Journal of Electrical Engineering, Electronics, Control and Computer Science – JEEECCS, Volume 5, Issue 16, pages 1-4, 2019

Application for Collecting Data from Sensors and Transmitting Them Through the Network

Alexandru CONSTANTIN, Valeriu Manuel IONESCU
Department of Computer Science
University of Pitesti
Pitesti, Romania
alexandrustelian13@gmail.com, valeriu.ionescu@upit.ro

Abstract **– Remote controlling robots is often necessary when operations need to be executed in a dangerous environments for humans. The information read from the monitoring of a real user's arm movement can be used, for example, to control the actions of a robotic arm. The needed sensor information can be taken from a mobile phone that is equipped with sensors which provide information about the user actions. The sensor reading can be easily done using modern software libraries and the data sending over the internet can be done with reduced latency using protocols such as WebSocket. This allows the low cost implementation of a remote control system over internet. This paper presents the implementation of a hardware and software control system for remote robotic arm control using the Android accelerometer sensor data and the WebSocket protocol for internet communication.**

Keywords-robotic arm; WebSocket; control; Arduino; sensors

I. INTRODUCTION

The first industrial robot was used in the automotive industry and it was a massive device built in the United States in the late 50s [1]. As the technology improved, the size and cost of robots has decreased making them useful for automating work in many domains ranging from manufacturing to healthcare. The use of robots is advantageous in environments where the use of human workforce is not recommended: environments that have safety hazards, situations that require the execution of repetitive tasks and places where it is expensive to use human workforce.

Robots produce safe and precise work; they can perform applications with more repeatability than the human workforce while being able to work 24/7 with the same production speed. Robots only make mistakes if they are defective and are more accurate than human labor. Finally robots can be monitored constantly and a faulty operation can be detected and corrected in a very short amount of time.

The operation of a robot can be fully automated or it can be controlled remotely by a human operator. For the remote operation many existing protocols are be used but sometimes custom protocols are created, depending environment conditions (such as distance, errors, latency). The requirements often include the need for real time bidirectional communication. Using the internet infrastructure for sending and receiving data from the robot can be an advantage as it reduces the implementation costs and allows long distances.

There are many internet protocols and APIs that can fulfill these characteristics: WebSocket, HTTP/2 or WebRTC API. Out of these, HTTP/2 [2] is the most forward looking protocol and it allows the server to push content, but is currently supported only by 30% of the websites, while WebRTC API uses multiple protocols for its implementation, including WebSocket. For the implementation from this paper we have used the WebSocket protocol.

WebSocket is a communication protocol that provides full-duplex communication channels through a single TCP connection. In this way, after the connection is established, both server and client can send messages without a request. WebSocket is an upgrade of an HTTP connection and is supported by browsers, allowing system control from a web page. This protocol was proposed in 2009 and adopted in December 2011 by Internet Engineering Task Force as RFC6455 [3]. The established connection can be secured optionally with Transport Layer Security (TLS) [4]. The WebSocket protocol defines a prefix *ws://* to indicate a WebSocket connection and *wss://* to indicate a WebSocket Secure connection, respectively. WebSocket is used in games, applications that need frequent updates, multi-user applications and robot control [5, 6].The advantages of this protocol over the classic HTTP connection are a reduced traffic due to the header size and a reduced number of connections. The implementation problems usually relate to the change of the traffic pattern compared to classic HTTP, which may affect the results of network traffic monitoring tools.

In related literature, many low cost solutions for robotic arm remote control were found, using different communication protocols, but many focus on short distance control:

- [10] uses a Arduino system to control the robotic arm while the commands are sent via an infrared remote. The disadvantage is that the communication is unidirectional and the advantage is that the cost of the hardware is very small.

- [11] uses Arduino Nano as the Bluetooth transmitter board and an Arduino Uno as the robot attached receiver. The hand movement is monitored by a GY-521 6DOF MPU6050 3 Axis Gyroscope with Accelerometer Module while the finger flexing information is taken from a Spectra Symboflex Sensor. The design is more complex but it does not offer the flexibility of transmitting the data over longer distances (internet). Similarly, in [12] the Arduino ATMEGA-328 micro-controller is via android app using a Bluetooth HC 05 Module.

- [13] is also using an Arduino Clone (RevIO) that communicates using two XBee Modules (one connected to the robotic arm and the other connected to a laptop). Again, the remote control via internet is not insured.

- Project [14] is using Android to control the robotic arm and the hardware that makes the communication possible is the 1Sheeld (Android shield for Arduino). The disadvantage is the use of dedicated hardware to connect the Android platform to the robot.

This paper presents the implementation of a control system for an Arduino powered robotic arm that uses accelerometer data from and Android mobile device. The Android server communication is made via WebSocket while the Server – robot communication uses a RS232 serial line. This allows reduced latency in the local control and the possibility to control the application form anywhere on the internet. This remainder of the paper is structured as follows: In Chapter II the system architecture is presented, in Chapter III the results and various implementation problems are presented, while the Conclusions will discuss future project improvements.

II. System Architecture

The system is composed of three components, as illustrated in Fig. 1:

-the Android application developed in Java that reads the user acceleration sensor information or the commands that are selected;

-the code running on the Arduino platform connected to the robot, that reads the serial line commands from the server commands and controls the robot motors.

-the server (written in Java), that creates the WebSocket communication with the Android application on one hand and gives commands to the robot via serial lines on the other hand. The Java Simple Serial Connector library was used for the serial line communication [7].

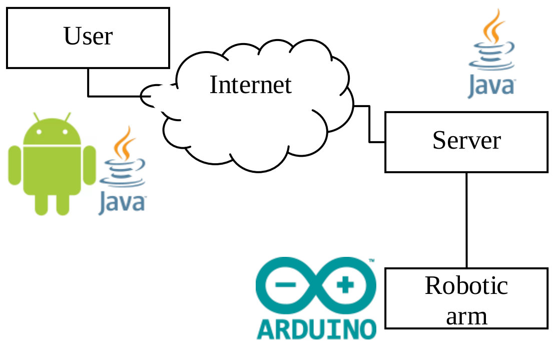

Figure 1. The system's architecture

III. System implementation

In the following section, the implementation of the various components is discussed and the problems that had to be resolved.

A. *The Android application*

The application was created in Android Studio using the Java programming language. To access the sensors the class *SensorManager* was used. The Sensor framework is part of the *android.hardware* package and includes the following classes and interfaces:

- *SensorManager* - Used to create a service instance for sensors. This class provides several methods for accessing and listing sensors, recording and unsubscribing sensors events listeners and getting orientation information and offers means to report sensor accuracy, set data acquisition rates, and sensors calibration.

- *Sensor* - this class provides different methods to determine the capabilities of a sensor and is used to create an instance of a specific sensor.

- *SensorEvent* - The system uses this class to create an event object of the type sensor, which provides information about a sensor event. An event object of the sensor type includes the following information: raw sensor data, sensor type that generated the event, data precision and time stamp for the event.

- *SensorEventListener* - this interface can be used for receiving notifications (sensor events) when the sensor values change or when sensor accuracy changes.

The OKHttp library [8] was used to communicate with the server using WebSocket, library for Android 2.3 and above that was initially designed to work for HTTP, HTTPS and HTTP/2. Since version 3.5 the OkHttp library supports the WebSocket protocol. The Android application will send short messages that actually represent hand robot commands. The data encapsulated in the packets was designed to have two components: the servomotor identifier and the number that identifies the position of the servomotor. These are enough to control each robot servo individually.

In order to implement the interface, the graphical design tools form Android Studio were used in a Constraint Layout. The interface has buttons to control the individual motors, and a single button to read the movement from the accelerometer. The A+/A-, B-/B- buttons were used to send manual commands through the WebSocket interface, for testing purpose. The Android application interface is presented in Fig. 2.

The application operates by first establishing from the client a WebSocket connection to the server through a process known as WebSocket handshake. This process begins with the client sending an HTTP request common to the server. An upgrade header is included in this request that informs the server that the client wants to establish a WebSocket connection. WebSocket implemented in this application uses the ws:// URI schema, meaning unsecure connection.

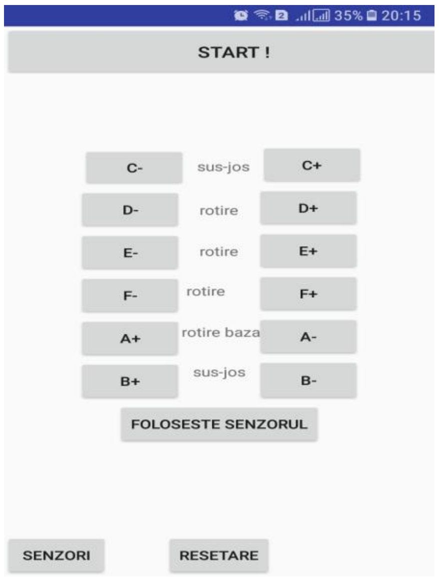

Figure 2. The Android application interface

To send a message via the WebSocket connection, it is necessary to call *send*() on the WebSocket instance. The parameter will be the data that is wanted to be transmitted (whether it is a message or a String, as in a WebSocket channel, both text data and binary data can be sent). To detect the rotation of the phone used to move the robot the data from accelerometer was used. It offers with double data about the 3 axes of the phone x, y, and z. So if there is a movement on the x axis I can say that the phone is moving left or right, and if the phone detects a motion on the y-axis, that means that the phone has a back and forth motion. The sending of data is only made is the "*Foloseste senzorul*" button is pressed in the user interface (Fig. 2).

B. The server application

The Java server application uses the org.java_websocket class in order to support the ws communication. The implementation is:

ServerSocket sv = new ServerSocket(80);

Socket client = sv.accept();

InputStream in = client.getInputStream();

OutputStream out = client.getOutputStream();

out.write(response, 0, response.length);

The server will relay the messages to the Arduino device using the serial line encapsulation. The implementation is as follows:

SerialPort serialPort = new SerialPort("COM5");

serialPort.openPort();

serialPort.setParams(2000000,SerialPort.DATABITS_8, SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);

serialPort.writeBytes(mesaj.getBytes()));

As seen in the code above, the serial communication speed is set as high, because the distance between the system and the robot was very small (< 1m). If the distance increases, then the communication speed will have to be reduced.

C. The robot control application and robot hardware implementation

The center of the robot implementation is the actuator (of type TowerPro MG996R [9]) which has a three-pin female connector: power, ground and PWM signal. The power line is connected to a separate power source as the Arduino cannot give enough power to move the servomotors, while the PWM line is connected to the Arduino pins (Fig. 3).

The robot uses six servomotors: one that controls the base rotation, two that control the arm extension and two that control the arm rotation and finally one that controls the robot tip. This gives to the robot 6 freedom degrees (Fig. 4).

Finally the robot control is done by the Arduino board, that has the purpose to send the commands received form the server to the six motors. The Arduino servo library was used for the implementation. For sending commands the myservoA.attach(x) was used to attach the Servo objects to a specific pin. After this, the myservoA.write(y) is used to rotate the servo motor.

While the WebSocket only gives the bidirectional communication channel, the messages of the communication protocol used between the Android system and the server had to be implemented. Also, a WebSocket ping/pong message had to be implemented in order to insure that the Android client was still connected to the server. Other implementation problems were mainly related to the sensor reading. When the accelerometer was used to control the robot, a certain hysteresis was necessary to be implemented as the data values can vary around certain values because on the normal (small) hand movements. This means that the control had to be implemented as a detection of quick (deliberate) hand movements and therefore it was necessary to detect a significant change in the device's position but it was also necessary to prevent sending commands to the robot multiple times. The acceleration level necessary to detect the voluntary movement of the user's hand with no error (false alerts) was chosen to be 2 through testing (Fig. 5). Also, a delay was implemented after each command was sent, as to prevent multiple commands.

Other problems that had to be solved regarded the correct handling of various network related operations (such as connection problems, errors, disconnects) or timing problems because after sending a command the serial line needs time to serialize the data and the robot needs time to execute it (meaning that a command delay is necessary to be implemented for each command to have time to be executed).

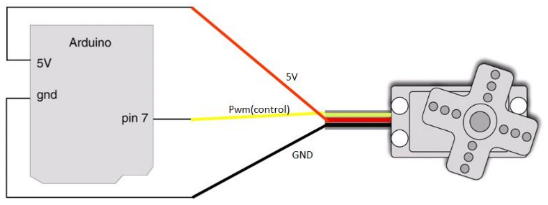

Figure 3. The actuator pins

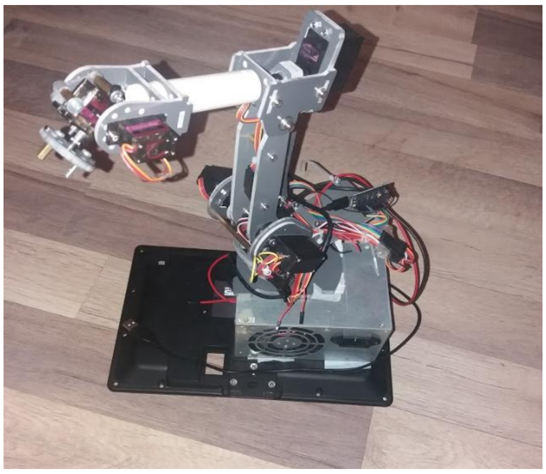
Figure 4. The six servomotor robot controlled by the system

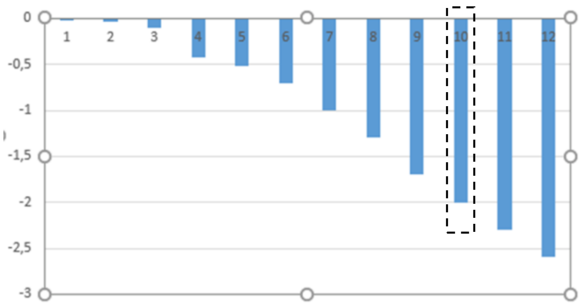

Figure 5. The evolution of the sensor x axis when tilted to the right has 2 as the level needed to trigger the robot command sending

Finally, a reset command had to be implemented so that when the user presses a reset button, the robot reverts to the initial position. The command includes reset positions for all six servomotors. As it is possible to continuously command the robot to move in one direction, a limit had to be implemented, in order to avoid any uncontrolled behavior when the robot receives a command that signals one more step even if the maximum value has been reached. Finally, for the purpose of saving the robot movement (for later playback and analysis), all the commands were record and stored in a SQLite database in Android.

IV. CONCLUSIONS

This paper presents the implementation of a control system for a hand robot using the WebSocket protocol. Compared to existing solutions, the implementation problems of robot control through internet connectivity are discussed. The results were presented and various implementation challenges were discussed.

The usage of WebSocket protocol for robot control over internet is simple to implement and has the advantage there are no major changes when switching from HTTP and firewalls will allow the traffic without special rules. The main problems with the implementation are: the need to increase the serial line speed for decreasing command latency via the serial line (between the server and the Arduino board on the robot); the need to implement a ping/pong protocol to let the server know that the client is still present; setting a motion detection level for the Android acceleration sensor data application (because of the inherent movement of the user's hand).

In the future we also intend to implement the robotic hand control with WebSocket, as Arduino has libraries that provide this support. This will also mitigate the problem that the server system had to be placed very close to the robotic hand, in order to have a high serial communication speed. In the future better motion detection will have to be implemented, as in the current version only abrupt hand movements are detected.

REFERENCES

[1] Nof, Shimon Y (1999). Handbook of Industrial Robotics (2nd ed.). John Wiley & Sons. pp. 3–5. ISBN 0-471-17783-0

[2] W3Techs, "Usage of HTTP/2 for websites". World Wide Web Technology Surveys. Retrieved October 17, 2018.

[3] IETF, WebSocket, RFC 6455

[4] IETF, Transport Layer Security, RFC 5246

[5] ROS CONTROL CENTER, web, https://github.com/pantor/ros-control-center, Accessed: 2018.10.10

[6] Sudar Muthu, Using Websockets and Android to Control Robots in Realtime, Nov 2, 2012, https://www.slideshare.net/Sudar/using-websockets-and-android-to-control-robots-in-realtime, Accessed: 2018.10.10

[7] Official jSSC (Java Simple Serial Connector) repository, https://github.com/scream3r/java-simple-serial-connector, web, Accessed: 2018.10.10

[8] Square, Inc "An HTTP & HTTP/2 client for Android and Java applications", web, http://square.github.io/okhttp/, Accessed: 2018.10.10

[9] Handson Technology, MG996R High Torque Metal Gear Dual Ball Bearing Servo, Datasheet, web: http://www.handsontec.com/dataspecs/motor_fan/MG996R.pdf, Accessed: 2018.10.10

[10] –, "Remote Controlled Robotic Arm", web, https://www.instructables.com/id/remote-controlled-robotic-arm/, Accessed: 2018.10.10

[11] –, "Wave Your Hand to Control OWI Robotic Arm... No Strings Attached", Web, https://www.instructables.com/id/Wave-your-hand-to-control-OWI-Robotic-Arm-no-strin/, Accessed: 2018.10.10

[12] Sharun Mendonca, Khalid Mohammad Zulqurnain, K. M. Abdul Razack, Mohammed Zohair, Rolwin Wilston Carlo, Multi Functional Android Controlled Robotic Arm for Drilling, Cutting and Cleaning Application, *Journal of Mechanical Engineering and Automation*, Vol. 7 No. 3, 2017, pp. 89-93. doi: 10.5923/j.jmea.20170703.06.

[13] –, "Make Wired Robotic Arm Edge to Wireless With DIY Arduino + XBee", web, https://www.instructables.com/id/Make-Wired-Robotic-Arm-Edge-to-Wireless-with-DIY/, Accessed: 2018.10.10

[14] Ammar Atef Ali, "CONTROL ROBOT ARM WITH YOUR ANDROID PHONE", March 1, 2016, web, https://create.arduino.cc/projecthub/ammaratef45/control-robot-arm-with-your-android-phone-adbfb3, Accessed: 2018.10.10