Designing a Low-Cost Web-Controlled Mobile Robot for Home Monitoring

David Espes, Yvon Autret, Jean Vareille and Philippe Le Parc
Université Européenne de Bretagne, France
Université de Brest
Laboratoire en Sciences et Techniques de l'Information (LabSticc UMR CNRS 6285)
20 av. Victor Le Gorgeu, BP 809, F-29285 Brest
E-mail: {david.espes, yvon.autret, jean.vareille, philippe.le-parc}@univ-brest.fr

***Abstract*—In this paper, we focus on a web-controlled mobile robot for home monitoring. The key point is low-cost. The robot is built from standard components to reduce the cost of the hardware. A large part of the system is deported to the cloud to minimize the required software on the robot. A minimal positioning system is provided to make the robot usable. The result is a small robot wich can be used from the outside or the inside the house.**

Keywords–Mobile robot; Home monitoring; Web control; WebSocket.

I. INTRODUCTION

Web-controlled mobile devices are more and more used in ubiquitous environments [1][2][3][4]. Small monitoring robots such as the Rovio WowWee can be used [5]. Web control is not really new, but recent improvements of network performance has led to the emergence of Service Robotics [6]. Services Oriented Architectures (SOA) [7] start to be used to control physical devices [8].

Our aim is to use these approaches to control mobile home robots designed for Human Ambient Living (HAL) environments. For us, a typical application is helping elderly people who live in their houses and sometimes have difficulties to move. A mobile home robot carrying a camera could help them monitoring their house either indoors or outdoors, for example to watch the dog or to see what is going on. The mobile home robot could also be used by care helpers or relatives, as a moving phone to communicate with the inhabitants of a house from the outside.

In such an HAL environment, the total cost of the mobile home robot is the first key point. It must be kept as low as possible especially if it is a HAL environment for elderly people who often have tight budgets. This means that the mobile home robot must be built by using low-cost commercial components. Moreover, we always keep in mind that mechanical failures are unavoidable and reliability is a major key point as much as ease to repair. The basic mobile home robot is nothing but than a mobile robot which carries a camera. More sophisticated sensors such as positioning sensors are optional.

The second main key point is software and network configurations. The mobile home robot should be plug an play. This means that software and network configurations should be reduced as much as possible. Deporting a part of the system to the cloud can be a solution if it helps to get a reliable plug and play system.

The third key point is security and access control. A Web-controlled mobile home robot can be used from anywhere in the world, but the interior of a house must not be seen by unauthorized users. It is necessary to avoid any intrusive access. In case of network failure, the mobile home robot should also be able to properly stop its current action and wait for a new order.

The fourth key point is the autonomy of the battery. The robot should have an autonomy close to one hour when moving, and automatically come back to a charging dock when the battery is low.

In this paper, the second part presents a mobile home robot solution based on a commercial low-cost robot and we discuss the advantages and the disadvantages. This lead us to the design of a mobile home robot built from commercial components such as a low-cost robotic platform and a smartphone to control it. In the third part, we present the cloud control system and its performance. In the fourth part, we add video monitoring capabilities to the robot, and in the last part, we present a low-cost positioning system.

II. DESIGNING A HOME ROBOT

A. Commercial home robots

Several commercial robots such as Miabot [9] are available. The Miabot robot is rather small (about 10cm long) and fast (3.5 m/s). It has a built-in bluetooth connection and must be connected to a local central computer to be web-controlled. Even if it was not really designed for that, it can carry a small camera or other sensors.

A better robot from our point of view is the WoWee Rovio [5]. It includes a mobile base, a mobile camera and a Wi-Fi connection. Its size is 30 x 35 x 33 cm. It can be remotely controlled from anywhere in the world. When the battery is low, it automatically comes back to its charging dock. The almost 300 euros cost is acceptable.

The WoWee Rovio is an interesting robot for a HAL environment, but a weak point is the reliability and the ease to repair [10]. The WoWee Rovio can not be considered reliable. For example, sunlight may interfere with the infrared bean of the WoWee Rovio and prevent it returning to its charging dock. In case of failure, the WoWee Rovio is difficult to repair. We have used several WoWee Rovio. One of them had and infrared led problem and all of them had battery problems after one year use. This was a real problem because we had no easy solution to replace the failing components.

 ISBN: 978-1-61208-353-7

B. Using a Smartphone, an Arduino, and a basic robotic platform

Using a smartphone may help simplifying the building of a home robot because it usually includes a webcam, Wi-Fi and Bluetooth. When connected to an Arduino micro-controller [11], a smartphone can also be used to control the motors of a mobile home robot.

We use an open robotic platform which includes two tracks. It is a 4WD Rover 5 from RobotBase. The size is close to that of the WoWee Rovio. When powered, it can move forward or backward and turn. The maximum speed is 1km/h. The Rover 5 is strong enough to carry up to two kilograms.

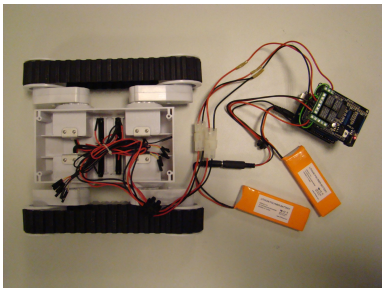

Figure 1. Components of the Web-controlled home robot.

The robot is controlled by the Arduino. Several Arduino shields are available to monitor the working speed and direction of the motors. We can use either a relay shield including four relays, or a motor shield based on a voltage regulator such as 78M05. An additional Arduino shield is required to allow Bluetooth communication between the Arduino and the smartphone.

The main advantage our solution is its simplicity. The home robot only includes six commercial components (Fig. 1 and Fig. 2):

- A mobile Rover 5 robot used as robotic platform (60 euros)
- An Android Smartphone (less than 80 euros)
- An Arduino UNO (20 euros)
- A Bluetooth shield (20 euros)
- A motor command shield (20 euros)
- Two batteries (one for the Arduino, one for the motors)

The total cost, smartphone included, is comparable to that of a WoWee Rovio. We can also use an old smartphone which has became useless. When used as a mobile home robot controller, a recycled smartphone significantly reduces the total cost.

The reliability of our mobile home robot is significantly higher than that of a Rovio. In case of failure, we only need to replace one component. Moreover, the diagnosis is very easy because each component can be individually tested.

When using 2000mAh lithium batteries, we have a 30mn autonomy when the robot is continuously moving. We have several hours of battery life when the robot is waiting for commands. Automatic battery charging is not available on our prototype.

Additional sensors can be added on the robot, but as it is a non autonomous Web-controlled robot, they are not essential. Moreover, it would increase the total cost.

III. A CONTROL SYSTEM IN THE CLOUD

We propose to deport a large part of the robot control system to the Cloud to reduce home configurations and installations. A user interface running on a standard Web Browser should make the robot usable without any special installation.

Using HTTP (Hypertext Transfer Protocol) is a solution to communicate with a distant server in the Cloud. Efficient HTTP Web servers such as Apache or Apache Tomcat are available. If the standard HTTP protocol lets easily handle problems such as client identification, it has severe limits when used for real-time monitoring.

A. The HTTP limitations

The HTTP protocol is a stateless protocol which was originally designed to get access to static HTML pages. Later, some web applications have implemented server-side sessions by using HTTP cookies. A Web server implementing sessions receives an HTTP request, establishes a connection with the server, executes the request, sends an HTTP response back, may keep a track of the HTTP request, and finally, releases the connection.

If a Web server is running on the robot, an identification sequence which gives the right to monitor the robot through the Web server can be easily implemented. The communication can be secured by using the HTTPS protocol. The main problem is the execution time of a command sent to the robot. Let us take the example of an HTTP request which should make the robot move for several seconds. As soon as the HTTP request is received on the server, the robot starts moving. If the robot moves for more than a few seconds, the HTTP response must be sent back before the robot has finished moving. In this case, the robot can get out of control.

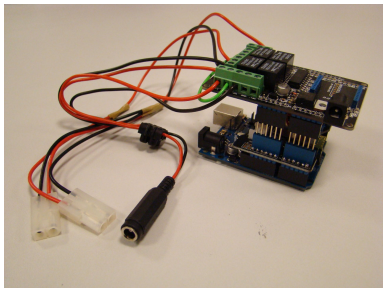

Figure 2. The Arduino command module.

This is a major problem because we must monitor a robot by using commands which execution lasts about one second. A one meter trip would require sending at least three commands to a Rover 5 moving at 1km/h. Touring a house would require hundreds of commands. When a command is sent to a distant robot, a permanent connection is required. A moving robot left

 ISBN: 978-1-61208-353-7

unsupervised just a few seconds can be dangerous. Presence and obstacle detectors working on the robot are never 100% reliable. This means that anyone who is monitoring from the outside or inside the house must have a permanent full control of the robot through the network. Moreover, the robot should be able to detect the smallest network failure, and to automatically adapt its behaviour, for example by reducing its speed.

This means that sending HTTP requests to a Web server running on the robot is not a good solution. We must continuously send HTTP requests to the robot to be able to detect network failures. That is a misuse of HTTP. Second, establishing a new connection from outside can be time consuming and sometimes takes several seconds. That is a risk we can not take. That is why we have chosen the WebSocket solution.

B. Using a WebSocket server

The WebSocket protocol was standardized in 2011 [12]. The communications are established by HTTP servers, and the communications may use TCP port 80 (or 443 when using secured communications). The client is responsible for making the connection by using an URL, consisting of a protocol, host, port, path, and optionally one or more additional parameters.

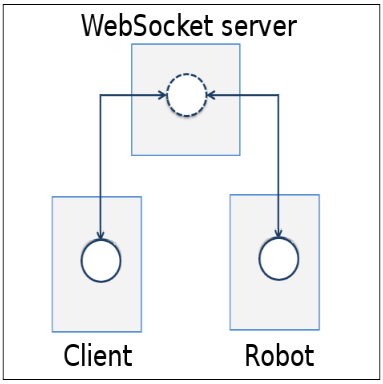

Figure 3. The WebSocket servlet.

The main advantage of WebSockets for our purpose is the fast responses coming from the server. That is due to the single connection that is established at the beginning of the communication. As soon as a connection is set, a bi-directional communication remains available. Full duplex communication over a single socket allows true real-time communication.

A standard Web browser can be used to monitor a robot through WebSockets. Most Web browsers now support WebSockets. Both the client and the robot send and receive information to and from the Web server through WebSockets. When a command is sent from the client to the Web server by using WebSocket, as soon as it is received on the server, it can be forwarded to the robot and executed. During the execution of the command on the robot, WebSockets are still used to send periodic aknowledges from the robot to the client, and from the client to the robot.

Thus, if the robot does not receive any acknowledge, or receive them too late, it can modify its state. For example, it can reduce its speed if the network is too slow. If the network is no more working, the robot can stop properly, and remain waiting until the network is working again.

C. A WebSocket server in the cloud

A WebSocket server in the clould greatly simplifies the installation of a Web-controlled home robot. The home robot just have to connect to the WebSocket server (Fig. 3). This does not require any special home configuration. An ordinary Wi-Fi connection can be used.

The well known Apache Tomcat Webserver now implements WebSockets. This means that we can use both the advantages of a standard Web server and those of WebSockets. A standard Tomcat application manages client and robot identification. The client uses an HTML form to ask for a robot. As soon as identification is successful on the server, a WebSocket communication becomes available between the client and the robot.

On the Tomcat server, we have a servlet to manage identification and robot allocation. We have also a WebSocketServlet to manage communication between the client and the robot.

The main elements of the WebSocket Servlet are shown in Fig. 4.

```
public class RobotWebSocketServlet
        extends WebSocketServlet {

  protected StreamInbound
    createWebSocketInbound(String subProtocol,
          HttpServletRequest request) {
      Manager manager = ...
      return new ClientRobot(manager);
    }
}

public class ClientRobot
     extends MessageInbound {
  Manager manager = null;
  private RobotCommunication
       (Manager manager)
       { ... }
  protected void onTextMessage
       (CharBuffer message)
       throws IOException
       { ... }
}
```

Figure 4. The WebSocket Servlet.

The ”manager” object is instantiated by the WebSocket server. From the robot point of view, it contains information about the client which is using the robot. From the client point of view, it contains information about the robot to control. The manager is stored as a Tomcat session object. It is a persistant object whose life duration is that of a session. A ”manager” object is instantiated during the identification phase, when the client asks for a robot. Another ”manager” object is instantiated when the robot connects to the WebSocket server When the WebSocket communications are set, the ”manager” objects can be retrieved and modified to help clients and robots communicate. One client is allowed to send messages to one robot, and one robot is allowed to send messages to one client.

Both the client and the robot exchange messages by sending lines of text. For example, the client sends a line containing ”forward” to make the robot move forward. Parameters can

 ISBN: 978-1-61208-353-7

also be added in the line, for example to make the robot move forward for n seconds.

D. WebSockets on the robot

As seen above, the robot is controlled by the Arduino and the Arduino is controlled by an Android smartphone using a Bluetooth communication. We use the Tyrus API to connect the smartphone to the WebSocket server. The main Java elements of the Android WebSocket connection are shown in Fig. 5.

```
final ClientManager client =
    ClientManager.createClient();
client.connectToServer(
  new Endpoint() {
    public void onOpen(Session session,
        EndpointConfig EndpointConfig) {
      session.addMessageHandler(
        new MessageHandler.Whole<String>()
          public void onMessage
              (String message) {
            ...
          }
        },
        ClientEndpointConfig.Builder.
           create().build(),
           URI.create("ws://.../robot"));
    }
  });
```

Figure 5. The Android WebSocket.

We use the Tyrus ”ClientManager” class to set a connection between the robot and the WebSocket server. When messages come from the client, the ”onMessage” method is triggered. The message is decoded and forwarded to the Arduino. During the execution of the command by the Arduino, the client and the smartphone periodically exchange messages to stop or slow down the robot in case of network failure. This program has been tested on Android 2.3 and Android 4.

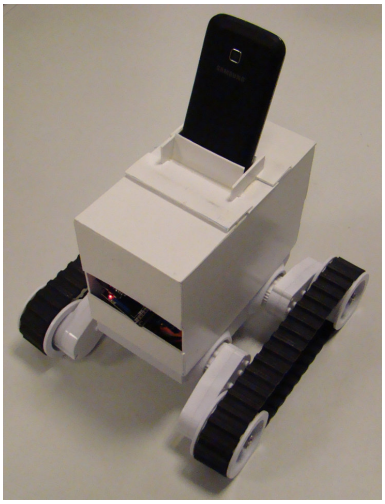

Figure 6. The Web-controlled home robot.

E. WebSockets on the client

A WebSocket connection from the client to the server is only possible if the identification phase and robot selection has been successful. This is taken into account by the standard Apache Tomcat Webserver. As soon as a client is successfully registered on the distant Web server, a WebSocket connection is established. The client uses a Web page as user interface. The only thing required to use the user interface is a WebSocket compatible Browser. The user interface is managed by the distant Web server. The main JavaScript elements of the WebSocket connection are shown in Fig. 7.

```
var client = {};
client.connect = (function(host) {
  client.socket = new WebSocket
     ("ws:// ... /client");
  client.socket.onmessage =
    function (message) {
      ...
    };
});
```

Figure 7. The client WebSocket.

The Javascript ”onmessage” function is triggered when a message comes from the WebSocket server. A widget such as a button in the user interface can trigger the ”sendMessage” function and send commands to the robot.

F. Performance

In this section, we present some experiments that illustrate the capabilities of our system. The server is connected to the local network of the laboratory, i.e., gigabit Ethernet network. It is hosted to a public address so any user are able to access it from anywhere using just a web browser. Beside the server, one robot is available. The robot is equipped with an arduino board, a bluetooth shield and a smartphone. The bluetooth shield is fully qualified to respect the Bluetooth version 2.0. Hence, the data rate is up to 2Mbps. The smartphone is connected to the local network through a WiFi connection. The wifi card on the smartphone is compliant to the IEEE 802.11g standard. Hence, the data rate is up to 54Mbps.

In order to show the performance of the system, we define the following performance metrics:

- the *Round-trip time between components* is the time to receive a response after sending a request without counting the delay due to other components. By example, if the arduino board sends a request to the smartphone, the round-trip time between these both components is the delay to receive a response without counting the delays imposed by smartphone-server connection and server-user connection.
- the *End-to-end round-trip time* is the time that the user receives a response after sending a response, i.e., it is the sum of the round-trip time between the whole components of the system. The increase of the end-to-end round-trip time degrades significantly the QoS of applications.

In order to test different scenarios, the user accesses to the robot from two different locations: our laboratory and

 ISBN: 978-1-61208-353-7

TABLE I. ROUND-TRIP TIME RELATED TO ENTITY CONNECTIONS

Entity connection	Round-trip time	
	Local (inside laboratory)	Distant (Romania)
User - Server	15 ms (±5 ms)	40 ms (±15 ms)
Server - Phone	35 ms (±10 ms)	35 ms (±10 ms)
Phone - Robot	125 ms (±40 ms)	125 ms (±40 ms)

the Military Technical Academy of Bucharest in Romania (about 2500 km from the laboratory). The user accesses to the robot through the LAN or Internet into the laboratory or the Academy of Bucharest respectively. In all the scenarios considered the server is inside our laboratory. However, due to the flexibility of our architecture, the server could be hosted in the cloud.

In Table I, we present the round-trip time related to component connections when the user accesses to the robot from different locations (laboratory and Romania). All times are expressed once the websocket connection is established.

It is interesting to see that the most important delay is added by the bluetooth connection between the arduino board and the smartphone. Indeed, the data rate of the bluetooth shield is quite low (2Mbps). The time to transmit the data from the robot to the phone, or inversely, is proportional to the data rate. This is the principal factor to this delay. Moreover, bluetooth system is a contention based system. Bluetooth systems are based on a combination of frequency-hopping and CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) [13] methods to access to the medium. The medium is shared between all the nodes belonging to the same user and other systems such as WiFi. The delay to access to a free medium or the retransmissions due to collisions increase the round-trip time significantly. However, it is interesting to use a bluetooth connection between the smartphone and the robot due to its low consumption. The mobile robot's operational time is limited before exhausting its battery power. Indeed, Bluetooth is much more power efficient than WiFi. As mentioned by Pering et al. [14], the power consumption of Bluetooth is 10 times lower than WiFi.

The round-trip time between the smartphone and the server is quite low. Unlike Bluetooth systems, WiFi systems have a high throughput. The transmission time is significantly reduced. The round-trip time induced by WiFi is roughly 4 times lower than the one obtained with Bluetooth.

The Internet delay, i.e., when the user is located in Romania, is almost negligible as compared with local access. The university of Brest, respectively Academy of Bucharest, has a guaranteed bandwidth of 3Gbps, respectively 1Gbps, on its national network. Hence, the difference in time is particularly due to the propagation time. Let us assume a propagation speed of 200,000 km/s, the round-trip propagation time is about 25 ms.

In Table II, the end-to-end round-trip time is analyzed under

TABLE II. END-TO-END ROUND-TRIP TIME

Protocol	End-to-end round-trip time	
	Local (inside laboratory)	Distant (Romania)
HTTP	600 ms (±120 ms)	730 ms (±100 ms)
Web Sockets	205 ms (±75 ms)	250 ms (±50 ms)

two different locations (local and Romania) and two protocols (HTTP and websocket). The end-to-end round-trip time is an important parameter because it is the main criteria to determine if real-time control is possible. To control a distant robot with an acceptable quality of experience, it is commonly accepted that the delay never exceeds 400 milliseconds. We can see the HTTP protocol cannot guarantee the delay bound. Indeed, the time to establish the connection, to send a request and receive a response significantly exceeds the delay bound. In case a system requires the establishment of a TCP connection for each transaction, the real-time control of the mobile robot is not possible. The websocket protocol is more suitable for real-time control. Being designed to work well in the Web infrastructure, the protocol specifies that the websocket connection starts its life as a HTTP connection, offering backwards compatibility with no-websocket systems. The handshake of the websocket protocol has slightly the same time than the HTTP protocol. Once the connection is established, control frames are periodically sent to maintain the connection. Hence, the time is significantly reduced as compared with the HTTP protocol. For all scenarios, the end-to-end round-trip time does not exceed 300 milliseconds which is quite acceptable to transmit QoS traffic.

IV. VIDEO MONITORING

If a video stream is sent from the robot to the client, the loss of some images is not critical. Thus, videos can be obtained from a standard video Web server running on the smartphone. We have used the IP Webcam application which works on Android 1.6 and up and broadcasts video and sound. The smarphone is placed on top of the robot (Fig. 6). It also communicate with the Arduino and the cloud server as seen above.

If security is required, videos can be sent to a distant Web server through a securized channel, and forwarded to the client.

V. TOWARD A LOW-COST POSITIONING SYSTEM

We also design a low-cost localization platform for 2D-positioning. Even if the robot is not an autonomous one, as it is web-controlled, information on the position of the robot is very useful to the user of the robot. Let us assume the robot only has to monitor flat floor, i.e., the relative z-coordinate is always equal to 0. In cases where different floors have to be monitored, a robot may be on each floor. They can communicate between them in order to extend the control in the whole habitation.

The positioning system involves 4 TelosB wireless devices. The 3 auxiliary sensors have a fixed position, being place in

 ISBN: 978-1-61208-353-7

strategic places of the room, in the corners for example. The places must be chosen in such way that the robot which will have the Main Sensor attached to be in permanent Line of Sight with this sensors. This way, the communication would be done with very little interference.

Fig. 8 shows the whole system and the interaction between the components. The auxiliary sensors send a message periodically. The main sensors do not know their position. After receiving a message from an auxiliary sensor, they gather information, such as receiver's Received Signal Strength Indicator (RSSI) and the identity of the sender. In order to optimize the energy consumption, the processing of the RSSI values is skipped in this moment, being the duty of the server application to make the necessary computations from which will result the distance approximation. Once the Main Sensor acquires a message from each of the 3 fixed sensors, it will create a data packet which contains the 3 pairs of ID - RSSI value for each sender, and will sent it through the USB interface to the arduino board. The arduino board forwards this message to the server that converts the raw values into physical distance, measured in meters. At this point, the server knows the distance between the main sensor and each auxiliary sensor.

In two-dimensional geometry, the trilateration technique uses three reference nodes to calculate the position of the target node. To be localized the target node should locate at the intersection of three spheres centered at each reference position. When the signal received from the reference nodes is noisy, the system is non-linear and cannot be solved. An estimation method has to be used. To get a satisfying approximation position of the mobile robot, we use the Newton-Raphson method [15]. This method attempts to find a solution in the non-linear least squares sense. The Newton-Raphson' main idea is to use multiple iterations to find a final position based on an initial guess (by example the center of the room), that would fit into a specific margin of error.

The first results show that RSSI values are not constant due to multipath components. Hence, the precision of our system is about 2 meters. Such a precision is sufficient to know approximately where the robot is. The localization will reduce the complexity of the control interface dedicated to the distant user. The web interface will contain the cartography of the house, and the robot will be able to reach a destination, only by clicking on the map.

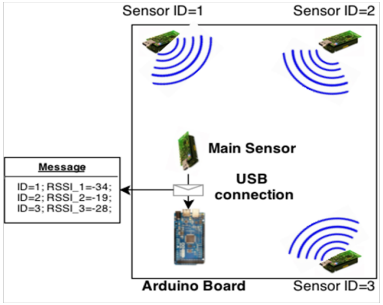

Figure 8. The positioning system.

VI. Conclusion

The smartphone and the Arduino micro-controller are the two main devices of the proposed home robot. The smartphone includes several important features such as network connection and webcam. It reduces the total cost and increases global reliability. By combining a standard Web server and WebSockets, we can deport a large part of the system to the cloud, and installing the home robot in a HAL environment becomes simple and cheap. The lack of sensors on the robot is not very important because it is not an autonomous robot. Moreover, it helps keep the price down. The positioning system is the weakest point of the system. The lack of accuracy make the robot more difficult to use. It is an important element which must be improved in the near future without increasing the total cost.

References

[1] A. Chibani, Y. Amirat, S. Mohammed, E. Matson, N. Hagita and M. Barreto, "Ubiquitous robotics: Recent challenges and future trends", *Robotics and Autonomous Systems*, Volume 61, Issue 11, November 2013, pp. 1162-1172, ISSN: 0921-8890.

[2] S. Nurmaini, "Robotics Current Issues and Trends", *Computer Engineering and Applications*", Vol. 2, No. 1, March 2013, pp. 119-122, ISSN: 2252-4274.

[3] A. Touil, J. Vareille, F. L'Herminier and P. Le Parc. "*Modeling and Analysing Ubiquitous Systems Using MDE Approach*". The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. Florence, Italy. October 2010.

[4] P. Le Parc, J. Vareille and L. Marce. "*Web remote control of machine-tools the whole world within less than one half-second*". ISR 2004: International Symposium on Robotics, Paris, France, March 2004.

[5] *WoWee Rovio, a Wi-Fi enabled mobile webcam*. http://www.wowwee.com/en/products/tech/telepresence/rovio/rovio. Online; accessed Apr. 15, 2014.

[6] *Robots With Their Heads in the Clouds*. IEEE Spectrum. Mars 2011.

[7] *Service-Oriented Architecture (SOA) and Cloud Computing*. http://www.service-architecture.com/articles/cloud-computing/service-oriented_architecture_soa_and_cloud_computing.html. Online; accessed Apr. 15, 2014.

[8] Y. Chen, Z. Du and M. Garcia-Acosta *Robot as a Service in Cloud Computing*. Fifth IEEE International Symposium on Service Oriented System Engineering. Nanjing, China, June 2010, pp. 151-158.

[9] Introduction to the Miabots & Robot Soccer, URL: http://eprints2.utem.edu.my/5831/1/Merlin_Miabot_Pro_Robot_Soccer_%282_Wheels%29_24_Pages.pdf. Online; accessed Apr. 15, 2014.

[10] *2009-01-Rovio-insecurity - Insufficient Access Controls - Covert Audio/Video Snooping Possible*. http://www.simplicity.net/vuln/2009-01-Rovio-insecurity.html. Online; accessed Apr. 15, 2014.

[11] *The Arduino micro-controller*. http://arduino.cc/. Online; accessed Apr. 15, 2014.

[12] *The WebSocket Protocol. Internet Engineering Task Force (IETF)*. http://tools.ietf.org/html/rfc6455. Online; accessed Apr. 15, 2014.

[13] M. Oliver and A. Escudero, "*Study of different CSMA/CA IEEE 802.11-based implementations*". In EUNICE, 1999, pp. 1-3.

[14] T. Pering, Y. Agarwal, R. Gupta and C. Power, "*Coolspots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces*". In Proc. ACM MobiSys, 2006, pp. 220-232.

[15] *The Newton-Raphson Method*. http://www.math.ubc.ca/ anstee/math104/newtonmethod.pdf. Online; accessed Apr. 15, 2014.

 ISBN: 978-1-61208-353-7