Automated System for Medication Stocks Management

Adrian MIREA
Faculty of Automation and Computers
Politehnica University Timisoara
Timisoara, Romania
mirea.adrian89@gmail.com

Adriana ALBU
Department of Automation and Applied Informatics
Politehnica University Timisoara
Timisoara, Romania
adriana.albu@aut.upt.ro

Abstract – **The resource management is one of the key factors for any successful and safe activity. This is even more applicable in medical field, where particular demands are necessary. Thus, the physicians (and other medical stuff) are not only responsible for specific medical interventions; they have to also provide the health care environment that includes many other aspects. For instance, one significant activity performed by the medical stuff is the medication stock management. The purpose of this paper is to describe an automated system aimed to help them with this activity. This application is able, at the time being, to receive information from peripheral equipment, to store it into a database and to display all the actions performed by different users of the system. This way, any placement or displacement of medication vials and small dimension medical objects is automatically detected and registered by the system.**

Keywords – medication stock management*; *data acquisition*; *data storage*; *medical informatics

I. Introduction

The automated system presented in this paper was created for hospital establishments, especially for intensive care units (but not only) and it is able to monitor medications with standard weight, and small medical supplies. It has two main sections:

— The data acquisition part is ensured by an Arduino Mega 2560 board that takes the data from peripheral equipment (based on weight sensors and RFID – radio-frequency identification), interprets it and sends this information to the server. This part has already been published by the same authors into a previous paper [1].

— The general context, the storage (into a database) and the use of the acquired data are the subject of the current paper.

As showed in [1], this system has been developed due to the lack of an efficient mechanism for medication stock management. The hospital units do not have hardware equipment able to detect the placement/displacement of medication. Currently, there are only some automated dispensers. Also, the stock management momentarily in use refers only to software tools that register the information into a database. There is no general solution for this issue, even though the subject is treated in recent scientific articles that describe these aspects ([2] and [3], for instance, justify this statement).

Regularly, at any end of a working shift, the medical staff has to spend significant amount of time to refill the medication stock that was used. The system discussed in this paper can be a real help, saving physicians' time. Other advantages may include the minimization of practice errors, omissions and unsafe procedures. More than that, unauthorized use of medication and medical supplies is impossible, thanks to an authentication system based on access cards. There are two types of users that can be identified through RFID tags; one is an administrator and the other type are the regular users. These user types define two states of the management system.

This paper has several sections that describe the proposed automated system for medication stock management. Therefore, the next section presents the architecture of the entire system. Then, some details regarding the implementation are provided in the third section. The results and the conclusions are emphasized into the last two sections.

II. The Architecture of the System

The entire project can be divided in two main parts, as Fig. 1 shows. One sub-system is responsible for data acquisition and interpretation, and the other one for the storage and use of this information.

The most important element of the acquisition part [1] is the Arduino Mega 2560 processing board [4], where the central component is the microcontroller Atmel ATmega2560. The medication vials with standard weight are monitored using two load cells connected to an electric circuit that includes the amplifier HX711 for weight sensors [5], which is also an analog digital converter. The RFID tags (used on some medical equipment and also for the authentication system) are monitored by a circuit that contains the RFID-RC522 chip, which is a RFID cards reader. In order to transfer the data between the Arduino board and a local network, an Ethernet Shield [6] is used. This way, the interactions with physical environment can be stored for further use.

The second sub-system of the medication stock management implies the storage and processing of relevant data and actions performed by different users. A XAMPP Apache distribution has been used for this purpose. It is an

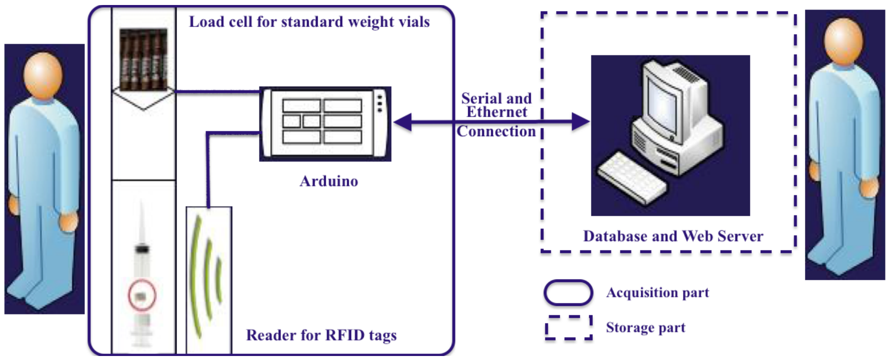


Fig. 1. The system's architecture.

open-source solution which contains the most common web development technologies in a single package [7]. Its name defines its components: X – cross-platform, A – Apache HTTP server (one of the most frequently used web servers in the world), M – MySQL (a management system for relational databases), PP – PHP and Perl (scripting languages). Therefore, this is a minimal configuration for web services development; nevertheless, it is very suitable for testing and distribution.

A database has been created in order to store the information of the management system. It contains four tables (Fig. 2 describes their fields and the way they are related). These tables can be grouped in three categories, according to the main function they are responsible for:

— one table stores the information received directly from the monitoring system: `arduino.med`;
— two tables contain general data about the management system: `arduino.user` and `arduino.sensor`;
— one table registers the actions that are performed inside the system: `arduino.actions`.

A notable feature of any medication, besides its commercial name, is the concentration of the active substance. For this reason, the primary key of `arduino.med` table contains these two elements. The table also has a foreign key

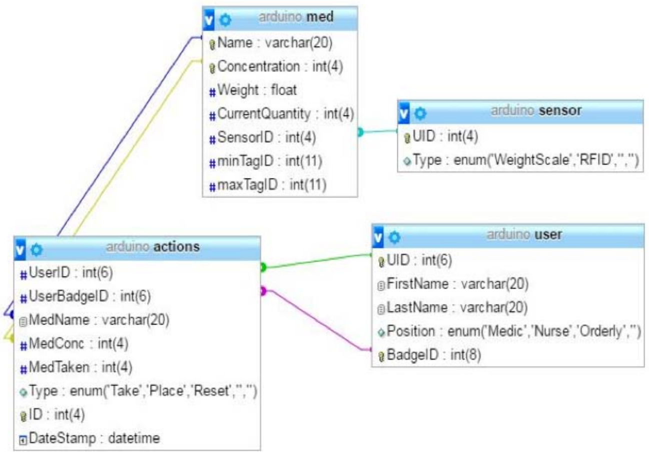


Fig. 2. The storage sub-system.

that makes the connection to the sensors, in order to automatically identify the records, if a sensor is removed from the system. These characteristics can be observed in the following SQL code:

```
-- Indexes for table `med`
ALTER TABLE `med`
  ADD   PRIMARY   KEY    (`Name`,`Concentration`)
USING BTREE,
  ADD KEY `SensorID` (`SensorID`);
-- Constraints for table `med`
ALTER TABLE `med`
  ADD   CONSTRAINT   `med_ibfk_1`   FOREIGN   KEY
(`SensorID`)  REFERENCES  `sensor`  (`UID`)  ON
DELETE CASCADE ON UPDATE CASCADE;
```

The `arduino.actions` table stores the most information about the medication that are used or added, and about the users who perform these actions. Therefore, it has the most foreign key dependences. Moreover, the primary key is auto-incremented.

```
-- Indexes for table `actions`
ALTER TABLE `actions`
  ADD PRIMARY KEY (`ID`),
  ADD KEY `UserID` (`UserID`),
  ADD KEY `UserBadgeID` (`UserBadgeID`),
  ADD    KEY    `MedName`    (`MedName`,`MedConc`)
USING BTREE;
-- Constraints for table `actions`
ALTER TABLE `actions`
  ADD  CONSTRAINT  `actions_ibfk_1`  FOREIGN  KEY
(`UserID`) REFERENCES `user` (`UID`) ON DELETE
CASCADE ON UPDATE CASCADE,
  ADD  CONSTRAINT  `actions_ibfk_2`  FOREIGN  KEY
(`MedName`,`MedConc`)      REFERENCES      `med`
(`Name`, `Concentration`) ON DELETE CASCADE ON
UPDATE CASCADE,
  ADD  CONSTRAINT  `actions_ibfk_3`  FOREIGN  KEY
(`UserBadgeID`)  REFERENCES  `user`  (`BadgeID`)
ON DELETE CASCADE ON UPDATE CASCADE;
-- AUTO_INCREMENT for table `actions`
ALTER TABLE `actions`
  MODIFY  `ID`  int(16)  NOT NULL  AUTO_INCREMENT,
AUTO_INCREMENT=183;
```

These two tables (`med` and `actions`) are updated by PHP scripts stored on the Apache web server when specific commands are received from the Arduino board via Ethernet connection. The other two tables (`user` and `sensor`) require direct access to the database in order to add records.

III. The Implementation

The application has a design that is based on a state machine; the current state depends on the previous one and, also, it will determine the next one. This is due to the fact that there are two types of users and the allowed actions are different for each of them. Fig. 3 is an UML model of these actions. The transitions between states are executed only if the conditions written in square brackets ('`[...]`') are accomplished. Next, the actions written in curly brackets ('`{...}`') are executed once. Inside the three possible states (`LOGIN`, `ADMIN_SUPPLIES` and `MONITOR_SUPPLES`) there are some keywords that specify the moment and the way the functions are executed: `'entry'` – at the beginning of that state, `'do'` – repeatedly, while the application is into that state, and `'exit'` – when the state is left.

The structure presented in the Fig. 3 is matching an Arduino sketch, which has two main functions: `setup()` – called once, when the applications is powered on or after a reset, and `loop()` – repeatedly called during the execution of the application. These two functions have been described by [1] from data acquisition point of view. Here are the functional operations.

Following the design from the Fig. 3, the first functional operations load from EEPROM persistent memory the weight measured by the sensors (`ReadEEPROMData()`). The weight values stored into the database are also necessary in order to establish if, during a system power failure, some unacceptable actions have been performed. The pseudocode function `checkDBSensors()` executes the real function `queryDB_Weight(weightValue, sensorNo, operation, CardID)`, which interrogates the database, returning the weight value registered for that sensor. If the difference between the two values (EEPROM and database) is greater than 1.5g (`EEPROMSuppliesDataNOK`), then the application is lead to `ADMIN_SUPPLIES` state, in order to adjust the inaccurate values. If the weight values are all right (`EEPROMSuppliesDataOK`), then the last logged user is verified (`checkLastLoggedUser()`), to see if one is authenticated and to determine what type of access that user will have. There are three possible results here, also emphasized by LEDs: `lastLoggedIsAdmin` (yellow LED), `lastLoggedUnknown` (red LED) and `lastLoggedIsUser` (green LED).

Once the setup is completed, the application starts the repeatedly execution of its main functions. Most of the time, the system will be in the `LOGIN` state, due to inactive times, that request a re-authentication (after 60 seconds without

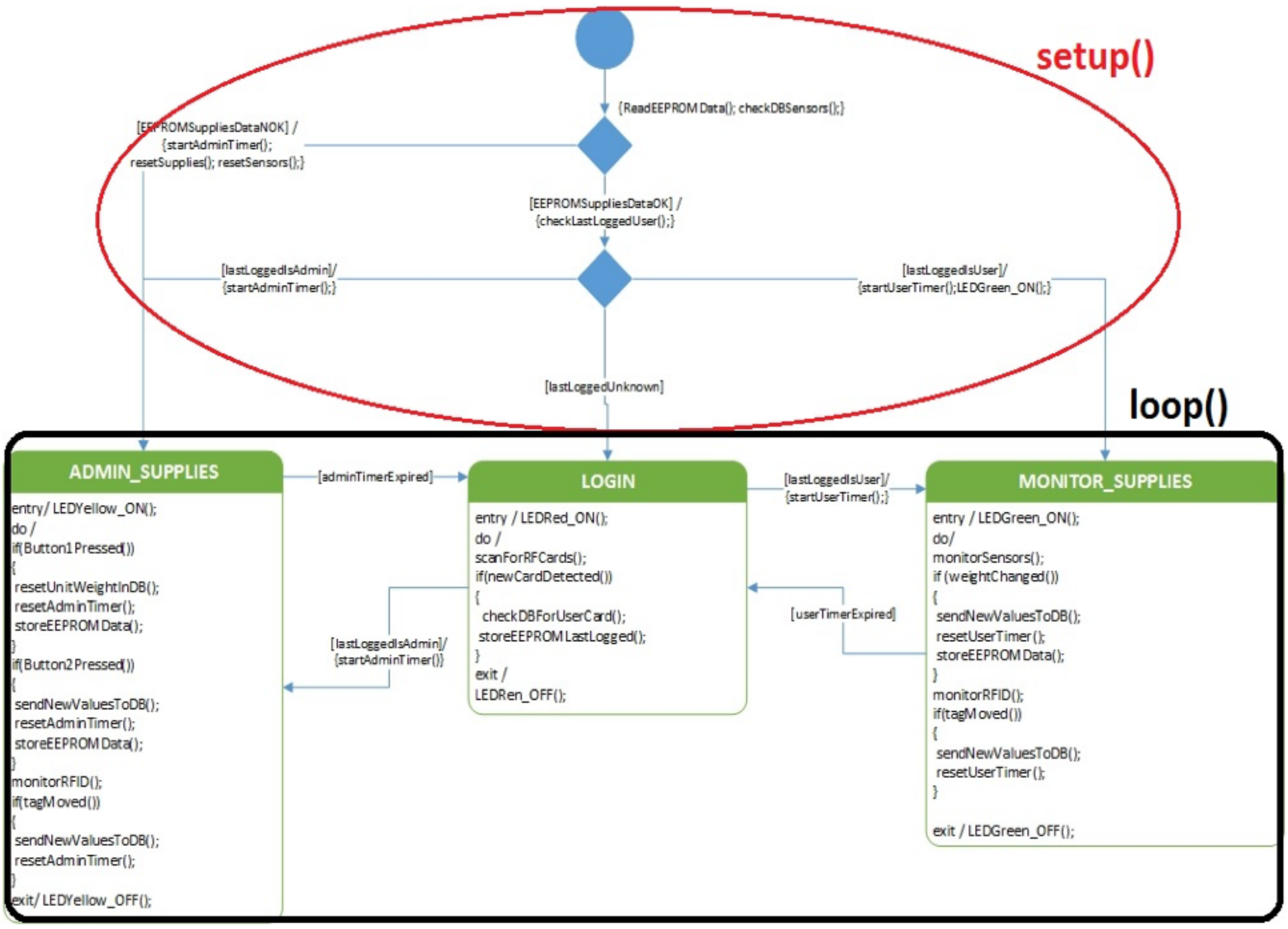

setup()
{ReadEEPROMData(); checkDBSensors();}
[EEPROMSuppliesDataNOK] / {startAdminTimer(); resetSupplies(); resetSensors();}
[EEPROMSuppliesDataOK] / {checkLastLoggedUser();}
[lastLoggedIsAdmin]/ {startAdminTimer();}
[lastLoggedIsUser]/ {startUserTimer();LEDGreen_ON();}
[lastLoggedUnknown]
loop()
ADMIN_SUPPLIES
entry/ LEDYellow_ON(); do / if(Button1Pressed()) { resetUnitWeightInDB(); resetAdminTimer(); storeEEPROMData(); } if(Button2Pressed()) { sendNewValuesToDB(); resetAdminTimer(); storeEEPROMData(); } monitorRFID(); if(tagMoved()) { sendNewValuesToDB(); resetAdminTimer(); } exit/ LEDYellow_OFF();
[adminTimerExpired]
LOGIN
entry / LEDRed_ON(); do / scanForRFCards(); if(newCardDetected()) { checkDBForUserCard(); storeEEPROMLastLogged(); } exit / LEDRen_OFF();
[lastLoggedIsAdmin]/ {startAdminTimer()}
[lastLoggedIsUser]/ {startUserTimer();}
[userTimerExpired]
MONITOR_SUPPLIES
entry / LEDGreen_ON(); do/ monitorSensors(); if (weightChanged()) { sendNewValuesToDB(); resetUserTimer(); storeEEPROMData(); } monitorRFID(); if(tagMoved()) { sendNewValuesToDB(); resetUserTimer(); } exit / LEDGreen_OFF();

Fig. 3. The software design.

activity, a user is automatically logged off). The authentication is performed using functions from `MFRC522.h` library, which implement the pseudocode instructions `scanForRFCards()` and `newCardDetected()` from the state diagram:

```
// Look for new cards
if(!mfrc522.PICC_IsNewCardPresent()){
  return;
}
// Select one of the cards
if(!mfrc522.PICC_ReadCardSerial()){
  return;
}
//Make sure scanned RF is not a TAG
if(mfrc522.uid.sak == 0){
  return;
}
```

There is an aspect that should be analyzed here: the difference between an access card and a medical supply with RFID tag, both of them being recognized the same way. This is solved using the byte `sak`, which identifies a receiver of type `PICC_TYPE_MIFARE_UL` (value `0`) or `PICC_TYPE_MI-FARE_1K` (value `8`) or other card types, unused into this project.

If the object is recognized as an access card, not a medical supply, it has to be searched into the database in order to identify its presence into the system and to decide if it is an admin or a regular user. The serial communication is changed from RFID mode to Ethernet Shield mode. Then, a connection to Apache web server is established (for the actions that involve the database). The IP address of the server and the port 80 are used for this connection. There are two methods available for sending information to a web server: `GET` and `POST`. The `GET` one was used by this project because it is not necessary to communicate secure information (passwords, images, medical documents). It is also more suitable for the PHP language. At the end of this part, the communication is set back to RFID mode. The entire function is presented below:

```
unsigned   char   search_IDinDB(unsigned   char
CardID){
  unsigned char returnValue=0;
  char c[1];
  /*ChipSelect the ETHERNET SHIELD*/
  ActivateSPIChip(0x01);
  delay(10);
  if(Ethernet.begin(mac_addr)==0){
    Ethernet.begin(mac_addr, my_addr);
  }
  if(client.connect(server_addr,80)){
    Serial.println("Connection OK");
    client.print("GET
/Arduino/checkCardID.php?");
    client.print("value1=");
    client.print(CardID);
    client.println(" HTTP/1.1");
    client.println("Host: 192.168.1.100");
    client.println("Connection: close");
    client.println(); //Empty line
    client.println(); //Empty line
    delay(100);
    if(client.connected()){
      if(finder.find("CardPos ")){
        returnValue = finder.getValue();
      }
    }
    if(returnValue==ADMIN_CARD_ID){
      returnValue = 0xFF;
    }
    client.stop();
  }
  else{
    Serial.println("Connection failed.");
  }
  delay(10);
  ActivateSPIChip(0x02);
  return returnValue;
}
```

According to the answer returned by `search_IDinDB()` function, the application goes to `ADMIN_SUPPLIES` state (if the returned value is `255 – 0xFF`) or to `MONITOR_SUPPLIES` state (for any other valid value that is found into the database). Otherwise, the authentication process is repeated.

The application detects the changes of weight sensors. A change that is more than 3.5g on any of the two weight sensors is sent through the web server to the database and it is also stored into the EEPROM memory. Besides the weight, the function that performs this task will also send the ID of the sensor and the operation that has to be executed by the PHP script. There are three possible actions here: (i) the value of that sensor is the weight of a medication unit and it is used for initialization (in setup part), (ii) the value of that sensor is the total weight and the number of units can be calculated, knowing the unit weight (in `MONITOR_SUPPLIES` state or `ADMIN_SUPPLIES` state), and (iii) the unit weight associated to that sensor is reset (available in `ADMIN_SUPPLIES` state only). The difference between the two available actions in `ADMIN_SUPPLIES` state is made using the two buttons placed on the development board.

Then, the presence of a RFID tag is scanned. There are two possible operations, according to the current state of the system: in `ADMIN_SUPPLIES` state the medical equipment is added and in `MONITOR_SUPPLIES` state it is subtracted.

All this information (from weight sensors and from RFID tags reader) is sent to the database in order to be stored for further use. The table `arduino.med` is updated with data about medication and the table `arduino.actions` registers all the activities performed into the system.

The script that updates the quantity of medical equipment identified by RFID tags is hereby presented as example. First, the SQL command that searches the RFID tag into the `arduino.med` table is created. If the tag is found, the necessary information (medication name and quantity) is selected. Also, the system time is stored in order to be added into the `arduino.actions` table.

```
$sql = "SELECT currentQuantity,name FROM med
WHERE    med.sensorID    =".$_GET["sens"]."    AND
$tagUsed    >    med.minTagID    And    $tagUsed    <
med.maxTagID";
$retval = mysql_query($sql);
if( $retval == FALSE ){
      echo 0xFF;
}
```

```
while($row    =    mysql_fetch_array($retval,
MYSQL_NUM)){
       $currentQ = $row[0];
       $MedName = $row[1];
}
$time = date("Y-m-d H:i:s");
```

With this information, the value for medication quantity is calculated. Then, according to the operation that is received from Arduino platform (adding or subtracting), the SQL command that updates the `arduino.med` table is created.

```
$finalQ = $currentQ - 1;
$sql = "UPDATE med SET currentQuantity=$finalQ
WHERE  med.sensorID  =   ".$_GET["sens"]."  AND
$tagUsed   >   med.minTagID   And   $tagUsed   <
med.maxTagID";
$retval = mysql_query($sql);
if( $retval == FALSE ){
       echo "NO UPDATE done";
}
else   {
       echo "UPDATE done";
}
$action = 'Take';
```

Once all this data is available and the server successfully finished the previous command, the new SQL command that updates the `arduino.actions` table can be created.

```
$sql   =   "INSERT   into   actions   (   UserID,
UserBadgeID, MedName, MedConc, MedTaken, Type,
DateStamp   )   VALUES   ($UserID,   $UserBadge,
'$MedName', 0, 1,'$action', '$time')";
```

These type of SQL functions are used into the entire project for different actions. The database is manipulated by updating or interrogating it.

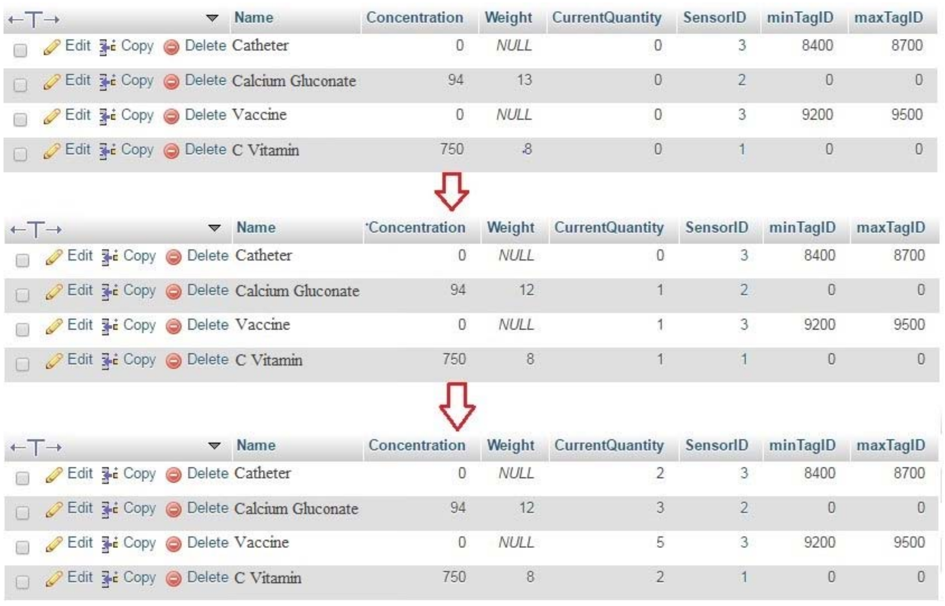
IV. Results

This prototype of an automated system for medication stock management is functional now. In order to be used, an initial configuration is necessary. It refers to the records of tables `med`, `sensor` and `user`. Then, the application can be used to manage the medication stock. Once the setup is completed, the red color LED shows that nobody is authenticated.

The first user should be an admin, in order to set the unit weight for the medication monitored by the two weight sensors. After a successful authentication, the yellow color LED informs that the system is in the `ADMIN_SUPPLIES` state. A unit of medication is placed on each load cell and the first button in pressed. The information is sent to the database, registering the unit weight of each medication. Then, the medication vials with standard weight can be added to the system, placing them to the load cells and pressing the second button. The total weight is measured, and, using the unit weight, the number of units can be calculated and registered into the database. The medical equipment with RFID tags can be added to the database bringing it in the magnetic field generated by the RFID reader. The evolution of `med` table during these actions performed in the `ADMIN_SUPPLIES` state is presented in the Fig. 4.

A regular user is then authenticated to the system. The green color LED indicates that the application is in the `MONITOR_SUPPLIES` state. Now, any change that is notified by the weight sensors is sent to the web server and the number of medication taken from or placed on that load cell is calculated and registered into the database. Also, if a medical equipment with RFID tag is placed near the reader, then a subtraction is executed from the total number of units of the identified object.

All the actions performed in the system are also displayed on the screen by a serial monitor. Fig. 5 emphasizes the correlation between the displayed messages (the top part of the figure) and the information registered into the database (the bottom part of the figure).

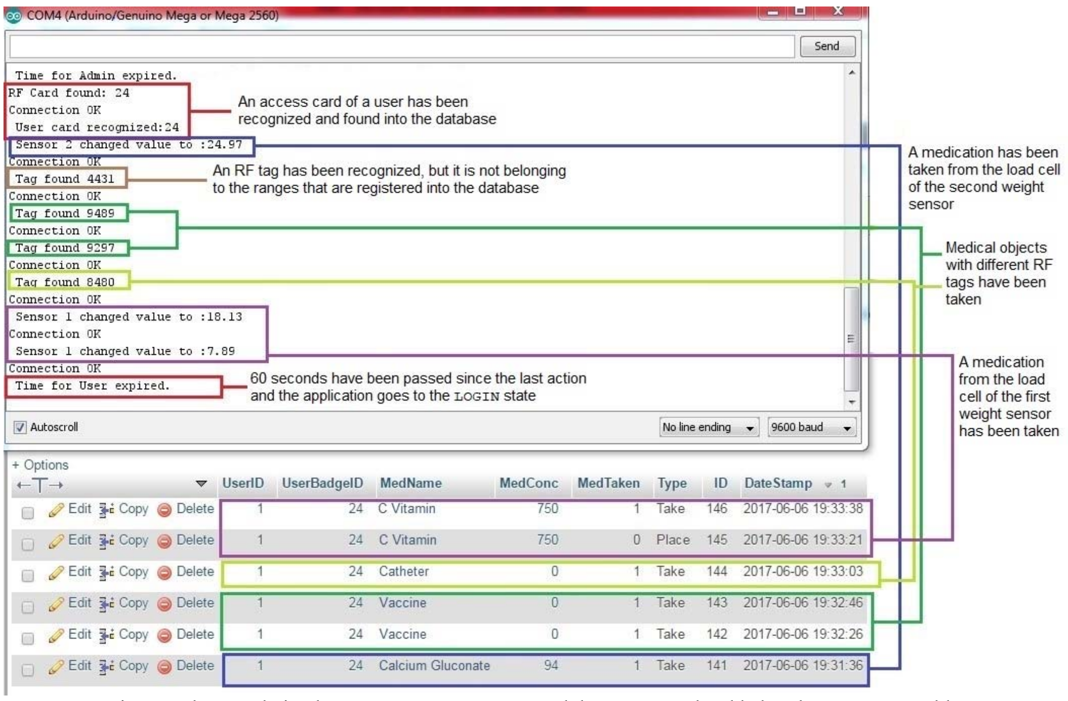


←T→	Name	Concentration	Weight	CurrentQuantity	SensorID	minTagID	maxTagID
Edit Copy Delete	Catheter	0	NULL	0	3	8400	8700
Edit Copy Delete	Calcium Gluconate	94	13	0	2	0	0
Edit Copy Delete	Vaccine	0	NULL	0	3	9200	9500
Edit Copy Delete	C Vitamin	750	8	0	1	0	0

←T→	Name	Concentration	Weight	CurrentQuantity	SensorID	minTagID	maxTagID
Edit Copy Delete	Catheter	0	NULL	0	3	8400	8700
Edit Copy Delete	Calcium Gluconate	94	12	1	2	0	0
Edit Copy Delete	Vaccine	0	NULL	1	3	9200	9500
Edit Copy Delete	C Vitamin	750	8	1	1	0	0

←T→	Name	Concentration	Weight	CurrentQuantity	SensorID	minTagID	maxTagID
Edit Copy Delete	Catheter	0	NULL	2	3	8400	8700
Edit Copy Delete	Calcium Gluconate	94	12	3	2	0	0
Edit Copy Delete	Vaccine	0	NULL	5	3	9200	9500
Edit Copy Delete	C Vitamin	750	8	2	1	0	0

Fig. 4. The evolution of table `med` during admin's actions.


COM4 (Arduino/Genuino Mega or Mega 2560)
Send
An access card of a user has been recognized and found into the database
An RF tag has been recognized, but it is not belonging to the ranges that are registered into the database
A medication has been taken from the load cell of the second weight sensor
Medical objects with different RF tags have been taken
A medication from the load cell of the first weight sensor has been taken
60 seconds have been passed since the last action and the application goes to the LOGIN state
Autoscroll
No line ending
9600 baud
+ Options
Edit
Copy
Delete

```
 Time for Admin expired.
RF Card found: 24
Connection OK
 User card recognized:24
 Sensor 2 changed value to :24.97
Connection OK
 Tag found 4431
Connection OK
 Tag found 9489
Connection OK
 Tag found 9297
Connection OK
 Tag found 8480
Connection OK
 Sensor 1 changed value to :18.13
Connection OK
 Sensor 1 changed value to :7.89
Connection OK
 Time for User expired.
```

UserID	UserBadgeID	MedName	MedConc	MedTaken	Type	ID	DateStamp
1	24	C Vitamin	750	1	Take	146	2017-06-06 19:33:38
1	24	C Vitamin	750	0	Place	145	2017-06-06 19:33:21
1	24	Catheter	0	1	Take	144	2017-06-06 19:33:03
1	24	Vaccine	0	1	Take	143	2017-06-06 19:32:46
1	24	Vaccine	0	1	Take	142	2017-06-06 19:32:26
1	24	Calcium Gluconate	94	1	Take	141	2017-06-06 19:31:36

Fig. 5. The correlation between screen messages and the new records added to the `actions` table.

V. Conclusions

The automated system for medication stock management presented in this paper brings significant benefits for medical staff and also for patients. It has intuitive functions and can be easily used by physicians and nurses. Using it, they will improve their professional life that, at this moment, is overloaded with activities which are time and energy consuming. One of these activities is the medication manipulation. At the end of a working shift the current stock must be verified and manually refilled. By the proposed system, these actions are automatically performed because it can detect and register the movement of standard weight medication and of medical objects with RFID tags.

The prototype described here proves that such a management system can actually be developed. There are some features of this system that could be improved in the next versions (from hardware and software point of view). For instance, at the time being, it has only two load cells, but other stands can be added. They also can have an expanded form, in order to support a larger number of medication and more accurate weight sensors can be used to react when small pills are managed. Other improvements refer to some tools (graphical interfaces) used by the admin when sensors or users should be added, modified or eliminated. Right now, these actions are performed directly on the database. The patients could be also involved into this system if, for each medication that is administrated to them, that information is sent directly to their electronic healthcare record, too. It would be also useful for such an application to be able to generate different reports.

The system described by this paper is functional (the data acquisition from peripheral equipment is accomplished, the communication between the development board and the storing equipment is working, and all the information is available for further use). As the present paper shows, this automated system for medication stock management has many advantages that make it a reliable tool for use in medical establishments. This project could be implemented on a wide-scale, its necessity being proved by the every day activities that are overloading the medical personnel with tasks that can be faster and safer performed by such a system.

References

[1] A. Mirea, A. Albu, "Acquisition of physical data in an automated system for monitoring medication stocks", The IEEE 12-th International Symposium on Applied Computational Intelligence and Informatics, pp. 179-182, Mai 2018.

[2] J. Labuhn, P. Almeter, C. McLaughlin, P. Fields and B. Turner, "Supply chain optimization at an academic medical center", American Journal of Health-System Pharmacy, vol. 74 (15), pp. 1184-1190, August 2017.

[3] L. Louden, J. M. Mirtallo, M. Worley, R. Naseman, A. Hafford and N. V. Brown, "Efficiency analysis of a barcode-enabled and integrated medication-tracking system", American Journal of Health-System Pharmacy, vol. 74 (23 Supplement 4), pp. S84-S89, December 2017.

[4] Steven F. Barret, "Arduino Microcontroller Processing for Everyone!", Third Edition, Morgan & Claypool Publishers, 2013, ISBN: 9781627052542.

[5] Load Cell Amplifier - HX711, SparkFun Electronics, available from: https://www.sparkfun.com/products/13879, accessed: 2018.

[6] Getting Started with the Arduino Ethernet Shield, available from: https://www.arduino.cc/en/Guide/ArduinoEthernetShield, last up-dated: 2017, accessed: 2018.

[7] D. Dvorski, "Installing, Configuring, and Developing with XAMPP", Skills Canada-Ontario, March 2007.