HOME AUTOMATION SYSTEM BASED ON ESP8266

Dimitar Minchev
Faculty of Computer Science and Engineering
Burgas Free University
62 San Stefano Str., Burgas, Bulgaria
e-mail mitko@bfu.bg

Atanas Dimitrov
Faculty of Computer Science and Engineering
Burgas Free University
62 San Stefano Str., Burgas, Bulgaria
e-mail atatas@bfu.bg

***Abstract*—This paper describes the implementation of an Internet of Things Home Automation System with energy measurement capabilities. The basic hardware components are the ESP8266 micro-controller by Espressif and energy meter chip HLW8012 by HLW Technology. The Software works over the TCP/IP stack and uses the MQTT protocol to communicate between the devices. The minimum configuration of the HAS includes one control device called Gateway and two or more manageable Host devices. The user friendly, simplified interface is web based. The platform has open software and hardware.**

Keywords— IoT; HAS; ESP8266; HLW8012; MQTT

I. INTRODUCTION

Internet of Things (IoT) is becoming more and more a part of our lives on a daily basis. IoT devices are primarily used to control, monitor, and manage the technology that we use every day. This means that the devices are typically designed to be easily installed and managed by the consumer. Industry analysts estimate the number of connected devices to be 50 billion by 2020. This paper describes the implementation of Internet of Things Home Automation System with energy measurement capabilities.

II. ARCHITECTURE

A simplified architecture of the proposed HAS is shown in Fig. 1. The proposed system architecture model is modular and consists of multiple end devices, called Hosts, capable of incorporating a variety of household electrical appliances. Each of these end devices has a wireless network interface that allows the device to connect to the wireless home network using one of the popular 802.11 b/g/n wireless networking standards. A single control device called Gateway controls the operation of all Hosts. This control device provides a command connection to the terminal devices, thus providing their remote control.

The Message Queue Telemetry Transport (MQTT) [1] communication protocol is used to transmit data between Hosts and Gateway devices. The choice of this communication protocol is due to its advantages.

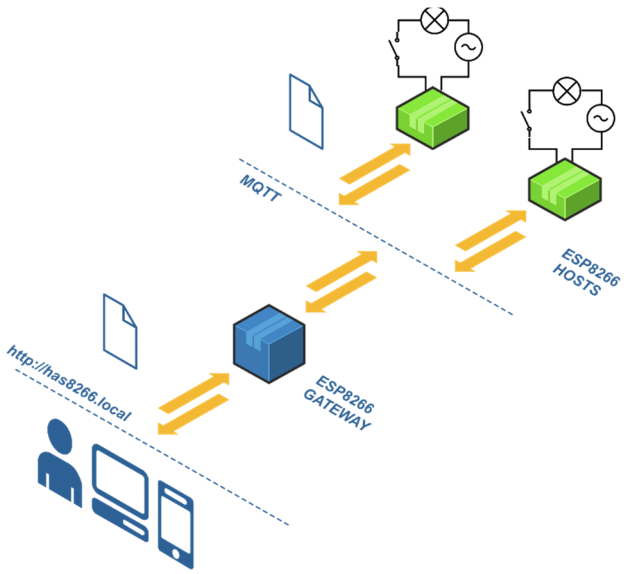


Fig. 1. A simplified architecture of the proposed HAS.

It is lightweight, open to the public, as described in the OASIS [2] specification, standardized by ISO / IEC PRF 20922 [3] and it working over the Transmission Control Protocol / Internet Protocol (TCP/IP).

III. HARDWARE

As it was mentioned in the previous section, for communication over local Wi-Fi network, HAS uses Espressif micro-controller - ESP8266 [4], integrated in ESP-12 Wi-Fi module developed by the Ai-Thinker team. The measurement of the consumed energy of the managed devices is accomplished with a single-phase energy monitor chip HLW8012 [5] (Fig. 2). The ESP8266 chip integrates an ultra-low power 32-bit MCU architecture. The core of the processor can work with clock speed of 80MHz or 160MHz. The module supports RTOS, IEEE802.11 b/g/n standard and complete TCP/IP stack, which is making it ideal for adding to an existing network device or building a separate network controller.

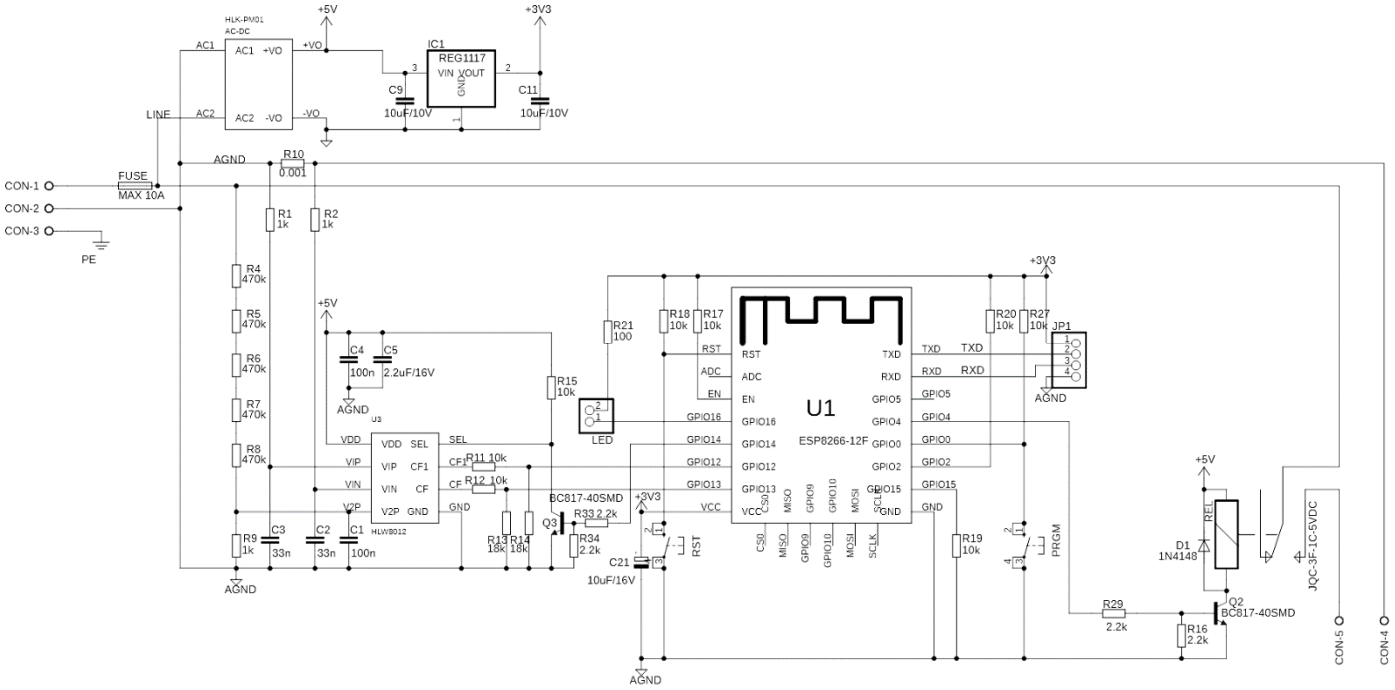


Fig. 2. Electrical scheme of the proposed Home Automation System.

The chip can be used as an Access Point (AP) to extend an existing network, or as an end-user Host device. Serving as a Wi-Fi adapter, ESP8266 can be added to any micro controller with integrated SPI/SDIO or I2C/UART interfaces.

HLW8012 can measure RMS values of current, voltage, and active power with an internal clock and a PWM interface. To operate, HLW80112 chip needs 5V DC voltage. It generates 50% duty square waves with frequency that depends on the magnitude of the measured parameter (power, current, or voltage).

The measurement of the consumed current is done by a differential potential measure across a milliohm copper-manganese or similar resistor, connected in series with a load (see Fig. 2) The differential potential is fed into pins V_{1P} and V_{1N}, where the potential of the V_{1P} must always be more positive than V_{1N}. The value of the shunt resistor must be selected with respect to the maximum peaks that the current will create on these pins (43.75mV). A one-milliohm resistor is well suited to measure currents up to 30A with a dissipation of less than 1W.

Voltage is measured on V2P pin, which supports peaks up to ±700mV. This imposes to scale down the potential of the measured voltage with a simple voltage divider. The HLW8012 datasheet recommends using a voltage divider of five 470kΩ resistors in the upper and a 1kΩ resistor in the lower lines. In this configuration, the scale factor is about 2821, which will convert a 230V RMS into 82mV. In our case, to minimize number of the components on the printed circuit board (pcb) plate, in the upper line of the divider are connected five resistors in series. The changed scale factor is 2351 and it will produce voltage of about 98mV on V_{2P} that falls way below the limit.

The HLW8012 has two output pins, assigned as CF and CF1, where PWM frequencies with 50% duty square waves is generated. CF pin is for the measured active power and CF1 is for the current (0) or voltage (1) depends on the logic state of pin SEL (see Fig. 2). According to the datasheet of the chip, the frequency generated on the CF pin for the measured power is equal to (1). For the current and the voltage these frequencies are related to equation (2) and (3) accordingly.

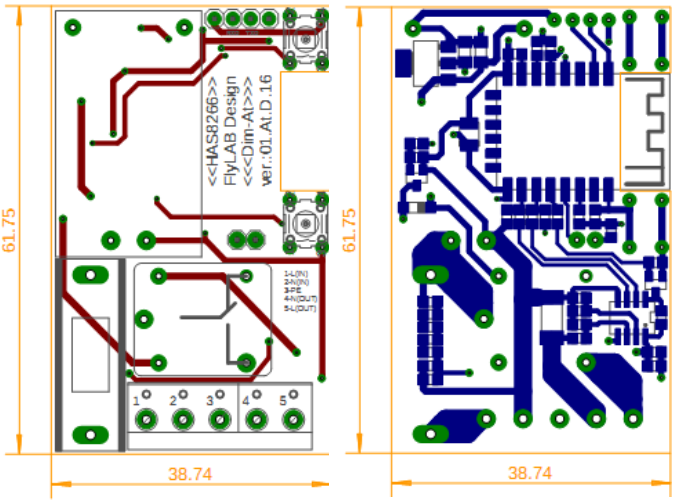
$$f_P = \frac{V_{1P}.V_{2P}.48}{V_{ref}^2} \cdot \frac{f_{osc}}{128} \tag{1}$$

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

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

where, f_P, f_I and f_V are the frequencies of the generated square waves for the measured active power, current and voltage, V_{1P} and V_{2P} are the measured potentials respectively on pins V_{1P} and V_{2P} on the chip, V_{ref}=2.43V is the internal reference voltage and f_{osc}= 3.579MHz is the frequency of the build on the chip clock generator.

The graphical originals of the created pcb of HAS is shown on Fig.3 (Fig3.a: top layer and Fig3.b: bottom layer).

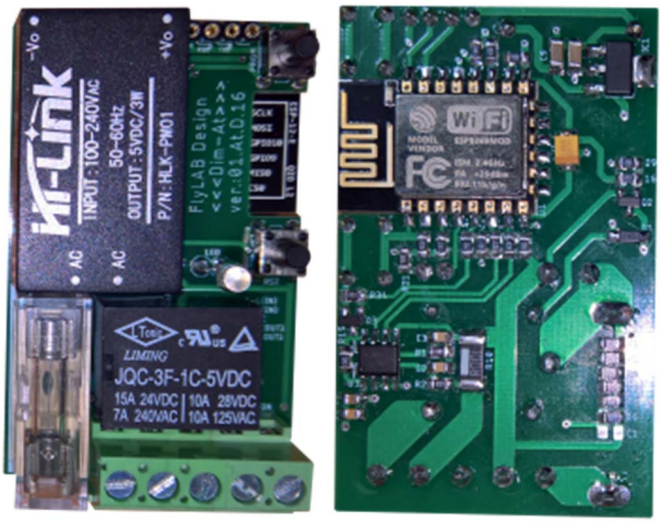


a) component side (top layer) b) solder side (bottom layer)

Fig. 3. Graphical originals of proposed Home Automation System.

As it can be seen on Fig3, the dimensions of the constructed HAS are very small so it can be easily embedded in any home appliance which is in interest to be controlled. At the moment, HAS can only switch ON or switch OFF managed devices via an embed relay in the pcb board. It is planned to extend this basic functionality in the future.

The realized device is shown on Fig.4.



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

Fig. 4. Photos of the created HAS device.

IV. SOFTWARE

The source code is written in C/C++ using Integrated Development Environment (IDE) Microsoft Visual Studio Code [6] and PlatformIO [7]. During the development of the firmware, depending of the device type (Gateway or Host), additional external libraries were used as follows:

- Implementation of Asyncronious TCP/IP MQTT broker inspired by original Arduino MQTT broker [8] was developed for the Gateway.
- ESP Async WebServer [9] is used for both Gateway and Host devices to add support of asynchronous TCP/IP Web Server.
- JSON protocol to send and receive command for both Gateway and Host devices is processed using ArduinoJson [10].
- MQTT client functionality for the Host device requires PubSubClient [11].
- Energy measurement for the Host device is ensured from the library HLW8012 [12] by Xose Perez.
- Connected Hosts device list on GateWay is stored in QList [13] by Martin Fagarin.
- Pinging Host devices from Gateway is executed by ESP8266Ping [14] by Daniele Colanardi.
- Timer functionality for both Gateway and Host devices is provided from Ticker [15] by Stefan Staub.

The view of the control web interface is shown on Fig. 5.

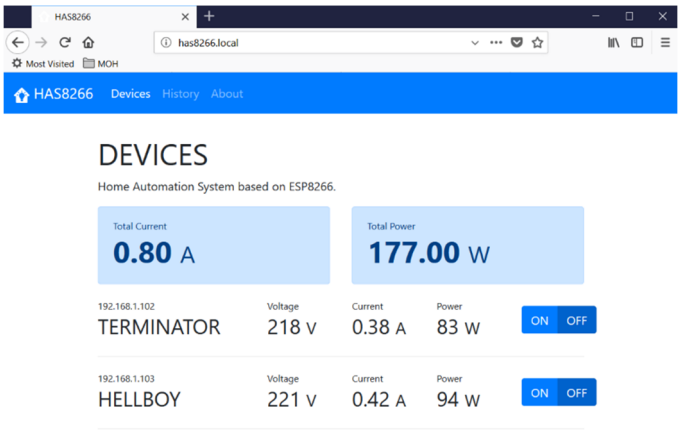


Fig. 5. The view of the control web interface.

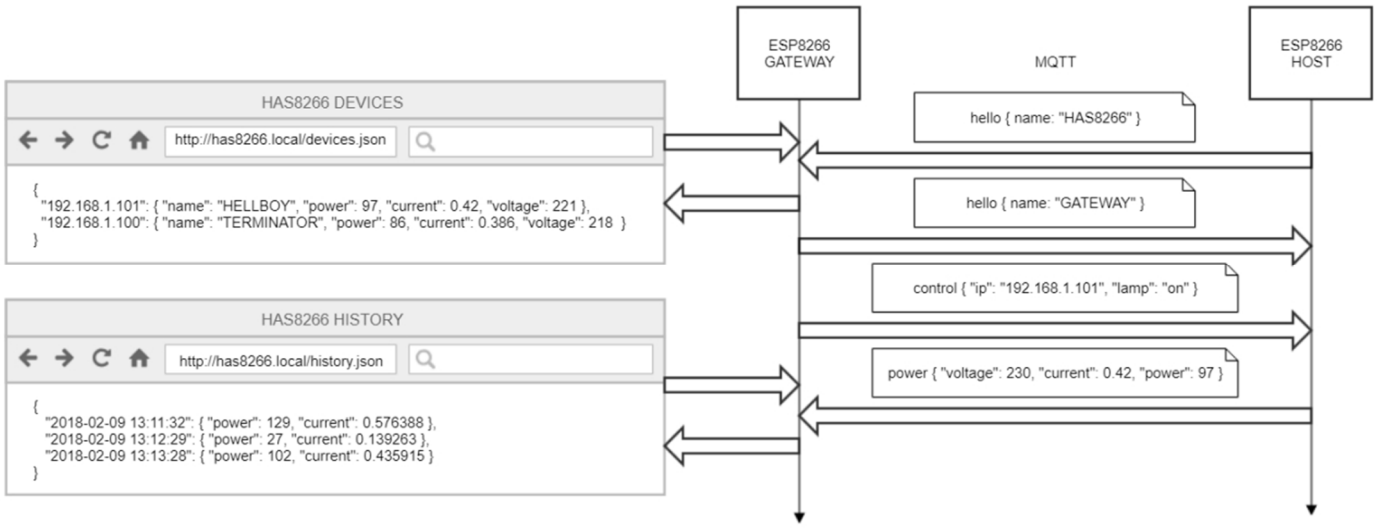


Fig. 6. HAS8266 Communication protocol

Communication between the Hosts and Gateway devices is realized as MQTT messages.

When is powered, a single Host introduces itself to the Gateway with a “Hello” message containing its name. The Gateway response back to the “Hello” message with its name. This “echo” from the Gateway not only confirms the newly connected host, but also adds Internet protocol address to the routing table. The connection is established after triple “Hello”messages.

When the Gateway receive user command to turn-on a specific device, it send “Control” message to the device IP address from the routing table to power desired Host. Receiving “Control” message, the Host periodically respond with “Power” message. It consists of information for power consumption, measured by HLW8012, and contains three fields: voltage of the power grid, consumed current and active power from managed device.

The full source code of the HAS is available as open software in GitHub Repository and can be freely downloaded on the Internet at address:

https://github.com/dimitarminchev/HAS8266

Conclusion

This paper describes the implementation of an Internet of Things Home Automation System with energy measurement capabilities. The basic hardware components are the ESP8266 micro-controller by Espressif and energy meter chip HLW8012 by HLW Technology. The Software works over the TCP/IP stack and uses the MQTT protocol to communicate between the devices. The minimum configuration of the HAS includes one control device called Gateway and two or more manageable Host devices. The user friendly, simplified interface is web based. The platform has open 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] ESP8266EX, Espressif System, https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

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

[6] Microsoft Visual Studio Code, https://code.visualstudio.com/download

[7] PlatformIO IDE for VSCode, http://platformio.org/platformio-ide

[8] Arduino MQTT broker, https://github.com/xDWart/MQTTbroker

[9] ESP Async WebServer, http://platformio.org/lib/show/306/ESPAsyncWebServer

[10] ArduinoJson, http://platformio.org/lib/show/64/Json

[11] PubSubClient, http://platformio.org/lib/show/89/PubSubClient

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

[13] M. Dagarin, QList, https://github.com/SloCompTech/QList

[14] D. Colanardi, ESP8266Ping, https://github.com/dancol90/ESP8266Ping

[15] S. Staub, Ticker, https://github.com/sstaub/Ticker