Zigbee Associated Network Based Dynamic Updation

Nilekh Chaudhari, Bhaskar Bandyopadhyay, Uddhav Arote, Swapna Borde

(Department of Computer Engg. Vidyavardhini's College of Engg. & Tech., University Of Mumbai)
Email: nilekh.chaudhari@gmail.com
(Department of Computer Engg. Vidyavardhini's College of Engg. & Tech., University Of Mumbai)
Email: bhaskarsuper9000@gmail.com
(Department of Computer Engg. Vidyavardhini's College of Engg. & Tech., University Of Mumbai)
Email: aroteuddhav@gmail.com
(Department of Computer Engg. Vidyavardhini's College of Engg. & Tech., University Of Mumbai)
Email: swapnaborde@yahoo.com

ABSTRACT

In this world of ever growing technologies, robots and the robotic technology have gained space here. Days are not far when we, humans, will be using robots for daily chores. It will be possible to control these robots from anywhere in the world just like ubiquitous web applications. In this literature we have described a way of controlling the robots from over the network, probably over the Internet. The robots are present at a remote location and these robots can be controlled from our home via Internet. Present day robots are either remote controlled or autonomous, but there are few implementations regarding robot being controlled over the Internet. The literature gives a brief idea of what system is, how the system works and its advantages. The best part of the system is a scalable protocol which communicates between two different standard communication protocols. The literature also gives an overview of the features the system provides with.

Keywords - **Client , Interpreter, Robot, Server, Zigbee**

I. INTRODUCTION

The system proposed helps in controlling in Spark V, Firebird V and its variant from the remote location over the Internet It provides a consistent and unambiguous GUI to control the robot. The protocol is independent of underlying communication technology. The system is optimized to allow for scalability. It provides onboard interpretation of the Java programs is allowed as a result of which java programs can be easily executed on the robot. The system consists of a web interface which help the user to control the motion of the robot, sets its IO port values and also gives them the chance to send batch scripts which will be executed at the robot side. The communication will be done via Zigbee protocol (IEEE 802.15.4) working on IEEE 802.15 standard of communication.

The client, a web browser, requests the Web server via HTTP protocol. The server interprets the action from the user at the client side and accordingly takes the necessary step. At the server, action packet is created and is sent via serially connected Zigbee module to similar module on the robot side. The robot interprets the action in the packet and acts accordingly.

The robot, then sends the packet to the server via same Zigbee module. The packet contains the information necessary to the user at the client side. After server receives the packet, it sends this packet data to the client in the form HTTP response. The client side user uses this data for some good reason.

II. ARCHITECTURE

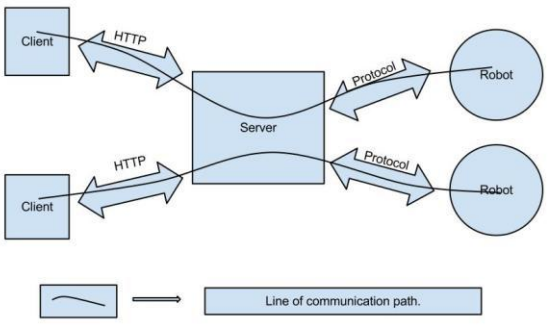


Fig.1 Architecture

The above fig represents the architecture for the proposed system. As seen in the figure, there can be multiple clients as well as multiple robots. At a time one robot can be controlled by one client only.All clients communicate with the web server

using HTTP. The data packet is then propagated to the robot via designed protocol. The reverse communication from the robot to the client happens in the same way, from robot to web server via protocol and further from server to the client via HTTP. The practical implementation has been conceived by using Zigbee as the protocol of choice while communicating between server and robot. For this, a Zigbee module has been used at server side and robot side.

III. IMPLEMENTATION

The implementation of the system consists of a server, one or more clients and one or more robots. The detailed explanation of each of these is given as under:

Client

The client consists of an HTML 5 compliant web br owser. No other added functionality is required at the client side.

Section I:

It is aligned to the left of the user interface and contains 3 divisions as follows -

Division 1: It will contain the number of robots online and who have registered to the system.

Division 2: It will contain the robots which are currently in use. These robots cannot be utilized by the other users who try to use these robots

Division 3: It will contain the messages from the server..

Section II: It is aligned to the right of the user interface and contains 5 divisions as follows -

Division 1: It will contain the canvas which graphically displays the position from the robot.

Division 2 to Division 4: It will contain the sensor values and distance vector received from the robot ,l ike IR sensors , distance sensors.

Division 5: It will contain the value related to the voltage of the battery set up on the robot.

Section III: It is in the middle of the user interface and contains tabs for various functionalities .The number of tabs is at least 3, as follows

Tab 1 [Motion Control]: It will help in controlling the motion of the robots using the key events .The velocity of the robots will also be controlled in this tab.

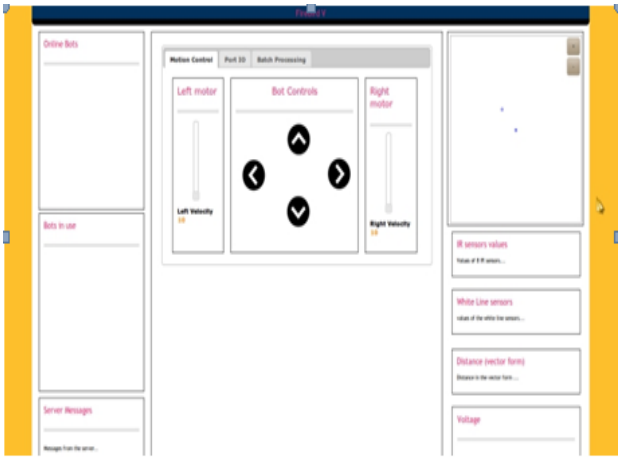
Tab 2 [IO Port manipulation]: It will help to manipulate the IO port values of the robots.

Tab3 [Batch Processing]: It will help the user to write batch scripts so that it can be executed on the robot.

Server

The server interfaces with both, the clients as well as the robot. PHP is used as the server-side scripting. A zigbee module is also connected to the server. A daemon service, programmed in C, runs on the server. The messages sent by various clients are buffered into a database by the daemon.

The buffered messages are sent serially to the zigbee module through the USBtty interface.



User interface

Code snippet shown below :

/*Communication between Server Xbee and Robot Xbee`s*/

```
main(
)
{
    Linked_List  L  ;
    Thread T1 ;
    Thread T2 ;

    startThread(T1  ,  L  ,'createNode');
    startThread(T2 , L,'processNode');

    while(true);
}

procedure   createNode()
{
    while(true)
    {
      if(dataAvailable())
```

```
{
  L.addNodeToLast(nextData());
}
    }

}

procedure
processNode()
{

while(true)
{
data    =    L.first()    ;
process(data) ;
      L.removeFirst()    ;
}
}
```

Robot

Spark V has been used as the standard robotic platform for implementing this system. NanoVM interpreter has been used for interpreting the commands sent by the server.

IV. FEATURES

1. Graphical representation of robots

The system attempts to show the real-time position of the bots with respect to a reference point in two dimensions. This is achieved by using vector math along with the timer and encoder inside the robot. The feedback obtained from robot regarding the encoder shaft gives an estimate of the distance and direction traversed by the robot. These values are then used to calculate the resultant vector and plot the position of the robot according to the equation,

The system uses HTML 5 canvas to render the position of the robots. This makes it easy to support operations like zooming and panning. The only requirement is for the browser used by the clients to support HTML 5.

2. Robot Registration

The proposed system supports many robots .The system knows about all these robots by the registration feature that has been included ,so that the clients come to know about the available of the robots.

In these the robot sends a ID packet to the Zigbee module attached to the server, as self identification. The server Zigbee module continuously senses as the TCP socket does. After the server receives the ID,it makes the entry into the MySQL database. The server after this acknowledges the request of the robot. Since the entry of the robot is made in the MySQL database, the robot name appears in the left column of the web UI.

3. Onboard Java interpretation

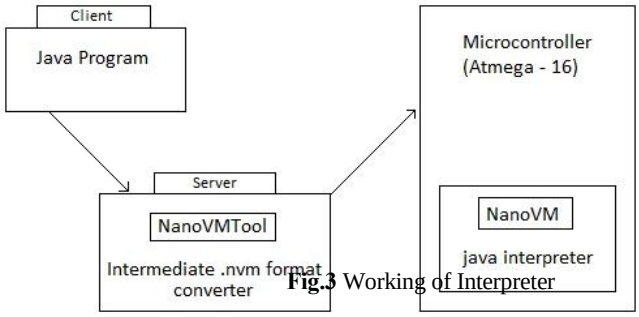


Fig.3 Working of Interpreter

The proposed system provides the feature of onboard java interpretation which use the concept the java virtual machine. The tiny version of virtual machine named NanoVM is used on the robot for interpreting the java program. Instead of regular C program here we use the java programs to specify robotic task.

We write the normal java program and upload it on the sever. The sever compiles the java program to create the CLASS file. The class file is converted into the internal file format of nanovm named .nvm file format. This converted file is then send to the robot using Zigbee module via bundles of packet. This packets are received on the robot & interpreted to perform the specified task.

4. Multiple clients

The system is designed in such a way that multiple clients can use it at a same time. The client needs to register to the system before using it. After the user registers, he gets a user-ID and a password to login into the system. The user logs in and can find the available and busy robots from the web UI provided, the left column of the web UI.

The available robots are those which are free and can be used. The busy robots are those which are already in use by some other client.

The restriction on the clients is that, a single robot can be used by only one client. But a client can use multiple robots.

5.Batch processing

The proposed system provides this feature under the tab of Batch Processing in the web UI. The main purpose of this feature is to execute bunch of task one after another instead of specifying them separately. These tasks are written in java , translated & send on the robot where they are interpreted by NanoVM. This feature can be useful for executing scripts, as well as saving and running entire bunch of statements written specifically for interfacing with custom hardware.

As the java code is compiled into bytecode by JVM, it guarantees machine independent optimization. Unfortunately the NanoVM implementation is not large enough to include a JIT compiler which could further optimize the bytecode at run-time.

Batch Processing in the web UI. The main purpose of this feature is to execute bunch of task one after another instead

of specifying them separately. These tasks are written in java , translated & send on the robot where they are interpreted by NanoVM. This feature can be useful for executing scripts, as well as saving and running entire bunch of statements written specifically for interfacing with custom hardware.

As the java code is compiled into bytecode by JVM, it guarantees machine independent optimization. Unfortunately the NanoVM implementation is not large enough to include a JIT compiler which could further optimize the bytecode at run-time.

V. CONCLUSION

In this process, we learn different elements of project building and use all techniques of programming that we have learnt in the past few years. The project makes use of programming techniques, MAC techniques, Interfacing with micro-controllers and their programming, Compiler design and web designing.

VI. FUTURE SCOPE

Implementing the concept mentioned above for all the different Firebird platform available. Enabling batch scripts support in different programming languages. To support media streaming over the robot. 3D mapping of the arena.

ACKNOWLEDGEMENTS

The authors of this paper would like to thank Prof.Kavi Arya and the staff from ERTS lab, CSE department IIT Mumbai for providing us an opportunity to work with standard robotic platforms and sharing their knowledge with us.

REFERENCES

[1] (2012) IO port programming. [Online] Available: http://www.faqs.org/docs/Linux-mini/IO-Port-Programming.html

[2] (2012) Serial communication. [Online] Available: http://www.easysw.com/~mike/serial/serial.html

[3] (2012) XBee / XBee Pro datasheet. [Online] Available: http://www.nexrobotics.com/images/downloads/Manual_xb_oem-rf-modules_802.15.4.pdf

[4] (2012) Java Virtual Machine specification. [Online] Available: http://java.sun.com/docs/books/jvms/second_edition/html/VMSpecTOC.doc.html

[5] (2012) The NanoVM-Java for the AVR. [Online] Available: http://www.harbaum.org/till/nanovm/index.shtml