Bulletin of the *Transilvania* University of Braşov • Vol. 14 (63) No. 1 - 2021
Series I: Engineering Sciences
https://doi.org/10.31926/but.ens.2021.14.63.1.2

CONTROL AND MONITORING OF A SOCKET SYSTEM

M. BURTEA[1] **G. PANĂ**[2]

Abstract: *The purpose of this paper is to present the design and implementation of a system that combines the hardware with the software resulting in an intelligent system of sockets that can be controlled and monitored remotely. The Arduino Mega 2560 performs the basic function and provides the information gathered from the sensors (Hall effect current sensor, motion sensor, dual temperature and humidity sensor). General system information is displayed locally on an LCD screen. The system protects the connected equipment by disconnecting the power supply in case of hazard; but can also be controlled remotely via a web application by the user.*

Key words: *Internet of Things, Arduino, Mqtt.*

1. Introduction

Recent technological discoveries have changed the perception of the way of life, producing major changes in the way people spend their free time, communicate with each other, and simplifying their activities at work and beyond.

With the help of technology, human work has been made easier and the processes of data collection and processing are more and more independent of the human factor, these being stored and analyzed by a computer. Basically, we can say that the changes brought by technology have led to the formation of a new type of society, a digital society.

The module to be presented in this paper is represented by a system of intelligent sockets that can control and monitor the connected electronic equipment.

The socket is a device by means of which the electrical connection is made, by means of a plug, of a consumer to an electrical network. There are major differences between normal and smart outlets, and the biggest difference and advantage of the latter is that they can be controlled remotely with the help of a phone or computer.

The following chapters will show how a smart socket system was designed using the Arduino development board and how it can be controlled remotely from the graphics page created for users.

[1] Master student in Electronic and Communication Integrated Systems, Transilvania University of Braşov.
[2] Dept. of Electronics and Computers, *Transilvania* University of Braşov, Romania.

1.1. Current state of the market

Over time, various systems have been developed that are based on this concept of "smart home":

- Applications for the control and energy management of an Arduino microcontroller socket (flexible system with low costs [1]);
- Socket energy control and management system based on ZigBee [4];
- MorSocket can be controlled by smartphones via Wi-Fi or Bluetooth; the system allows the control of multiple sockets (the maximum number is 30 sockets for the current implementation [2]);
- Hardware system based on STM32F103 (STM32F103 devices use the Cortex-M3 core) for the development of an intelligent socket for power response management and related economic losses [3].

2. Implementation

2.1. Hardware Implementation

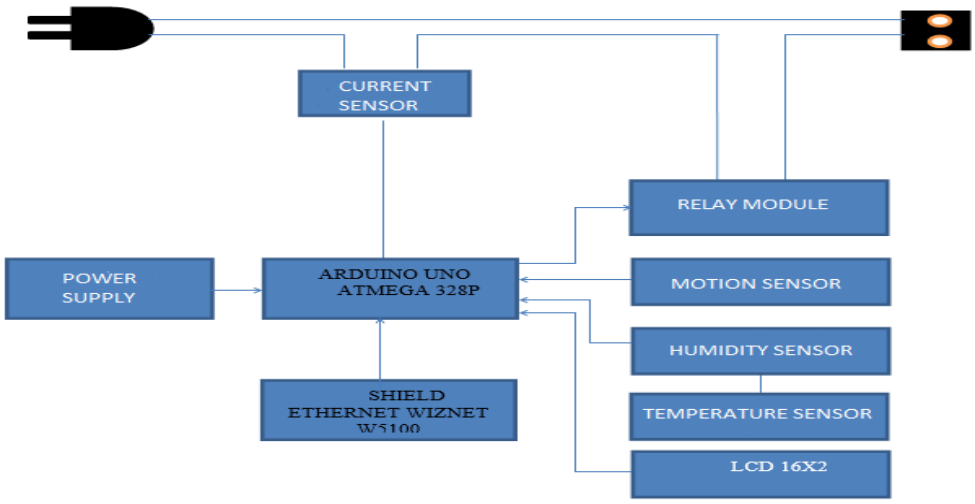


Fig. 1. *The hardware structure of the system*

The developed system (see Figure 1) focuses mainly on the Arduino development board that performs the basic functions and processes the information collected from the sensors and received from the user. The Arduino Mega 2560 was chosen because of the need for a microcontroller with more memory than the Arduino Uno to store the source code. The development board is based on the Atmega2560 microcontroller, has 54 digital pins (of which 15 can be used as PWM-Pulse Width Modulation) and 16 analog pins [5].

The DHT1 sensor is a digital temperature and humidity sensor that incorporates a capacitive humidity sensor and a thermistor to perform measurements on the environment and provide a digital signal on the data pin.

The ACS712 current sensor measures currents using the Hall principle. The Hall effect consists in the appearance of a potential difference on the sides of a sample when it is in magnetic field and is crossed by electric current. The output voltage of the Hall effect sensors, called the Hall (VH) voltage of the Hall base element is directly proportional to the intensity of the magnetic field passing through the semiconductor material.

The PIR motion sensor HC-SR501 is based on IR (InfraRed) technology, is reliable and has a low operating voltage.

4-channel relay module. With such relay module, different IoT (Internet of Things) projects can be realized for the automation of different activities.

The display screen used is 16x2 LCD being the most used in projects like this and has 16 columns and 2 rows, on which are displayed locally the data that characterizes the operation of the system.

The Wiznet W5100 chip Ethernet shield [6] allows an Arduino development board to connect to the Internet. Arduino communicates via the SPI bus (via the ICSP header) with the W5100 chip and the SD card. The responsible pins are 11, 12 and 13 on Uno and 50, 51 and 52 on Mega. On both boards, pin 10 is in charge of selecting the W5100 chip, and pin 4 is in charge of the SD card.

2.2. Software Implementation

The software for the control and monitoring system of the socket system is based on the Arduino programming language. The system was programmed to once display, from a 5V source, to display, locally on the LCD, the message "Smart socket" on the first line, and on the second the status of the sockets, which is initially 1, meaning all four sockets are powered.

After querying the sensors, the first line will display information about the temperature and humidity of the environment, the second still displaying the status of the sockets.

The system is connected to the Internet via the Ethernet shield, and with the help of an application development program based on MQTT (Message Queuing Telemetry Transport) [7], the system communicates with the web interface.

The sensors are interrogated and if for 120 seconds no movement is recorded around the system, the power supply to the sockets is automatically stopped by activating the relays, the same being done if the ACS712 current sensor detects a current that exceeds the set limit of 10A.

From the graphical interface you can send simple start/stop commands for each socket of the system, but you can also set a work schedule for socket number four. If the latter has a program set and is in the execution period, the PIR sensor has no effect on it, but only the current sensor. If socket four is in working time and the period set for the motion sensor has expired, the microcontroller will disconnect only the first three sockets, the fourth continuing its priority activity being the user's command.

During operation of the system, if it is powered by USB to the computer, the data transmitted in series can be viewed. Information regarding the connection of the system to the server, data connected from the temperature and humidity sensor and from the

motion sensor is transmitted. Information on orders received from the web interface, such as the socket number and the order for it, as well as data on operating times are also displayed (see Figure 2).

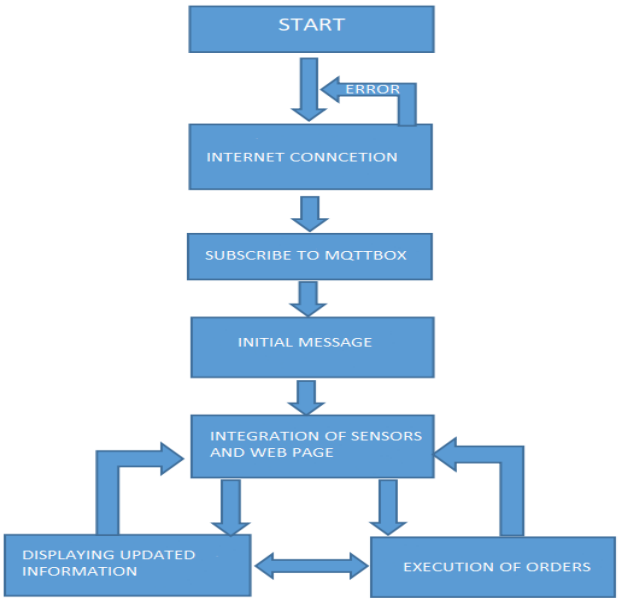


Fig. 2. *The logical scheme of the system*

2.3. MQTT

The communication between the web page available to users and the Arduino Uno and Mega development board was made using the MQTT (Message Queuing Telemetry Transport) protocol which transports messages via the Internet between a server and several clients, on the "publishing and subscribing" model. It is suitable for machine-to-machine messaging, ideal for mobile applications due to low power consumption, minimized data packets and efficient distribution of information to one or more receivers.

MQTT is a simple messaging protocol, a perfect solution for IoT (Internet of Things) applications that allows you to send commands to control outputs, read and publish data received from sensors and much more. It is very easy to establish a communication between several devices and that is why it is recommended for its use in a home automation project and more.

MQTT has some basic concepts:

- Publish / Subscribe
- Messages
- Topics
- Agent

As I said before, MQTT works so that one or more clients connect to a server and publish their messages on a certain topic, and other clients will subscribe to that topic and take over the information transmitted.

The customer who takes over the information within this project is represented by the development board, which subscribes to the subject "license" and decodes the data for the on / off order of the four outlets or the data on the operating period for outlet number four. The exchange of messages is done with the help of an MQTT Broler that can be viewed as a router in a local network.

The “web page” customer will publish the message that will contain the information used to order the four outlets. The data to be published are serialized using a JSON (JavaScript Object Notation), the socket number and the state it will have (on or off), or the operating program for the last socket (relay number, start date and time operation and the period for which he must be active).

```
function setAutoStart(relay_no, start_delay, run_time){
    var jsonMessage = {
      "t" : "sa",
      "r" : ""+ relay_no,
      "sd" : ""+ start_delay,
      "rt" : ""+ run_time
    };
    var strJsonMessage = JSON.stringify(jsonMessage);
    console.log("setAutoStart: "+ strJsonMessage);
        message = new Paho.MQTT.Message(strJsonMessage);
    message.destinationName = "licenta_ard";
    client.send(message);
  }
function dateChanged(relay, start_picker, end_picker){
   var start_date_ms start_picker.data("DateTimePicker").date();
var end_date_ms end_picker.data("DateTimePicker").date();
    console.log("start = " + start_date_ms);
    console.log("end = " + end_date_ms);
      if( start_date_ms == null || end_date_ms == null){
      return;
    }
    if(start_date_ms >= end_date_ms){
      return;
    }
      var timestamp_now = new Date().getTime();
      if(start_date_ms < timestamp_now || end_date_ms < timestamp_now){
      return;
    }
      start_date_ms -= timestamp_now;
    end_date_ms -= timestamp_now;
    var run_time = end_date_ms - start_date_ms;
    setAutoStart(relay, start_date_ms, run_time);
  }
```

The “Arduino” customer subscribes to the same topic on which the “web page” customer published, takes over the serialized data, decodes them and makes decisions based on the orders received.

```
// mqtt defs
#define SEPARATOR "/"
#define MACHINE_ID (char*)"licenta_ard"
#define MQTT_SERVER (char*)"broker.hivemq.com"
#define MQTT_USERNAME NULL //(char*)""
#define MQTT_PASSWORD NULL //(char*)""

// connect the mqtt client to broker
void reconnectMqtt() {
 if(!mqttClient.connected()) {  // if is not connected
  Serial.println("Attempting to connect MQTT...");

  if (mqttClient.connect(MACHINE_ID, MQTT_USERNAME, MQTT_PASSWORD)) { // if
int state = json["st"].as<int>(); // what state to write for the relay
    int hi_low = state == 1? RELAY_OFF : RELAY_ON;
    // conversion to the default type //0?
  if(hi_low == RELAY_ON){
   String setTime(String(SET_TIME_PREFIX) + relay_number);
   bufferr.remove(setTime);
  }

  digitalWrite(FIRST_RELAY_PIN + relay_number, hi_low);
  Serial.println(String(relay_number) +"="+ hi_low);
  lastPirActivity = millis();  // timer reset to avoid
```

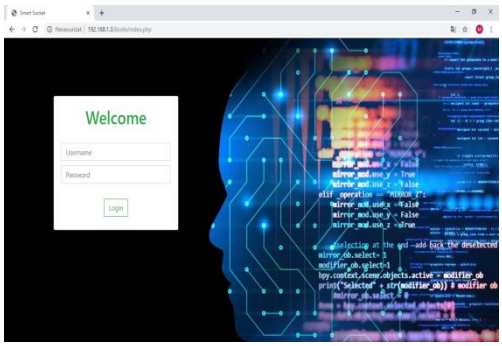
```
      the connection was successful
    mqttClient.subscribe(MACHINE_ID);  // subscribe
      to messages for your current device
    Serial.println("conectat");
   }
   else {
    Serial.println(String("Eroare,          rc=")          +
      mqttClient.state() + ", reincearca in 5 sec");
    delay(5000);
   }
  }
}
void OnIncoming(const JsonObject& json){
 if(!json.containsKey("t")){
  return;
 }
 String type = json["t"];

 if(type == String("sr")){  // if json(the json message)
      is of the expected type
  int  relay_number  =  json["r"].as<int>()  -  1;  //
      obtain the relay for which the action will be
      taken
      sudden start and stop if the timer has
      previously expired
 }
 else if(type == String("sa")){
  int relay_number = json["r"].as<int>() - 1;  //
      obtain the relay for which the action will be
      taken
  unsigned long start_delay =
      json["sd"].as<unsigned long>();
  unsigned long run_time = json["rt"].as<unsigned
      long>();
  unsigned long set_time_ms = millis();
  bufferr[String(SET_TIME_PREFIX) + relay_number]
= String(set_time_ms);
  bufferr[String(START_TIME_PREFIX)          +
      relay_number] = String(start_delay);
  bufferr[String(RUN_TIME_PREFIX)          +
      relay_number] = String(run_time);

  Serial.println(String("autostart in ") + start_delay +
      " for " + run_time);
 }
}
```

2.4. Web Interface

The point of interaction between the user and the socket monitoring and control system is represented by the graphical interface, which consists of an authentication page, a page for administrators and an order page (see Figure 3).



(a) 0th Web page for authentication

```
<?php
session_start ();
$conn=mysqli_connect("localhost","root","","socket") or die("connection failed:".mysqli_error());

if(isset($_POST['sub']))
{
        $a = $_POST['Name'];
        $b = $_POST['Password'];
        $_SESSION['Name']  = $a;
        $admin_class = 1;
        $user_class = 0;
        $sql = "select* from login where Name='$a'and Password='$b'";
        $sql1 = "select* from login where Name='$a'and Password='$b'";
        $result = mysqli_query($conn, $sql);
        $result1 = mysqli_query($conn, $sql1);
        $result=mysqli_fetch_array($result);
        if($result)
        {

                while($row = mysqli_fetch_array($result1))
                {
                        $type = $row['Type'];
                        if($type == $admin_class)
                        {
                                $_SESSION['auth_type']  = 1;
                                $_SESSION["login_1"]="1";
                                header("Location: /licsite/admin_page.php");

                        }
                        else if($type == $user_class)
                        {
                                $_SESSION['auth_type']  = 0;
                                $_SESSION["login_1"]="1";
                                header("Location:/licsite/user_page.php");
                        }
                }
        }
        else
        {
                header("Location:/licsite/index.php?err=1");
        }

}
?>
```

(b) 1st Log-in menu code

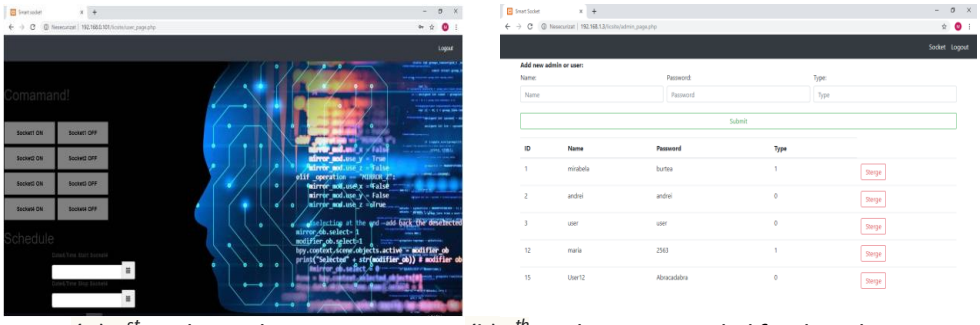
Fig. 3. *Main web page*

The main page in Figure 3 has a login menu that allows the authentication of two types of users: administrators and simple users, stored in a database made with MySQL (Structured Query Language). Administrators have additional access to a table with the data of all persons authorized to control the socket system, being able to add other users or delete from the existing list, but also have access to system control.

```
<?php
        if(isset($_POST['deleteItem']) and is_numeric($_POST['deleteItem']))
          {
            $delete = $_POST['deleteItem'];
            $sql_read = "DELETE FROM login where `ID` = '$delete'";
             echo("<meta http-equiv='refresh' content='1'>");
          }
        $result1 = mysqli_query($conn, $sql_read);
        while($row = mysqli_fetch_array($result))
        {
           $id = $row['ID'];
           $name = $row['Name'];
           $pass = $row['Password'];
           $type = $row['Type'];
           echo "<tr><td scope='row'>".$id."</td><td scope='row'>". $name . "</td><td scope='row'>". $pass
. "</td><td scope='row'> ". $type ."</td><td scope='row'> ";
           echo '<td><button type="submit" class="btn btn-outline-danger" name="deleteItem"
value="'.$row['ID'].'" />Sterge</td></tr>';
        }
        echo "</table>";
        ?>
```

The page where the four sockets of the system can be controlled has eight buttons, two for each socket, which represent the on / off commands (see Figure 4).

Socket number four has the possibility to set an operating program, by specifying the date and time for starting and ending the operating program.



(a) 1st Order web page *(b) 0th Web page intended for the administrator only*

Fig. 4. *The web interface of the system*

3. Conclusions

In this paper we proposed and implemented a project that allows the control and monitoring of a socket system. We realized this project by using low-cost devices and developing an easy-to-use web interface. The system can stop the supply of connected equipment when the data collected from the sensors exceeds the limits set in the source code, but can also be controlled by the user according to his needs.

References

1. Chandramohan, J., Nagarajan, R., Satheeshkumar, K., Ajithkumar, N., Gopinath, P.A., Ranjithkumar, S.: *Intelligent smart home automation and security system using Arduino and Wi-fi.* In: *International Journal of Engineering And Computer Science (IJECS)*, 2017, Vol. *6*(3), p. 20694-20698.
2. Lin, Y.B., Huang, C.M., Chen, L.K., Sung, G.N., Yang, C.C.: *Morsocket: An expandable iot-based smart socket system.* In: *IEEE Access*, 2018, Vol. *6*, p. 53123-53132.
3. Ma, M., Huang, B., Wang, B., Chen, J., Liao, L.: *Development of an energy-efficient smart socket based on STM32F103.* In: *Applied Sciences*, 2018, Vol. *8*(11), p. 2276.
4. Singaravelan, A., Kowsalya, M.: *Design and implementation of standby power saving smart socket with wireless sensor network*. In: *Procedia Computer Science*, 2016, Vol. *92*, p. 305-310.
5. https://www.robotshop.com/media/files/pdf/arduinomega2560datasheet.pdf. Accessed: 20.03.2021
6. https://www.mouser.com/catalog/specsheets/a000056_datasheet.pdf. Accessed: 20.03.2021
7. https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/A. Accessed: 20.03.2021.