System Requirements Specification
CS 308 Embedded System Lab

09005003 : Saif Hasan
09005010 : Chinmay Chauhan
09005013 : Sagar Chordia
09005015 : Hemant Gangolia

February 18, 2012

1 Introduction

This project essentially enables person to control Firebird5 robot remotely from any place in the world. Live video and audio streaming from bot will be available to assist this remote controlling.

1.1 Definitions and Abbreviations

- GUI- Graphical User Interface
- TCP - Transmission control Protocol
- UDP - User Datagram Protocol
- IP - Internet Protocol
- USART - Universal Synchronous/Asynchronous Receiver/Transmitter
- USB - Universal Serial Bus
- FB5 - FireBird5 robot
- SDK - Software Development Kit
- ADK - Google Accesory Development Kit
- API - Application Programming Interface

1.2 Requirements

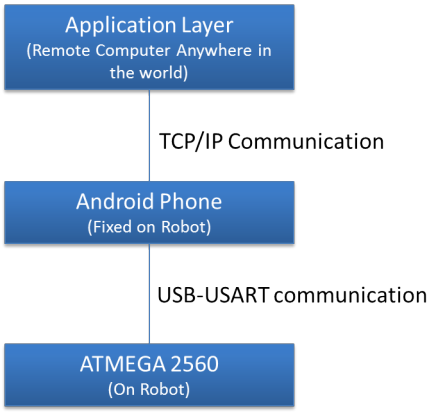
- FireBird5 Robot with Atmega2560
- IOIO board acting as USB host and providing USB to UART communication
- Android mobile phone with camera, speaker and mic functionalities
- Android version should be greater than 2.3.4 and shold support USB client mode.
- Internet connectivity to android phone.

2 General Description

2.1 Overall Description

Remote user will run an Application having interface to control various movements and sensors of robot. These messages will be passed to robot through internet. Robot will have Android enabled device to receive and process these messages. Finally Android will communicate to Atmega2560 which will regulate the bot.

2.2 Modules and intercommunication between them



Three Layer of Processing

- Application layer: Application installed on remote computer, user will interact with this application.
- Android phone: Remote application will be interacting with Android based application installed on mobile device. Android acts as Wifi gateway. Compactness of android device with camera, mic makes it best candidate as wrapper device to Atmega.
- Atmega2560: Android phone will communicate with Atmega2560 and control robot in this case FB5.

Two level of Communication:

- TCP/IP communication: Will be used between remote application layer and android layer. Video control and audio control will be passed through this using UDP packets. Signals to control bot from remote application will be passing through this.
- USB/USART communication: Between Android layer and Atmega layer. Signals to maneuver robot and activate sensor values will be passed through this layer.

2.3 Product Functionalities

The communication channels used are TCP/IP connection and USB-UART communication, These are very generic and hence various other applications using this kind of architecture can be developed Some typical applications are:

1. Remote manuevering of robot
2. Spy over enemies using bot. We can also listen to their communication
3. Real time survelliance in critical regions.

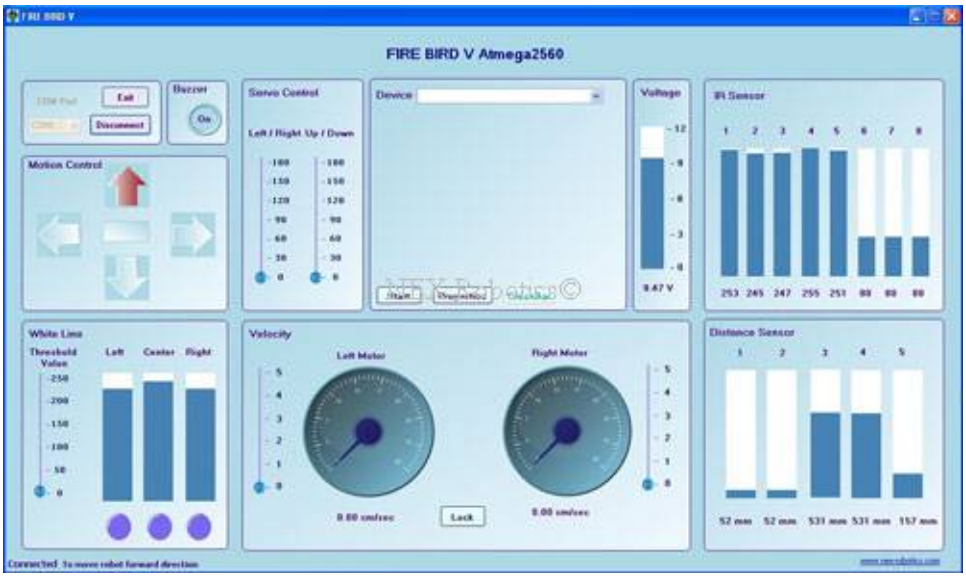
2.4 General Constraints

Firebird5 must be in range of Wifi or 3G to enable communication to remote client. Video and Audio transfer depends on strength of available network bandwidth.

3 Details

3.1 Remote client application

We plan to implement GUI to control bot having following functionalities

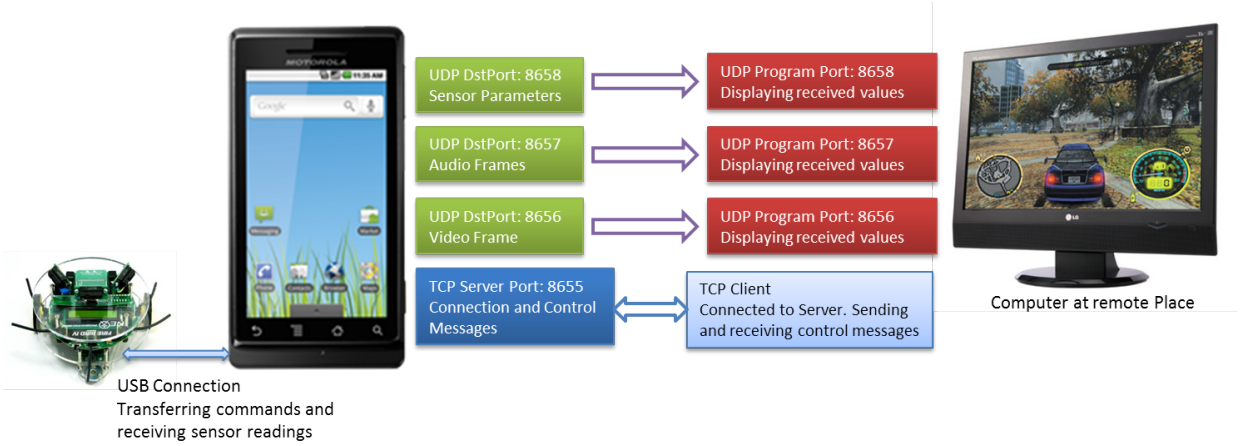


- Connect/Disconnect
- Motion control buttons (left, right, front, back)
- Buzzer on/off
- left/right motor velocity control
- All Sensor readings. Sensors would be classified as per their utilities
- **Option to upload a custom program on the bot**
- Voltage reading
- Servo Control
- Video Feed that the user will get back from the android phone attached on the bot
- Audio input channel, to communicate sound to robot.

Some sample programs which are used most frequently would be built-in the Atmega. And depending on message passed particular function on Atmega will be invoked. There would be live video feed taken back from the bot using the android phone attached. We plan to implement it based on the guidelines given here

`http://developer.android.com/guide/practices/ui_guidelines/index.html`

3.2 Networking aspects of Client Application (Foreign Computer)



Green and Red Packets are Sent and received UDP packets
Dark blue box represent TCP Server socket program
Light blue packet represent TCP Client which initiate connection to TCP Server socket program

- User has to enter IP address of robots android phone in order to establish connection. Application will establish TCP/IP connection to remote host on port 8655 where one server process will already be listening
- This application basically spawns three sub processes which controls the whole application.
- Application has four main network components which are :
  - **TCP/IP Connection** : This is the same connection as prior one (server port : 8655). Using this connection user sends control messages like. E.g start video, stop audio, set sensor on, set sensor off, take left turn, increase speed
  - **Video Processing Thread** : This thread listens on port 8656 for UDP packets. This UDP packet contains the video frame. Thread process the frame, converts it to appropriate byte array and display to user using Java's Image IO library
  - **Audio Processing Thread** : Thread listens on port 8657(UDP) for audio UDP packets, which will be sent by android phone. Thread reads the packet process it and converts to appropriate format which is then played using Java Sound API
  - **Parameter Sensing Thread** : Thread listens on port 8658 (UDP) for special packets which contains sensor parameters like motor speed, direction

- We are sending sensor values to user using UDP cause we need to send it at a fixed rate and loss of some packets is tolerable

3.3 Networking aspects of Host Application (Robots Android)

- This Application has one TCP Server Socket program which listens to port 8655 for incoming connection. This program allows only one connection at a time, so all other request wont be entertained while someone is connected

- Once some request is accepted, this application starts receiving commands from user on this Connection itself. It also sends some Acknowledgements packets like video started etc.

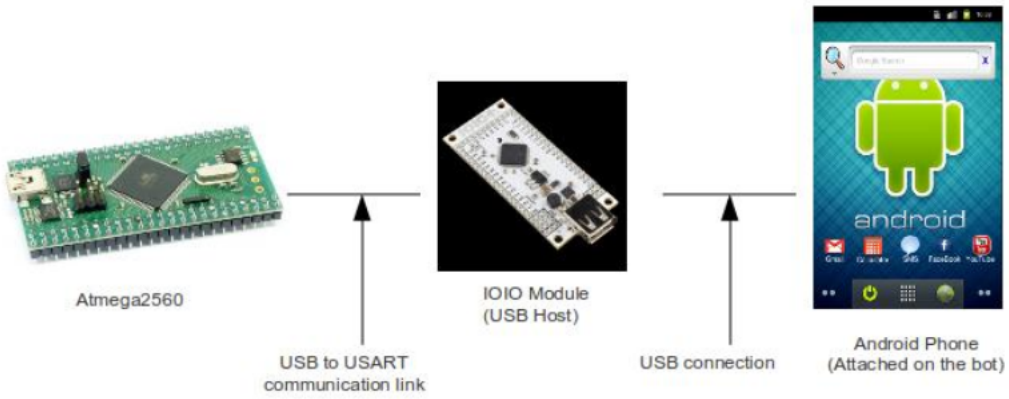
- Simultaneously following threads may be started

  - **Video Recorder** : When user request for video streaming then video recorder will be initiated. This thread first initiate camera on android, starts receiving raw video data in mpeg frames, encapsulate in UDP packet and send to user's computer on port 8656.
  - **Audio Recorder** : On user demand main thread initiate this thread. This thread initiate mic on android and starts reading voice data in byte format. It sends this data to host on port 8657 using UDP protocol which is then played on remote computer
  - **Parameter Sender** : This thread continuous keep sending the sensor readings from robot to user on port 8658 in UDP format

- For reading raw frames of camera and audio data, Application uses advanced library provided by Android in API-15

- Whenever user disconnects the slave, all threads will be killed

3.4 Android to Atmega communication



Option1 - Bluetooth communication

1. Bluetooth module needs to be attached to Firebird5.
2. Android supports Service Discovery Protocol (SDP) for blutooth communication, but bluetooth module on FB5 does not provide SDP service.
3. So raw data transfer mechanism using RFCOMM is needed.
4. **Problems** High latency, Lower bandwidth, consumes more power, Not reliable.

Option2 - USB-Usart Communication

1. Android phone communicates to external USB hardware (an Android USB accessory) in "accesory mode".
2. USB to USART converter.(External Hardware)
   - **Microbridge** Supports only Arduino boards. Tricky since we need to burn arduino code on atmega.
   - **IOIO board** from sparkfun supporting Android ADK - We will use this.

Concept

Android ADK platform The Android 3.1 platform (also backported to Android 2.3.4) supports Android Open Accessory. The IOIO board contains a single MCU that acts as a USB host and interprets commands from an Android app acting as USB client. IOIO has PPSo pins which will communicate to Atmega using UART or SPI communication. `https://github.com/ytai/ioio/wiki/Getting-To-Know-The`

3.5 Atmega application

- Application will contain Data structure (tentatively decided as an array) containing instructions obtained from the Android phone.
- The data will be obtained in the form of signals as and when occurred on the bot.
- The respective signal handlers will be dispatched and the data will be transmitted/received via UDR2 USART2 I/O Data Register.
  - SIGNAL SIG_USART2_RECV
  - SIGNAL SIG_USART2_TRANS
- The array will be of the following types
  1. Motion Control - To navigate the bot
  2. Sensor Control - To modify sensor values
  3. LCD Control - To display messages on LCD screen
- Motion Control array will be a unsigned char array which will contain values as follows:
  - 1st - Forward/Reverse bit
  - 2nd - Left/Right bit
  - 3rd - Left Motor Velocity Control bit
  - 4th - Right Motor Velocity Control bit
- Sensor Control array will contain boolean values for turning sensors on/off. These sensors can be IR sensors, Proximity Sensors, Sharp Sensors, etc.
- LCD Control array will contain a message to print on the LCD screen. The type of the array will be decided by the header of the main data array.
- In the main function, inside the infinite loop, these values will be used to call appropriate functions to manuever the robot.
- The functions will contain the code for moving the bot, lcd display, reading and writing registers and other purposes.

4 Quality Control

- The TCP messages sent/received provide reliability.
- Usage of UDP protocol for video feed ensures real-time displaying of video on the remote computer.
- The design will provide robustness to the bot and ensures the safety of the Android phone as well as other peripherals.
- Mic enabled Android phone will provide speech-recognition and also helps in acknowledgements of buzzer signals.

5 Risk Management

- The TCP connection between server and client is not encrypted so somebody might tamper with control messages as well as spy on them. Fall Back Plan: Use authentication and encryption
- The data of video frames, audio frames and robot state variables are being sent to foriegn host using UDP and they are unencrypted so somebody can read those value as well as may tempter with values too. Fall Back Plan: Use session key for encryption which is unique per session
- The camera feed from the android phone will require a good bandwidth to transmit the raw video feed to the computer in order to allow user to view this feed.
- The UDP connection should have enough bandwidth to send/receive the feed in real-time.

6 Design Constraints

- Clarity of camera and mic
- Android must have 512MHz processor to support this application (since multiple threads are running)
- WiFi range limit

7 References

References to some documents

http://eyantra.com
http://developer.android.com/guide/topics/usb/adk.html
http://code.google.com/p/microbridge/
http://www.sparkfun.com/products/10585
https://github.com/ytai/ioio/wiki/Getting-To-Know-The%20Board