Building network-enabled smart sensors and actuators

Ivan Pavić[(1)], Josip Puškar[(1)], Ivan Soldo[(1)], Ivan Spasić[(1)], Hrvoje Džapo[(1)], Dražen Ćika[(2)]

Faculty of Electrical Engineering and Computing[(1)]
Department of Electronic Systems and Information Processing
Unska 3, 10000 Zagreb, Croatia
E-mail: ivan.pavic2@fer.hr, josip.puskar@fer.hr, ivan.soldo@fer.hr, ivan.spasic@fer.hr, hrvoje.dzapo@fer.hr

Research and Engineering Center d.d.[(2)]
Ulica gradaVukovara 284, 10000 Zagreb, Croatia
E-mail: drazen.cika@rec-global.com

Abstract –The advent of the new Internet-of-Things (IoT) paradigm in last few years redefined the ways how the small embedded systems can enhance smart sensors and actuators to provide new possibilities in ubiquitously interconnected global world. Although there is a rising tendency in modern low-power embedded system design to provide some kind of IoT functionality out-of-the-box due to the availability of network-aware microcontrollers and system-on-chip (SoC) solutions, there is also a great need to upgrade legacy solutions to fit them into this new paradigm shift.

At the lowest level of the system design, in most cases there is a requirement for some computationally capable low power microcontroller, long autonomy from a single battery power supply, embedded network interface (wired or wireless, typically Ethernet, Wi-Fi, and Bluetooth), embedded communication protocol stack, persistent data storage, and some scalable real-time operating system (RTOS). Moreover, there is typically a need to provide access from a smart client device with rich user interface, such as tablets or smart phones. This is a typical scenario for upgrading legacy systems to provide Internet connectivity, remote monitoring, data logging, crowd sensing etc.

In this article we describe a simple generic framework for upgrading legacy devices, which was built upon the state-of-the-art low cost and low power off-the-shelf hardware components, and open source software. The solution can be easily adapted for various types of smart sensors and actuators. We describe a common set of requirements for building such systems, design decisions reasoning, comparison and justification of HW/SW components choice, and sample demonstration of adaptation and upgrade of the chosen commercial equipment, a device for bubble production at children fun events and similar applications.

I. INTRODUCTION

The embedded devices based on microcontrollers have been in use for many years. The result is that majority of applications and devices today have some sort of intelligence in a form of software built in at least in the part of the device functionality.

It could be concluded that all those devices were capable of communication with their environment in elegant and transparent manner, but this is not always the case. The devices designed and put into operation before the proliferation of low power microcontrollers and low cost wireless interfaces are very often isolated when it comes to the communication with external world. There may be various interfaces available for users or system maintenance, but those are often based on the wired infrastructure such as the legacy RS232 port, USB connection, Ethernet port and similar. It is not always an easy task to enable such devices to communicate with their environment over the Internet.

Older devices which have not been designed based on the microcontroller or microprocessor could be completely cut out of the Internet of Things (IoT) future driven by most silicon vendors and technology adopters today.

In this article the possibility is explored to enable simple devices, even those originally containing no firmware and digital electronics, to exchange the information with their environment using the wireless communication and Internet. In other words, the attributes ‘smart’, and ‘networked’ are added to some low cost equipment possibly having only few simple functions.

II. MICROCONTROLLER PLATFORM

The STMicroelectronics™ microcontroller (MCU) used on the development board chosen for this demonstration project belongs to the STMF32F4 family, which is based on the ARM Cortex M4 core.

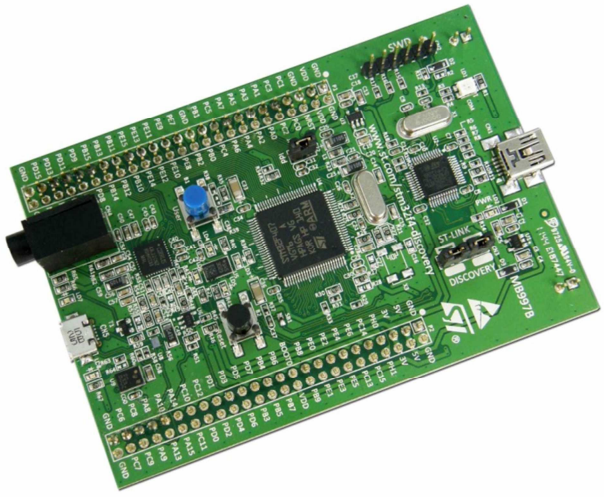

Figure 1: STM32F4 Discovery evaluation board

The STM32F4 Discovery evaluation board is shown in the

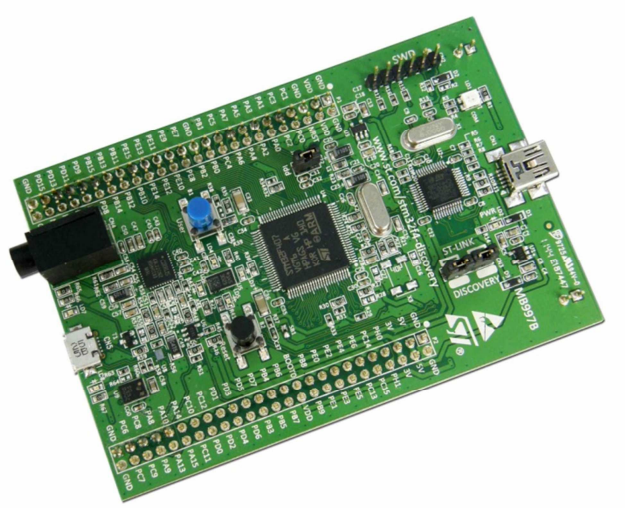

Figure 1. It is used for this project because of the low cost, expandability, and convenient form factor enabling easy design of the necessary carrier and expansion board with power supply and wireless and I/O interfaces. The carrier board custom designed for the purpose of this project, and populated with the STM32F4 Discovery, and all interfaces is shown in the Figure *2*.

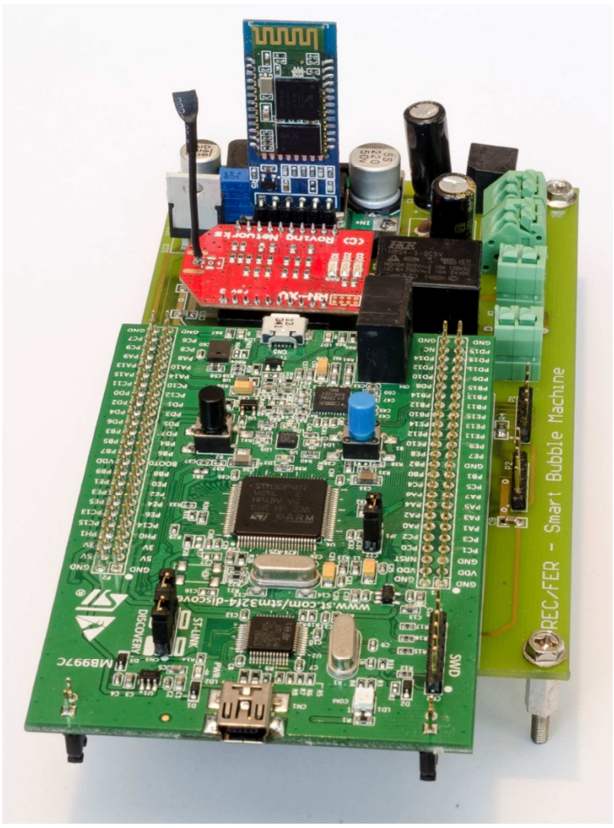

Figure 2: Carrier board with power supply and I/O interfaces

The specific DSP instructions and extensions available on Cortex M4 platform have not been used here, so that some other low power microcontroller core like the ARM Cortex M0 could also be used. Key factors for the platform selection are often device availability and support by the tool vendors, manufacturers and community. The STM32F4 platform is well supported and easily available on the market.

III. PERIPHERAL LIBRARIES AND DEVICES

The software (firmware) for the STM32F4 Discovery board is implemented using various peripheral and system core libraries.

Functions for controlling and interaction with peripheral devices, such as GPIO pins, USART interface and I2C interface are written using the Standard Peripheral Drivers library supplied by the STMicroelectronics™, but any other functionally equivalent library could be used.

There are two different set of libraries currently supported by the manufacturer: the older ST Firmware Library (listed now as the STM32F4 DSP and standard peripheral library) which is still active and supported, and the new STM32CubeF4 which was not used for this project. Note that library names might be slightly different for other ARM cores like M0 or M3, and older libraries might no longer be supported in the future.

The example project device described in this article has two actuators for which the output control is implemented:

- AC motor relay is driven by the general purpose input / output (GPIO) pin which is configured as output pin with pull up resistor.
- DC motor speed is controlled using pulse width modulation (PWM), therefore appropriate timer channel had to be configured for PWM.

The Real-Time Clock (RTC) is the clock oscillator with support needed to keep track of the current time. Although RTCs are often used in personal computers, servers and embedded systems, they are also present in almost any electronic device that requires accurate time keeping. Use of the separate RTCs avoids the necessity for microprocessor or microcontroller to run all the time just in order to keep time, so the sleep mode of operation is possible if needed. RTC can be realized as external chip from the same or different manufacturer, or it could be available within the silicon of the main MCU, which is the case in this project.

RTC on the STM32F4 series offers additional features like the calendar, the alarm, the periodic wakeup unit, tamper detection, and support for the time stamp and calibration applications. RTC uses external oscillator and Static RAM (SRAM) memory which are powered by Li-On battery connected to dedicated MCU pins. Configuration and initialization of RTC is done using the Standard Peripheral Library.

IV. WIRELESS INTERFACES

Bluetooth is the wireless technology standard for exchanging data over short distances, usually between fixed and / or mobile devices. It is mainly used for peer to peer communication, but could also be useful for building personal area networks (PANs). It was originally conceived as a wireless alternative to the RS-232 data cables.

Bluetooth can connect several devices while overcoming their synchronization problems. There is large number of ready-made bluetooth modules on the market, which simplifies the design of the device. One of the most popular modules is the HC-05. It offers serial (USART) interface for communication with MCU, and configuration is done using AT commands. The module itself is not

suitable for direct soldering on the through hole PCB, so the custom carrier board was designed for the project using only seven pins as an interface, as shown in the Figure *3*.

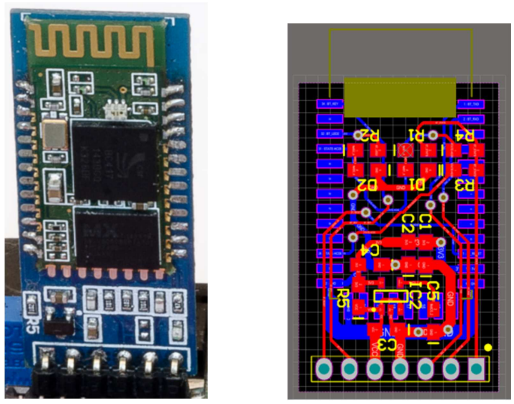

Figure 3: Bluetooth module with its custom carrier board

Wi-Fi is a local area wireless technology that allows an electronic device to participate in computer networking using 2.4 GHz UHF and 5 GHz SHF ISM radio bands.

Wi-Fi devices can connect to a network resource such as the Internet via a wireless network access point. Such an access point (or hotspot) has a range of about 20 meters indoors and a longer range outdoors. Hotspot coverage can comprise an area as small as a single room with walls that block radio waves, or as large as many square kilometers achieved by using multiple overlapping access points.

Wi-Fi has adopted various encryption technologies. The early encryption (WEP) proved to be easy to break. Higher quality protocols (WPA, WPA2) were added later.

There are various pre-built Wi-Fi modules on the market, such as the WiFly from the company Roving Networks™ shown in Figure 4.

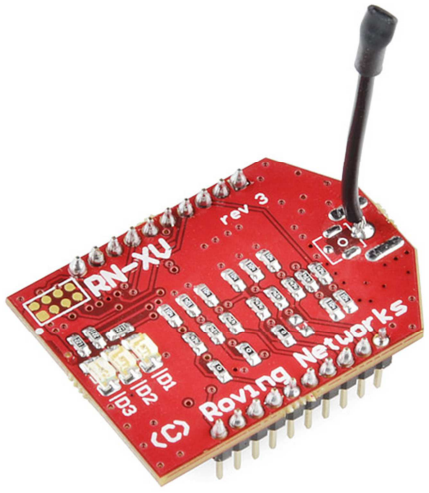

Figure 4: Wi-Fi module RN-XV WiFly by Roving Networks

Wi-Fi module is actually a small communication computer incorporating 802.11 b/g radio, 32 bit processor, TCP/IP stack, the real-time clock, crypto accelerator, power management unit and analog sensor interface.

The module is pre-loaded with Roving firmware to simplify integration and minimize development time of the application. In the simplest configuration, the hardware only requires four connections (PWR, TX, RX and GND) to create a wireless data connection.

V. DATA LOGGING AND FILE SYSTEM

Each field device with sensors and actuators can have requirement for data logging, so the storage space for larger amount of data might come handy.

SD card functionality is implemented to fulfill such requirements, and the Secure Digital Input/Output (SDIO) Interface was chosen for this purpose and enabled using the Standard Peripheral Driver Library. The SD card connector had to be added to the custom designed carrier board, as it was not present on the basic evaluation board.

For handling and managing files the well known and popular Chan's FatFs is used. FatFs is generic FAT file system module for small embedded systems. It is written in compliance with the ANSI C standard, and is separated from the I/O layer.

Since the FatFs module is a file system layer, it is completely separated from the physical devices, such as memory card, hard disk or any other type of storage device. The low level device control module is not a part of FatFs module and is provided by implementer of the file system functionality.

VI. REAL TIME OPERATING SYSTEM

When confronted with the requirements to handle several asynchronous communication tasks, while keeping the responsive user interface and handling measurement data from the sensors in the background, developers usually opt for the usage of some operating system to help partition tasks in manageable chunks of code.

Also, the embedded devices often have real time response requirements, and specifically IoT devices have to behave economically and be able to go to sleep when there are no tasks to be executed left. The support of the Real Time Operating System is needed, or at least preferred.

Multitasking and the real time requirements are in the case of this application achieved by FreeRTOS.

FreeRTOS is real-time operating system kernel for embedded devices. It is distributed under the GPL with an optional exception. FreeRTOS is designed to be small and simple. The kernel itself consists of only three or four C files.

To make the kernel code readable, easy to port, and maintainable, the FreeRTOS is written almost entirely in C, and there is only few assembly functions included where needed, mostly in the architecture-specific scheduler routines.

The FreeRTOS provides functions for multiple threads, mutexes, semaphores, software timers, and other common RTOS objects. It also supports thread priorities.

The FreeRTOS based software structure of the demo application shown in this article can be seen in the Figure 5. Various colors mark tasks, interrupts, HW resources, global variables, or the RTOS objects such as semaphores or message queues.

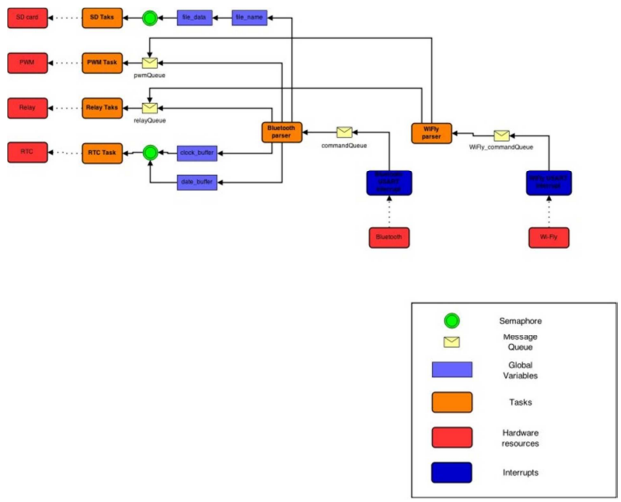

Figure 5: Firmware block diagram

Data received over USART is handled by interrupts in the background. Every byte is forwarded by the interrupt to its message queue and then handled by the task responsible for the particular message queue content.

The task checks the message queue one byte at the time and sends them to the parser function which sets some control signals and semaphores, accordingly to the command parsed from the string received from the message queue.

III. EXAMPLE PROJECT DESCRIPTION

The example project chosen for the demonstration purposes is called the Smart Bubble Machine. Its purpose is purely entertainment, and as its name suggest its main function is to produce bubbles.

In its original form shown in the Figure 6 it is very simple device which basically can only be turned on or off, and when turned on, it produces the soap bubbles in a constant stream until turned off.

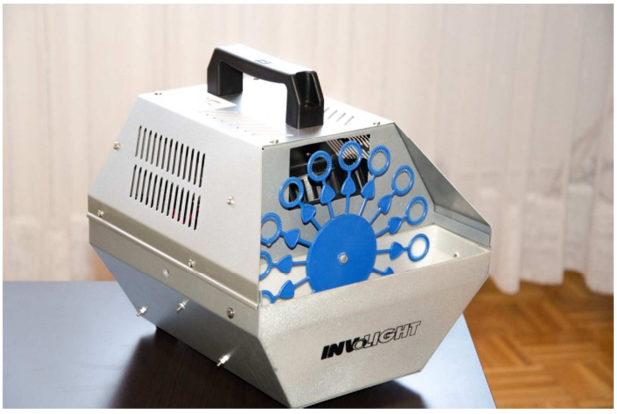

Figure 6: The exterior of the Bubble Machine

The mechanics and electronics built initially into the machine by the manufacturer are really simple: the transformer, some basic power supply, and AC and DC motors. Figure *7* shows the hardware components added to the bubble machine so that it could earn the ‘Smart’ attribute.

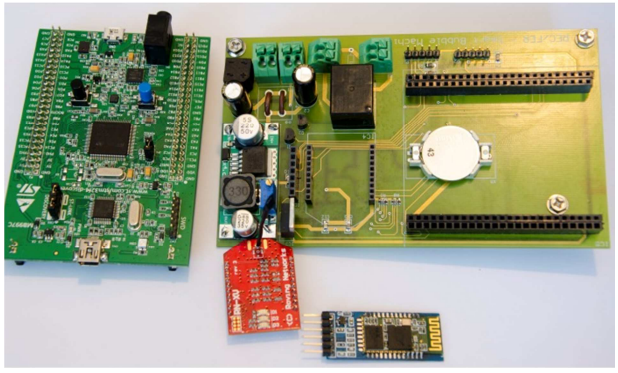

Figure 7: Components added to the Bubble Machine

The basic principle of operation is very simple. The AC motor is connected directly to the 220 V AC mains power supply, and it was used to rotate the big blue perforated disc. After modification, this motor can be also started and stopped under software control using the on-board electromechanical relay.

The transformer is used to transform 220 V AC to 12 V AC which is then fed to the full wave rectifier circuit. The resulting 12V DC voltage was originally used only to constantly drive the DC motor (actually the PC CPU type ventilator) which blows air into the perforated disc, thus creating bubbles. Before exposure to the air stream, the plastic disc rotates through the liquid soap (detergent) filled into externally mounted reservoir.

After modification, the step down switching convertor module was added instead of simple basic rectifier supplying 5 V DC, and additional low drop out linear voltage stabilizer has taken care of the 3,3 V DC voltage needed. All components needed for the project are placed on before mentioned carrier board shown also in the Figure 7. The PC type ventilator is now driven by the PWM output, so that fine control over the ventilator speed is possible and implemented in the software.

The remote control is possible from the near distance and over the Internet using the custom Android application on the tablet or Smartphone, or via the web based user interface from any web browser.

IV. CONTROL SOFTWARE USER INTERFACE

The software control for the Smart Bubble on the remote side is implemented twice: as the Android application and as the Web application. Both applications enable users to monitor and control the device using simple graphical user interface, as shown in the Figure 8 and the Figure 9.

Figure 8: The Android application user interface

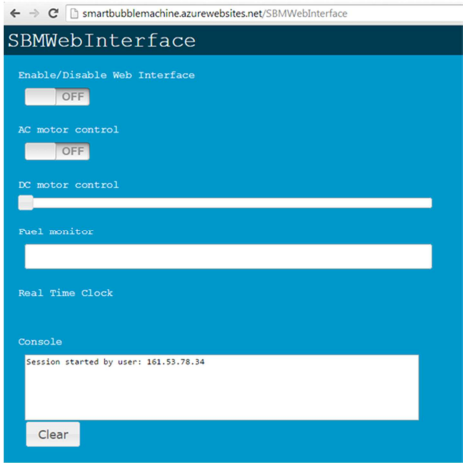

Figure 9: The Web application user interface

The Android application connects to the device using the Bluetooth interface, so that user can change additional parameters and configure device also for the web interface access. It resolves the situation in which the Wi-Fi access has to be configured for new network.

Additionally the Android application offers some debugging features, like the terminal for monitoring transmitted and received data. The terminal application and permission to change the device parameters such as Bluetooth or Wi-Fi connection passphrase are administrator privileged functions.

Parameters which can be changed are divided in three groups:

- Bluetooth connection parameters: Bluetooth module name and passphrase
- Wi-Fi module parameters: Network SSID, WEP encryption, WPA passphrase
- Real-time clock parameters: time and date

Web application is developed using the ASP.NET framework. Application runs on the Microsoft Azure webserver, and interacts with the Smart Bubble device using the HTTP GET method.

It also enables multiple users to control and exchange data with the device at the same time. This is done using the SingalR notification hubs and jQuery Ajax library.

Application has console which is updated every time change is made by any user. Additionally it contains debugging messages which are useful in developing or upgrading the device firmware.

Both applications can't be used at the same time, since the Android application is event driven, but device must periodically send http requests to refresh. Therefore the web interface can be disabled using the *Enable/Disable* control shown in the Figure 9.

Furthermore, Android application sends and receives data independently. For example, the real-time clock data is sent every second, while sensor data such as the fluid level sensor (not implemented jet) could be sent when the sensor readings are changed.

The web application keeps the TCP connection open and sends data after device requests data, and posts sensor data and real-time clock data to the web application.

V. CONCLUSION

This project is demonstration of basic principles of microcontroller usage to control and monitor device remotely using the wireless connection, actuators such as AC or DC motors, and standard off-the-shelf client device such as the tablet or smart phone.

Possibility is added for the user to remotely turn simple legacy device or machine on and off, to regulate the speed of the ventilator thus influencing the bubble production rate, and it would be easy to program the machine to operate independently and log data during operation.

It is also shown how to implement and use some widely used technologies, such as the Bluetooth, the Wi-Fi, the SD card, the Flash file system, or the RTC. The Real Time Operating System (RTOS) has been applied successfully.

Using the resources and techniques described, any legacy machine can be controlled and monitored remotely from the close range or over the internet using two different wireless options, data from sensors can be collected and sent to the server, new machine configuration can be sent and information stored on the MCU or SD card, and the machine can connect itself to the Internet based on some predefined events.

Therefore, control and monitoring can be performed without being anywhere near the machine, and without usage of any proprietary technology or components.

All modules, voltage convertors, actuator drivers and microcontroller development board itself are mounted on the custom designed PCB, and some more custom boards have been designed where necessary, or feasible.

The authors would like to thank the RESEARCH AND ENGINEERING CENTER d.d. company for the support and resources during this development.

REFERENCES

[1] Wang, K.I.;Salcic, Z.; Atmojo, U.D.; Pediredla, B.; Hadi, M.;Daji, C.:IP-enabled smart sensor and actuator node for ambient intelligence systemsIntelligent Sensors, Sensor Networks and Information Processing,

2013 IEEE Eighth International Conference on DOI: 10.1109/ISSNIP.2013.6529793

[2] Zug, S.; Schulze, M.; Dietrich, A.; Kaiser, J.:Programming abstractions and middleware for building control systems as networks of smart sensors andactuators, Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on DOI: 10.1109/ETFA.2010.5641341

[3] Aoki, S. ;Kirihara, Y. ; Nakazawa, J. ; Takashio, K. ; Tokuda, H. : A sensor actuator network architecture with control rules,Networked Sensing Systems (INSS), 2009 Sixth International Conference on DOI:10.1109/INSS.2009.5409934

[4] Derbas, A.M. ; Al-Aubidy, K.M. ; Ali, M.M. ; Al-Mutairi, A.W. : Multi-robot system for real-time sensing and monitoring, Research and Education in Mechatronics (REM), 2014 15thInternational Workshop on DOI: 10.1109/REM.2014.6920442

[5] Cheng Jing ;Guo-jun Zhao : Systems and Informatics (ICSAI), 2014 2nd International Conference on DOI: 10.1109/ICSAI.2014.7009313