Wireless smart monitoring energy system: a proof of concept

Marco Costantino
Department of Mathematics
University of Padova
Padova, Italia
marco.costantino1@studenti.unipd.it

Francesco Magarotto
Department of Mathematics
University of Padova
Padova, Italy
francesco.magarotto@studenti.unipd.it

***Abstract*—This document describes a prototype of domotic smart plugs. The system developed includes the smart plugs, a server, and a web interface that allow the user to monitor the power consumption of each individual appliance connected and power them on and off remotely. Furthermore, the system, provided that its settings are correct, keeps the user from turning on enough appliances to trigger the circuit breaker. The prototype has been developed keeping in mind issues of power consumption and reliability of communication between components of the system while keeping the traffic light.**

***Index Terms*—Domotic plugs; Reliable UDP;**

I. Introduction

Nowadays, keeping track of domestic energy usage is a common scenario in domotic applications that aim to monitor the real-time watt consumption of home appliances. In most cases, the solution is a sensor connected to the main safety switch yielding the total energy consumption. Our approach, to get more detailed information and to have control over the use of power by the appliances, is to have an energy sensor for each plug connected to the 220V line. In the following paragraphs, we present our *proof of concept* (POC) which, functionally, consists of three main components: the smart plugs, the server, and the web interface. In our POC, we have modified the architecture described in [**?**] obtaining a specific low-energy consumption-driven infrastructure. This consists of the smart plugs, a computer (for the server), and a device (for the web interface). In a home scenario, these devices are connected to the home wireless router. The communication between smart plugs and server occurs over UDP with a simple form of reliability implemented at the application level and tested using Wireshark [**?**]. The architecture implemented differs from the most common ones in two major aspects: first, most homes aren't equipped with this kind of system, instead, a circuit breaker triggers for safety reasons when too much power is being consumed. Second, when a similar system is used it usually relies on cloud computing or external services. In contrast, our system offers the functionalities already described and both computing and data storage are performed locally. This can be an advantage in terms of both privacy and security. Furthermore, this is an advantage in terms of reliability: our system doesn't need an internet connection to work properly, instead, it just needs a working WLAN. However, while this aspect is significant, the development of the prototype hasn't focused on the security aspects and some issues will be discussed in the appropriate section. The architecture of the system is such that if need be it can be easily modified to use third-party services. The smart plugs have been prototyped using Arduino and the server has been written in Java. Let's now consider a use case example: the user sets the maximum power consumption possible in the home and then turns on two appliances. When this happens for the first time the system decides, based on the type of appliance connected, a default value for the maximum power usage of the appliance. This value is then updated to reflect the real maximum power usage of the appliance. Now that the system is running and some appliances are turned on, the user will be able to switch on only the appliances that consume less than the available power. Our POC covers this use case and shows that such a system can be practical and useful and that the traffic generated isn't enough to sensibly worsen the performances of the typical home WLAN. To summarize, the system keeps the power usage from reaching the maximum limit set by the user by forbidding the power-on of appliances that consume more than the available power. It also gives granular information about the energy consumption of the device and allows the user to turn on or off appliances remotely (via web interface). Aside from the main use case, in the age of climate awareness, such a system could be interesting to an environmentally conscious user that can monitor the power usage of appliances but also could set the maximum power consumption to be lower than the real one to use less power or save money.

II. General architecture

The general architecture has 3 main components: the smart plugs, the server, and the web interface.

The smart plugs have been prototyped using Arduino. The module drives a sensor, needed to gather data on the power consumption and communicates with the UDP server

After describing some architectural design aspects,

A. Smart plugs

The smart plugs is a small hardware plug which consists of a WeMos D1 R2 board, a 220v relay that can be controlled

using 5 volts digital signals, and coil connected to a PZEM-004T sensor which provides the readings to WeMos through its digital pins. The connection scheme is represented in Figure 1, and has a cost about €25, so it is not so expensive. The WeMos D1 R2 board is equipped with a wireless module working at 2.4GHz called EPS 8266 and the wireless connection settings are hard coded in the software, since to store a configuration file an SD module and an SD card are needed. The board is programmed in C/C++ using Arduino IDE and uses an UDP library to create the packets which data are serialized using the JSON format. In every Arduino-like board, there is a function which allows our program to wait for UDP package arrivals from the server, affecting the relay status. Another thread, once the board has recognized and registered the server, sends update periodically if there is a variation greater than the 10% of the energy consumption. Firstly, smart plugs wait for an *INIT message*, sent by the server via broadcast. Once the packet has been received, each plug saves the information about the server and is not replying to this type of messages for a certain amount of type, or until a hard reset is performed. Then, the plug starts to send *UPDATE messages* to inform the server about a energy consumption variation. To do so, a thread keeps watch over the readings provided by the PZEM sensor. In order to switch correctly the relay, the board waits for *ON/OFF messages* from the server. After those requests has been received correctly, the correct action is performed and then a *ACK message* is sent to inform the server about the successful receipt of the command.

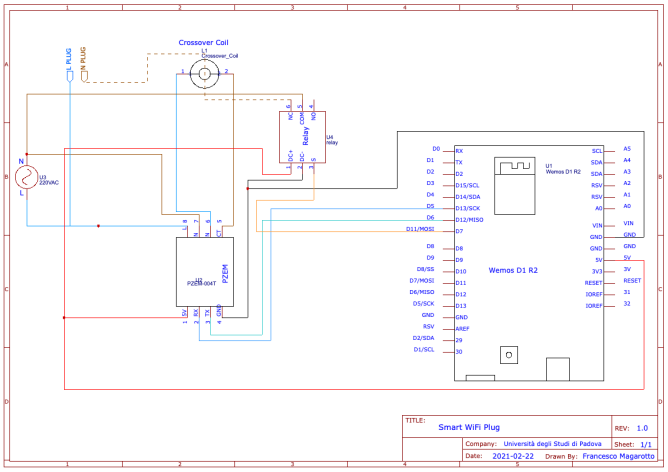

Fig. 1. Electric schema of the wireless smart plug

B. Server

C. Web interface

III. System results

IV. Consumption