An I2C and Ethernet based open-source solution for home automation in the IoT context

Lucian Nicolae Pintilie, Titus Pop, Ioana Cornelia Gros, Adrian Mihai Iuoras
Department of Electrical Machines and Drives
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Lucian.Pintilie@emd.utcluj.ro

***Abstract*—This paper presents a solution for a custom home automation application, with emphasis on its advantages: open-source system – it may be reprogrammed by the user, modularity – integrated IC's with specific tasks, and Internet of Things (IoT) based context – web server access. The proposed system is implemented based on a development platform, with embedded microprocessor that, due to its OS, allows the management of the tasks needed for this application. The Ethernet based system give the user the opportunity of managing and controlling the application through a web page user interface. The hardware implementation brings out also the utility of I2C communication protocol for the used components. The novelty of the subject is not based on the interfaced components, but in combining the communication protocols and the software tools in order to obtain a reconfigurable, open-source, custom application.**

***Index Terms*—open-source, I2C, Ethernet, home automation**

I. INTRODUCTION

Recently, home automation solutions are offered by strong technology companies in a large suite of commercial products, proposing modern, high technology devices that communicate with the user's daily routines. The increased number of intensively implemented standards and protocols (KNX, Zigbee, X10 etc.) cause uncertainty in choosing a solution, while the demand of secured, accessible, energy efficient solutions primes for domestic users [1], [2]. The integration of the solution in larger compatible systems, along with its capability of re-configuration the controlled devices are also aspects not to be neglected.

In the context of home automation solutions [3], the main objective of this article is to present a manner of extending the house hold grid functionality, taking into account the reduction of energy consumption and the possibility to work on open-source environments. The results of a custom home automation application are presented within this work. The implementation focuses on using an embedded micro-computer (Raspberry Pi, Intel Galileo etc.) with OS, with elements of human-machine interfaces, I2C communication of devices [4], sensors and actuators and converters for the execution units, all of them in the range of house holding application.

Although the IoT concept for home automation system is related to Wi-Fi communication [5], a more reliable approach is used, using Ethernet, as it offers a more secure network and also energy efficiency issues, because it uses the same power source [2]. The Ethernet media offers high flexibility in therms of extending the capability of the employed network, because it uses the TCP / IP protocol. So, the wired Ethernet network can be shared via Wifi, LiFi, and even Power Lines. Also, making use of Ethernet adaptors within the development platform used, the implementation is done using software tools that allow the interaction of the user with the application through the dedicate web page. In this scope, UNIX based Python language is used, inter-operating with HTML language through dedicated modules (Flask, NodeJS, MQTT etc.) [6]. In this manner, there is possible for the final user to run the application in real-time, by remote debugging.

The flexibility of the application is explained in Section II, revealing modularity and human interaction aspects of the application. Section III presents the communication protocols and the technologies used in order to implement the custom home automation application, presented gradually in Section IV, with the results and the final user interface. The conclusion section emphasize the importance of this approach in terms of flexibility and re-configuration/customization.

II. MODULARITY AND HUMAN INTERACTION ASPECTS IN CUSTOM MADE HOME AUTOMATION APPLICATION

Real-time IoT systems are widely experimented and implemented in a various range of industrial or commercial solutions, including automations, smart homes, cities, transportation, cars etc. but lack of homogeneity driven from the hardware used: different manufacturers of sensors, actuators, processing devices, software tools and final users' demands [7]. Flexible architectures are needed in this context, in order to make a contribution to several applications, or of the same domain, with minimum efforts from the embedded designer. Aspects of *modularity* at the hardware architecture level are proposed within this work, by integrating into the experimental back an Atmel 8-bit ATtiny MCU [8], with

978-1-7281-3349-2/19/$31.00 ©2019 IEEE

reduced on-board capability, but with high potential of software customization.

The *human interface* element is also easy configurable, using accessible, novel software tools of web design combined with embedded processing. The web server is hosted on the development platform, containing basic control elements for the devices in the circuit.

The proposed hardware architecture is presented in Figure 1, and briefly introduced in this section, highlighting the special functionalities of the overall system.

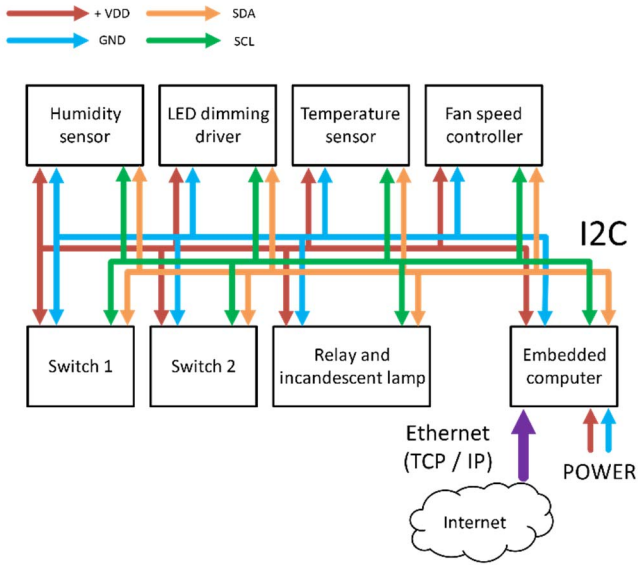

Figure 1. Architecture of the I2C network

The main components of the proposed network topology can be classified into four major categories:

- Central unit (Embedded computer);
- Sensors (temperature, humidity);
- Execution elements with drivers (light, heath, motion);
- Human interaction / human – machine interface (switches);

There are six key features that make this combination of communication protocols and hardware resources a versatile and flexible solution:

- ✓ *modularity* – a custom – programmable Atmel 8-bit ATtiny MCU can be configured to perform a specific task in the hardware assembly. In the present work the ATtiny MCU controls the execution elements (light and ventilation), and the human – machine interface switches (dimmer and on / off switch), according to the instructions sent via the I2C bus from the main control unit.
- ✓ *open-source* – the custom user made modules that compound the network can be software configured (re - programmed), related to a specific application, offering flexibility by comparison with industrial, or commercial solutions that are preset or pre – programmed;
- ✓ *open-source programming* – the Wiring programming language, Arduino IDE based is used, as it offers accessibility for users;
- ✓ *Ethernet based communication* – relatively secure and reliable network for home automation application. The Ethernet infrastructure relies on TCP / IP protocol. This means that the communication can be traversed over other media means (ex. WiFi, Power Line);
- ✓ *I2C communication* for devices and sensors – yielding in a compact hardware architecture. Nowadays, most commercial variants of sensors and low power actuators are I2C compatible and ready to use. It means that, in such a configuration, these devices can be used by simply attaching them to the same shared I2C bus along the house. Another advantage of using this protocol consists in the fact that the devices can be custom made, by the user;
- ✓ *human interface elements* – The switches are custom addressable, so that can be associated with the execution element at user's choice - for example, turning on/off a light bulb that is not located in the same room where the switch was installed. There is also a web server used for controlling the elements in the application, using an embedded low – power microprocessor, instead of a personal computer. The web interface offers a great user experience in terms of human – machine interaction;

III. CURRENT TECNHOLOGIES AND COMMUNICATION PROTOCOLS

Among the technologies used in this approach, Inter Integrated Circuit (I2C) communication represents an intensively, easy to use, "little communication" protocol [9]. It has become a communication standard, especially for integrated circuits and microcontrollers; it represents a hard–wired communication protocol that enables digital information exchange between two digital electronic terminals (ex. microcontroller units) via only two wires. Being a synchronous serial communication, it requires the two wires for data exchange SDA-Serial Data and for synchronization SCL–Serial Clock. The I2C protocol uses a "master–slave" communication model, which means that, a main unit (master) is required, to control one or more secondary "slave" units: sensors, microcontrollers, execution units, interfacing elements etc.

The I2C compatible devices are connected to a shared I2C bus and they are also powered from the same power supply, like the embedded computer's output power supply. The power bus consists of +VDD and GND wires, and the I2C bus consists of SDA and SCL wires. All the devices are wired in a parallel connection via the buses. Most embedded computers use +5V or +3.3V voltage levels. Thus, in order to use the 3.3V

device together with 5V devices, an auxiliary electronic circuit is usually required (level shifter).

In this paper, the embedded computer acts as master I2C communication device and the microcontroller units from the sensors, execution elements and human interaction devices act as slaves.

One of the main objective of the work is constructed around the design and programming of an embedded system and then command it in a local network. Any low resources embedded microprocessor can become a basic web server that responds to external requests; thus, the Ethernet protocol is related to sending and receiving data about the created circuit over the local network, based on a web page. Ethernet communication works the same at the computer level and TCP/IP protocols as an embedded processor with Ethernet capability [10]. The software resources combined in this application in order to obtain a real-time control through the web page are Python/Flask modules, with dedicated socketio package.

An automated numerical process in areas like energy production and distribution, electric traction etc. usually implies a complex Supervisory Control and Data Acquisition (SCADA) system. Figure 2 presents an assembly of a complex numerical control system, containing three types of embedded systems (microcontroller based, DSP based and microprocessor OS based), with focus on the micro-computer that allows web server hosting and control.

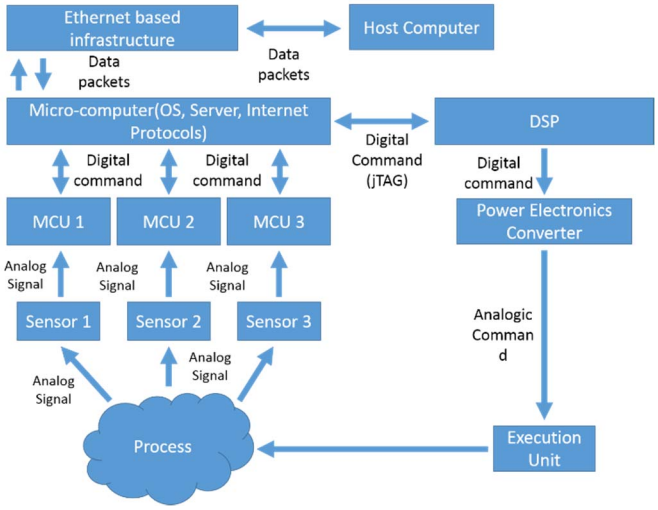

Figure 2. Assembly of a complex numerical assisted process

IV. IMPLEMENTATION OF AN I2C AND ETHERNET HOME AUTOMATION APPLICATION

The present application targets an approach to a complex numerical assisted process, as detailed in Section III, specifically, a custom home automation process.

Home automation process involve more than one specific physical application domain: light, heath, and motion control. The home thermal regulation process can be assisted by controlling the heath (gas pressure at the burner) and ventilation or air flow by controlling the speed of the exhaust fan, inside the rooms. In order to control such a complex automation implementation, a network or device constellation is needed.

I2C can be considered as a solution to this need. Because of this fact, the devices that belong to the considered I2C network can be classified into four classes: central units, sensors, execution elements or actuators and human interaction devices.

The implementation strategy adopted in this paper follows this classification by grouping the I2C device addresses into several categories:

- Execution elements: – Digital relay (0x07);
  – Digital variators (0x08 and 0x09);

- Human interaction devices: – Digital dimmer (0x13);
  – Digital switch (0x14);

- Sensors: – Digital temperature sensor (0x48);
  – Digital humidity sensor (0x5C);

After establishing the grouping strategy the right topology has been chosen. The corresponding network topology follows the standard I2C parallel networking method (two wires for communication and two wires for power supply) indicated in Figure 1. The wiring was done using multiple screw terminals.

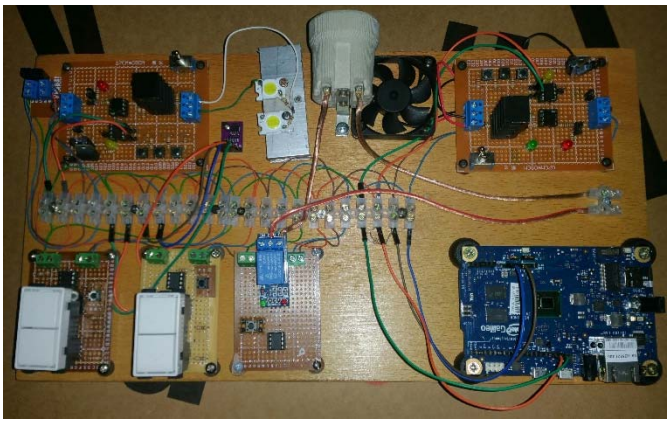

Figure 3. The design and implementation of the I2C network

Each addressable device on the network can be considered as a module. Each module has an analogic-physical device (switch, relay, voltage variator/chopper), that is interfaced to the digital I2C communication interface by an ATTiny 45 microcontroller. These microcontrollers are low power, low cost, and easy to program by using Arduino IDE and a standard programming device like USB Tiny ISP.

The program that is flashed inside the memory of the ATTiny 45 allows the I2C module device to be fully functional and accessible in the I2C network, because, the flashed program contains the unique I2C address, and all the control strategy of the specific task that the module was is intended to do inside the I2C network. By the upper classification each module in the network has a specific role in the system.

The commercial I2C sensors are designed to work almost on the same principle, so they can be used easily in the network. Excepting the temperature and humidity sensors all the rest of

the modules are user custom made devices, using the ATTiny 45 MCU.

The I2C is a "master–slave" communication protocol model, which means that a main control unit that initiate the communication process is involved and a secondary device that waits for commands to be communicated is also needed. The slave units are the modules that are connected to the I2C network: switch, relay, chopper, sensor.

The master or main control unit is an Intel Galileo, that runs Debian 7 Wheezy Linux. The Linux distribution also contains the MRAA low level skeleton library. This library is used to interact with the low level devices (ex GPIO, ADC, PWM, I2C) from a Linux operating system compatible application. Intel uses this kind of library to manage all the hardware resources from the running operating system.

The Linux operating system uses Python as one of the primary programming language. Python can manage this set of low level hardware resources of Intel Galileo, via the MRAA library. To link everything together (ex. relay and switch, dimmer and voltage chopper etc.), a Python application was developed on Intel Galileo.

Python offers also great advantage and versatility, because the modules can be adapted to the user's application. A intensively used module for Python is Flask - a Web server that can be initiated from a Python application. This means that the variables used in the code of the application, can be easily externalized to a HTML page. Having this opportunity, means that a hardware - interactive Web page can be developed. The elements found on that page: sliders, buttons or checkboxes can control the GPIO ports or the PWM output of the Intel Galileo, or even the devices attached to the I2C interface and network.

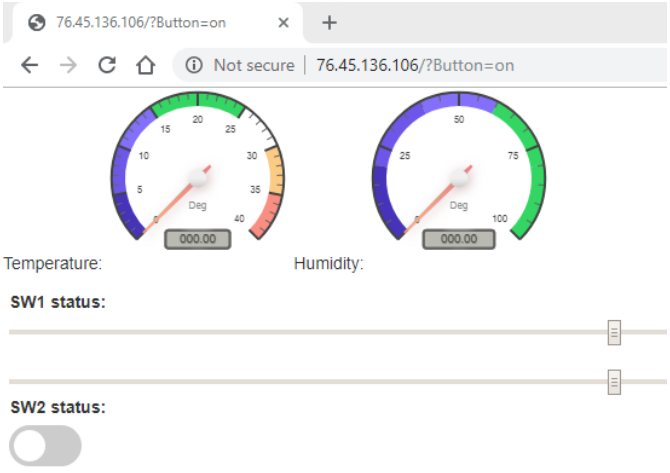

Figure 4. An example of HTML interactive user interface

The Web page is interactive and it controls the I2C hardware attached to the bus. The gauges, sliders and buttons were created using JAVA script. The real-time interaction aspects of the Python application from the HTML page is achieved by using the SocketIO library.

The Ethernet connection allows the TCP / IP protocol to be used for remote assistance (Secure Shell - SSH remote command line) and HTTP WEB connection.

V. CONCLUSIONS

The present paper describes an implementation of a custom home automation application, using I2C and Ethernet communication protocols. The considered I2C network involved in this home automation highlights advantages listed above:

- The network topology is custom and modular - custom modules can be created using integrated circuit MCUs like ATtiny and commercial I2C compatible sensors and execution elements can be attached to the bus;
- The tasks of the component modules can be re-assigned from the master unit through the Python application, using the addressability of the I2C modules;
- The software used is fully Open–Source (Arduino IDE, Debian Wheezy, MRAA, Python and Flask);
- The Ethernet connection allows remote access and management of the system;
- The I2C network uses only two wires for data exchange; thus, it may reduce additional costs of implementation and hazard of using high voltage for control elements such as user interaction switches and dimmers.

ACKNOWLEDGMENT *(HEADING 5)*

This paper was supported by the project " High power density and high efficiency micro-inverters for renewable energy sources" - MICROINV - Contract no. 16/1.09.2016, project co-funded from the European Regional Development Fund through the Competitiveness Operational Program 2014-2020.

REFERENCES

[1] https://www.icontrol.com/blog/2015-state-of-the-smart-home-report/. "2015 State of the Smart Home Report."

[2] J. P. S. Monteiro, "I2C Network for Home Automation", Mestrado Integrado em Engenharia Electrotécnica e de Computadores, dissertation, 2017, open-acces.

[3] Chinju P., A. Ganesh, C. Sunitha, "An overview of IoT based smart homes", *IEEE 2nd International Conference on Inventive Systems and Control (ICISC)*, 2018.

[4] Leens, F., An introduction to I2C and SPI protocols, *Instrumentation & Measurement Magazine, IEEE, 2009*.

[5] N. V. Rajeesh Kumar, B. S. B. Praveen, A. V. Sarveswara Reddy, B. Baron Sam, „Study on IOT with reference of M2M and WiFi", *International Conference on Information Communication and Embedded Systems (ICICES)*, 2017.

[6] Yuvraj Upadhyay, Amol Borole, D. Dileepan, "MQTT based secured home automation system, IEEE Symposium on Colossal Data Analysis and Networking (CDAN), 2016.

[7] Yelamarthi K., Aman Md. S., Abdelgawad A., „An Application-Driven Modular IoT Architecture", „Wireless Communications and Mobile Computing", 2017.

[8] Atmel ATtiny25, ATtiny45, ATtiny85 Datasheet - Farnell

[9] Sarns S., Woehr J., "Exploring I2C" Embedded Systems Programming, vol. 4, Sept. 1991.

[10] Axelson J., "Embedded Ethernet and Internet Complete Designing and Programming *Small Devices for Networking"*, Lakeview Research, June 1, 2003, pp.14 -32.