Indian Institute of Technology Bombay

CS 684 - Embedded Systems

Course Project Report

Internet based Control Suite for Firebird and an Android Implementation

Group - 11

Alex Poovathingal	-	113050035
Arpit Jain	-	113050028
Rahul Kumar	-	113050066
Vibhor Kulshrestha	-	113050031

November 19, 2012

1 Introduction

Greenhouse automation was the general theme for all course projects done as part of 2012 Autumn offering of CS 684 - Embedded Systems course at IIT Bombay. Different project groups were asked to develop robotic applications using Firebird V robots that could automate any of the activities involved in Greenhouse Agriculture. A user at a remote location should also be able to control the robots by issuing commands to them over Internet.

We observed that this would result in a lot of work duplication as each group would have to program a Zigbee handler part in the Firebird robot and in a server computer. They also have to create a server application and remote user interface program that can be used to issue commands to the Firebird robot. Another problem with this approach would be that each projects would use their own protocols which would render impossible a possibility of seamless integration of different projects at a future stage.

Therefore we decided to create a common interface program that would allow the other projects to ignore all the abovesaid concerns and instead concentrate on their Firebird specific program as long as they meet a specific set of conditions. We also integrated all the projects that followed our standards with this common interface and demonstrated that it can be controlled over Internet using an Android app.

1.1 Definitions, Acronyms and Abbreviations

FireBird A Robotic platform based on ATMEGA2560.

Zigbee High level communication protocol using small, low-power digital radios based on IEEE 802 standard for personal area networks.

Xbee A family of compatible radio modules.

TCP/IP Transmission Control Protocol/Internet Protocol. A set of communication protocol used in Internet.

Android A open source Linux based Mobile operating system developed by Google.

GPRS General Packet Radio Service. Mobile data service on 2G and 3G cellular connection system.

2 Problem Statement

We plan to create a common interface program which can be used to control remotely any projects done on Firebird platform which follows the communication standards and template code that we provide.

The final product will have three parts :-

- Template Code for Firebird Program.
- Server Code for the Interface Program between Firebird Robot and Remote User Interface.
- Remote User Interface implemented as an Android app.

The template code for Firebird robot shall contain functions and Interrupt handlers for Zigbee communication. The server code is responsible to handle incoming requests from the remote user and pass these requests to the Firebird robot. The remote User Interface is an Android App which will display the list of robotic functions that can be executed and let's the user select one. It communicates this request to the server program.

3 Requirements

3.1 Functional and Non-Functional Requirements

- Robot should be controllable over the Internet. This is a functional requirement and a constraining metric. The aim is to facilitate a user who is situated in a remote location, to control the robot. The use of Internet is an optimizing metric, the system could also have used an SMS based system, but Internet was preferred for the higher communication speed and ease of development.
- The robot should be controllable by an Touch based Android phone. This is an optimization metric. Controlling the robot using touch screen would be more natural and intuitive for an end-user.
- The server should be able to register and support any robot which follows the communication standards. This is inherently satisfied if our template code for Firebird and our Android app are used. If a project requires not to use them, then they should satisfy that the remote program or Firebird program that they make satisfies our communication requirements.
- The communication delay involved in the transactions between remote user and server program should be minimized.

3.2 Hardware Requirements

- Firebird Robot
- Xbee Module
- Android Mobile with Internet Connectivity
- Server system with Internet Connectivity and a Xbee module

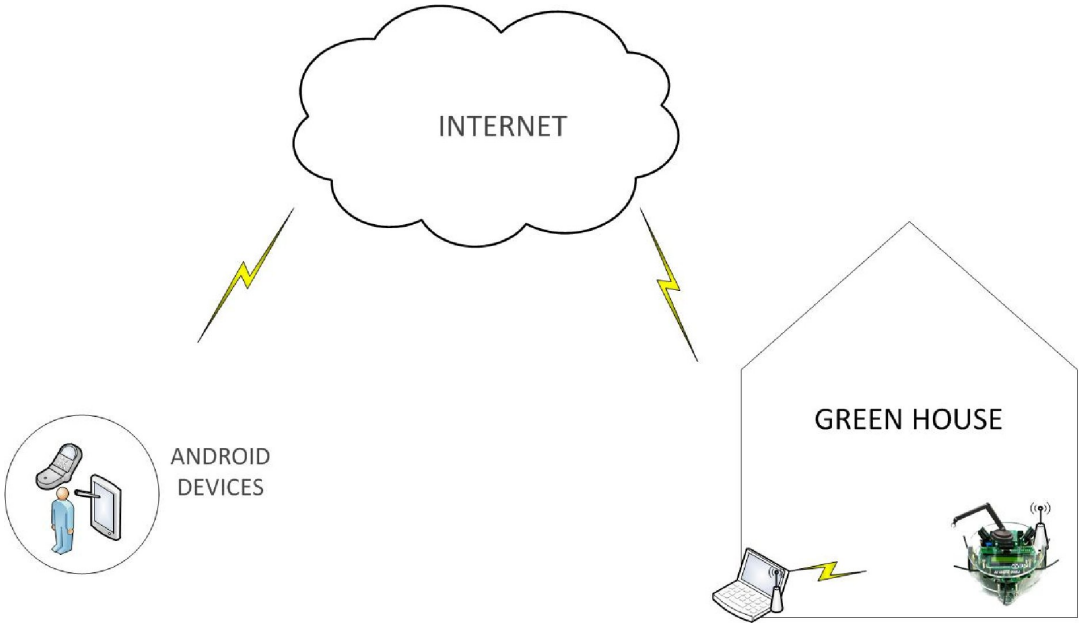

Figure 1: System Design

4 Implementation

The system design involves three interacting parts - the firebird program, a server program and an android app for remote user.

4.1 Communication Protocol

4.2 Firebird Template Code

The Firebird Template code, initializes a global variable botId to a unique number assigned to each Firebird robot. This botId is also used for registering the bot with the server. The robot then initializes all the devices that it uses including UART0 for Zigbee communication. Then the robot goes to an inifite loop waiting for a request to appear from the server through the Xbee module.

The standard communication format for the message from Server to the Firebird Robot is "botId$funCode$par1$par2$par3#" where botId is a unique number assigned to each bot. Using this number, a bot verifies whether a message it received was actually directed at it. funCode is a number corresponding to function on that bot. When an input arrives through the Xbee module, the Xbee Interrupt handler populates an array with the values received, until it receives a '#'. When '#' is received, it confirms whether the botId in the message is same as the botId of the bot and if it is, the function corresponding to funCode is called with parameters par1, par2, par3 ,etc. This is performed by the function *function_caller()*. If the botId was not the same, then the message is discarded and the robot waits for the next message.

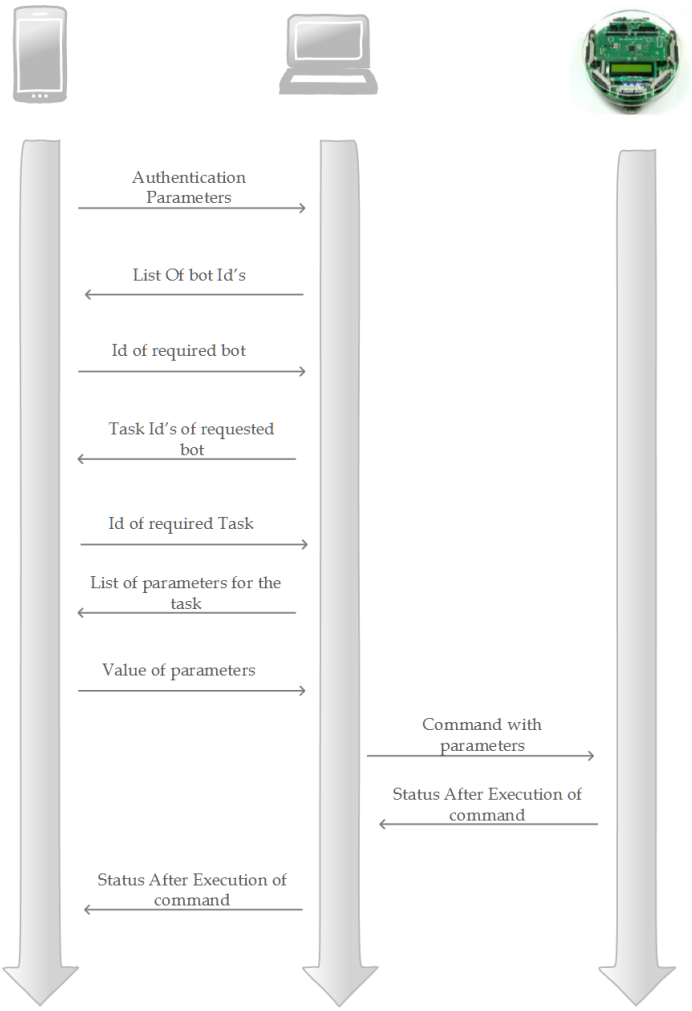

Figure 2: Communication Protocol among app, server and bot

4.3 Server Program

- **Main.php:**
  This php script when called, establishes a connection to database which resides in server.

- **authentication.php:**
  This php script authenticates the user using the username and password supplied by the user. Main.php and authenticate.php together checks whether the user is authentic user or not. If yes, then session is proceeded to Bot selection page, else error message is displayed.

- **bot_id_send.php:**
  This php script when called, sends all the available Bot IDs to the android mobile. From these options, user chooses the Bot he needs to control.

- **task_send.php:**
  This php script when called, sends all the available Task IDs for the selected Bot to the android mobile. From these options, user chooses the Task which he wants the Bot to perform.

- **param_send.php:**
  This php script when called, sends list of all the parameter names corresponding to the selected Bot and Task to the android mobile. User is provided with text boxes to fill in the values for the required parameters.

- **Message_pass.php:**
  This php script takes the string containing Bot ID, task ID and parameter values which are separated by '$' and passes on to newface.py which instructs the corresponding Bot to do the required task with the supplied parameters using Zigbee module.

- **Interface.html:**
  This provides the web-interface to control the Bots over internet using a browser.

- **register_page:**
  This provides the web-interface to register new Bots and their respective tasks to the system. This also needs the parameter of task as input and stores them to a database maintained at server.

- **newface.py:**
  This python script is called with the message to be sent command line argument. A socket connection is established with the XBee module on the Firebird robot and this message is passed to it. If any status messages are returned, the database containing status messages is updated. When the robot returns a '#' at task completion, the script terminates.

4.4 Android App

App consists of following 5 files, each of which represents a user screen on the phone.

- **MainActivity.java:**
  This page simply present a login screen to user and take the input from the user. It then creates an http connection with the server. Then it calls the main.php file from the server that in turns authenticate user and sends back a list of bot is retrieved from the database.

- **Interface.java:**
  This page presents the user with the list of bot id's, one of them can be selected and the selected id is sent back to server after establishing an connection again with the server by calling bot_id_send.php. The php file retrieves the tasks associated with that particular bot id.

- **task_Id.java:**
  This page presents the user with the list of task id's, one of them can be selected and the selected id is sent back to server after establishing a connection again with the server by calling task_send.php. The php file retrieves the parameters associated with that particular task id.

- **getParameters.java:**
  This page presents the user with the list of parameters, user fills in the values of the parameters and full command (comma separated bot id, task id , parameter, parameter value, ...) is sent back to server after establishing an connection again with the server by calling message_pass.php.
- App then waits for the finish status from the server. After getting the status it comes back on getParameters.java page.

5 Deployment

5.1 Firebird Robot

Given below are the places in the Firebird template code where modifications might be needed.

- Add the project specific functions and to the part marked with *PROJECT_CODE* in the template code provided.
- Modify function_caller() function as per the requirements of the project.
- In function init_devices() call the port and device initialization functions needed by the project.
- In main(), modify the value of botId to the unique botId using which the robot was registered.
- In uart0_init() function, modify the value UBBR0L from 0x5F to 0x4F if your Firebird Frequency is 11059200Hz and not 14745600 Hz.
- If any of your tasks needs more than 5 parameters or needs parameters with size more than 5 characters, then modify the array declaration of fcall to the required value.

Also, the Xbee module in the Firebird robot will have to configure the server Xbee module's address as its destination address.

5.2 Server

- Install Apache2 and php5. For ubuntu, use following command:
  sudo apt-get install apache2 php5
- Make a directory in home directory of apache(which is /var/www).
- Place all the php and python files in that folder.
- Give read/execute permissions to the folder and all the files in it. For ubuntu use the following command:
  sudo chmod 755 -R /var/www/folderName
- Connect the zigbee module to USB and check which USB port it is attached to. Give read/write/execute permissions to that port. For ubuntu use:
  sudo chmod 777 /dev/ttyUSB0 (if USB port is USB0)

- Provide a static public IP to the server machine so that is may be accessed from anywhere using Internet.
- Now setup is ready to execute the command using the android mobile phone.

5.3 Android App

Phone requirement :

- Touch screen android phone
- Installed OS : Minimum Android 2.3.3
- 2G/3G or wi-fi Connectivity

- Change the source files with IP of the server running in Green house.
- Copy the apk file on phone (via bluetooth or data-cable, internet etc.)
- Install the file and give it internet connection permission.
- Launch

6 Testing

Our interface program was integrated and tested successfully with all the projects that wanted to integrate (6 no's). The testing was done by the following methods.

- Control the robot from the Android App using 2G connectivity.
- Control the robot using the web interface from Institute's internet network and from an external network.
- Several bots were controlled simultaneously making sure that a message to a particular bot was ignored by the other bots.
- Verified that projects following our Firebird code template could be seamlessly integrated to the interface program.

7 Discussion of System

7.1 Things that worked as planned

- **Integration** - Integrating our interface with the projects that followed the guidelines didn't bring about any challenges.

7.2 Things that did not work as planned

- **Zigbee communication** - The XBee communication does not provide inbuilt buffering. This led to loss of data when data was written sequentially one after the other without delay. This was corrected by monitoring the flags and delaying the write operation until the previous write was completed.

- **Serial Communication** - We planned to program the server as PHP scripts only. This gave rise to some issues. Considering the devices as files and doing I/O as file operations weren't efficient because the file descriptor had to be closed for a write to complete. This wasn't the behavior that was required. Hence, we used a external library - phpSerial which did just that. This too wasn't a suitable solution as they supported only write operation to serial port and don't support read operations. We needed to read from the serial port to confirm whether an action was performed by the robot and to receive the status messages sent by robot. Therefore, we went for a Python based solution, where the Python scripts that handles serial communication was called the PHP script.

- **Status Message Update** - We wanted the Android mobile app to display in real-time any status message that has been passed on to it from the firebird robot. This couldn't be done in a straight forward way in our design because we used a PHP based system. The PHP scripts, even if it gets a status message from the robot, it sends the response back to the user only after the entire process is completed. To overcome this, we created a new database for storing the status messages. The Python scripts reads any status message and writes it to this database. These messages are not passed on to the PHP script. The android app could poll this database for new status messages by calling another PHP script. The final android app would send a request, wait for a response from the server and in mean time, it will keep calling this PHP script to check if there are any new status messages. Only database specification and PHP script for this have been developed at this point of time. This functionality has not yet been added to Android application yet.

- **Interrupt Handlers** - The UART0 signal handler disables all other global interrupts while it executes. As the tasks are called by this, these tasks were also run in an environment where interrupts were disabled. Due to this, some projects that relied on interrupts for tasks like handling wheel encoders, didn't work in our system. Therefore, we had to artificially enable the interrupts before the tasks were called.

- **Multiplexing Zigbee channel** - In the set of standards to be followed to interface with our projects, we had specified that the Zigbee channel will be used extensively by our code and cannot be used for any other purposes. This wasn't a fair requirement because there were some projects which required to communicate with other devices through the Zigbee module. Due to this, it wasn't possible to interface one project with ours. As we had already specified the standard beforehand, it was not possible to change it to suit their requirements too. This problem should be handled later.

8 Future Work

There are several features that should be added to this product before it becomes worthy to be deployed in an actual greenhouse.

- **Porting** - The user interface should be ported to other popular devices like iPhone, Aakash tablet, etc.
- **More Abstractions** - Instead of selecting a robot and a task as is being done right now, the user should be able to select a trough and select on a task to be done on that trough. The server system can figure out which robot does the required task and execute the task.
- **Multiplexing Zigbee Channel** - The Firebird program should be allowed to use the XBee module to execute the tasks.
- **More Interactive UI** - The Android App can be made more interactive by displaying real-time status messages and making the Firebird robot responsive to commands even while executing another task.
- **General Server** - This project was required to create a server for a particular greenhouse having several robots. This can be modified to a server that can different greenhouses, each of which will have a local system for Zigbee communication. This shall require multiple users logging in at the same time into the server.
- **Template Code Generator** - A web interface which would let you download Firebird Template code specific to the project if the robot and project details are provided to it.

9 Conclusions

The project aimed to put forward a common platform for unified control of heterogeneous robotic systems working in a space that comes in the range of a XBee module (e.g. green house, house automating etc.) from anywhere around the globe. We created a basic version of a such system and demonstrated its working. This presents an opportunity to create higher abstraction over the system and support higher level primitives (e.g. in a greenhouse, we can create a primitive that "a selected turf should be irrigated or seeded" without fiddling with the lower level details as selecting bot id, task id etc.). As this project gets refined in successive iterations over the coming years, we expect that verything related to the user control can be abstracted out completely and the users shall need to program only the logic for their robotic task.

10 References

1. **Android Developer's Page** - http://developer.android.com/sdk/index.html
2. **XBee/XBee-PRO RF Modules Datasheet** - http://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf