Putting Cloud 9 IDE on the Wheels for Programming Cyber-Physical / Internet of Things Platforms
Providing Educational Prototypes

Andrej Skraba[1], Vladimir Stanovov[2], Eugene Semenkin[2], Andrej Kolozvari[1], Radovan Stojanovic[3] and Davorin Kofjac[1]

[1]University of Maribor, Cybernetics & Decision Support System Laboratory, Faculty of Organizational Sciences, Kidriceva Cesta 55a, Kranj, Slovenia

[2]Siberian State Aerospace University, Computer Science and Telecommunications Institute, 31, Krasnoyarsky Rabochy Av., Krasnoyarsk, Russian Federation

[3]University of Montenegro, Faculty of Electrical Engineering, Dzordza Vasingtona bb., Podgorica, Montenegro

Keywords: Cyber-Physical System, Internet of Things, Iot, Computer Cloud, HTML5, JavaScript, ECMAScript, Node.Js, Robotic Platform, ROS, Robot Operating System.

Abstract: The paper describes the development of educational Cyber-Physical Robotic Platforms, remotely controlled via cloud technologies. The platform is implemented using Parallax ActivityBot kit (only for mechanical part), controlled by Arduino, and includes a Quad core ARM-processor Mini PC MK802 V5 LE running Xubuntu Linux. The programming on the developed platform could be performed in JavaScript and HTML, providing the web interface for controlling the system through Wi-Fi. Programming the platform is possible through the Cloud9 IDE web interface, enabling rewriting the code or running different programs by the user. Four equal platforms were implemented to address the need for the easily accessible educational Cyber-Physical Robotic Platforms / Internet of Things hardware for students and tested in the experiment.

1 INTRODUCTION

When engaging the students to Cyber-physical & Internet of Things (IoT) robotic platforms development one usually faces many problems connected to the difficluties with low-level programming required for the hardware. Besides the problems of different software solutions which are needed for the platforms to work might also make things harder. In this paper we present our implementation of a simple Cyber-physical & IoT platform, which uses a fully-functional Mini PC with Linux onboard, Arduino, Node.js server and cloud9 IDE. Programming in JavaScript/ECMAScript is much easier than programing in C, C++ or Java and other languages, and enables designing a straightforward web interface. Important aspect of such approach is, that user interface could easily be developed while being reliable and thoroughly tested. Once writen such software could be run on any hardware as long as it supports html / JavaScript (ECMAScript).

The resulting robotic platforms represent a Cyber-physical system by integrating the physical processes with processes of software and communications. Cyber-physical systems connect together several disciplines; computers, communications, software and mechanical engineering. Regarding the term “Cyber”, we would suggest, that the platform could easily be controlled, monitored and programmed on the internet (Kraijak and Tuwanut 2015) with easy possiblity to participate in e.g. social networks, harvesting the cloud solutions, such as speech recognition (Škraba et al., 2014, 2015b), motion tracking (Škraba et al., 2015a) etc.

In the next section we will describe the hardware architecture of the Cyber-Physical Robotic Platform (CPRP), and in the second part we will move to the software part. The main component of the robotic platform is a Mini ARM A17 processor PC RKM V5 LE with a Quad-core Rockchip RK3288 (RKM-V5 2014.11.19) processor running with 2Gb RAM and Xubuntu Linux (Ubuntu, 2016) version 14.10.

Skraba, A., Stanovov, V., Semenkin, E., Kolozvari, A., Stojanovic, R. and Kofjac, D.
Putting Cloud 9 IDE on the Wheels for Programming Cyber-Physical / Internet of Things Platforms - Providing Educational Prototypes.
DOI: 10.5220/0005985204280435
In *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2016) - Volume 2*, pages 428-435
ISBN: 978-989-758-198-4

The Mini PC is connected to Arduino MEGA (Arduino, 2016) based on ATMega 2560 microcontroller and the Logitech C270 HD 720p webcam through USB. The operator can connect to the platform through Wi-Fi using local address or from the WAN.

2 SYSTEM ARCHITECTURE

2.1 Hardware Architecture

The main element on the platform which is shown in Figure 1 and 2 is ARM based MK802 V5 LE Mini PC with Xubuntu Linux installed (1). The Quad Core MK802 V5 LE has significant processing power and is a great enhancement form the previous versions of MK802 versions and GK802. One has to mention, that the power consumption of MK802 V5 LE is around 0.2A with fully operational Linux running.

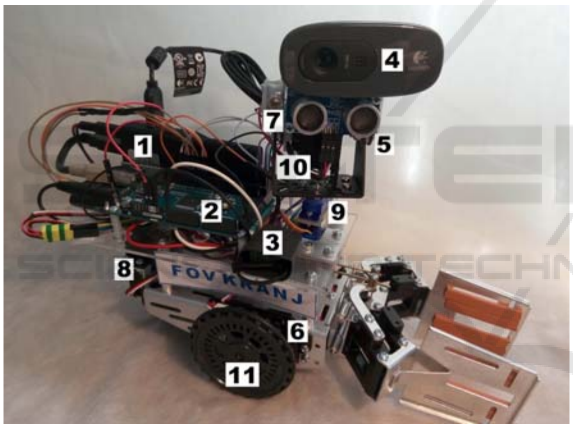


Figure 1: Side view of cyber-physical robotic platform.

The boot-time is also measured in seconds which is significant improvement comparing to previous versions of MK802. The MK802 V5 LE is connected to the Arduino MEGA 2560 microcontroller over the USB (MK802 V5 LE has two USB ports). The power to the Mini PC (1) is provided using a Pololu step-down voltage regulator (13), giving exactly 5 volts output, which drains the current from five AA-sized NiMH accumulators (12). The typical voltage of five accumulators is around 6 volts. The servo motors, as well as the DC motors receive power directly from the batteries. The DC motors are driven by the TB6612FNG H-bridge (3) connected to the Arduino (2). The platform has two infrared sensors (6) to read the frequency of each wheel, Parallax Ping))) (5) ultrasonic sensor 28015 is mounted under the camera (4) to measure the distance, together with a laser pointer (7). CPRP includes two DC motors (15, 16) for the wheels (11), one for each of the two wheels, one servo motor for triggering the gripper (8) so that the platform is capable of grabbing things, and two servos for adjusting the camera tilt, in horizontal (9) and vertical (10) axis. Camera position control mechanism was added in order to provide better ability to monitor the surrounding without the need to move entire platform. This contributes to the lower power consumption and better durability of the platform.

Figure 3 shows the hardware architecture of the system, including connections between the devices. The MK802 V5 LE Mini PC is the heart of the system with Xubuntu installed it provides a solid performance running needed software stack and communicating with Arduino MEGA 2560 Microcontroller over USB. WebCam is also plugged directly to the Mini PC providing the live video stream from the CPRP. The Voltage regulator provides +5V to Mini PC. Arduino is powered by +6 V. CUR. SENS. is the current sensor, required to measure the power consumption of the system. Arduino controls the H-Bridge needed to run Left and Right DC motors. Optical encoders on the wheels of Activity-Bot are used as the feedback about the wheels speed. By the encoders the precise control is enabled giving the platforms the ability to perform complicated navigation tasks. Here DIG means digital signal, ANA – analog signal, PWM – pulse-width modulation. ENC. L and ENC. R are the left and right encoders respectively. The Mini PC could be accessed via Wi-Fi connection. The web router provides the interface for the smart devices, such as phones, tablets, TVs and PCs. When exposing the IP of the Mini PC to the outside world, the platform could be accessed from the internet. It is important, that not only the control and monitoring of the CPRP could be performed but also the programming, without the need to install software on the client side. If you want to program ROS, for example, all you need is a phone. By connecting the CPRP to the cloud, the system could harvest the cloud technology for example for speech recognition (Škraba et al., 2014, 2015b) etc. The typical power consumption is around 200mA when the robot is not moving.

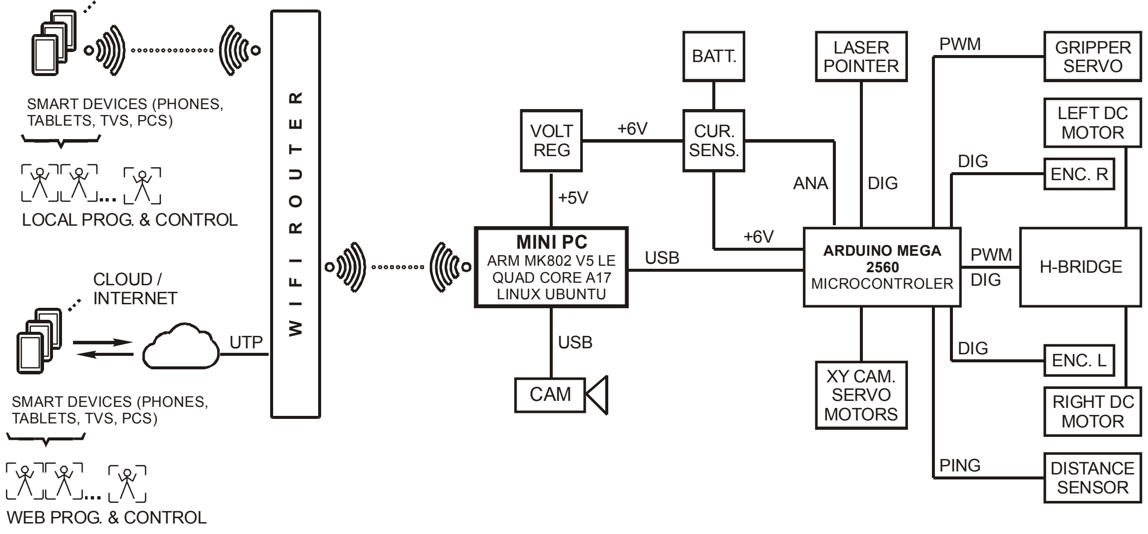


Figure 3: System hardware architecture.

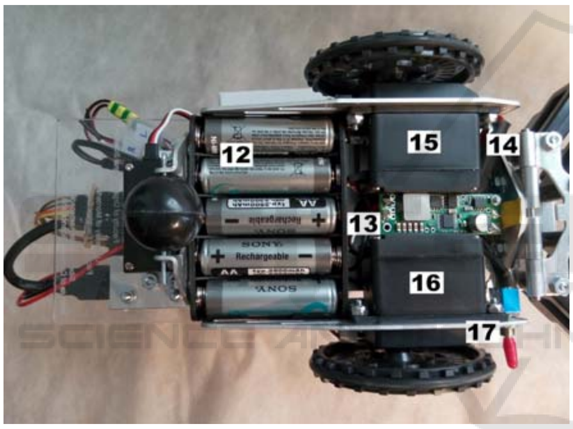


Figure 2: Bottom view of cyber-physical platform.

2.2 Software Architecture

The program part includes the pre-installed Xubuntu Linux system (based on Ubuntu Linux) installed on the MiniPC, Node.js (node.js, 2015) for running the server, and the following node packages: socket.io (socket.io, 2016), firmata (firmata 2016) and johnny-five (johnny-five, 2016). The last package is required to handle the ultrasonic sensor, as this functionality is not included with standard firmata. On the Arduino the firmata C library is installed in order to enable communication between Arduino and software on the Mini PC, in our case over USB. The video streaming is performed with the mjpg-streamer library (mjpg, 2016). The mjpg-streamer saves the jpeg-file on a disk every 200 ms, and then this image is transferred to the user's browser via node.js server.

The Node.js server, providing the robot control interface is started at port 8080, via socket.io package. Node.js enables us to run JavaScript / ECMAScript code on the server side using the Google V8 engine. It is asynchronous event driven framework designed to enable building of scalable network applications. The johnny-five JavaScript Robotics programming framework and firmata are used to communicate with the Arduino, through USB connection. The Arduino has pre-installed PingFirmata, which allows using Parallax Ping))) sensor.

For programming the platform the Cloud9 IDE (Cloud9, 2016) web interface has been installed, so that the operator can run different programs or modify the code without connecting peripheral devices, such as keyboard and monitor, and even without seeing the platform. This enables new opportunities in the field of cloud computing. In our case, we have Cloud9 installed on wheeled platforms and users could program them remotely without the need to install any additional software, the web browser on the client side is sufficient. Cloud9 is open source online integrated development environment.

In Figure 4 it is important to note, that the entire platform is exposed to html5 / JavaScript / ECMAScript for programming and control. Form the user side only JavaScript is needed to program advanced hardware solutions and even ROS (Codd-Downey R. and Jenkin, 2015). There are certainly limitations in latency, however for the educational and even professional purposes (Škraba et al., 2015b, Jiang et al., 2015) the proposed platform could be fully functional. By the pace of the network hardware development the platform could be also suitable for most demanding applications therefore it would be worth to examine the capability of JavaScript / ECMAScript for hardware development (10).

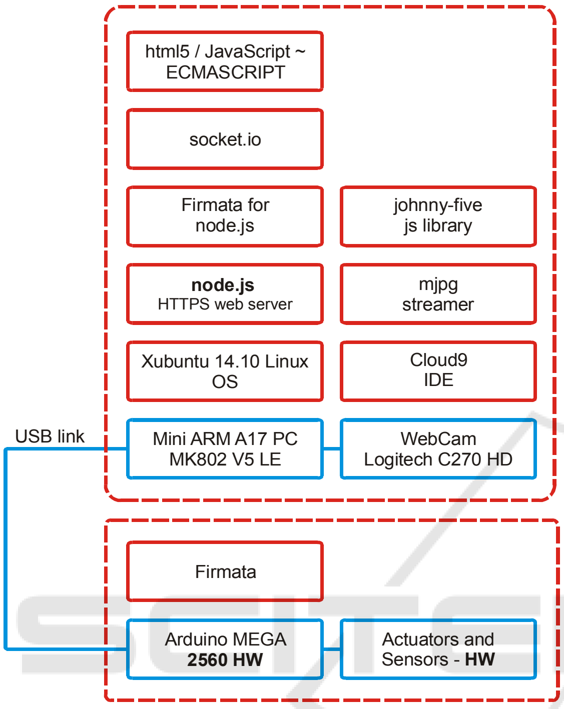


Figure 4: Software stack of the system.

Figure 5 shows the concept of programming several CPRP over the web browser. In our case, we have four CPRP with installed Cloud9, node.js, firmata and Arduino. Mini PC is providing Wi-Fi connectivity to the Wi-Fi router. User, from the client side access all four platforms from the web browser. This, as mentioned, significantly reduces the effort of handling hardware, such as keyboards, mice and monitors. User can only with Alt-Tab switch between four platforms and work with all of them from one computer. When developing ROS, it is important, that the hardware is easily accessible. Besides, if one uses Chrome, the superior debugging capabilities are available for JavaScript. Node.js has also good capabilities for debugging code on the client side.

When all four platforms are active, they are accessible through the Cloud9 web interfaces. Figure 6 shows the interfaces accessed from a local network, with each robot having addresses 192.168.1.128-131 at port 8181. One could observe four opened web-browsers, marked with numbers from 1-4. If one would like to access the CPRP, one should enter the address in the address line of the web-browser, for example:
http:// 192.168.1.128-131:8181
and the user interface of Cloud9 would be accessed as shown in Figure 6. Therefore, only by entering the IP address of the platform, the fully functional IDE Cloud9 is invoked and the programming of the hardware CPRP could be started, by easily accessing not only one but several CPRPs.

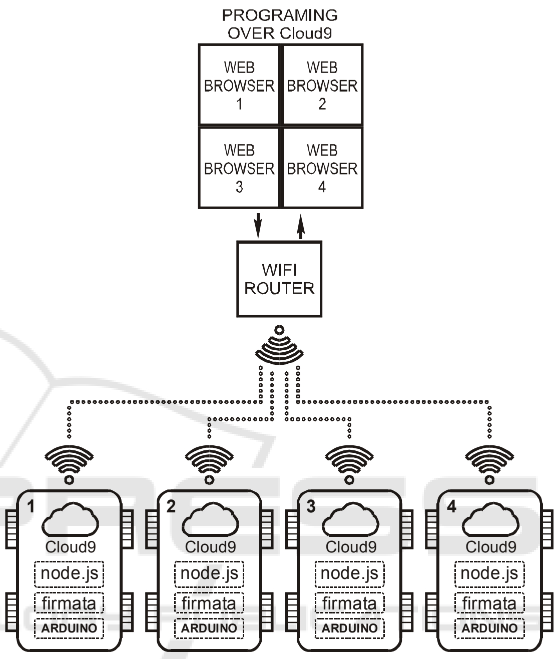
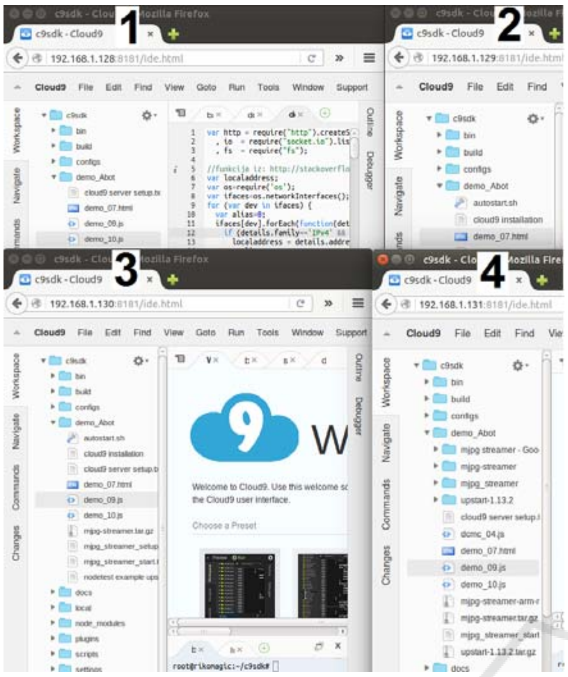


Figure 5: Programming of four cyber-physical platforms over web browser with Cloud9 IDE.

Programming the platforms can be performed from both local net and internet, by several users at the same time, as this functionality is implemented in Cloud9.

In figures 7 and 8 the scripts for the automatic start of the Cloud9 server are presented. Each Cloud9 server starts automatically on power up using upstart, which runs a simple script in c9sdk folder, so there is no need to connect peripheral devices and input any commands to access the web interface. The interface can be accessed by any smart devices, i.e. smartphones, tablets, PCs, laptops and even smart TVs. This easies the development process and requires less equipment.



```
var http = require("http").createS
  , io  = require("socket.io").lis
  , fs  = require("fs");

//funkcija iz: http://stackoverflo
var localaddress;
var os=require('os');
var ifaces=os.networkInterfaces();
for (var dev in ifaces) {
  var alias=0;
  ifaces[dev].forEach(function(det
    if (details.family=='IPv4' &&
      localaddress = details.addre
```

Figure 6: Four Cloud9 interfaces open with web browsers.

```
description "Cloud9 server"
env HOME="/home/cloudsto"
env USER="cloudsto"
start on filesystem or runlevel [2345]
task
exec /home/cloudsto/c9sdk/autostart.sh
stop on shutdown
```

Figure 7: Upstart script, in /etc/init/nodetest.conf.

```
#!/bin/bash
cd
cd c9sdk
sleep 10
node server.js --listen 192.168.1.128 -p 8181 -a
username:password
```

Figure 8: Script to start Cloud9 server in file /home/cloudsto/c9sdk/autostart.sh.

The wheels frequency is controlled automatically via PWM values for each motor independently by the PID controller. We have used the standard implementation for PID and adjusted the parameters by hand to achieve the desired smoothness and swiftness of movements. Figure 9 shows the JavaScript code realization.

```
function GetPWMfromPIDLeft(desiredFLeft,frequencyLeft)
{
    if (IntegralCounterLeft < 3) {
        ErrorLeft.unshift(desiredFLeft - frequencyLeft);
        IntegralCounterLeft++;
    }
    else {
        ErrorLeft.pop();
        ErrorLeft.unshift(desiredFLeft - frequencyLeft);
    }
    if (IntegralCounterLeft == 1) {
        PWMleft += KiLeft*ErrorLeft[0];
    }
    else if (IntegralCounterLeft == 2) {
        PWMleft += KpLeft*(ErrorLeft[0] - ErrorLeft[1]) +
            KiLeft*ErrorLeft[0];
    }
    else {
        PWMleft += KpLeft*(ErrorLeft[0] - ErrorLeft[1]) +
            KiLeft*ErrorLeft[0] +
            KdLeft*(ErrorLeft[0] - 2*ErrorLeft[1] + ErrorLeft[2]);
    }
    return PWMleft;
}
```

Figure 9: JavaScript code implementing PID regulator.

The PID algorithm parameters were the following: $K_p = 1.5$, $K_i = 0.75$, $K_d = 0.05$.

```
<script src="/socket.io/socket.io.js"></script>

var robotIP00 = "192.168.1.131:8080"; // robotIP00 is IP address
var robotIP01 = "192.168.1.130:8080"; // robotIP01 is IP address
var robotIP02 = "192.168.1.129:8080"; // robotIP02 is IP address
var robotIP03 = "192.168.1.128:8080"; // robotIP03 is IP address

var socket0 = io.connect(robotIP00);
var socket1 = io.connect(robotIP01);
var socket2 = io.connect(robotIP02);
var socket3 = io.connect(robotIP03);

buttonForward.addEventListener('click', function() {
    socket0.emit("commandArduinu", {"noOfCommand": 1});
    socket1.emit("commandArduinu", {"noOfCommand": 1});
    socket2.emit("commandArduinu", {"noOfCommand": 1});
    socket3.emit("commandArduinu", {"noOfCommand": 1});
});

buttonStop.addEventListener('click', function() {
    socket0.emit("commandArduinu", {"noOfCommand": 0});
    socket1.emit("commandArduinu", {"noOfCommand": 0});
    socket2.emit("commandArduinu", {"noOfCommand": 0});
    socket3.emit("commandArduinu", {"noOfCommand": 0});
});
```

Figure 10: Part of html file with JavaScript code on the master CPR platform.

A simple robot operating system (ROS) has been implemented to control several CPRPs with one interface. The main idea of this is that the main, or host CPRP produces the web user interface, through which the commands are sent to other CPRPs. A simple example is shown in figures 10 and 11, where there are only two commands: move forward and stop. In html file the connection to all four CPRPs is established. In order to do this, the socket.io library is used. For each robot, unique IP with port is assigned. After the sockets have been established, the event listener on the particular button triggers the delivery of the Instructions to all robots in the network.

In the JS file the flags represent the commands to the control algorithm to start moving the platform. These commands are the same for all four platforms and are trigged at the same time. This makes the platforms move in exactly the same manner, while the user works only with one web interface.

```
socket.on("commandArduinu", function(data) {
// when the socket is ON and "commandArduinu" is sent

    if (data.noOfCommand == "1") { // FORWARD
        StopFlag = false;
        ForwardFlag = true;
    }

    else if (data.noOfCommand == "0") { // STOP
        StopFlag = true;
        ForwardFlag = false;
    }

}
```

Figure 11: Part of JavaScript cod on each CPRP.

Figure 10 shows four robotic platforms (Numbers 1-4) with output from the on-board cameras, which is on the right side (Numbers 5-7). Besides, one could observe real time camera stream on the mobile phone together with user interface (Number 9). This provides good possibility to interact with the CPRP, since the outputs of the platform could be easily conveyed to the internet.

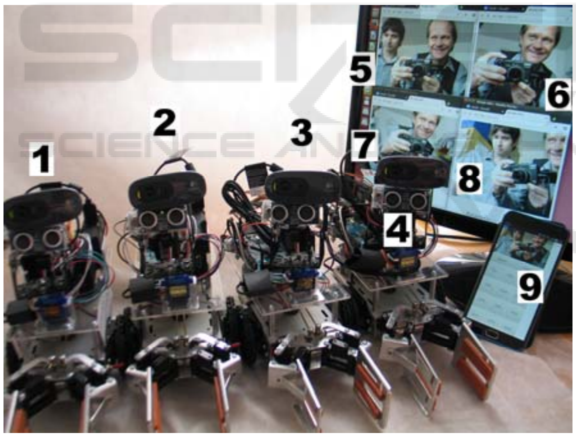


Figure 12: Four robotic platforms with output from the on-board cameras on monitor and smartphone.

Controlling four independent platforms provides various possibilities for the cyber-physical system, such as: moving objects, by the one robot, grabbing objects, moving objects by cooperative behaviour of several platforms, searching for some particular place or object, interacting between the platforms, and others. The platforms may be equipped with additional software or hardware, such as gyroscopes, magnetometers, LIDARs, infrared sensors, speakers and any other sensors compatible with Arduino.

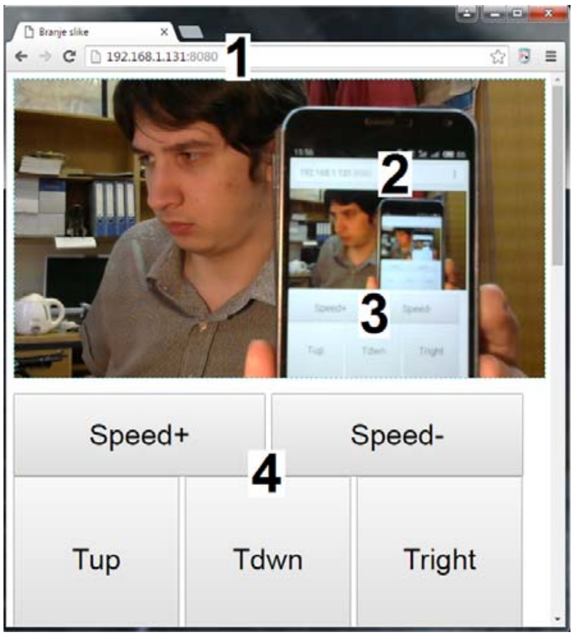


Figure 13: Example of a simple web interface used in experiments.

An example of a simple web interface for controlling the cyber-physical system is presented in figure 13. This interface allows simple controlling of the cyber-physical robotic platforms with several buttons (3 and 4), allowing to adjust the camera tilt, trigger the gripper, moving forward, backward, left forward, left backward, right forward, right backward, and spins right and left. Also a simple speed control is available. The streaming from the camera placed at the top of the interface. In figure 13 the web interface is accessed from the smartphone (2) and the laptop (1) at the same time, giving an opportunity to make a screenshot. The development of entire user interface was done in html5 and JavaScript / ECMAScript.

The main advantages of developed platform are:

a) The platform consist of Mini PC and common webcam that might be used for several other purposes not only for presented platform. Besides, such an ARM Mini PC and webcam might be already available to the students (as smart TV dongle etc.) This lowers the cost of the implementation. We have already tested the platforms in classroom in laboratory exercises giving the promising results.

b) The javascript (ECMAScript) with Cloud 9 IDE solves many problems with HW especially at coding and usage. With multiple platforms this showed to be essential. This is also useful for development of simple Robot Operating Systems

(ROS). The platform was partially used in classroom and the combination of technologies showed to be very well designed for the educational purposes.

3 EXPERIMENTAL RESULTS

To evaluate the effectiveness of the developed platform, a set of experiments has been performed by four operators over the web interface performing a simple task of finding an object, grabbing it and bringing it back to the start position. Some of the operators were experienced persons, who developed the web interface and the platform, while the others had never had an opportunity to control the robot. The operators could only see the image from the web camera, but not the robot itself. The starting position and the position of the object was fixed, and the object was out of sight from the starting position. All experiments were performed with the same platform. Figure 14 shows the test area schematics.

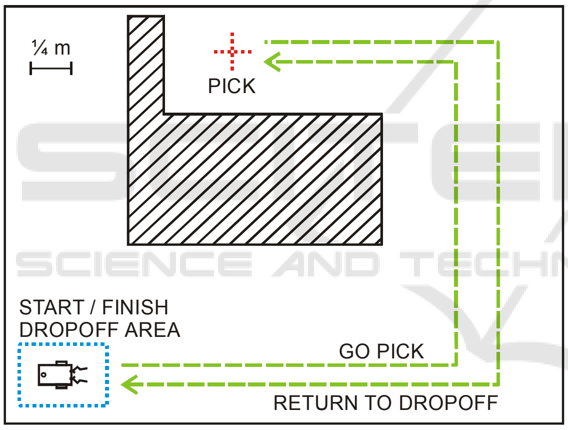


Figure 14: Test area for the experiments.

The parameters which were measured for every task are the total time, required to finish the task, and the amount of energy consumed by the robot. The energy consumption was calculated as the sum of products of voltage, current and time interval between measurements (150 ms) over the whole running time. The running time was estimated as the time between the first press of a button on the web interface and the last press. Table 1 contains the time intervals, measured for four users.

Table 1: Time required to perform the task, s.

Time	User 1	User 2	User 3	User 4	Average
Test 1	223,2	270	338,4	314,08	286,42
Test 2	214,8	236,7	262,2	211,6	231,32
Test 3	206,4	252,2	249,3	239,6	236,88
Average	214,8	252,96	283,3	255,1	251,54

As table 1 shows, the first test was the least successful for all participants, while the second one was most successful (M1=214.8, SD1=8.4, M2=260.7, SD2=36.6, t(13)=-3.95, p=0.005). This happened because the participants have learned the best way to control the platforms, avoid obstacles, and grab the object as fast as possible. Table 2 contains the results on power consumption.

Table 2: Energy required to perform the task, kJ.

Energy	User 1	User 2	User 3	User 4	Average
Test 1	0,828	1,034	1,242	1,176	1,079
Test 2	0,804	0,87	0,918	1,005	0,899
Test 3	0,807	0,927	0,981	0,933	0,912
Average	0,813	0,944	1,047	1,038	0,960

The energy consumption is highly correlated (r=0.92) to the time required to perform the task, as the main system component consumer is the miniPC, which drains around 0.4 amps at 6.1 volts. However, these results show that some operators required more power to finish, for example user 2 on test 2 consumed 0.87kJ, while user 4 on test 2 – 1kJ, which is a 13% difference, while the time spent was almost the same, 236,7 and 239,6 seconds respectively. Figure 15 shows the graph of power consumption for one of the tests.

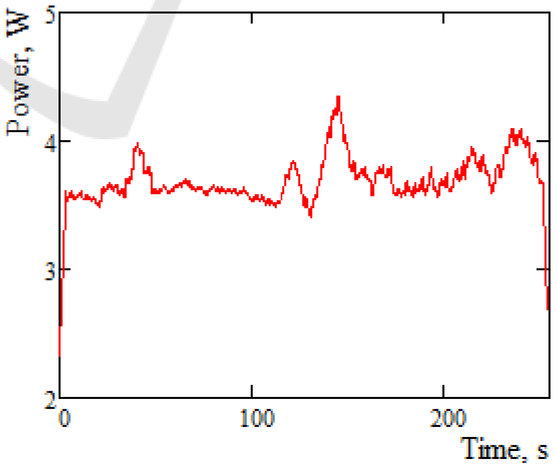


Figure 15. Power consumption for test 3, user 2.

The graph has been smoothed, and the high peak at the middle of time period corresponds to grabbing the object, as the grabbing servo consumes quite a lot of power while activated. Other peaks correspond mainly to camera movements and changing direction.

4 CONCLUSIONS

The developed four robotic platforms show a simple way to create a complicated cyber-physical system, capable of performing various tasks. The main task of this particular system is to engage students into development (Mostefaoui and Benachenhou, 2015), providing them with hardware and software that is easy to get and understand. Nevertheless, the presented system is capable of solving more sophisticated problems, as described in (Škraba et al., 2015b) and (Škraba et al., 2015a) where a speech-controlled wheelchair uses a similar system architecture.

The main contribution of presented prototypes is hardware-software combination that enables us to easily program CPRPs in JavaScript / ECMAScript and html5. Since the students are usually familiar with mentioned technologies this might accelerate the learning process as well as development of professional solutions. Certainly, there are limitations to professional use however, for particular cases; the proposed architecture could be the right way to perform some real world implementations. The conducted experiments showed the possibility to study the complex cyber-physical interactions with subjects. It is interesting, how only with the browser, one could access and program sophisticated CPRPs and conduct exciting experiments; what else could one want in order to engage students?

ACKNOWLEDGEMENTS

This work was supported in part by the Slovenian Research Agency (ARRS) (Research program “Decision support systems in electronic commerce”, program No.: UNI-MB-0586-P5-0018), ARRS SI-RF bilateral project »Development of Speech Controlled Wheelchair for Disabled Persons as Cyber-Physical System« Proj. No.:BI-ME/16-17-022 and Russian Federation Presidential Scholarship No. 16-in-689.

REFERENCES

Arduino, 2016, http://www.arduino.cc/ (Accessed, 21.3.2016).

Cloud9, 2016. https://github.com/c9/core/ (Accessed, 21.3.2016).

Codd-Downey R., Jenkin M., 2015. RCON: Dynamic Mobile Interfaces for Command and Control of ROS-enabled Robots. *ICINCO Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics*, 66-73, 2015, Colmar, Alsace, France.

Firmata, 2016. https://github.com/jgautier/firmata (Accessed 21.3.2016).

Intel, 2016. “Why Intel Loves HTML5,” http://software.intel.com/en-us/videos/why-intel-loves-html5 (Accesed, 21.3.2016).

Jiang W., Zhou B., Zhang M., 2015. Architecture Analysis and Implementation of 3D Theatre Display System Based on Node.js. *International Conference on Network and Information Systems for Computers.*

johnny-five (2016) http://johnny-five.io/, (Accessed, 21.3.2016).

Kraijak S., Tuwanut P., 2015. A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and future trends. *Proceedings of ICCT2015.*

mjpg, 2016. http://sourceforge.net/projects/mjpg-streamer/ (Accessed 21.3.2016).

Mostefaoui H., Benachenhou A., 2015. Design of a remote electronic laboratory. *2015 International Conference on Interactive Mobile Communication Technologies and Learning* (IMCL), 160-162.

node.js, 2016. http://nodejs.org/ (Accessed, 21.3.2016).

Škraba A., Koložvari A., Kofjač D., Stojanović R. (2014) Prototype of speech-controlled cloud-based wheelchair platform for disabled persons. *Embedded Computing* (MECO), 2014 *3rd Mediterranean Conference on*, pp. 162 – 165.

Škraba, A., Stojanović, R., Koložvari, A., Kofjač, D. 2015a. Wheelchair maneuvering using leap motion controller and cloud based speech control: Prototype realization. *Embedded Computing (MECO), 2015 3rd Mediterranean Conference on*, pp.391-394.

Škraba, A., Stojanović, R., Zupan, A., Koložvari, A., Kofjač, D. 2015b. Speech-controlled cloud-based wheelchair platform for disabled person. Microprocessors and Microsystems, Vol. 39/8, Pages 819–828.

socket.io, 2016. http://socket.io/ (Accessed, 21.3.2016).

Ubuntu, 2016. Ubuntu http://www.ubuntu.com/ (Accessed 21.3.2016).