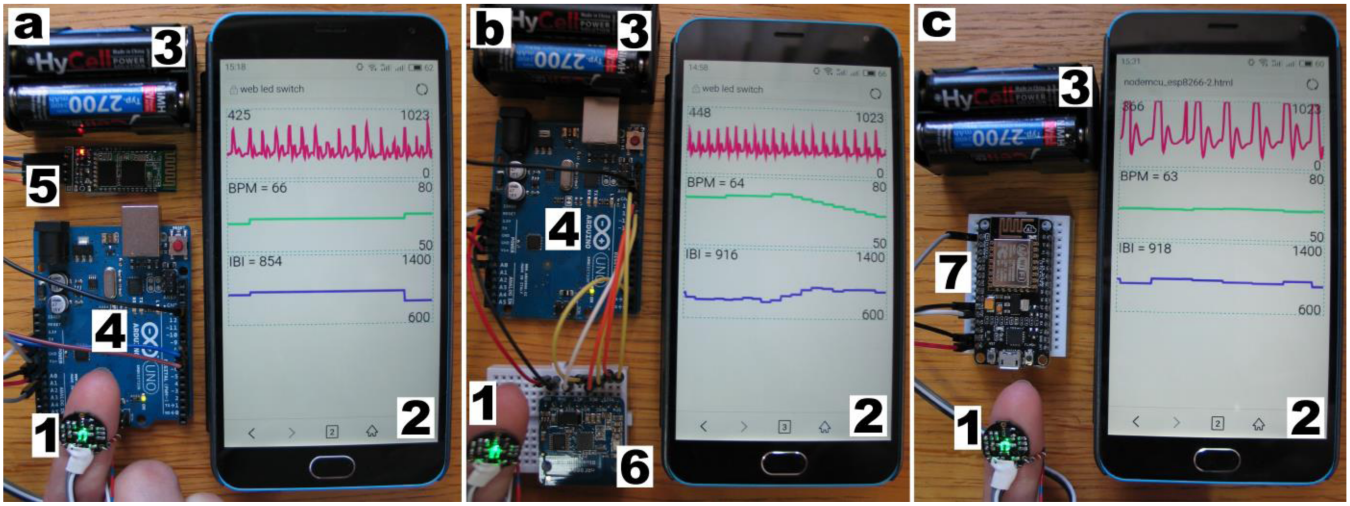

Figure 4: Example of sensor setup with HC-06 BT Module, (1) is SparkFun Pulse Sensor, (2) Smartphone, (3) 4AA Battery Pack, (4) Arduino UNO, (5) HC-06 Bluetooth module, (6) LowEnergy Bluetooth module nRF8001 from AdaFruit, (7) NODEMCU ESP8266.

The Arduino board runs a little bit modified code provided by sensor producer. This code processes the raw data from the sensor, and provides beat rate data, as well as the average time between beats. This data is sent over serial interface via Bluetooth and then decoded by the server. Fig. 4b shows the example of sensor setup with LowEnergy Bluetooth module nRF8001 from AdaFruit (6). The sensor (1) data is preprocessed on the Arduino UNO (4) which also executes the code for the UART communication between nRF8001 and Arduino UNO. The data could be observed on the smartphone (2). The power is provided by the battery pack (3).

Fig. 4c. Shows the realization of the sensor setup with NODEMCU ESP8266. This realization is the minimalistic regarding the hardware components, however, as we will see in the next part of the paper, the power consumption is a major drawback. The main component is NODEMCU ESP8266 (7) module to which the analog data form the pulse sensor (1) is feed. The real-time data could be then observed on the smartphone (2). The system is powered from the battery pack.

TABLE I. MEASURED CURRENTS

Configuration	Current [mA]*
BT HC-06 + Arduino UNO SMD + Pulse sensor SEN 11547	**18**
BT HC-06 Bluetooth module only	7.5
BT LE Adafruit nRF8001 + Arduino UNO SMD + Pulse sensor SEN 11547	**11**
BT LE Adafruit nRF8001 Bluetooth module only / when transmitting	0.4
BT LE Adafruit nRF8001 Bluetooth module only / when not transmitting (search)	0.55
NODEMCU Amica ESP8266 MOD 80Mhz 4Mb RAM + Pulse sensor SEN 11547	**30.5**

* Measured at voltage of 5.04 V

Table I. shows the currents measured for all three configurations, with HC-06 BT, BT LE nRF8001 and NODEMCU ESP8266. Bold underlined current measurements show the values for each of the configurations respectively. One could observe, that the highest current consumption is measured at NODEMCU ESP8266, approximately half of this is consumption with HC-06 and even half less consumes the BT LE nRF8001 based configuration. The Bluetooth Low Energy nRF8001 consumes only 0.4 mA i.e. 19 times less than BT HC-06 module. Here we consider the current consumption only for the communication modules.

Although the NODEMCU ESP8266 is easy to use, the current consumption is the highest being a major drawback if the low energy consumption is of primary importance. Current measured on the pulse sensor SEN 11547 was less than 1mA at 3.3V. By entering the IP address and port, i.e. https://192.168.1.132:8080 one gets the response from the node.js server and the real time graph in the upper left corner is drawn. Under the graph the interactive log of the values is shown in order to monitor the data gained from the Bluetooth module.

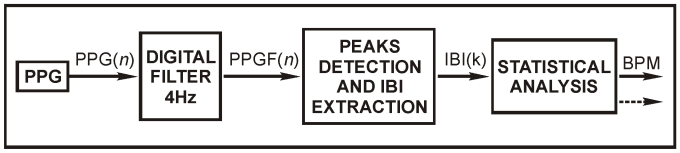

Figure 5: The signal processing algorithm steps, implemented in JavaScript.

The algorithm for processing the raw data, calculating the average number of Beats Per Minute (BPM), as well as the InterBeat Interval (IBI) in case of using NODEMCU was implemented on client's side in JavaScript, as NODEMCU is not fully Arduino-compatible. This implementation is basically the same as C implementation, which includes 4Hz (250ms) digital filter, as well as IBI-based filtering, so that the peaks within 3/5 of the last IBI are not detected. The peak detection stage keeps track of the lowest and highest sensor

value after the last beat, and the peak is detected if the exceeded. The algorithm keeps track of the 10 last IBI values to average them and calculate the BPM. In case if there were no beats for 2.5 seconds, all values are set to their defaults. Figure 5 shows the main steps of the algorithm.

Developed user interface is convenient for implementation, in our case showing only the graphical and numerical representation of the near real-time data flow. The values on the user interface include the time between two beats in ms, as well as the average number of beats per second (GUI in Fig. 4). In our case we have used smartphone with installed Chrome. By entering the IP of the server, the real time data stream could be observed. One should also consider the security issues; in our case the secure connection is established over https. The web page shows the results of the sensor output in real time on the mobile device. The IP used is internal however, if one would provide external IP to the server the results could be observed from the web.

V. CONCLUSION

Development of sensor monitoring by the application of the JavaScript / ECMA Script over Bluetooth and Wi-Fi with the use of the node.js and node-serialport library provides promising platform. Important advantage of proposed interface development is usage of the single language and technology on the software side of the implementation, i.e. JavaScript /ECMA Script, which is growing in its importance in the field of hardware [17]. If one would like to expose the hardware to the Cloud / Internet [18], then one of the possibilities would be described software stack with unifying language and approach based on the Linux and node.js.

BPM / IBI algorithm implementation in JavaScript on the client side successfully distributed the processing burden between client and server.

Three different prototypes vary significantly in a) system complexity and b) in power consumption. If one would strive for minimal power consumption, then the Low energy Bluetooth implementation would be best of the three solutions considered. Regarding the complexity, the application of the NODEMCU ESP8266 is most elegant solution, having only two components, module and the sensor. Further development should address Discovery, Management and Repository issues. This will be of significant importance in the future, where the number of sensors grows and good management of sensor orchestra will be needed in order to obtain the needed data.

ACKNOWLEDGMENT

This work was supported in part by the Slovenian Research Agency (ARRS) (Research program “Decision support systems in electronic commerce”, program No.: UNI-MB-0586-P5-0018), ARRS SI-RF bilateral project «Development of Speech Controlled Wheelchair for Disabled Persons as Cyber-Physical System» Proj. No.: (pending) and Russian Federation Presidential Scholarship No. 16-in-689.

REFERENCES

[1] Z. Tafa and R. Stojanovic. Bluetooth-based approach to monitoring biomedical signals. WSEAS Transactions on Business and Economics. 2006/2.

[2] K Perakis, M. Haritou, R. Stojanovic, B. Asanin and D. Koutsouris. Wireless patient monitoring for the e-inclusion of chronic patients and elderly people. Proceedings of the 1st international conference on Pervasive Technologies Related to Assistive Environments. ACM 2008/7/16.

[3] Kovacevic, Jovan; Stojanovic, Radovan; Karadaglic, Dejan; Asanin, Bogdan; Kovacevic, Zivorad; Bundalo, Zlatko; Softic, Ferid, "FPGA low-power implementation of QRS detectors," in Embedded Computing (MECO), 2014 3rd Mediterranean Conference on , vol., no., pp.98-101, 15-19 June 2014 doi: 10.1109/MECO.2014.6862667

[4] A. Škraba, R. Stojanović, A. Zupan, A. Kološvari, D. Kofjač. Speech-Controlled Cloud-Based Wheelchair Platform for Disabled Persons. Microprocessors and Microsystems, Elsevier, 2015.

[5] A. Škraba A. Kološvari, D. Kofjač, R. Stojanović, Wheelchair maneuvering using leap motion controller and cloud based speech control: Prototype realization. Embedded Computing (MECO) 4th Mediterranean Conference on, Budva, Montenegro. 14-18 June 2015, pp. 391 – 394.

[6] A. Škraba A. Kološvari, D. Kofjač, R. Stojanović, Prototype of speech controlled cloud based wheelchair platform for disabled persons. Embedded Computing (MECO) 3rd Mediterranean Conference on, Budva, Montenegro. 15-19 June 2014, pp. 162 – 165.

[7] A. Zupan, “Sophisticated Wheechairs,”, Rehabilitacija 2007, vol. 6 supl. I, Inštitut Republike Slovenije za rehabilitacijo, Linhartova 51. 1000 Ljubljana, pp. 15–18.

[8] Karagoez, Mehmet Fatih; Turgut, Cevahir, "Design and Implementation of RESTful Wireless Sensor Network Gateways Using Node.js Framework," in European Wireless 2014; 20th European Wireless Conference; Proceedings of , vol., no., pp.1-6, 14-16 May 2014

[9] Kliem, A.; Koner, M.; Weissenborn, S.; Byfield, M., "The Device Driver Engine - Cloud enabled ubiquitous device integration," in Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on , vol., no., pp.1-7, 7-9 April 2015, doi: 10.1109/ISSNIP.2015.7106921

[10] Biswas, J.; Maniyeri, J.; Gopalakrishnan, K.; Shue, L.; Eugene, P.J.; Palit, H.N.; Foo Yong Siang; Lau Lik Seng; Li Xiaorong, "Processing of wearable sensor data on the cloud - a step towards scaling of continuous monitoring of health and well-being," in Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE , vol., no., pp.3860-3863, Aug. 31 2010-Sept. 4 2010, doi: 10.1109/IEMBS.2010.5627906

[11] Carlos, R.; Coyle, S.; Corcoran, B.; Diamond, D.; Tomas, W.; Aaron, M.; Stroiescu, F.; Daly, K., "Web-based sensor streaming wearable for respiratory monitoring applications," in Sensors, 2011 IEEE , vol., no., pp.901-903, 28-31 Oct. 2011 doi: 10.1109/ICSENS.2011.6127168

[12] www.nodemcu.com (Accessed, 15.4.2016)

[13] node.js, http://nodejs.org/ Accesed, 9.3.2015

[14] https://github.com/voodootikigod/node-serialport, Accesed, 10.10.2015.

[15] Ariff, M.H.; Ismail, I., "Livestock information system using Android Smartphone," in Systems, Process & Control (ICSPC), 2013 IEEE Conference on , vol., no., pp.154-158, 13-15 Dec. 2013, doi: 10.1109/SPC.2013.6735123

[16] S. Koceski, N. Koceska and I. Kocev, “Design and Evaluation of Cell Phone Pointing Interface for Robot Control,” International Journal of Advanced Robotic systems pp. 1-9. Vol. 9, 135:2012.

[17] “Why Intel Loves HTML5,” http://software.intel.com/en-us/videos/why-intel-loves-html5 (Accesed, 9.3.2014).

[18] R. Šafarič, M. Debevc, R.M. Parkin, S. Uran S., Telerobotics experiments via Internet. Industrial Electronics, IEEE Transactions on, 2001, 48(2), 424-431