Overview of architectures with Arduino boards as building blocks for data acquisition and control systems

Vladimir Cvjetkovic
Faculty of Science
University of Kragujevac
Kragujevac, Serbia
vladimir@kg.ac.rs

Milan Matijevic
Faculty of Engineering
University of Kragujevac
Kragujevac, Serbia
matijevic@kg.ac.rs

***Abstract*— Standard SBCs (Single Board Computer) with number of standard shields and sensors can be used as building blocks for rapid development of network of intelligent devices with sensing, control and Internet access. Arduino family of boards having high popularity and large number of sold units featuring open access, reliability, robustness, standard connections and low prices, possesses large potential for implementation of autonomous remote measurement and control systems of various levels of complexity. As Arduino boards can function independently, they are complete small computer platforms that can perform various tasks requiring some kind of interaction with the outer world. Arduino boards can be used and programmed in various ways, and can be arranged in various combinations forming some typical implementation architectures that this paper discusses. Starting from basic and simple configurations, more advanced are gradually considered from the aspects of chosen way of programming and combining with other boards. Special attention is devoted to NodeJS as programming platform for Arduino boards and considerations of libraries used with Arduino boards like Johnny-Five, Galileo-io firmata equivalent, mraa library and other ways of program access to GPIO like Linux Sysfs. As typical representatives of Arduino boards' family, the Arduino Uno, Arduino Due and Arduino Galileo were selected, with justification that all other not mentioned boards are somewhere between those selected, according to official hardware specifications.**

***Keywords*— Arduino; NodeJS; Johnny-Five; configurations**

I. INTRODUCTION

Appearance and development of various SBCs like Arduino [1, 2], BeagleBone [3], RaspberryPi [4], RIoTboard [5], PandaBoard [6], OLiMEX [7] and others together with mobile phones, created enormous potential for building various devices capable of interaction with environment, data processing and network communication. Such devices are nowadays also seen and classified as being part of Iot (Internet of Things). Required functionalities of IoT devices are usually:

- M2M (Machine to Machine) communication
- Some kind of data acquisition using adequate sensors
- Local processing of acquired data
- Control of some local system
- Upload of acquired and processed data to some cloud network storage

Some or all of these functionalities can be present including also some specific not mentioned here. Mentioned functionalities are quite general and do not pose limitations by themselves, as the real limits of IoT devices are mainly determined by processing power, speed, available memory, power consumption and similar characteristics. If the task for some IoT device is too demanding, the possibility of logically redefining the task so that more than one IoT device could be used and combined to fulfill the given task, should be considered. That further suggests the use of a number of inexpensive IoT devices with small computing power in a network of devices possessing a significant net effect not possible with a single IoT device.

Regardless of used architecture, including a single device or many devices, each device should operate reliably and in predictable way.

Following chapters are organized in this way:

Chapter 2 Overview of Arduino family, with currently available Arduino general purpose boards grouped according to processors on board.

Chapter 3 Overview of programming modes and connections with Arduino boards, enabling various acquisition and control configurations.

Chapter 4 Overview of architectures with Arduino boards

Chapter 5 Conclusion

Acknowledgement and References

II. OVERVIEW OF ARDUINO FAMILY

Arduino family of boards was selected among others for its popularity resulting in a large number of users and a number of boards to choose from. Besides boards that are more like general purpose computer devices, there are also boards called "shields" that extend functionalities of boards for purposes like Ethernet, WiFi and GSM communication, use of SD cards, motor and relays control, space orientation and other.

Arduino boards are based on Atmel microcontroller units (MCU). On some more powerful devices there is an additional microprocessor computer providing greater processing power and network communication.

Arduino boards [8] Uno, Nano, Mini 05, Mega 2560, Leonardo, Micro, Robot, Esplora are based on Atmel MCUs with AVR architecture.

- ATmega328 – Uno Nano and Mini 05
- ATmega2560 – Mega 2560
- ATmega32u4 – Leonardo, Micro, Robot, Esplora

Some of the boards have variants with added functionalities:

- Arduino Ethernet (ATmega328) based on Uno, Ethernet enabled, Micro SD card
- Mega ADK (Accessory Development Kit) (ATmega2560) for use with Android phones
- Leonardo ETH (ATmega32u4) based on Leonardo, Ethernet enabled, Micro SD card

All previously mentioned Arduino devices have AVR MCU operating at 16MHz frequency.

Arduino Due is different, as it is based on Atmel SAM3X8E MCU with 32-bit ARM Cortex-M3 CPU (Central Processing Unit) core running on 84MHz. Also, Due has significantly larger memory – SRAM (Static RAM) for program and flash memory for uploaded code.

Arduino M0 and M0 Pro are advanced versions of Uno based on SAMD21 MCU, with 32-bit ARM Cortex M0 core running at 48MHz.

Arduino boards Industrial 101, Tian, Yun, Yun Mini, are MCU based, but also have additional MIPS processor [9] based computer supporting Linino [10] Linux distribution based on OpenWRT [11]. Additional Linux computer provides extra processing power for support of MCU acquisition and control tasks.

Boards Intel Galileo and Intel Galileo Gen 2 are based on Intel Quark SoC X10000, a 32-bit Intel Pentium processor-class system on a chip (SoC). That processor runs both the Linux and Arduino acquisition code. Intel Galileo boards are pin to pin and software compatible with other Arduino boards.

Finally, the Intel Edison board is based on two processors, the Intel Atom 500MHz dual-core, dual-threaded CPU and an Intel Quark 100MHz MCU.

From this brief overview of current Arduino boards, it can be seen that there are two kinds of boards:

- With MCU as single processor capable of running Arduino code, called sketch
- Boards with added processor for running Linux which provides additional processing power and supports acquisition and control tasks of MCU
- Intel Galileo and Intel Galileo Gen 2 boards as a special case of boards running Linux where Intel processor executes both Arduino sketch and Linux

Arduino Uno board is a typical representative of MCU only based boards, as an effort was made for other boards to be pin to pin and software compatible.

Arduino Due is a top representative of high performance MCU only based boards, as it has highest MCU clock frequency and available memory.

Intel Galileo is a representative of MCU boards with added Linux computer for additional processing support of acquired data and network communication.

III. OVERVIEW OF PROGRAMMING MODES AND CONNECTIONS WITH ARDUINO BOARDS

A. *Features of Arduino boards*

Arduino boards [12, 13] are designed to provide interaction of a computer system with some environmental physical quantities using appropriate sensors. All boards possess general purpose inputs and outputs (GPIO). Presence of GPIO is the main difference from usual computer systems for everyday use – desktop, laptop, tablets and smart phones. GPIO has two main types of input / output system, analog and digital. All Arduino boards have analog inputs (AI) for voltage measurement. Number of analog input pins and resolution varies for different boards. Analog output (AO) can be implemented with digital outputs as PWM (Pulse Width Modulation) or with ADC (Analog Digital Converter) circuit. Digital pins can be used both as input and output (DIO). Arduino boards support USB communication with external computer running Arduino IDE (Integrated Development Environment) for programming in language resembling C. Other supported communications types are UART (Universal Asynchronous Receiver Transmitter) TTL (Transistor Transistor Logic), I^{2}C (Internal IC) / TWI (Two Wire Interface) and SPI (Serial Peripheral Interface). SPI is mainly used for connecting boards with various shields as it provides very fast communication. UART is hardware implemented for USB on digital lines 0 and 1, but it can be also quite easily software implemented with provided SoftwareSerial library using digital I/O. Serial communication is convenient for data exchange between boards. TWI can also be used for communication between boards or other devices with provided Wire library and using SDA (Serial DAta) and SCL (Serial CLock) lines.

Besides GPIO, very important programming aspect is available memory which is limited as it is part of MCU, and organized as flash memory for code, SRAM (Static RAM) for program execution and EEPROM (Electrically Erasable Programmable Read-Only Memory) as permanent storage for data. Some boards like Yun, Galileo, Arduino Ethernet and shields like Ethernet, also provide program access to SD memory cards. Dynamic characteristics of boards like maximum rate of measurements depend on many factors, with MCU operating frequency as one of the important but rough indicators. Rates for digital IO operations are higher than for analog measurements which use analog digital converter (ADC) circuit that in general requires many processor cycles for conversion.

TABLE I. FEATURES OF SELECTED BOARDS

Board / features	**Uno**	**Due**	**Galileo**
AI	6, 10 bits	12, 12 bits	6, 12 bits
AO	6, PWM, 8 bits	2, DAC, 12 bits	6, PWM 8 bits
DIO	14, 6 PWM, 8 bits	54, 12 PWM 8 bits	14, 6 PWM 8 bits
Processor	ATMega 328	AT91SAM3X8E	Intel Quark SoC X1000
Clock	16 MHz	84 MHz	400 MHz
Flash	32 KB	512 KB	8 MB / 512KB
SRAM	2 KB	96 KB	512 KB
DRAM	-	-	256 MB
EEPROM	1 KB	-	-
Micro SD	- (on shield only)	- (on shield only)	Up to 32 MB
Ethernet	- (on shield only)	- (on shield only)	10/100 Mb/s

Features of selected boards are compared in Table 1.

B. *Programming modes of Arduino boards*

Arduino and compatible boards can be programmed in a number of ways. Depending on selected way of programming, Arduino boards can be used in a variety of configurations ranging from simple with only one board, to configurations including two or more boards and other devices. This paper will consider and discuss using of JavaScript (JS) programming language with various configurations primarily consisting of, and based on Arduino boards. Use of JS can also include communication with Arduino boards programmed in Arduino language (AL) resembling C. Programming modes of Arduino boards that will be considered in this paper are:

- Programing in AL from Arduino IDE
- Using JS with Standard Firmata and Johnny Five framework
- Using Sysfs on boards with Linux
- Using NodeJS on boards with Arduino-IO library and Johnny Five framework

1) *AL and Arduino IDE*

All Arduino boards including compatible boards like Galileo, can be programmed in Arduino IDE which is used for program development in AL. IDE creates project with skeleton code, a starting point for development of user code arranged and saved in unit called sketch. It performs syntax check of sketch code, compiles and uploads compiled code to selected board using USB connection.

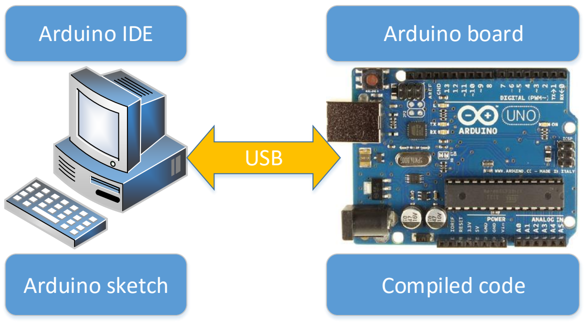


Fig. 1. Arduino board connected with PC

IDE comes with large number of example sketches covering wide range of possible applications with sensors, actuators, displays, communication, extension boards, SD cards and support for some specific boards like Robot, Esplora and others.

Fig. 1 illustrates the basic connection of Arduino board with PC using USB cable. Once the compiled program code is uploaded to board flash memory, program execution automatically starts and functions as independent system.

After start of the program, data from Arduino board can be obtained using monitor program available from IDE which displays data sent from running program. Data display from monitor program can be a very useful debugging tool.

User developed custom sketches can include many libraries available from IDE that support various functionalities of Arduino boards:

- SPI
- Ethernet
- WiFi
- GSM
- TWI - Wire
- UART – software serial

Some libraries are supplied from a sensor manufacturer, like for instance DHT library for Digital Temperature and Humidity sensor [14] from AdaFruit [15].

Advanced boards that have besides MCU also the on board Linux system, like Yun, can communicate with Linux system from sketch code using Bridge library from AL.

2) *JS with Firmata and JohnnyFive library*

Java Script (JS) is best known as language for web clients scripting, providing functionalities for web pages. JS is quickly gaining popularity and spreading to web server platforms due to appearance of NodeJS [16] which is JS platform based on Chrome V8 JavaScript engine. NodeJS offers use of the same language and technology both on web clients and servers. Besides, NodeJS on web servers offers good performance and execution of asynchronous JS code. NodeJS quickly spreads to various computing areas with development of NPM [17] (Node Package Manager) modules that are installed in NodeJS and extend available functionalities. JS as a language of web enters the fields of IoT and robotics with development of Johnny-Five [18] (JF) JS programming framework. As the Arduino boards cannot be directly programmed in JS, the matching software layers are required consisting of the mentioned JF framework and Arduino StandardFirmata [19] (ASF) library. ASF is installed in Arduino board flash memory as sketch implementing the Firmata protocol [20] (FP) for communication between MCU and application on host computer. JS code running on NodeJS platform on host PC using JF framework communicates with Arduino board over USB cable and FP. Such a configuration is presented in Fig. 2. JF currently introduces some limitations when accessing boards with advanced features comparing to standard Arduino Uno board. 12 AI pins of Arduino Due, comparing to 6 of Uno, can be accessed by JF but

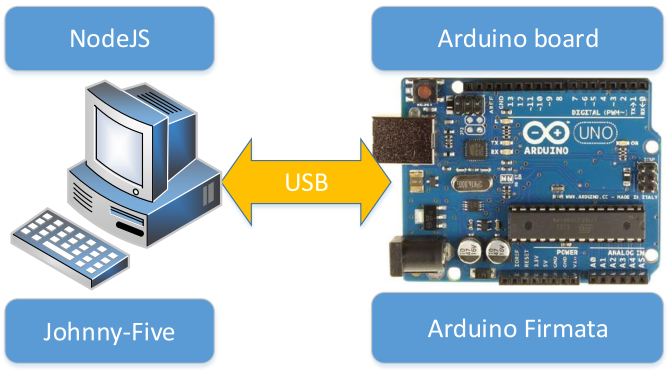


Fig. 2. Arduino programming with JS

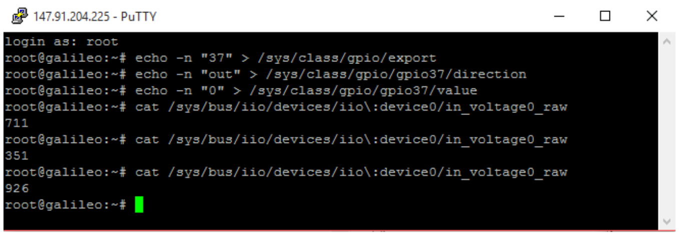
with analog resolution of only 10 bits as is for Uno, although Due has analog resolution of 12 bits. Programming elegance of JF has to be paid with some limitations when accessing advanced boards features, which makes JF use acceptable only for basic boards. Further development of JF may bring improvements.

3) Programming with Sysfs on boards with Linux

Advanced boards that have an extra processor in addition to MCU can be programmed with standard Arduino IDE in AL, the same as boards without extra processor, and with tools generally available from Linux operating system (OS). Additional processor comes with significantly larger memory and higher operating frequency. Besides higher processing power, Linux computer with MPU usually also comes with integrated network adapters and memory card readers. Basic difference between MCU and MPU is a real time operation which is possible with MCU, but not with MPU. MPU systems on Arduino boards host some Linux distribution like Linino which is based on OpenWrt on Yun board, and Yocto [21] Linux distribution for Intel Galileo board. The basic Linux distributions are preinstalled on Yun and Galileo boards. Galileo board supports installation of other Linux distributions from Intel, allowing that way for installation of the latest version which also has additional packages. Alternate Linux distribution for Galileo is created from the image to micro SD card, and allows for booting the new version if the SD is inserted to board, if not, factory default will be active. Yun can also update Linux from SD card.

On board Linux system can be reached in various ways. Yun has a so called "bridge", which is a library for passing information between processors and allows reaching Linux from IDE sketches and vice versa. There are two IDEs for Galileo Linux programming, the Intel XDK IoT Edition [22] primarily for NodeJS programming in Linux and Intel version of Eclipse IDE [22] for Java and C++ programming. IDEs also support and enable project files upload to board.

Besides using IDEs, on board Linux system can be reached with serial communication and SSH client like Putty [23] providing access to Linux from terminal. Yun can be reached through built in network adapters, while Galileo also offers connection through RS 232 serial cable which is convenient for initial Linux system configuration. File transfer with Linux system can be accomplished with SCP (Secure CoPy) clients based on SSH, like WinSCP [24].



```
login as: root
root@galileo:~# echo -n "37" > /sys/class/gpio/export
root@galileo:~# echo -n "out" > /sys/class/gpio/gpio37/direction
root@galileo:~# echo -n "0" > /sys/class/gpio/gpio37/value
root@galileo:~# cat /sys/bus/iio/devices/iio\:device0/in_voltage0_raw
711
root@galileo:~# cat /sys/bus/iio/devices/iio\:device0/in_voltage0_raw
351
root@galileo:~# cat /sys/bus/iio/devices/iio\:device0/in_voltage0_raw
926
root@galileo:~#
```

Fig. 3. Configuration and reading of A0 with Sysfs

Galileo GPIO ports can be accessed from Linux shell using Sysfs interface [25]. Sysfs is a virtual file system in Linux which exposes devices as files, and operations with devices as file operations which is presented in Fig. 3.

In order to be used, GPIO port has to be exported to Sysfs. For analog input A0, the GPIO port 37 which controls the multiplexer channel 0 is exported as:

```
echo -n "37" > /sys/class/gpio/export
```

After exporting, the direction is specified:

```
echo -n "out" > /sys/class/gpio/gpio37/direction
```

A0 input is connected to ADC AD7298 circuit used in Galileo with:

```
echo -n "0" > /sys/class/gpio/gpio37/value
```

After setting up, analog value from A0 can be obtained with:

```
cat /sys/bus/iio/devices/iio\:device0/in_voltage0_raw
```

Fig. 3 displays previous commands and results of 3 readings of A0 connected to function generator. In a similar way, digital IO can be accessed and controlled, including PWM. If executed from within a program in Linux, these commands can be regarded as a GPIO API.

Fig. 4 illustrates concept of program development with Sysfs that requires no special IDE for Arduino or Galileo.

Program using Sysfs can be developed on host PC, then uploaded to Galileo and started from shell using SSH based clients. It is also possible to develop program on Galileo only, by creating all files from shell.

4) Hosting NodeJS on board

Boards with Linux system can host applications requiring use of GPIO. NodeJS applications hosted on board and using JF framework have many advantages comparing to NodeJS applications hosted on PC and managing MCU systems via USB and Firmata protocol.

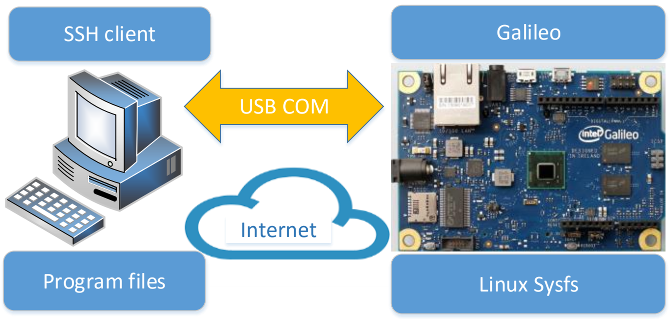


Fig. 4. Program development with Sysfs

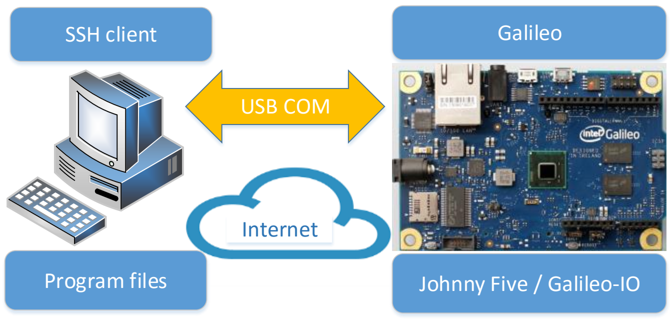


Fig. 5. Program development with JF / Galileo-IO

Board that hosts JF based application can operate independently, not requiring extra PC. Consequently, applications based on single device tend to be more robust in some situations, as the automatic restart after power failure for instance. It is also expected that higher signal rates will be possible by omitting USB and Firmata protocol communication.

As the JF framework was designed to work with Firmata protocol, additional software layer with same interface is required when executing on the host board. That additional software layer is Galileo-IO [26], an IO plugin for JF and also standalone module, which can be used independently from JF in NodeJS applications. Fig. 5 illustrates concept of onboard program development with JF / Galileo-IO.

Besides JF and Galileo-IO, the mraa [27] library can be used for access to GPIO on Galileo boards from NodeJS, Python and Java. Intel XDK IoT IDE uses mraa and provides complete environment for development and upload of NodeJS projects to Galileo boards including Linux shell access. Fig. 6 illustrates configuration with Intel XDK IoT IDE and mraa library.

IV. Overview of Architectures with Arduino Boards

A. *Requirements for system design*

Arduino boards can be used for various tasks requiring interaction with environment. Due to small dimensions, low power consumption, significant computing potential, ability to read and generate analog and digital signals, possessing network connections, these boards are convenient computing platforms for environment monitoring and control of artificial systems. As the environment is complex, characterized by many physical quantities, development of corresponding applications for monitoring and control requires adequate design in broader sense that consists of standard phases of software design or software life cycle phases.

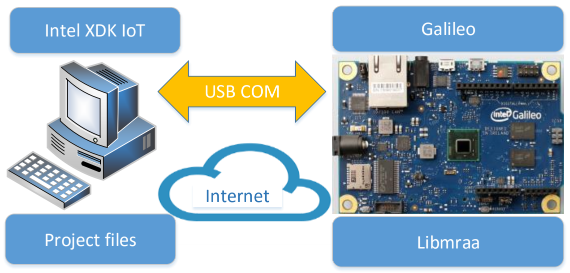


Fig. 6. Program development with Intel XDK IoT IDE and mraa library

Analysis of the environment and system to be controlled creates model that includes physical quantities to be measured and aspects to be controlled. Physical quantities are of analog nature with continuous values requiring use of adequate measurement converters or sensors which output standard electrical signals measured by analog inputs on boards which can also generate analog and digital signals for control of devices and actuators, and communication with other systems.

Depending on complexity of required task, systems with various configurations can be designed. Configuration aspects that are of interest for this paper are choice of the board, board programming, communication with other systems, single or multi board system, and scalability of the monitoring and control system. Configurations of measurement and control devices attached to GPIO board pins will not be considered.

B. *Single board configuration*

Board with uploaded sketch and power supply operates as independent device. Fig. 1 illustrates such simple configuration. PC is required for sketch development and upload to board, but after that board functions on its own. Computing resources and GPIO pins on one board may be enough for monitoring and control application, and if the task of the board is to serve as autonomous controller or regulator for some system, such simple configuration is quite adequate. Possible problem with such simple configuration may arise if the communication and transfer of acquired data to other computer systems is required for processing and storage by applications working on other devices. In some simple cases, the LCD device for display attached to board as extension may be enough, but in general for data transfer, a connected PC with running application for data processing and storage is necessary. Connected PC can further provide for acquired data transfer and also for extra data processing and return of processed signals for control purposes.

C. *Single board with network communication*

Utility of a single board configuration significantly depends on communication with other computing devices required for acquired data processing and storage. Boards directly support various wired serial communications such as UART, SPI, TWI and USB. Real tasks for monitoring and control frequently require remote placement of sensing devices attached to board, with board being also placed near points for monitoring and control. In such cases an adequate communication channel between board and other computing device must be provided over arbitrary distance. A good solution candidate is existing computer network or an extension of network being quite satisfactory with the exception of some rare extreme conditions, such as extreme electrical interference, extreme temperatures, pressures, chemically active agents and similar that require advanced solutions. Fig. 7 illustrates single board configuration using Ethernet adapter called shield in Arduino terminology.

Instead of Ethernet shield, WiFi or GSM shields could be used depending on specific conditions that may favor one over the others. Selection of shield also affects the sketch with code, as it must programmatically support used shield in addition to acquisition and control tasks. Client symbol in Fig. 7 stands for any client using the board, being it a software or a human client using browser.

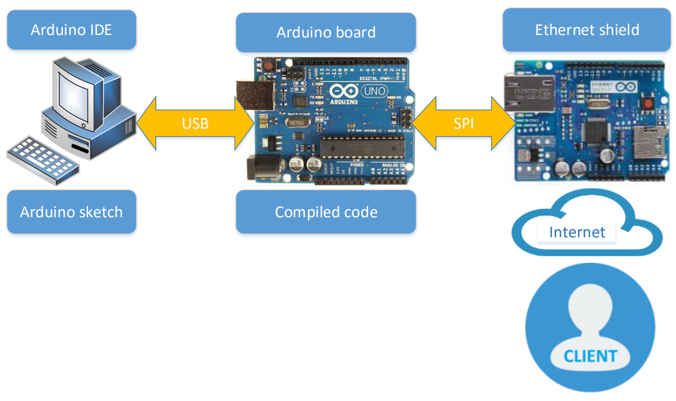


Fig. 7. Single board configuration with Ethernet adapter

D. *Single board with Ethernet shield and web server*

Simple configuration with a single board and Ethernet shield can provide basic and simplified web access due to limited memory of microcontroller board. Provided web access is best used for communication with remote client for the purposes of sending acquired data from sensors at client request, or receiving control data from client. In this configuration the client is software implemented and acts as a bridge between Arduino board and main web server on a PC computer. User as the main client directly communicates with web server on PC which can be arbitrarily complex. Fig 8 illustrates such configuration. Although any web server can be used, NodeJS is convenient for good performances, and using JS for code on software client, server and web page.

E. *Single board with Firmata and Johnny-Five*

Configuration in Fig. 8 provides web access to Arduino board using PC based server and network enabled board. In general case it requires two computers, one for programming, and other for web, although the same computer could be used for both tasks, with web server being also used for specific task of board programming in that case. Board being programmed in AL deviates from JS programming paradigm. Modified configuration with Firmata code on board allowing programming with JS exclusively is illustrated in Fig. 9. That configuration also does not require network adapter, as the web server is used for access to Arduino board. Formally, the configuration in Fig. 9 is equivalent to configuration in Fig. 8 from the aspect of client. Firmata configuration is with less devices, but AL configuration has two important advantages.

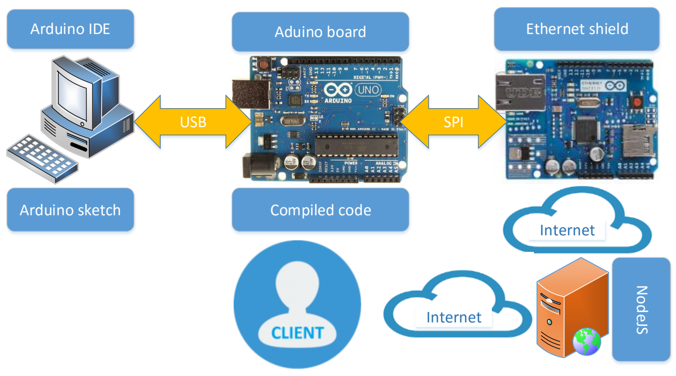


Fig. 8. Arduino with web client access

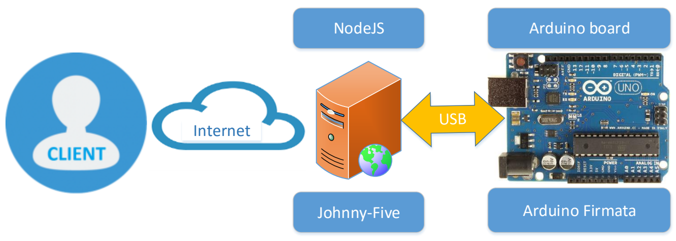


Fig. 9. Single Arduino board with Firmata and Johnny-Five

First advantage is that Johnny-Five library poses various limitations for full use of potentials of some boards, and does not cover some specific sensors for which AL libraries exist. Second advantage is system scaling, when more boards are added to the system. Adding of new AL boards might require some changes of web server software, while adding of Firmata boards would require new USB hardware connections and adding of software for serving each USB connected board.

All USB connected boards must be placed within few meters distance from server, while for AL board with network adapter distance of physical placement is irrelevant.

F. *AL programmed Galileo board*

Galileo board although significantly different from boards with controller is software and pin to pin compatible with Arduino UNO, so it can be connected, programmed and used the same as Arduino UNO board. Although more expansive than UNO, the price of UNO plus the price of network adapter is comparable to price of Galileo board, suggesting that Galileo board may be a better investment than Arduino board with network adapter, bearing in mind Linux OS on Galileo board. Fig. 10 illustrates equivalent configuration of Galileo board with configuration in Fig. 7. Important advantage of Galileo configuration is a significantly larger memory.

G. *Galileo Linux programming*

Full potential of Galileo board can be unleashed with programming for its Linux platform. Fig. 11 illustrates the generic configuration for Galileo Linux programming. Variants for Galileo Linux programming may differ depending on used development IDE, programming technology and used support libraries. As mentioned in previous chapter, three basic configurations for Galileo Linux programming were considered:

- Sysfs Linux programming, Fig. 4
- NodeJS with JF / Galileo-IO, Fig. 5
- NodeJS with Intel XDK IoT based on mraa library, Fig. 6

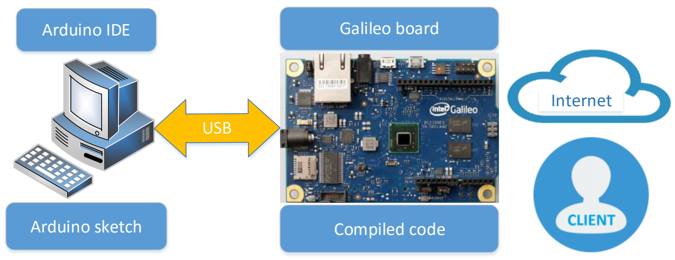


Fig. 10. Galileo board used as UNO board with network adapter

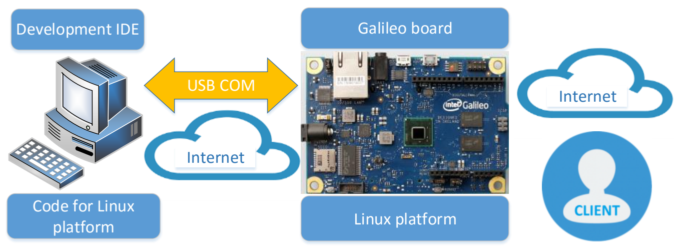


Fig. 11. Configuration for Galileo Linux programming

Each of the configurations can further have its own sub variants.

Generic programming configuration in Fig. 11 comprise three configurations with added client. In order to obtain optimal solution for given requirements, mentioned programming approaches can be combined.

H. *Web Interface for Arduino board*

Web communication between Arduino board and software client can be standardized with definition of appropriate interface. Fig. 12 illustrates the concept of web interface [28]. Structure of web interface is illustrated in Fig 13. Web interface specifies allowed commands and operations depending on configuration of the board.

Main web server communicating with Arduino board in Fig. 12 is implemented as NodeJS on Galileo board suggesting that whole hardware configuration for monitoring and control can consist only of Arduino and compatible boards, without PC computers, with no serious limitations. Arduino board with Ethernet shield plays a role of the so called acquisition web server that performs required operations with attached sensors and devices and sends acquired data, upon received request. Main web server has software implemented client that sends requests to acquisition server as a result of user requests to main web server. Galileo board implementing main web server has more memory and processing capacities comparing to Arduino controller boards. Besides hosting a main web server, Galileo board can also be used for measurement and control.

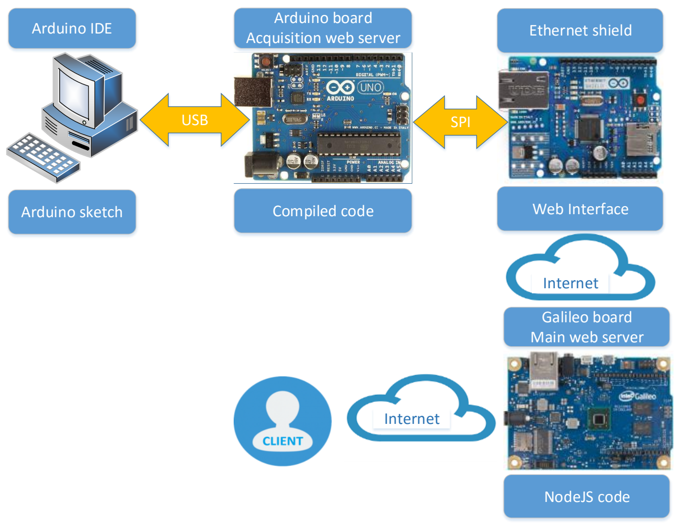


Fig. 12. Concept of a web interface

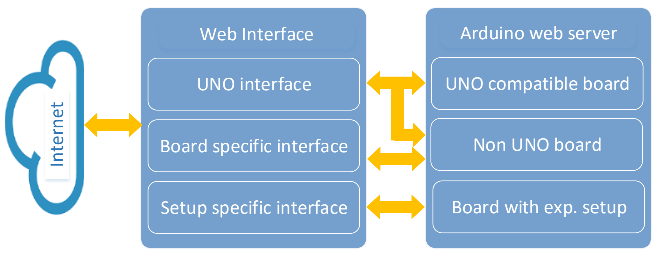


Fig. 13. Web interface structure

Web interface in Fig. 13 specifies three possible modes of communication depending on board being communicated with. The first mode named “UNO interface” is the most general, as it specifies standard communication with Uno board that other boards are compatible with. Specified communication includes IO pins and if required, access to UART, SPI and TWI communication can also be specified. Second mode named “Board specific interface” in addition to first, specifies communication with particular board having more advanced configuration than Uno. Separate interface is required for each non UNO Arduino board. Third interface mode is the most specific, as it specifies communication with actual setup configuration connected to board pins, the sensors and actuators.

Arrows in Fig. 13 signify that “UNO interface” as more general interface can also be used with more specific non UNO boards, as they are UNO compatible.

I. *Scaling of Arduino based system*

Various considered configurations with Arduino and compatible boards may offer adequate solution for measurement and control tasks on systems which can be located at any place provided with adequate computer network infrastructure. Besides network requirement, systems to be monitored and controlled should be concentrated with components relevant for measurement and control being relatively close and within reach of cables connecting sensors and devices with boards. For systems consisting of components distributed at greater distances, and with large number of measurement and control points, an adequate extension of considered basic configurations is required, that will follow and resemble the configuration of the system.

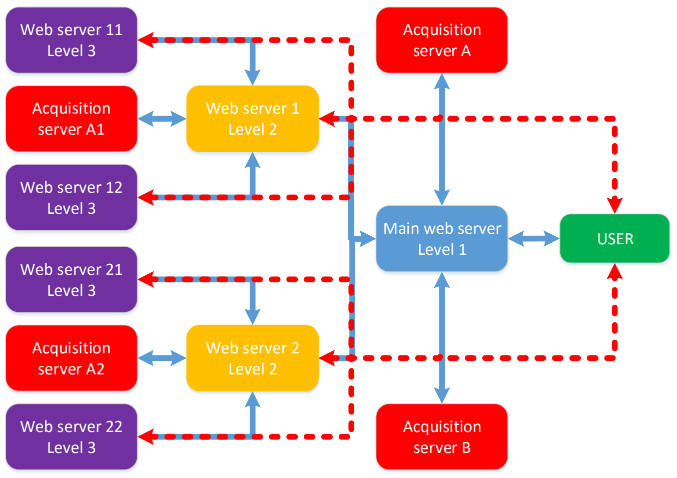


Fig 14. Scaling of measurement and control system

Fig. 14 illustrates the schematic of extension structure based on considered elementary configurations. Remote web user represented as green block directly access the main web server which is on the level 1 of extended configuration. Main web server at level 1 is a direct entry point to scalable monitoring and control system that can be extended as required.

Configuration only with main web server and acquisition server A in Fig. 14 resembles configurations in Fig. 8 and 12. Blue solid lines designate primarily computer network connections which could also be USB as well, in a special case when connecting web server at some level with acquisition server. A number of remote acquisition servers which are boards with network adapters, can be attached to main web server. Acquisition servers A and B on Fig. 14 represent acquisition servers directly connected to configuration level 1 with possibility of adding more servers on the same level, C, D, etc. as required. If the structure of distributed system is of hierarchical type, then it may be more convenient to add new acquisition servers not directly on level 1, but to also add web servers which are called by software clients from main web server. Web servers 1 and 2 in Fig. 14 are examples of web servers on level 2. Servers at level 2 can be accessed by software clients from main web server, but could in principle be also reached directly from the user, which is designated by dotted red lines. Dotted red lines designate alternate direct access by the user to web servers on the lower configuration levels. Blue lines are regular access paths that follow and respect hierarchy, and are intended to be used by ordinary regular users that access the system in a safe and prescribed way. Red dotted lines are direct paths to lower levels that some users can be authorized for, in order to use it for special purposes, such as maintenance, troubleshooting, software configuration changes that can require such direct access. Designation of servers in Fig. 14 follow hierarchy logic, and in addition, web servers at the same level have the same color, while all acquisition servers are red. Hierarchy configuration in Fig. 14 can be extended with arbitrary number of web servers, configuration hierarchy levels, and acquisition servers as required.

V. CONCLUSION

SBCs for acquisition and control based on single microcontroller or with additional processors are quickly becoming important platforms for various monitoring and control tasks, due to small dimensions, reliability, enough computing power, large existing software support, easy integration with other larger computer systems and low price. Arduino family of SBCs is particularly interesting for its large number of users, variety of boards, board extensions, programming modes, software support and influence on products of other manufacturers resulting in Arduino compatible boards. Three representative boards were selected for consideration and comparison according to hardware characteristics and programming modes. Boards with additional processor are particularly interesting as they allow various programming modes as support for acquisition and control. Various boards and programming modes result in various possible combinations called configurations in this paper. Single board configurations were considered and compared according to available programming modes with particular attention to NodeJS and libraries for NodeJS supporting Arduino boards. Single board configurations are applicable for certain tasks with real world systems at single location. Applications for remote and distributed systems require use of network architecture obtained as direct extension and generalization of considered basic single board configurations and concept of web interface. Presented network configuration can be further extended with arbitrary number of network nodes and hierarchical levels. Obtained results are directly applicable to configurations with nodes being some other, non-Arduino boards.

ACKNOWLEDGMENT

Work on this paper was partly funded by the SCOPES project IZ74Z0_160454 / 1 “Enabling Web-based Remote Laboratory Community and Infrastructure” of Swiss National Science Foundation

REFERENCES

[1] “Arduino” [On line] Available: http://arduino.cc

[2] “Arduino” [On line] Available: http://arduino.org

[3] “BeagleBone Black” [On line] Available: http://beagleboard.org/BLACK

[4] “Raspberry Pi” [On line] Available: https://www.raspberrypi.org/

[5] “RIoTboard” [On line] Available: http://riotboard.org/

[6] “PandaBoard” [On line] Available:http://pandaboard.org/

[7] “OLiMEX” [On line] Available: https://www.olimex.com/

[8] “Arduino boards” [On line] Available: http://www.arduino.org/products

[9] “MIPS processors” [On line] Available: http://imgtec.com/mips/

[10] “Linino” [On line] Available: http://www.linino.org/

[11] “OpenWrt” [On line] Available: https://openwrt.org/

[12] M. Švaljek, Arduino Succinctly, Syncfusion Inc., 2501 Aerial Center Parkway Suite 200 Morrisville, NC 27560 USA, 2015, http://www.syncfusion.com/

[13] A. D’Ausilio, Arduino: A low-cost multipurpose lab equipment, Behavior Research Methods, vol. 44, 2, pp 305-313, 2012, http://dx.doi.org/10.3758/s13428-011-0163-z

[14] “DHT 22” [On line] Available: https://www.adafruit.com/products/385

[15] “AdaFruit” [On line] Available:https://www.adafruit.com/

[16] “Node.js” [On line] Available: https://nodejs.org/en/

[17] “Node Package Manager” [On line] Available: https://www.npmjs.com/

[18] “Johnny-Five” [On line] Available: http://johnny-five.io/

[19] “Standard Firmata” [On line] Available: https://www.arduino.cc/en/Reference/Firmata

[20] “Firmata protocol” [On line] Available: http://firmata.org/wiki/Main_Page

[21] “yocto Linux” [On line] Available: https://www.yoctoproject.org/

[22] “Intel XDK IoT Edition” [On line] Available: https://software.intel.com/en-us/iot/software/ide

[23] “putty” [On line] Available: http://www.putty.org

[24] “WinSCP” [On line] Available: https://winscp.net/eng/download.php

[25] “Linux Sysfs” [On line] Available: http://www.malinov.com/Home/sergey-s-blog/intelgalileo-programminggpiofromlinux

[26] “Galileo-IO” [On line] Available: https://github.com/rwaldron/galileo-io

[27] “mraa” [On line] Available: http://iotdk.intel.com/docs/master/mraa/

[28] C. Salzmann, S. Govaerts, W. Halimi, D. Gillet, The Smart Device Specification for Remote Labs, REV 2015, 25-27 Feb. 2015, Bangkok, Thailand