Laboratory Automation System Using IOT Devices

D.P. Minchev and A.I. Dimitrov
Burgas Free University/ Faculty of Computer Science and Engineering, Burgas, Bulgaria
mitko@bfu.bg, atatas@bfu.bg

***Abstract* - This paper describes a Laboratory Automation System using IoT devices. The main hardware modules consist of: ESP8266 micro-controller by Espressif and energy meter HLW8012 by HLW Technology. The communication between the IoT devices uses IEEE 802.11g WiFi network, TCP/IP stack and MQTT protocol. The minimal configuration of the LAS includes one control device called Gateway and one manageable device called Host. The devices can be controlled remotely by user-friendly, simplified web-based interface. The project is published as open source software and hardware.**

Keywords - IoT, ESP8266, Embeded system, Smart device

I. INTRODUCTION

Internet of Things (IoT) is turning out to be increasingly more and more a piece of our lives regularly. IoT gadgets are basically used to control, screen, and deal with the innovation that we utilize each day. This implies that the gadgets are normally intended to be effortlessly introduced and overseen by the costumer. Industry examiners estimate the quantity of associated gadgets to be approx. 50 billion before the year 2021. This paper covers the implementation of Laboratory Automation System (LAS) using IoT Devices with energy measurement capabilities.

II. ARCHITECTURE

The proposed framework design model is secluded and comprises of numerous end gadgets, called *Hosts*. The *Hosts* are equipped for joining an assortment of lab electrical apparatuses. Every one of these end-gadgets has a remote system interface which is associated with the single control gadget and is called *Gateway*. The *Gateway* controls the *Hosts*. It builds up a correspondence network channel between the costumer and *Hosts*, in this way giving their remote control and the executive commands.

The Message Queue Telemetry Transport (MQTT) [1] correspondence convention is utilized to transmit information among *Hosts* and *Gateway* gadgets. The decision of this correspondence convention is because of its focal points. It is lightweight, open to people in general, as portrayed in the OASIS [2] in particular, normalized by ISO/IEC PRF 20922 [3] and it works over the Transmission Control Protocol/ Internet Protocol (TCP/IP).

When controlled, a solitary *Host* acquaints itself with the *Gateway* with ceaseless "Hello" message. This distinguishing proof message contains the name of the *Host*. In the wake of getting the message, the *Gateway* reacts back to the "Hello" message with another "Stop" message. Along these lines, the *Gateway* affirms the *Host* that it is effectively associated with the system, reasoned by *Gateway* and adds its IP address to his directing table.

At the point when the *Gateway* gets its client order to turn on a particular gadget, it sends "Control" message to the gadget IP address from the directing table to control the *Host*. Upon receiving the "Control" message, the *Host* occasionally reacts with "Power" message until *Gateway* sends "Stop" message. The "Power" message comprises of data for power utilization, estimated by HLW8012, and contains three fields: voltage of the force network, expended current and dynamic force from the overseen gadget.

At the point when the *Gateway* gets the client order to kill a particular gadget, it sends a "Control" message to the gadget IP address from the directing table to turn off the *Host*.

The UML succession graph of the MQTT correspondence convention among *Gateway* and *Host* gadget is on Fig. 1.

III. SOFTWARE

The project firmware is written in C/C++ using Arduino IDE [4]. Additional external libraries were used during the development:

- **ESPAsyncTCP** [5] - Asynchronous TCP library, aimed at enabling trouble-free, multi-connection network environment for Espressif's ESP8266 MCUs.
- **ESPAsyncWebServer** [6] - Arduino compatible Asynchronous HTTP and WebSocket Server for ESP8266.
- **PubSubClient** [7] - This library provides a client with doing simple publish/subscribe messaging with a server that supports MQTT.
- **uMQTTBroker** [8] - Arduino compatible MQTT Broker library for Espressif's ESP8266 MCUs.
- **ArduinoJson** [9] - JSON message protocol library implementation in C++ for Arduino and IoT.
- **HLW8012** [10] – Arduino energy measurement library by Xose Perez.

978-1-7281-4346-0/20/$31.00 ©2020 IEEE

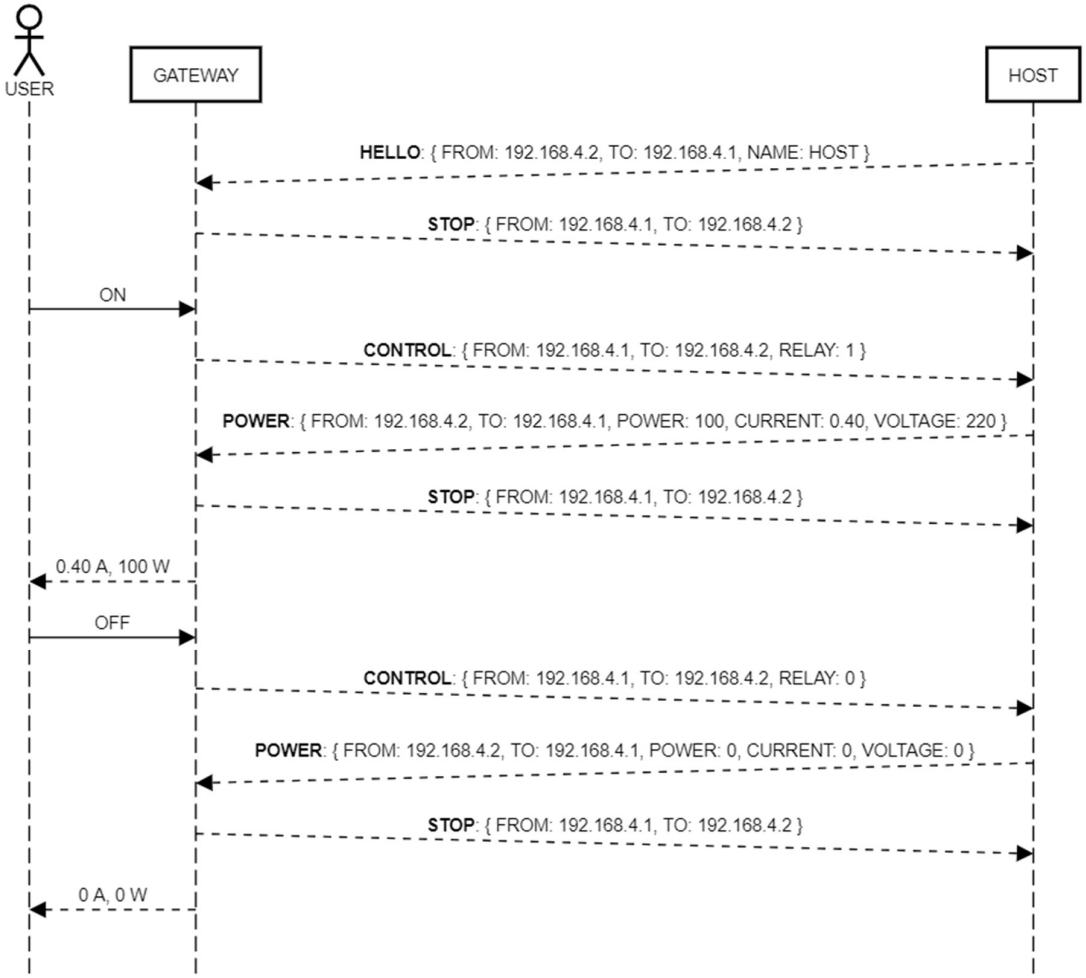
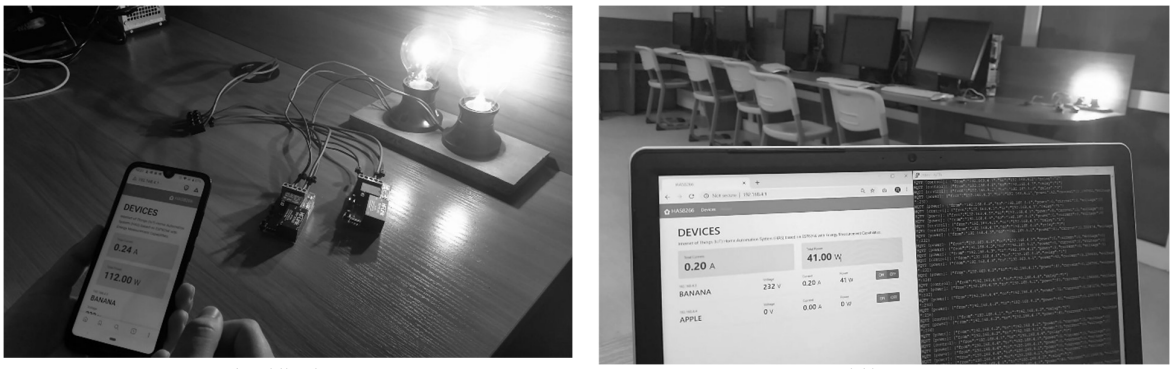


Fig.1. UML Sequence Diagram of the MQTT Communication protocol

The project firmware is open software and can be downloaded from the GitHub Repository: https://github.com/dimitarminchev/HAS8266/tree/master/2020

Testing the project firmware in laboratory-controlled environment on mobile device and laptop is on Fig.2.



a/ mobile phone b/ laptop

Fig.2. Testing the project software in laboratory-controlled environment.

IV. HARDWARE

As it was referenced in the past segment, for correspondence over neighborhood Wi-Fi, the task utilizes Espressif smaller scale controller - ESP8266 [11], coordinated in ESP-12 Wi-Fi module created by the computer-based intelligence Scholar group. The estimation of the devoured vitality of the overseen gadgets is cultivated with a solitary stage vitality screen chip HLW8012 [12] (Fig. 3). The ESP8266 chip coordinates a ultra-low force 32-piece MCU engineering. The center of the processor can work with clock speed of 80MHz or 160MHz. The module bolsters RTOS, IEEE802.11 b/g/n standard and complete TCP/IP stack, which makes it perfect for adding to a current system gadget or building a different system controller.

The chip can be utilized as an Access Point (AP) to broaden a current system, or as an end-client Host gadget. Having a Wi-Fi connector, ESP8266 can be added to any miniaturized scale controller with incorporated SPI/SDIO or I2C/UART interfaces.

HLW8012 can quantify RMS estimations of current, voltage, and dynamic force with an inside clock and a PWM interface. In order to work a HLW8012 chip needs 5V DC voltage. It creates half obligation square waves with recurrence that relies upon the extent of the deliberate boundary (power, current, or voltage).

The estimation of the devoured current is finished by a differential likely measure over a milliohm copper-manganese or comparable resistor, associated in arrangement with a heap (see Fig. 3) The differential potential is taken out of pins V_{1P} and V_{1N}, where the capability of the V_{1P} should consistently be greater than V_{1N}. The estimation of the shunt resistor must be chosen as for the greatest pinnacles that the current will make on these pins (43.75mV). A one-milliohm resistor is appropriate to quantify flows up to 30A with a difference of under 1W. The voltage is estimated on V_{2P} pin, which supports tops up to ±700mV. This forces to downsize the capability of the deliberate voltage with a straightforward voltage divider. The HLW8012 datasheet suggests utilizing a voltage divider of five 470kΩ resistors in the upper and a 1kω resistor in the lower lines. In this setup, the scale factor is around 2821, which will change over a 230V RMS into 82mV. For our situation, to limit the number of the parts on the printed circuit board (PCB) plate, in the upper line of the divider five resistors in arrangement are associated. The changed scale factor is 2351 and it will create voltage of about 98mV on V_{2P} that falls path underneath the breaking point.

The HLW8012 has two yield pins, appointed as CF and CF1, where PWM frequencies with half obligation square waves are produced. CF pin is for the deliberate dynamic force and CF1 is for the current (0) or voltage (1) relies upon the rationale condition of pin SEL (see Fig. 3). As indicated by the datasheet of the chip, the recurrence produced on the CF pin for the deliberate force is equivalent to (1). For the current and the voltage these frequencies are identified with condition (2) and (3) respectively.

$$f_P = \frac{V_{1P}.V_{2P}.48}{V_{ref}^2} \cdot \frac{f_{osc}}{128} \quad (1)$$

$$f_I = \frac{V_{1P}.24}{V_{ref}} \cdot \frac{f_{osc}}{512} \quad (2)$$

$$f_V = \frac{V_{2P}.2}{V_{ref}} \cdot \frac{f_{osc}}{512} \quad (3)$$

where, f_P, f_I and f_V are the frequencies of the produced square waves for the deliberate dynamic force, current and voltage, V_{1P} and V_{2P} are the deliberate possibilities individually on pins V_{1P} and V_{2P} on the chip, V_{ref}=2.43V is the interior reference voltage and f_{osc}= 3.579MHz is the recurrence of the expand on the chip clock generator.

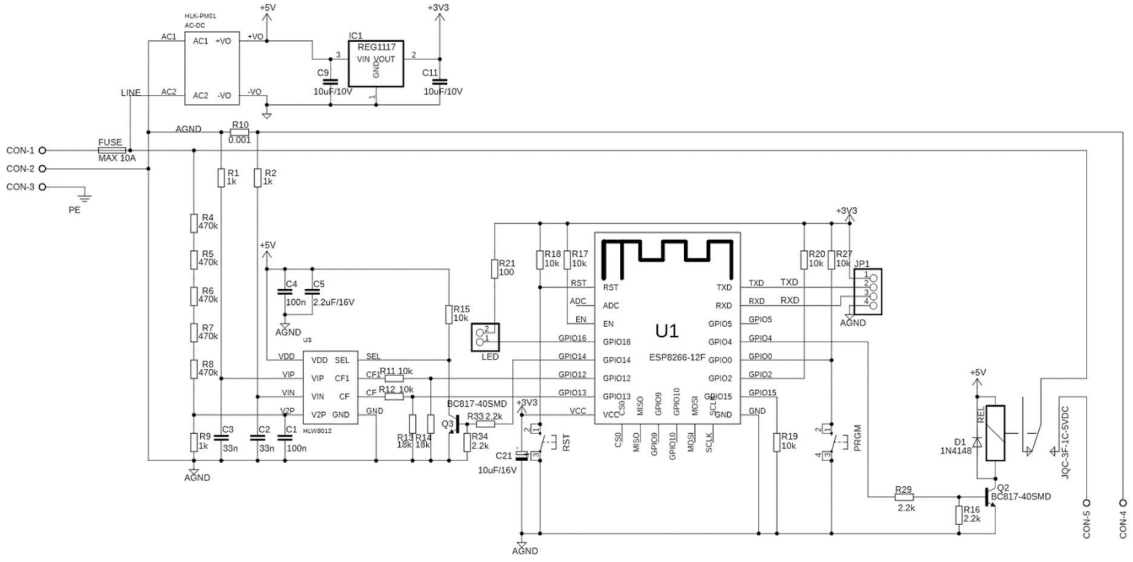
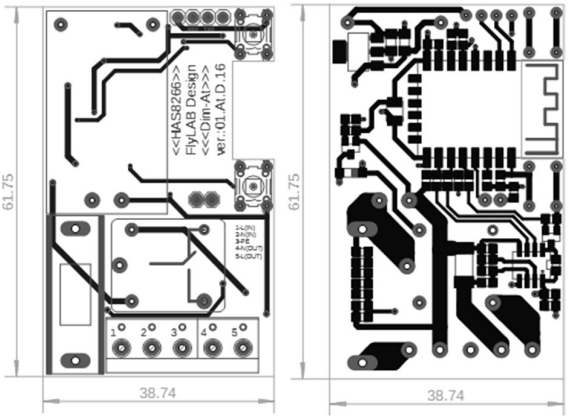


Fig.3. Electrical scheme of the proposed Laboratory Automation System.

Printed circuit board (PCB) of the project is shown on Fig.4 (*Fig.4a-component side/top layer) and (Fig.4 b-solder side / bottom layer).*

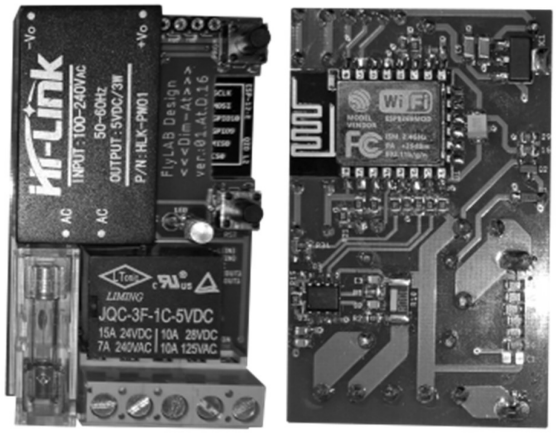


a) component side / top layer b) solder side / bottom layer

Fig.4. Printed circuit board of proposed Laboratory Automation System.

As it can be seen on Fig.4, the elements of the built item are exceptionally little so it can be effectively implanted in any home machine which is in enthusiasm to be controlled. Right now, the device can just turn ON or switch OFF gadgets through a relay on the PCB board. It is intended to broaden its fundamental usefulness later.

On Fig.5 a photo of a single device is shown *(Fig.5a-component side / top layer) and (Fig.5b-solder side / bottom layer).*



a) component side / top layer b) solder side / bottom layer

Fig.5. Single device

V. Conclusion

This paper describes a Laboratory Automation System using IoT devices. The main hardware modules consist of ESP8266 micro-controller by Espressif and an energy meter HLW8012 by HLW Technology. The communication between the IoT devices uses IEEE 802.11g WiFi network, TCP/IP stack and MQTT protocol. The minimal configuration of the LAS includes one control device called Gateway and one manageable device called Host. The devices can be controlled remotely by user-friendly, simplified web-based interface. The project is published as open source software and hardware.

References

[1] Message Queue Telemetry Transport (MQTT), http://www.mqtt.org/

[2] OASIS Standard Incorporating Approved Errata 01, MQTT Version 3.1.1 Plus Errata 01, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

[3] ISO/IEC 20922:2016, Information technology, Message Queuing Telemetry Transport (MQTT) v3.1.1, https://www.iso.org/standard/69466.html

[4] Arduino IDE, https://www.arduino.cc/en/Main/Software

[5] ESPAsyncTCP, https://github.com/me-no-dev/ESPAsyncTCP

[6] ESPAsyncWebServer, https://github.com/me-no-dev/ESPAsyncWebServer

[7] PubSubClient, https://github.com/knolleary/pubsubclient

[8] uMQTTBroker, https://github.com/martin-ger/uMQTTBroker

[9] ArduinoJson, https://github.com/bblanchon/ArduinoJson

[10] X. Perez, HLW8012, https://bitbucket.org/xoseperez/hlw8012

[11] ESP8266EX, Espressif System, https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[12] HLW8012, Hiliwi Technologies, www.hiliwi.com/products_detail/&productId=36.html