An IoT project for Vital Signs Monitoring

Felician ALECU[1], Paul POCATILU[1], Sergiu CAPISIZU[2]
[1]Bucharest University of Economic Studies, Romania
[2]Bucharest Bar Association, Romania
felician.alecu@ie.ase.ro, ppaul@ase.ro, sergiu.capisizu@yahoo.com

Nowadays, the Internet of Things (IoT) projects are very popular and they are developed for numerous fields. In order to detect various medical problems on time, it is required to monitor the subjects either human or non-human. This could be used on regular or specific activities, like sport or work. It is necessary to determine the factors that could lead to medical problems. Another important aspect is to quantify the factors, to monitor them, to collect data and to make the proper interpretation. This could be achieved using dedicated sensors, controlled by an application embedded on a development board. When a dangerous value is reached, the system has to inform the subject (if human) or someone else (if non-human). This paper presents an Arduino based IoT project used for monitoring the vital signs for human and non-human and the results based on its usage. The paper details the hardware and software components of this project.

Keywords: IoT, Arduino-based development, Vital Signs, Sensors, Communication.

1 Introduction

Vital signs are very important in detecting and monitoring various medical problems from early stages because they show the quality of the basic functions of the body, indicating the general status of a person's physical health. The acceptable values may vary depending on multiple factors, like the person's sex, age, weight, medical background, life style, geographical location, and so on.

The medical theory is consistently pointing out four primary vital signs: body temperature, pulse, blood pressure and respiratory rate. Vitals can be very useful for prevention, while the pain must be considered a sign of an already existing illness.

The body temperature and pulse are showing a general view over a person's health condition, while the respiratory rate and blood pressure are highly specialized indicating potential respiratory dysfunctions or heart failure risks [1].

This is the main reason why the most important vital signs are considered to be the body temperature (BT), and the pulse, also called heart rate (HR), so the continuous monitoring appears to be a must.

Arduino-based development platforms are very popular for the development of such systems due their low price and capabilities [2], being easily programmed and integrated [3].

One important aspect for this solution is the dimension, having in mind that such a device could be used by humans or non-humans.

The paper is structured as follows. The sections *Human Vitals* and *Vital Signs for Non-Human* presents the main factors that can be monitored in order to detect signs of illness or infection for human, respectively non-human. The section *Hardware Design* describes the proposed Arduino-based solution for monitoring the vital signs on human or non-human. *Software Design* deals with the software component of the proposed system. The results are presented in *Findings and Results* section. The paper ends with conclusion and future work.

2. Human Vitals

The normal body temperature recorded at skin level is about 36.5ºC for adults (and 37.0ºC for babies and children). A value between 36.5ºC and 37.5ºC is considered to be normal for adults. Fever (or hyperthermia) occurs when the value is higher than 37.5ºC and it usually indicates infection or illness. The severe fever

DOI: 10.12948/issn14531305/21.2.2017.03

is called hyperpyrexia and it appears when the body temperature is over 40.0ºC.

When the body becomes too hot, the following two critical conditions may occur – heat exhaustion and heatstroke, as described below [4]:

- heat exhaustion – because of the heat, the body starts losing salts and water, leading to degradation of the person's physical condition, like feeling dizzy, faint or sick, weakness, sweating, thirst, muscle cramps, etc. Severe symptoms may include seizures or the loss of consciousness.
- heatstroke – the body is not able anymore to cool down by itself so the temperature is going higher until a dangerous level is touched. A not spotted heat exhaustion may lead to heatstroke that can be life threatening.

A drop below 36.5ºC of body temperature is known as hypothermia (the reverse of hyperthermia), usually being life threatening so it should be addresses as a real medical emergency. Cold environments are usually producing hypothermia, like falling into cold water, having no enough heat in the house or staying outside for a long time in cold conditions without wearing proper warm clothes.

The second important vital sign is the pulse, also known as HR (heart rate), indicating the heart beat rate while pumping blood and usually measured in BPM (beats per minute). The normal adult resting pulse is between 60 and 100 beats per minute, while the athletes may have a resting rate between 40 and 60 beats per minute. The pulse is affected by the activity a person is doing, higher when doing exercises and lower when deep resting or sleeping. The maximum pulse rate a person may achieve without being in danger to lose consciousness can be simply computed by using the following formula:

$$\text{HR\_MAX} = 220 - \text{AGE} \quad (1)$$

So, for example, the HR of a 40 years old adult doing exercises should be under 180 beats per minute, going over is a clear sign of a dangerous situation that must be avoided by taking a few minutes of rest. Exercises are having great benefits over the body, leading to a healthier and happier life.

While it is very important to constantly monitor the vital signs, only a few people are doing this daily, even if it is very simple to quickly check the body temperature and the pulse.

3. Vital Signs for Non-Human

We used statistical analysis to check if the human most important vitals are also vital when discussing about non-humans. For this reason, we used various data sets concerning dogs, cats and other common animals. For example, based on data from Table 1, it is analyzed a data set containing details about horse colic (severe abdominal discomfort that must be treated as emergency because of the high rate of mortality) [7].

Table 1. Factors that influences non-human health conditions

Factor	ANOVA
CellVolume - number of red cells by volume in the blood	31.912
Pulse - heart rate in beats per minute	***31.876***
TotProtein - total protein in blood	28.959
AbdCenTotProt - abdomcentesis total protein (fluid from the abdominal cavity)	6.760
NasogReflpH - nasogastric reflux PH – indicating a gas cap in the stomach	3.3389
RR - respiratory rate	1.985
BodyTemp – higher values may occur due to infection	***0.967***

By applying ranking, we can easily see that the Pulse and Body Temperature are

DOI: 10.12948/issn14531305/21.2.2017.03

important factors explaining the health condition for the horse, as is depicted in Figure 1. The pulse clearly indicates a sign of illness, most of the horses with higher pulse values finally died due to the colic.

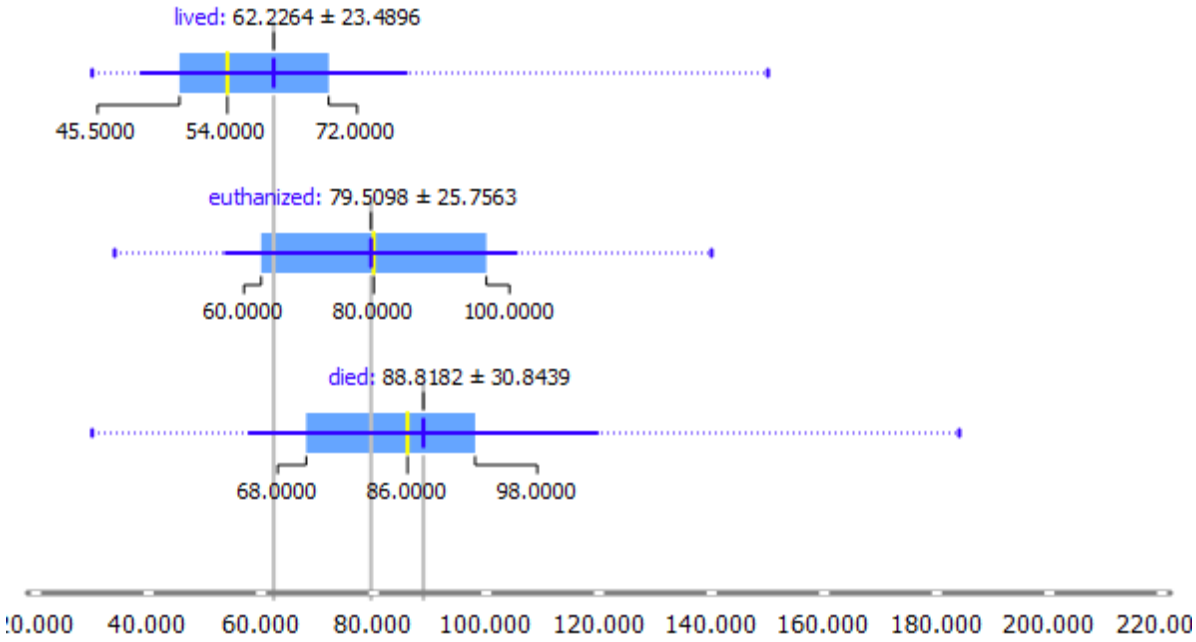

Fig. 1 Ranking results and the BoxPlot chart

PCA (Principal Component Analysis) shows the pulse is by far the most important component explaining the variations of outcome (lived, died or euthanized), Table 2. [7]

Table 2. Results of Principal Component Analysis

Feature name	PC1
Pulse - heart rate in beats per minute	***0.874***
RR - respiratory rate	0.289
CellVolume - number of red cells by volume in the blood	0.142
PeriphPulse=reduced – reduced peripheral pulse	0.006
AbdCenTotProt - abdomcentesis total protein (fluid from the abdominal cavity)	0.006
CapRefill=>=3s - capillary refill time, indicating a poor blood circulation	0.006
BodyTemp – higher values may occur due to infection	0.004

These results lead us to choose to monitor the pulse and the temperature for both human and non-human.

4. Hardware Design

There are several books and papers dealing with Arduino projects, like [10] and [11]. The projects can be developed for almost anything in real life, even for health monitoring.

We can easily notice the human vital signs seem equally important for non-humans as well, so the continuous monitoring becomes a very wise decision that allows to early notice any health issue signs.

In this respect, we propose a system that is depicted in Figure 2. The proposed system is

DOI: 10.12948/issn14531305/21.2.2017.02

based on an Arduino board and it gathers data from pulse sensor (heart rate), temperature sensor, accelerometer, GPS and, in case of emergency, it sends SMS using a GSM module. Also, the system shows the collected data on a display and it uploads data to a server using the cellular line. The board and the components are powered by an accumulator in order to assure portability and autonomy.

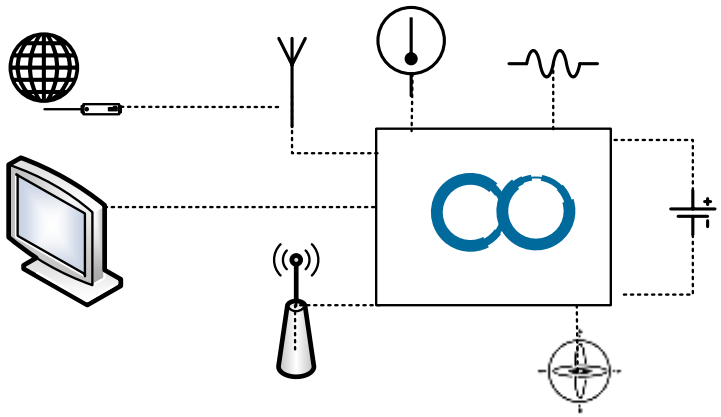

Fig. 2. The proposed system components

On the market, we can find devices being able to track such signs, like [5], or proposals like [6], but we were thinking about the opportunity to design an inexpensive mobile device to be used for continuous heart rate and body temperature monitoring for humans and non-humans, too.

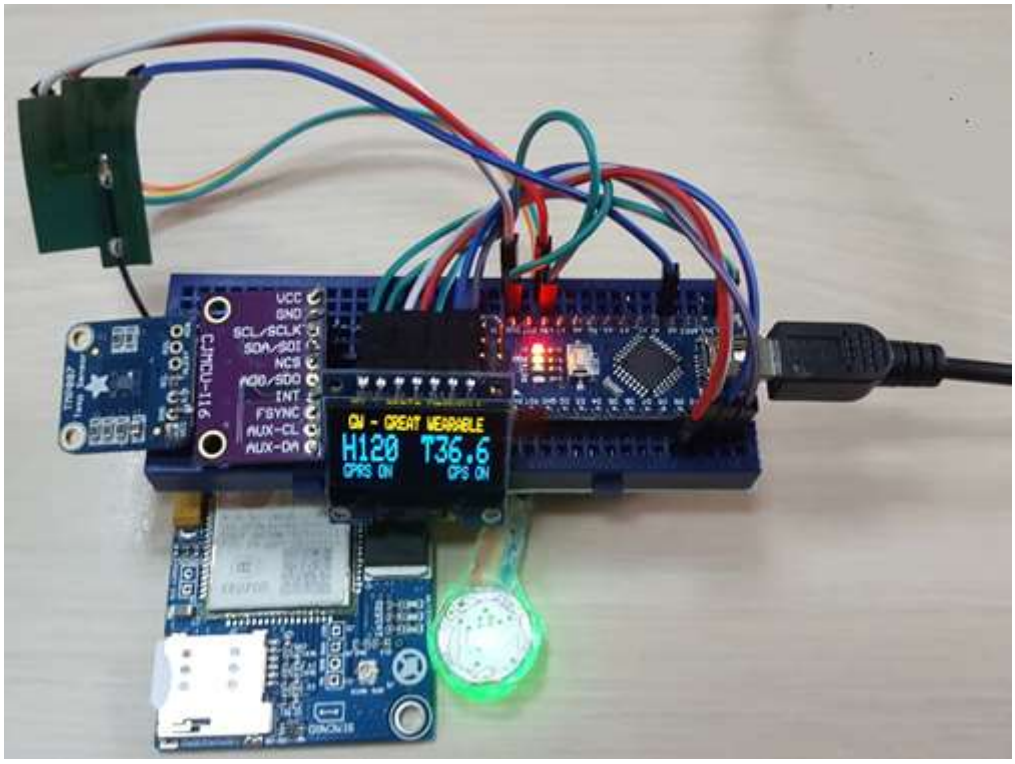

Fig. 3. The Arduino-based assembled device

Figure 3 represents the setup we did by using the components that are briefly presented below.

The *IR Temperature sensor*, TMP007, is an

DOI: 10.12948/issn14531305/21.2.2017.03

integrated microelectromechanical system (MEMS) thermopile sensor that contactless measures the temperature [9]. The sensor can read temperature between -40°C to +125°C and it has a supply voltage between 2.5V and 5.5V.

The HR sensor, XD-58C, works at a sample rate of 500Hz and it is powered at 5V.

The accelerometer is CJMCU-116, based on MPU-6500 Integrated Gravity 6-axis Gyro Acceleration Module with SPI/I2C interfaces. The maximum power supply is 3.3 V. It provides data for three axes.

The GPS, GSM and Bluetooth board is based on SIM808 integrated circuit. The GSM module is used for sending data over a GPRS connection. The location coordinates are provided by the GPS module and are collected by the application in order to track the subject. The supply voltage is between 5 and 18V.

*The OLED displa*y has 0.96 inches and it uses SPI/I2C. The display has a resolution of 128 x 64 pixels and it uses tow colors: yellow and blue.

The development board is *compatible with Arduino Nano* (ATmega328p and CH340).The board works at 16MHz and is powered at 5V. It has 14 I/O pins (6 PWM and 8 ADC).

All components are connected using a *breadboard*. The breadboard can be easily identified in Figure 3 as it connects the Arduino board and the other modules.

The Arduino-based board is collecting data about the heart rate, body temperature and location (one reading per second) and is uploading these details in the cloud by using the GPRS connection at a 3-minute interval (less than 500 entries during a day). This time interval could be customized according to the needs, but for the regular monitoring we consider a 3-minutes timer offers a good balance between the quantity of data and the vitals monitoring benefits.

The system is using the OLED display to show the current readings for the pulse and temperature together with the GPS and GPRS status.

5. Software Design

The collected data are processed by a module developed using the dedicated Arduino environment and third-party libraries when necessary.

Once the setup is finished, the data are uploaded into the cloud once at every 3 minutes in a dedicated so-called channel. The Great Wearable channel settings page is presented in Figure 4.

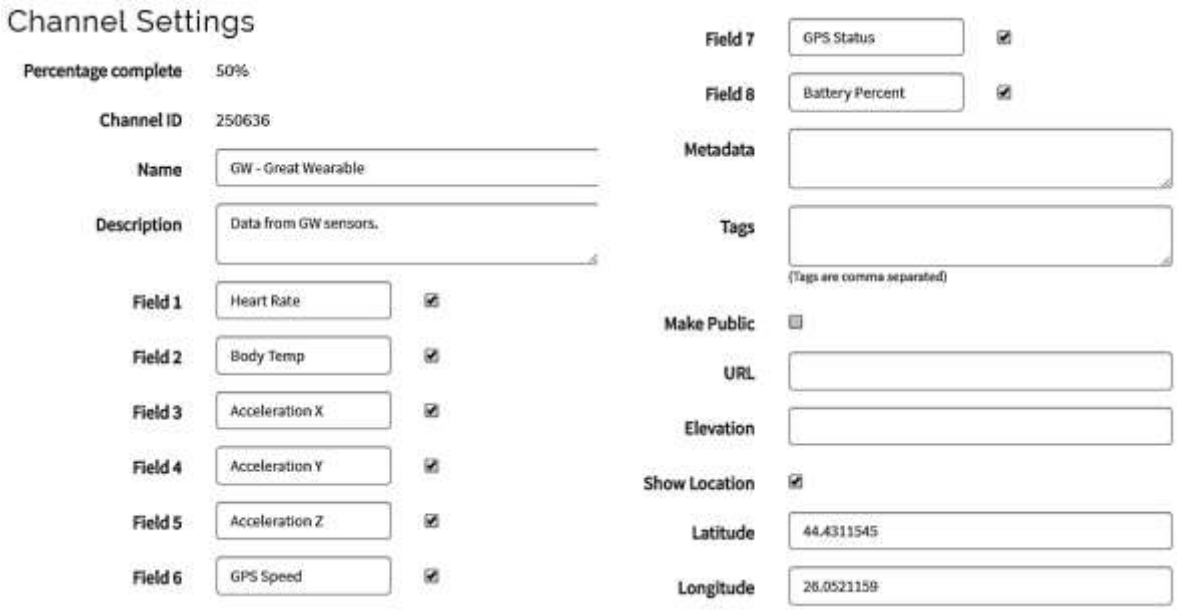

Fig. 4. Great Wearable channel settings page

DOI: 10.12948/issn14531305/21.2.2017.03

We are using the maximum number of fields the cloud platform is supporting for regular users (Field1 to Field8). Also, the location checkbox is on, meaning we can additionally upload the latitude, longitude and altitude for each reading.

The channel data upload is done by using a HTTP GET call directly executed in the code. The design of the Arduino sketch we implemented is presented in Figure 5, each component having its own dedicated module.

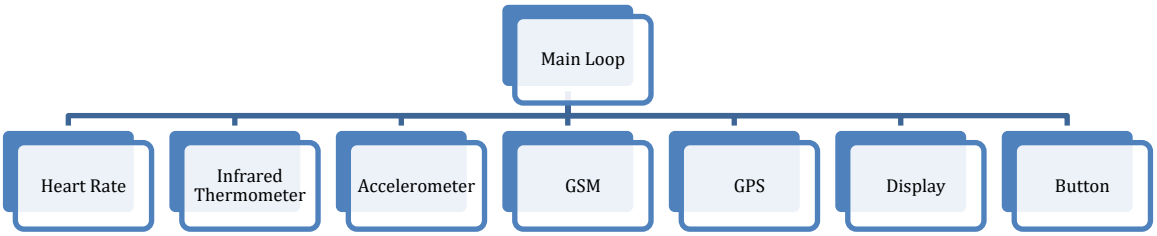

Fig. 5. The software components

Inside the main loop, the program is reading the heart rate for 30 seconds (for accurate results) and is also checking for any SMS received since the user can issue commands by sending text messages (like give me the current heart rate, GPS position or battery level).

For power saving reasons, the display is off all the time, so a button is used to manually turn it on to see the instant readings and the status of the sensors. There is also the option to turn on/off the display by sending SMS commands.

Listing 1. The Main Loop structure

```
void loop(){
  // check if the setup is finished
  if (Setup_in_Progress)return;
  // sms check
  unsigned long lastGSMCheck = millis();
  // 3 minutes for main loop (180 seconds)
  #define main_loop_secs 180
  unsigned long loopStart = millis();
  while(millis() - loopStart < (unsigned long)main_loop_secs * 1000){
    // BUTTON
    button_loop(); // check if pressed
    // PULSE
    if (HR_ON && HR_BPM == 0){
      // no pulse detected yet
      // 30 seconds action for hr update
      #define secs_to_search_for_hr 30
      getHR(secs_to_search_for_hr); // get hr for 30 seconds
    }
    // IR TEMP
    ir_temp_loop(); // get infrared temperature
    // ACCELELROMETER
    acc_loop(); // accelerometer details
    // SMS check at every 30 seconds
    #define GSM_check_in_secs 30
    if (GSM_ON)
      if(millis() - lastGSMCheck > (unsigned long)GSM_check_in_secs * 1000){
        gsm_loop(); // check for incoming sms
        // update the timer
        lastGSMCheck = millis();
        // restore the screen
        display_line_update(display_line_3, false);
      }
    // GPS
    gps_loop(); // gps details
    // DISPLAY
    display_loop(); // display details
```

DOI: 10.12948/issn14531305/21.2.2017.03

```
  }
  // finally, GPRS data transmission
  if (GSM_ON && GPRS_ON)
    sendGPRS();
}
```

The data upload is done by using a HTTP GET call to the following address: http://api.thingspeak.com/update?api_key=MY_API_KEY. The fields to be updated are specified as query parameters, like http://api.thingspeak.com/update?api_key=API_KEY&field1=85, where field1 is defined as being the HR value. The GPRS is turned on only at data transmission time, otherwise it stays disabled for power saving reasons. The GPS data has the following structure: mode, fixstatus, utctime (yyyymmddHHMMSS), latitude, longitude, altitude, speed, course, fixmode, reserved1, HDOP, PDOP, VDOP, reserved2, view_satellites, used_satellites, reserved3, C/N0max, HPA andVPA.

Listing 2 Source code for the sendGPRS() function

```
void sendGPRS(){
  // read gps status
  int8_t gps_status = GSM.GPSstatus();
  update_gps_status(gps_status);
  // get GPS data, if any
  #define gps_data_len 11
  char latitude[gps_data_len], longitude[gps_data_len];
  char altitude[gps_data_len], gpsspeed[gps_data_len];
  char gpsdata[220];
  //
  if (GPS_CONNECTED){
    // read gps data
    // check for GPS location
    GSM.getGPS(0, gpsdata, 120);
    // skip GPS mode
    char *tok = strtok(gpsdata, ",");
    // skip fixstatus and utctime
    tok = strtok(NULL, ","); tok = strtok(NULL, ",");
    // get latitude and longitude
    char *gps_latitude = strtok(NULL, ","); char *gps_longitude = strtok(NULL, ",");
    // get altitude and speed (km/h)
    char *gps_altitude = strtok(NULL, ","); char *gps_speed = strtok(NULL, ",");
    strcpy(latitude, gps_latitude); strcpy(longitude, gps_longitude);
    strcpy(altitude, gps_altitude); (gpsspeed, gps_speed);
  }

  // read battery percent
  VPRC = getBatteryPercent();
  // read acceleration data
  sensors_event_t event;
  if (ACC_ON)
    accel.getEvent(&event);
  // enable GPRS
  GSM.enableGPRS(true);
  // prepare the GET URL
  strcpy(gpsdata, "http://api.thingspeak.com/update?api_key=MY_API_KEY");
  #define field_data_len 11
  char url_data[field_data_len];
  // field1 - HR_BPM
  strcat(gpsdata, "&field1=");
  String(HR_ON?HR_BPM:0).toCharArray(url_data, field_data_len);
  strcat(gpsdata, url_data);
  // field2 - IR TEMP
  strcat(gpsdata, "&field2=");
  String(IR_TEMP_ON?tmp007.readObjTempC():0).toCharArray(url_data, field_data_len);
  strcat(gpsdata, url_data);
  // field3 - ACC_X
  strcat(gpsdata, "&field3=");
  String(ACC_ON?event.acceleration.x:0).toCharArray(url_data, field_data_len);
```

DOI: 10.12948/issn14531305/21.2.2017.03

```
  strcat(gpsdata, url_data);
  // field4 - ACC_Y
  strcat(gpsdata, "&field4=");
  String(ACC_ON?event.acceleration.y:0).toCharArray(url_data, field_data_len);
  strcat(gpsdata, url_data);
  // field5 - ACC_Z
  strcat(gpsdata, "&field5=");
  String(ACC_ON?event.acceleration.z:0).toCharArray(url_data, field_data_len);
  strcat(gpsdata, url_data);
  // field6 - GSP_SPEED
  if (!GPS_CONNECTED) strcpy(gpsspeed, "0");
  strcat(gpsdata, "&field6=");
  String(gpsspeed).toCharArray(url_data, field_data_len);
  strcat(gpsdata, url_data);
  // field7 - GSP_STATUS
  strcat(gpsdata, "&field7=");
  String(gps_status).toCharArray(url_data, field_data_len);
  strcat(gpsdata, url_data);
  // field8 - BATTERY_PERCENT
  strcat(gpsdata, "&field8=");
  String(VPRC).toCharArray(url_data, field_data_len);
  strcat(gpsdata, url_data);
  if (GPS_CONNECTED){
    strcat(gpsdata, "&location=true");
    // latitude
    strcat(gpsdata, "&lat=");
    String(latitude).toCharArray(url_data, field_data_len);
    strcat(gpsdata, url_data);
    // latitude
    strcat(gpsdata, "&long=");
    String(longitude).toCharArray(url_data, field_data_len);
    strcat(gpsdata, url_data);
    // altitude
    strcat(gpsdata, "&elevation=");
    String(altitude).toCharArray(url_data, field_data_len);
    strcat(gpsdata, url_data);
  }
  uint16_t statuscode; int16_t length;
  // send the data
  GSM.HTTP_GET_start(gpsdata, &statuscode, (uint16_t *)&length);
  GSM.HTTP_GET_end();
  // now close the GPRS connection
  GSM.enableGPRS(false);
}
```

According to the compiler feedback, the sketch uses 25228 bytes (82%) of program storage space (maximum is 30720 bytes). Apart the default web based visualizations, for the GPS data (latitude, longitude and altitude) we implemented our own Matlab code showing the GPS points on the map to generate the full geopath. Also, we added some JavaScript functions able to create a GeoChart showing the region (like thw country, province or state) containing all the Global variables use 1438 bytes (70%) of dynamic memory, leaving 610 bytes for local variables (maximum is 2048 bytes).
GPS readings in a time interval.

6. Findings and Results

The visualization of historical data is web based, so the data can be accessed by anyone from anyplace, as illustrated in figures 6 and 7. Figure 6 presents the time evolution of heart rate and body temperature.

DOI: 10.12948/issn14531305/21.2.2017.03

Fig. 6. Representation of data provided by sensors (heart rate and temperature)

Figure 7 presents the geolocation data showed on maps, using the same web interface. The geolocation data can be correlated with the sensors' data within any moment of time. This could be used for further analysis.

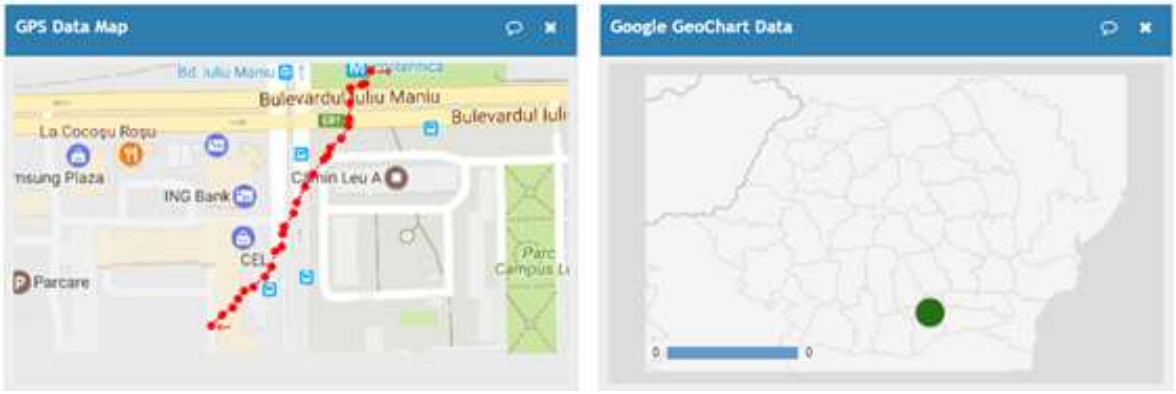

Fig. 7. Representation of geographical data collected by GPS receiver

Despite the web visualization, the application is also implementing a powerful alert system by sending SMS massages and calling to the emergency number the user assigned to the personal account. A SMS messages is sent each time the vital signs are outside the normal limits. So, when the body temperature drops under 36.5ºC or goes over 37.5ºC, an alert is quickly generated. For the pulse rate, the warning system is based on the formula from equation (1), so if the pulse is exceeding this value, the alert is instantly sent. The same for the situation when the pulse goes under 60 beats per minute, the lower limit of the normal range.

If the abnormal situation persists for 15 minutes or longer, the device is calling the user registered phone. If there is no answer, the next step is to call the emergency number defined on the user account.

All these normal range limits can be customized inside the user account in order to reflect person's fitness level and physical conditions, so the alerts can be easily personalized for any particular user.

7. Conclusions

The present paper tries to highlight the enormous importance of the continuous vital signs monitoring for human and non-human subjects. We designed a simple and inexpensive IoT project, a portable device that acquires the vital signs data and uploads the details in the cloud for a web based graphical friendly interface. The real power of this system resides in the alerting component that indicates very rapidly when a vital sign is going outside the normal limits, so the user can take very quick the appropriate measures, like resting, consulting a doctor or calling for an emergency crew.

Future steps involve more system testing and

DOI: 10.12948/issn14531305/21.2.2017.03

fine tuning and extended use in real conditions.
An important and useful future improvement would be to add a voltage regulator (like LM7805) to prevent the LiPo power source discharge to a level below 3V per cell that may permanently damage the battery capacity.

Acknowledgment
Parts of this research have been published in the Proceedings of the 16th International Conference on Informatics in Economy, IE 2017 [8].

References
[1] NHS Choices - Your health, your choices. Internet: http://www.nhs.uk [Mar. 20, 2017].
[2] A. Hayes, *Arduino: A Quick-Start Beginner's Guide*, Amazon Digital Services LLC, 2017
[3] C. Amariei. *Arduino Development Cookbook*, Packt Publishing, 2015
[3] D. L. Schriger. *Approach to the patient with abnormal vital signs* in L. Goldman, D. Ausiello, *Cecil Textbook of Medicine*. 23rd ed. Philadelphia, Elsevier; 2007, chap 7.
[5] e-Health Sensor Platform V2.0 for Arduino and Raspberry Pi [Biometric / Medical Applications]. Internet: https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical [Mar. 20, 2017]
[6] N. Ahmed, R. Banerjee, A. Ghose, and A. Sinharay, "Feasibility Analysis for Estimation of Blood Pressure and Heart Rate using A Smart Eye Wear," In Proc. of the Workshop on Wearable Systems and Applications (WearSys '15), New York, NY, USA, 2015, pp. 9-14.
[7] Horse Colic Data Set. Internet: https://archive.ics.uci.edu/ml/datasets/Horse+Colic [Mar. 20, 2017].
[8] F. Alecu, P, Pocatilu, S. Capisizu, "Human and Non-Human Vital Signs Monitoring," In Proc. of. the 16th International Conference on Informatics in Economy (IE 2017), pp. 128-133, 4-7 May 2017, ISSN 2284-7472
[9] TMP007 Infrared Thermopile Sensor with Integrated Math Engine, http://www.ti.com/lit/ds/symlink/tmp007.pdf Mar. 20, 2017].
[10 M. Geddes, *25 Practical Projects to Get You Started*, No Starch Press, 2016
[11] John Boxall, Arduino Workshop: A Hands-On Introduction with 65 Projects, No Starch Press, 2013

Felician ALECU has graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 2000 and he holds a PhD diploma in Economics from 2006. Currently he is lecturer of Economic Informatics within the Department of Economic Informatics at Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies. He is the author of several articles in the field of parallel computers, grid computing and distributed processing. He holds a Project Management Professional (PMP) certification from the Project Management Institute (PMI), and he is member of the Romanian chapter of PMI.

Paul POCATILU graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 1998. He achieved the PhD in Economics in 2003 with thesis on Software Testing Cost Assessment Models. He has published as author and co-author over 45 articles in journals and over 40 articles on national and international conferences. He is author and co-author of 10 books, (Mobile Devices Programming and Software Testing Costs are two of them). He is professor at the Department of Economic Informatics and Cybernetics

within the Bucharest University of Economic Studies, Bucharest. He teaches courses, seminars and laboratories on Mobile Devices Programming, Economic Informatics, Computer Programming and Project Quality Management to graduate and postgraduate students. His current research areas are software testing, software quality, project management, and mobile application development.

Sergiu CAPISIZU has graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 1997 and National University of Defense in 2005. He holds a PhD diploma in Economic Cybernetics and Statistics, having the title Models and techniques to perform the economic information audit. He is co-author of books and articles in information audit and ICT fields. Also, he has published articles in proceedings of national and international conferences, symposiums, workshops in the fields of data quality, software quality, information audit and juridical aspects in ICT field. He is evaluator of ANEVAR association.

DOI: 10.12948/issn14531305/21.1.2017.03