University of Applied Sciences Koblenz

Department of Electrical Engineering and Information Technology

The University of Western Australia

School of Electrical, Electronic and Computer Engineering

Wireless Networks for Mobile Robots

by

Christian Schmitz

A thesis submitted for the degree of

Diplom-Ingenieur Elektrotechnik (FH)

Supervisors: Prof. Dipl.-Inf. H. Unkelbach

A/Prof. T. Bräunl

February 2005

Declaration

I hereby declare that this project was done by myself and that I have not used any resources beyond the given.

Perth, 15.02.2005

_____

Christian Schmitz

Abstract

This thesis deals with wireless networks for mobile robots and their realisation in *EyeNet*. *EyeNet* was an already established network for robots, built with the *EyeBot* controller. To introduce newer technology with their advantages of higher data rates and included protocols to the robot network, Bluetooth and WLAN were used to build new versions of *EyeNet*.

The thesis firstly shows which modifications on the *RoBIOS* operating system and its system library had to be accomplished. For both Bluetooth and WLAN basics on these technologies are given. Furthermore, the used hardware, the network topology and lastly, a remote control and monitoring program are explained in each case. Finally, a comparison shows advantages and disadvantages of both network solutions.

The result of the project are two new versions of *EyeNet*, using either Bluetooth or WLAN as transport medium, which provides higher data rates, especially needed for image transmission. Both versions should provide the network functionality for mobile robots for future projects.

Acknowledgements

I would like to thank and mention some people for their help and support during this project.

First, I would like to thank my supervisor Associate Professor T. Bräunl for giving me the opportunity to write my thesis at the Mobile Robots Lab of the University of Western Australia. He also provided me with the required hardware, which was not inexpensive.

I am grateful to my supervisor Prof. Dipl.-Inf. H. Unkelbach, who introduced me to Prof. Bräunl. I also appreciated his supervision and guidance in some difficult aspects of the project.

For his help and the delivered information on the serial port, I want to thank Prof. Dr. N. Schultes.

I also would like to thank my friends and lab fellows for their help as well as for the good times we had besides the work. Thanks to Minh Tu Nguyen, Adrian Boeing and especially Louis Gonzalez.

A special thankyou to my girlfriend Melanie for her inexhaustable patience, friendship and love. She is a source of happiness and encouragement.

Finally, my major thanks goes to my parents, Werner and Doris, and my brother Marcus, who supported me in any possible way throughout this project as well as my whole study. I also want to apologize here to my mom for being abroad for another 7 months.

Table of Contents

List of Figures

List of Tables

Abbreviations

ACK	Acknowledgment
ACL	Asynchronous Connection Less
AP	Access Point
AR_ADDR	Access Request Address
ARPA	Advanced Research Projects Agency
ARQ	Automatic Repeat Request
BD_ADDR	Bluetooth Device Address
BPSK	Binary Pulse Shift Keying
BSS	Basic Service Set
BT	Bluetooth
CCA	Clear Channel Assignment
CCK	Complimentary Code Keying
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CTS	Clear To Send
CTP	Cordless Telephony Profile
DAC	Device Access Code
DCF	Distributed Coordination Function
DS	Distribution System
DSSS	Direct Sequence Spread Spectrum
DUN	Dial-Up Networking Profile
ESS	Extended Service Set
FDM	Frequency Division Multiplexing
FHSS	Frequency Hopping Spread Spectrum
FTP	File Transfer Protocol
GAP	Generic Access Profile
GIAC	General Inquiry Access Code
GOEP	Generic Object Exchange Profile
GUI	Graphical User Interface
HCI	Host Control Interface
HF	High Frequency
HR-DSSS	High Rate Direct Sequence Spread Spectrum
HSP	Headset Profile

IBSS	Independent Basic Service Set
ICMP	Internet Control Message Protocol
IntP	Intercom Profile
IP	Internet Protocol
ISM	Industrial Scientific Medical
LAN	Local Area Network
LAP	Local Area Profile
LBT	Listen-before-talk
LCD	Liquid Crystal Display
LLC	Logical Link Controller
LMP	Link Manager Protocol
LT_ADDR	Logical Transport Address
L2CAP	Logical Link Control and Adaptation Protocol
MAC	Media Access
MAN	Metropolitan Area Network
MSB	Most significant Byte
MSPP	Multi-Serial Port Profile
OBEX	Object Exchange
OFDM	Orthogonal Frequency Division Multiplexing
OPP	Object Push Profile
OSI	Open Systems Interconnection
PC	Point coordinator / Personal Computer
PCF	Point Coordination Function
PHY	Physical (Layer)
PLCP	Physical Layer Convergence Protocol
PM_ADDR	Park Mode Address
PMD	Physical Medium Dependant
PN	pseudo-random numerical sequence
PPP	Point-to-Point Protocol
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RFCOMM	Radio Frequency virtual Communications
ROM	Read Only Memory
RTS	Ready To Send
SCO	Synchronous Connection Oriented
SDAP	Service Discovery Application Profile
SDP	Service Discovery Protocol
SIG	(Bluetooth) Special Interest Group
SP	Synchronisation Profile
SPP	Serial Port Profile
SRAM	Static Random Access Memory
SSID	Service Set Identifier
TCP	Transport Control Protocol
TCS	Telephony Control

TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
UHF	Ultra High Frequency
WAN	Wide Area Network
WAP	Wireless Application Protocol
WECA	Wireless Ethernet Compatibility Alliance
WEP	Wired Equivalent Privacy
WiFi	Wireless-Fidelity
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network

Chapter 1

Introduction

One of the first technical communication networks was formed by the telegraph lines in the 19th century. Already at that time there was a need for long distance communication. When the first computers arose in the middle of the 20th century, a new age for information processing as well as for communication began. Firstly, computers were expensive and rarely and the few computer systems were centralized. In 1958, ARPA (Advanced Research Projects Agency) was founded as part of the US Department of Defense for research on communication issues. Their task was to create a reliable command-and-control network that could survive a nuclear war. As a result, the decentralized ARPANET emerged. Containing only 4 nodes in the beginning, the ARPANET grew very fast. In particular, many universities showed high interest in the network. Later the ARPANET and other networks on other continents were connected together and the collection became the Internet [33].

With the rapid increase in computers, the need for intercommunication between them grew, especially in the last two decades. Today, computer networks can be found almost anywhere. Whether at home, at universities, in office buildings or other places computers are connected to private networks or the Internet. The fact that the approximately 4 billion available IPv4 addresses are running out, and IPv6 will be the future version, shows the potential and importance of computer networks [27] [1]. Besides pure computer networks, a new need for connectivity arose in the past few years with today's numerous electronic data devices such as cell phones and PDAs.

Computer networks have now been established for a long time. Bringing the advantages of network communication, particulary wireless, to the mobile robot world was the

intention of this project.

The first chapter outlines real life scenarios for today's networks including stationary wired, stationary wireless and mobile wireless networks. A further section then links the world of networks and the world of robotics together and presents particular examples of multi-robot systems in networks. The tasks for this project are then enumerated, and finally a thesis outline is given.

1.1 Networks Today

As mentioned above, networks can be found almost everywhere nowadays. The following three sections will briefly present examples for stationary and mobile, wired and wireless networks.

1.1.1 Stationary Wired Networks

The first networks that were created belong to this category and examples include the telegraph network, the modern telephony network, and any kind of today's PANs, LANs, MANs and WANs. Personal area networks, as the name suggests, connect personal devices within a range of a few meters together. Local area networks connect computers and other IT accessories in environments such as office buildings and universities. Different LANs, i.e. from one company, form metropolitan or wide area networks. The expression MAN is used within ranges of cities (approx. 10km) and WAN covers the range of countries and even continents [33].

Copper cables or fiber optic cables are used as a transport medium in these networks. Fiber optic cables support higher bandwith but are more expensive than copper cables. Fiber optics are largely used in high performance backbones[1] whereas copper cable is used on the 'last mile' due to its robustness. Both have their specific advantages and disadvantages and more information on this aspect can be found in [9] [33].

[1] A central network connecting other networks and carrying high density traffic is also refered to as *backbone*.

1.1.2 Stationary Wireless Networks

The term ‘wireless networking’ is often used synonymously with mobile computing, but there is a difference [33]. Wireless networks do not require any mobility and can also belong to fixed stations. An example might be a direct radio link between two office buildings, that connects a building to the companies network. In old buildings the use of wireless network equipment might also be cheaper than the installation of wires. Even satellites in a geostationary orbit can be considered an example of stationary wireless networks.

In general the wireless technology is based on radio wave transmission. Infrared transmission is another method but is restricted to direct line of sight. The infrared waves are shorter and behave more like light and therefore do not pass through obstacles. Light wave transmission behaves in a similar fashion.

1.1.3 Mobile Wireless Networks

The importance and distribution of mobile wireless LANs has rapidly increased in the past few years rapidly. One reason for the success of mobile wireless networks is the increasing numbers of mobile IT devices such as Laptops, PDAs and cell phones. Hence, the number of applications for these networks is not surprising. The synchronization of information between wireless devices is a common task today. Wireless LANs are also used to set up small networks quickly within temporary environments such as a conference in an office. Recently, almost every airport started to provide a wireless network for their customers. The combination of wireless networking and mobile computing led to unbounded information processing. Most of today’s mobile wireless LANs are based on the IEEE 802.11 standards or Bluetooth. A small field also uses Infrared. Besides the mentioned networks, the mobile telephone networks such as GSM and UMTS, and also military communication networks represent further mobile wireless networks.

1.2 Cooperating Mobile Robots in Networks

The advantages of common networks can be and already have been [8] [7] used within robotics. A network between a group of mobile robots allows:

1. Communication between two or more robots.
   Position data for an example can be exchanged between robots.
2. Communication between a robot and a host PC.
   - PC provides higher computation performance, i.e. for image processing.
   - Sensor monitoring is easier on a PC rather than on a microcontroller.
   - Remote control program running on a PC allows full control of any robot.
   - Monitoring robots (position, orientation, status)
3. Inclusion and exclusion of robots from the networks.

Single acting robots are still more common than multi-robot systems and can be found in a wide range of applications nowadays. This might change in the future because a group of robots provide outstanding advantages such as [38]:

- Reliability
  The functionality is not based on only one robot. Therefore, if a single robot fails in the group, the task can still be processed by the remaining group members.
- Performance
  Tasks may be processed faster by several robots rather than by a single robot.
- Distributed functionality
  Some applications require sensor values from different places. An example for that is the creation of region maps.
- Costs
  A larger number of simpler robots can use serially manufactured parts and decrease the costs.

In particular the mobile wireless networks are of special interest in the field of robotics. The following sections show real life examples of cooperating mobile robots of today and tomorrow.

Figure 1.1: CIIPS Glory Robot Soccer Team

1.2.1 Robo Soccer

Probably the most famous events where several groups of robots cooperate is one of the several annual robot soccer competitions [2] [29]. 'RoboCup' is one of these where a different number of robots, depending on the league, have to interact. An intelligent interaction can only be established if a communication system is provided. Therefore, robots that belong to a group have to form a network to exchange information. The main information that is exchanged between the team members is:

- Own robot's position
- Position of team members
- Position of opponents
- Position of ball
- Driving commands

Figure1.1 shows the CIIPS Glory, the University of Western Australia's soccer team.

1.2.2 Industrial Robots and Agents

Industrial products, such as cars, are nowadays largely assembled by industrial robots. The material supply for the industrial robots can be performed by mobile robots called agents. The fixed industrial assembly robots send requests to the agents. Depending on the requested job the agent who is most appropriate to it will process the job. Criteria like the maximum load or the speed will decide whether an agent processes a command or not. In a working environment like this the robots are networked [35].

Another example for cooperating industrial robots is the solution of tasks that cannot be handled by a single robot. This may occur if heavy things have to be lifted and a robot has to request help. During the lifting process both robots have to give each other feedback. The start and stop times, position and velocity are the major information that are transmitted between them.

1.2.3 Exploration Robots

A group of robots has the ability to explore a larger range than a single robot. Equipped with communication modules such robot groups might be used in future space missions to map a planet's surface [36]. The development of a rescue robot group is also in its infancy. Similar to the robot soccer competition a rescue competition was created recently [2] [29]. The task is to locate injured people in a destroyed environment.

1.3 Project Tasks

As illustrated in the next chapter, a network solution for the EyeBots was already existent. Network routines were already implemented and put into the RoBIOS operating system library. Based on the preconditions, two new versions of the radio functionality had to be implemented using Bluetooth and WiFi as a base. Due to the numerous previously written applications using the RADIO routines, backward compatibility to those should be warranted. Therefore, the bodies of the RADIO functions included in the library had to be rewritten. To have the same facilities as before, a remote program had to be implemented to monitor and control any robot within a network. The project finally covered the following objectives:

1. Rewrite of the RADIO functions providing backward compatibility
2. Rewrite the background RADIO routine for the EyeBot
3. Implementation of an EyeBot setup menu for both Bluetooth and WiFi modules
4. Implementation of a router for routing messages within the Bluetooth network
5. Re-implementation of a monitor and control program (BT & WiFi)

6. Implementation of direct image transfer
7. Visual display of the robots' positions
8. Technical documentation for EyeNet

The memory space for the RoBIOS on an EyeBot is limited. Thus, another goal to achieve was the avoidance of two different versions for the RADIO routines which would have lead to two different RoBIOS versions.

1.4 Thesis Outline

This thesis starts with a short introduction about networks in general and robot networks and where they can be found in particular.

Chapter 2 describes then the EyeNet including a short outline of the EyeBot and its operating system RoBIOS, as well as the RADIO library.

Chapter 3 presents the previous EyeNet solution based on the 418/433MHz Radiometrix modules. Topology of the network and the remote program are parts of this chapter.

Chapter 4 and 5 are the main chapters of this report. In Chapter 4, all the relevant information to the Bluetooth solution for EyeNet are presented. Starting with Bluetooth basics, this chapter also covers the hardware and concludes with the remote program, that also includes the routing functionality. The same structure can be found in Chapter 5, covering the WLAN solution.

A comparison between Bluetooth and WiFi is discussed in Chapter 6. Advantages and disadvantages of both systems are discussed in general and as a solution for robot networks, especially EyeNet.

Finally, the whole project is summarized and concluded with an outlook on further work in Chapter 7.

The technical documentation for EyeNet BT and EyeNet WLAN is placed in the Appendix.

Chapter 2

Existing Network Solution

A precondition for this project was the already existent wireless robot communication network, called EyeNet. It is able to connect up to 16 agents and is self-configuring, which means that there is no need for a fixed master agent. EyeNet provides the user with a system library to use the radio functions (explained in the next chapter).

This section introduces you to the hardware, the network structure, the communication protocol, and a remote control program of the system. Further details on this EyeNet version can be found in [8].

2.1 Hardware

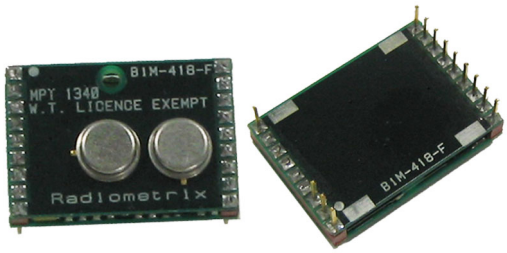


Figure 2.1: RADIOMETRIX BiM-418/433

Figure 2.1 shows one of the Radiometrix BiM-418/433 low power UHF data transceiver modules, which can be attached to the second serial port of the EyeBot. The Radiametrix modules are small UHF radio modules, integrating a low power transmitter, receiver, data recovery and TX/RX change over circuits. They allow bi-directional data transmission in

half-duplex operation. Serial data can directly be sent over the TXD pin and is recovered at the RXD pin of the receiving module. The following list gives the main features:

- Frequency: 433MHz or 418 MHz
- Range: 30 meters in buildings, 120 meters line of sight
- Power Consumption: max. 25mA
- Power supply: 4.5V - 5.5V DC
- Performance: up to 40kbit/s

Further details on performance data can be found in [31].

2.2 Topology

Because the modules work only on a single frequency, either 418MHz or 433MHz, it must be ensured that only one transmitter demands the channel at a time. Otherwise data transmission will be jammed. CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) or TDMA (Time Division Multiple Access) are two media access protocols to avoid transmission collisions. Both are principles that describe how more than one device can use a common transmission medium.
In a network that uses CSMA/CA, a device listens for a free channel before it starts transmission. An ingenious timing mechanism avoids collision and a handshake method secures transmission (cf. subsection 5.1.4).

TDMA, as the name suggests, divides the time in several time slots. Each time slot allows one device to send data and thus there is always only one transmitter occupying the channel. There are three basic ways to configure the network:

- Synchronous
  Each device gets a fixed time slot. If a device has no data to send, the time slot remains unused. The efficiency is not optimal.

- Polling (Master-Slave)
  A master device consecutively sends polling messages to the slave devices and gives them transmitting permission.

- Asynchronous (Random access)
  If a device has no data to send, the time slot is used by the next device. Because there is no coherency between a time slot and a device, such as in synchronous transmission, the send data needs a header with further information.

The EyeNet was implemented as a 'Virtual Token Ring' network which reflects an asynchronous time division network. The Virtual Token Ring network is called virtual, because the logical structure is a ring, whereas the real radio channel (physical structure) is applied to all the transceivers at any time. A special message, called token, is passed from node to node in the ring. A node has to wait until it gets the token before it is allowed to access the network medium. Once it receives the token, it will send its data and then pass the token to the next node. Within the network one node takes the master role. The master is responsible for:

- Creation of the token on initialization or when a token is lost
- Creation, maintaining and broadcasting of the ring structure
- Checking for new nodes that want to join the network
- Remove nodes that do not belong to the network anymore.

2.3 Protocol

To handle data communication and data transfer a protocol has to standardize how the transmitted bytes have to be interpreted. In EyeNet, the bytes given to the BiM have to form a special data packet. The protocol specifies the frame structure of a data packet which is shown in Figure 2.2. It consists of:

- Preamble
  The preamble itself consists of:
  - 20 bytes (0x55) to allow the data slicer in the BiM to settle

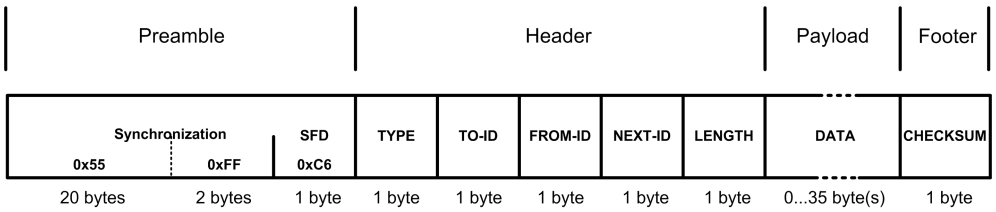


Figure 2.2: Message structure

  - 2 bytes (0xFF) to allow the receive UART to lock
  - 1 byte (0xC6) as a specific Start Frame Delimiter (SFD)

- Header
  - Message type

    Distinction between user data, system data or other. (1 byte)
  - ToID

    ID of the destination. Single robot, group of robots or broadcast. (1 byte)
  - FromID

    ID of the sending robot. (1 byte)
  - NextID

    ID of the next robot in the ring-structure. (1 byte)
  - Length

    Number of bytes of the message data (1 byte)
- Payload
  - Data

    The actual data to send. (0 to 35 bytes)
- Footer
  - Checksum

    A CRC over the whole frame. (1 byte)

Determined by the message type byte, a message can serve three different purposes:

- Application data message
  A message that contains data, exchanged between applications running on the robots.
  Message type: USER.

- Operating system message
  Contains remote control and monitor data.
  Message type: OS

- Network system message
  Messages that are used to manage the token ring network.
  Message types: TOKEN, WILDCARD, ADDNEW, SYNC.

Although EyeNet does not follow the OSI-model, the three major message groups can be seen as protocols, each defining how a message has to be interpreted. The application data and operating system messages refer to the application layer. The information carried in these messages are either used by a user application or the remote control program. The network system messages represent a mixture between a data link and a network layer protocol. It decides which node is allowed to access the network medium by passing a TOKEN. The master periodically sends WILDCARDs to allow new nodes to join the network. A new node responds with an ADDNEW to a WILDCARD and authenticates to the master by sending its own ID. To inform all nodes in the network about the new structure, the master broadcasts an updated list of the available nodes within a SYNC message. Network system messages are also used on initialization of the network. The lowest layer (physical) is represented by the BiM radio modules and their features.

2.4 Remote Control

One of the main advantages of EyeNet is the ability to monitor and remote control any of the EyeBots from a host PC. For that purpose, a remote control program has been implemented. The host system connects to the token ring network with the same kind of radio modules and behaves like any other robot. Therefore it can be switched on or off at any time, which is equivalent to adding or removing a robot.

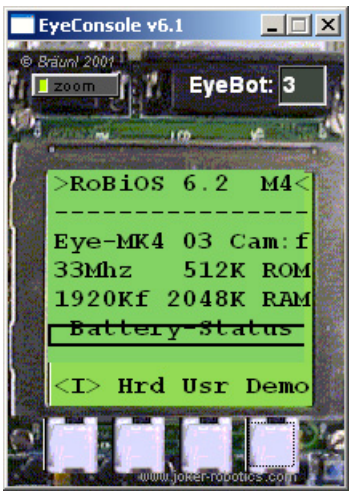


Figure 2.3: Eyeconsole

As a part of the remote software, the EyeConsole window provides a view of a connected EyeBot controller. After a successful connection has been established, an EyeConsole window is displayed along with the current status of the real controller. All LCD output will be sent as an OS message to the host system, where it will be displayed in the appropriate EyeConsole. Due to the cost of colour LCDs the EyeBot only provides a monochrom display, whereas the EyeConsole supports colour. Through the buttons on the EyeConsole, commands can be sent to the real EyeBot controller which will then react as if a real button had been pressed. Figure 2.3 shows an EyeConsole window.

Chapter 3

Network Interface Design

3.1 EyeBot

The central core of all robots in the Mobile Robot Lab is the EyeBot, currently in its fourth version called Mark IV. The EyeBot is a controller developed for the special purpose of creating autonomous mobile robots, covering walking, driving, flying and swimming vehicles. The controller consists of a Motorola 68332 32-bit microcontroller, running on 25MHz up to 33MHz. 512KB of Flash-ROM keep the operating system, called RoBIOS, and up to three user programs of a size up to 128KB. The available RAM is 2048KB. Visual feedback is given through the 128×64 pixel large monochrome LCD display. Four input buttons serve for user interaction. Video features are provided through an interface for a color or greyscale camera. Further parts of the controller comprise a wide range of ports, such as two serial ports, one parallel port, 16 digital inputs/outputs (8 each), 8 analog inputs, and 16 timing I/Os. These provide the ability to connect DC motors, servos, sensors, and other actuators and sensors to the EyeBot. The RADIOMETRIX radio modules as well as the Bluetooth and WLAN modules use one of the serial ports. A microphone and a speaker round out the whole EyeBot. Due to the low power consumption of 270mA at a voltage level of 7.2V DC, the EyeBot is appropriate to be battery-powered.

Figure 3.1: Mark IV EyeBot controller

3.2 RoBIOS

The operating system running on the EyeBot is called RoBIOS. It is in fact a mixture between a basic input/output system and an operating system. Together with the hardware description table (HDT), the RoBIOS completes the operating system. The HDT contains information about the connected hardware as well as EyeBot characteristics. The operating system provides a menu-based navigation to access the EyeBot's hardware settings as well as the user programs. User programs can be written in C/C++ or Assembler and the C/C++ files are compiled with a cross compiler, which is based on the open source GNU compiler. A system library containing several C functions is included in the RoBIOS and can be used within user programs. The functions are grouped according to special tasks such as key input, image processing, LCD output, system operations and many more. Functions for using the radio functionality are also comprised in the library and these will be examined more closely in the next section.

3.3 EyeNet Radio Function

There were eight user functions (listed below), three system functions as well as the background routines that had to be reimplemented for Bluetooth and WLAN.

3.3.1 Programming Interface

The radio functions give the user a programming interface to use the network features. To keep backward compatibility this interface had to work as before.

`int RADIOInit(void)`

Initializes the radio communication.

`int RADIOTerm(void)`

Terminates the radio communication.

```
int RADIOSend(BYTE id,
              int byteCount,
              BYTE* buffer)
```

Sends *byteCount* bytes stored in *buffer* to robot with *id*.

`int RADIOCheck(void)`

Checks if messages are waiting and returns the number of messages.

```
int RADIORecv(BYTE* id,
              int* bytesReceived,
              BYTE* buffer)
```

Receives the next buffered message and stores its source in *id*, the message length in *bytesReceived* and the message in *buffer*.

`void RADIOGetIoctl(RadioIOParameters* radioParams)`

Gets current radio parameter settings and puts them into *radioParams*.

`void RADIOSetIoctl(RadioIOParameters* radioParams)`

Sets radio parameter settings delivered by *radioParams*.

`int RADIOGetStatus(RadioStatus *status)`

Put the current network status, including the active nodes, into *status*.

3.3.2 System Functions

The following three functions are not accessible for users. They are used within the RoBIOS to send operating system messages (OS), basically for the remote control program.

```
int RADIO_SysSend (BYTE mestype,
                   BYTE id,
                   int byteCount,
                   BYTE* buffer)
```

Sends *byteCount*bytes from *buffer* to *id*. Up to now there is only one operating system message type. Therefore *mestype* is OS.

```
int RADIO_SysCheck()
```

Checks if an operating system message is waiting in the operating system message queue.

```
int RADIO_SysRecv (BYTE* id,
                   int* bytesReceived,
                   BYTE* buffer)
```

Receives an operating system message from the appropriate queue. The message's content is stored in *buffer*, content's length is stored in *bytesReceived* and the source of the message is stored in *id*.

`RADIO\_SysCheck()` and `RADIO\_SysRecv()` are frequently called in the RoBIOS to see if a remote control command has been received.

Two further internal functions have been implemented. The first is only enabled if the appropriate HDT entry is set to WLAN mode. The second function deals with the direct sending of messages.

```
int connectMSG(BYTE toID)
```

Sends a special string combination to establish a TCP connection with the wireless LAN modul. The parameter *toID* specifies the last octet of the destination IP address.

```
int sendMSG(BYTE mestype,
            BYTE toID,
            int len,
            BYTE content[])
```

Creates an EyeNet packet with *mestype*, *toID*, *len* as header entries and *content* as data. The message is then send to the RS232 interface. Messages are not longer queued in a message queue but send when they occur.

3.3.3 Background routine

The background routine runs every 100ms after the radio functionality was enabled by calling `RADIOInit()`. The task of this routine is to catch messages coming through the configured serial port and put them in either a message queue (FIFO) for received user messages or in a queue for operating system messages. That way, two different types of messages can be received transparently for the user at the same time.

In the previous version there was a third receive queue for network messages like the passed token. Using Bluetooth or WiFi the whole network management was sourced out and is handled by the devices themselves, leading to a far smaller background routine and to a smaller load for the CPU. The old routine also took care of sending user, operating system, and network messages, which were queued in analogous queues. Having no need for those anymore, they were replaced by a new function that immediately sends a message once `RADIOSend()` or `RADIO\_SysSend()` is called.

Figure 3.2 shows the activity diagram of the background routine. As long as bytes are available at the serial port, they will be received. In the incoming byte stream a special start byte along with other conditions has to appear. From then on the message will be buffered until the message is complete. The buffered message contains a small header (see next section) with information on the message type. Depending on the type, the message is put either into the operating system or the user message queue. Finally, the start detection variable is reset and the sequence starts all over.

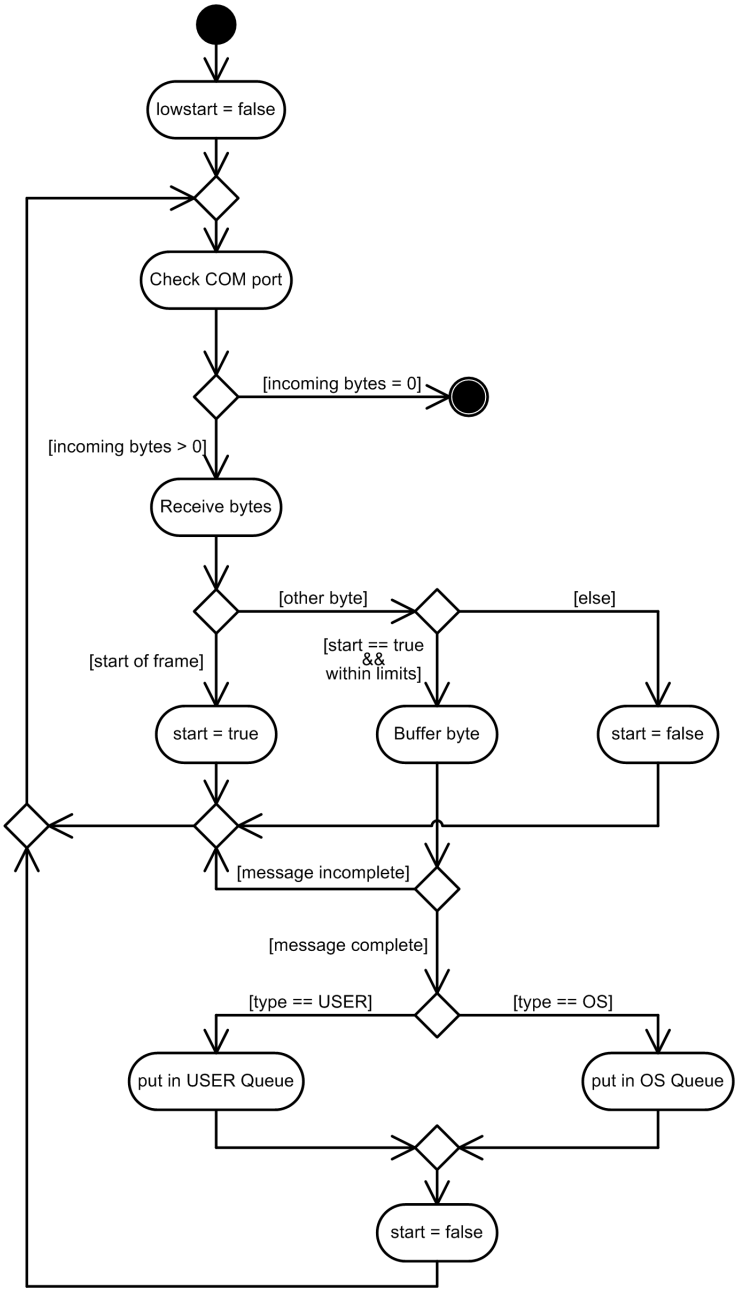


Figure 3.2: Radio background routine

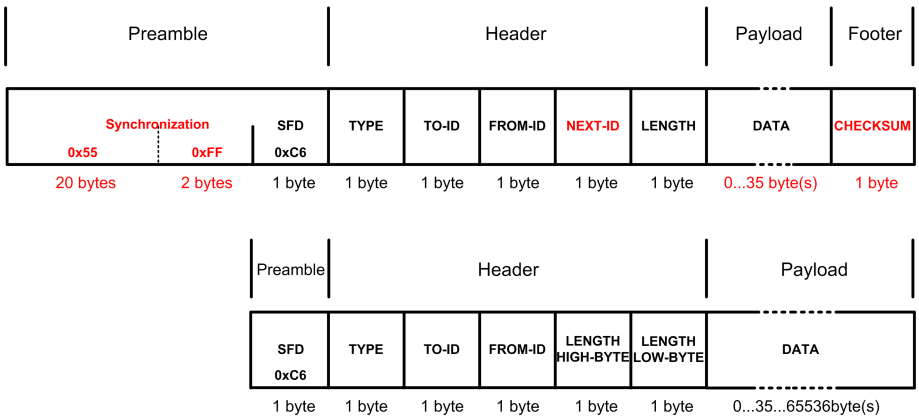


Figure 3.3: Old vs. new frame structure

3.4 Protocol

Because the RADIOMETRIX modules send plain data, there was a need for a large network management on the EyeBot. The advantage of using the new technologies Bluetooth and WLAN is that the devices come along with already implemented network protocols. Therefore, almost the whole network management messages and their routines had been removed, leaving only the user application, operating system, and synchronization messages.

Figure 3.3 shows the former and the new message frame structure. Unfortunately, Bluetooth does not support any routing mechanism.

Frame structure:

- Preamble
  - Start Frame Delimiter (0xC6)
- Header
  - Message type
    Distinction between user, operating system and network data. (1byte)
  - ToID
    Address (robot ID) of the destination. (1byte)
  - FromID
    Address (robot ID) of the source. (1byte)

  - Length
    Number of bytes of the message data(2 byte)

- Payload

  - Data
    The actual data to send. (0...65536 byte)

The first 22 bytes in the former preamble were hardware specific for the Radiometrix modules and got cut off, leaving only the start byte in the preamble. The second byte now represents the message type. Only USER and OS are left. Byte three and four carry the addresses of the destination and the source. The information on the next ID was based on the token ring system and was replaced by a second byte for the length. The length field consists of two bytes now and allow up to 65536 bytes of data following. Checksum as well as packet segmentation are done by the devices.

The length of a message was increased to fit a whole image (maximum of $3 \times 60 \times 80$ = 14400 bytes), caught from the robot's camera. This results in a very small overhead of header data[1] (less than 1%) for the EyeNet message whereas an image was previously split into 35 byte long pieces. Thus, the RoBIOS function `send_image()` (lcdfunc.c) had to be modified and now bypasses the radio background routine. Images are sent directly as soon as the serial port is free rather than queued in a message queue. Due to the limited space for RoBIOS on the EyeBot's memory it is not possible to implement the receiving message queues in a size capable for an image. User messages sent between robots are still limited but got extended to 50 bytes. Image transmission is basically used for the remote control program.

[1] The total overhead is higher because the different protocol layers have a header themselves.

Chapter 4

EyeNet Bluetooth Network Implementation

By now the basics of robot networks as well as the existing solution with the Radiometrix modules have been introduced. Due to the lower performance of the former solution by transmitting image sequences, provided by the EyeBot's camera, a newer technology had to be used. The up-to-date technologies for small networks are Bluetooth and WLAN, also known as WLAN. This chapter introduces the basics of Bluetooth, the used hardware and the final solution, whereas the next chapter deals with WLAN.

4.1 Bluetooth Technology

This section deals with the basics of the Bluetooth technology. Because Bluetooth was developed by several companies and hundreds of people, its technical details became very complex. Fully detailed information can be found in the official Bluetooth specification (see Appendix C), but the following pages give an overview of the basic concept.

4.1.1 Introduction

Bluetooth is an industrial specification for low costs short-range wireless networks. In 1994, Ericsson Mobile Communications started a study to find a low power and low cost solution to replace cable connections between mobile as well as fixed devices, such as a laptop and a printer. The solution had to match special criterias of cost, performance,

size and power consumption to also fit in small battery-powered portable devices, i.e. cell phone. Beside this the transmission of both data and speech had to be realized.

To make Bluetooth to a worldwide standard, Ericsson Mobile Communications, IBM, Intel, Nokia, Toshiba founded the Bluetooth Special Interest Group (SIG) in September 1998. The SIG developed the Bluetooth wireless technology and standard to be interoperable between different devices of different producers. The group growed larger and counts over 2000 membership companies today whom are allowed to use the open platform technology.

The Bluetooth core specification, covering the physical layer and the data link layer, was adopted by IEEE under the name WPAN (Wireless Personal Area Network) and can be found in IEEE 802.15.

Being only an internal code name first, Bluetooth became an official trademark later. The name Bluetooth derives from the Viking king Harald Blåtand, who united Norway and Denmark in the 10th century and brought Christianity to Scandinavia. The viking word 'Blåtand' translates to 'Blue Tooth' and refers to Harald's dark complexion rather than the folklore story of his affection for blueberries. According to his unification of two countries, the SIG founders believed Bluetooth to be an appropriate name for the unification of the companies in that project.

4.1.2 Functionality & Architecture

What started in 1994 as a replacement for cable, developed further to a network technique that establishes connections between several devices and forms small LANs. Bluetooth operates on the unlicensed ISM band between 2.402MHz and 2.480 MHz. This frequency range is divided into 79 channels of each 1MHz for the frequency hopping method. Frequency hopping is used to deal with radio interferences and is explained in a later section in more detail. As a robust, low power, low cost, and short-range communications system Bluetooth provides a maximum bit rate of 1Mbit/s within a range of up to 100 metres. The maximum data transfer rate is up to 721kbit/s. Two types of physical links are supported:

- SCO (Synchronous Connection Oriented)
  Supports symmetrical, circuit-switched point-to-point connections. Uses reserved timeslots at regular intervals. Data rate of 64kBit/s.

- ACL (Asynchronous Connection Less)
  Supports symmetrical and asymmetrical, packet-switched point-to-multipoint connections. Data rate of 721kBit/s in one, and 57.c6 kBit/s in other direction

The transmitting power between 1 and 100mW makes a Bluetooth unit suitable for small battery-powered devices.

Within the Bluetooth technology all transmission is done via the master unit. The so called master is the device that establishes a channel to a slave. Up to 7 slaves can be active in a piconet and an unlimited number can remain in a park mode. The devices in a piconet share one logical channel, which is defined by its hopping frequency sequence, defined by the master. To form larger networks scatternets are created over a common device. Further details on the network types can be found in the 'Topology' section.

Each Bluetooth device is identified through its BD_ADDR (Bluetooth Device Address), a unique 48 bit address comparable to a MAC address. The active nodes in a piconet are also assigned a 3 bit LT_ADDR (Logical Transport Address), the parked nodes are assigned a 8 bit PM_ADDR (Park Mode Address) and an AR_ADDR (Access Request Address).

The architecture of the Bluetooth controller consists basically of the three units shown in Figure 4.1

- Radio
  A HF-module with antenna for transmission and reception.

- Baseband Controller
  Deals with packet management, frequency-hopping, error correction, device discovery, physical link management.

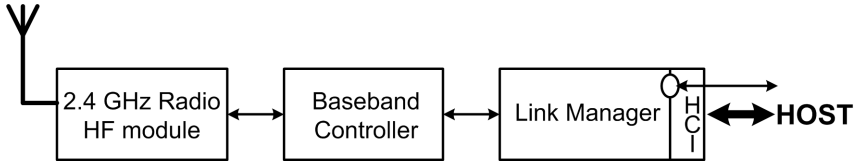


Figure 4.1: Bluetooth Controller

- Link Manager
  Responsible for creation, modification, and release of logical links. Manages transport attributes (i.e. security, transmit power, connection quality).

The controller can be accessed by the host (higher layers, protocols) either directly or through an optional Host Control Interface (HCI). Implementing the HCI allows the exchange of the controller or host sub-system by other controllers or hosts (see also section 4.1.6).

4.1.3 Frequency Hopping

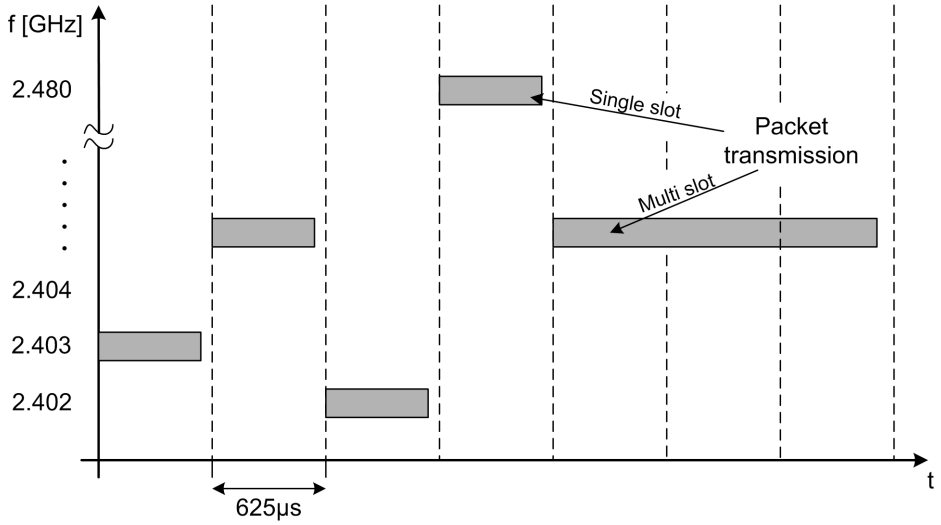


Figure 4.2: Frequency hopping

The frequency range for Bluetooth covers the ISM (Industrial Scientific Medical) band between 2.402 GHz and 2.480 GHz. This ISM band is worldwide unlicensed and therefore widely used by other devices like wireless LAN, garage openers or mircowave ovens. Due to the high usage and background noise, a lot of interferences occur in the ISM band. The frequency hopping method is used to avoid these interferences. The ISM band is divided into 79 channels of 1MHz each and the frequency hops every 625 microseconds (1600hops per second) in a pseudo-random order into one of these channels. Therefore, the traffic is spread over the entire frequency range converting the narrowband signal into a wideband signal. Hopping faster and using shorter packets than other devices in the

ISM band, makes the Bluetooth device less vulnerable to narrowband interferences. If a transmission is jammed, the chance to hop into another interfered channel is very low. Hopping into and out of interfered channels increases the chance to repair the transmission error rather than staying for a longer period in an interfered frequency. Error correction algorithms as well as ARQ (Automatic Repeat Request) are used in Bluetooth to correct faults by jammed transmissions. Figure 4.2 illustrates the frequency hopping method.

4.1.4 Packet Format

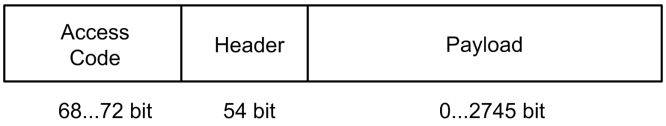


Figure 4.3: Packet format

Figure 4.3 shows the three parts of a Bluetooth packet. In the first 72 bits the access code contains information about the master's identity and the master's system clock used for synchronisation. The access code is unique for all packets on the specific channel. The access code is followed by a 54 bit long header carrying error correction, retransmission and flow control information. The real data of 0 bits up to 2745 bits completes the packet. Normally, one packet uses one of the 625 microsecond timeslots. A multi-slot packet mode is also specified (3 or 5 time-slots without hop).

4.1.5 Connection establishment & Device states

To establish a connection the paging method is used if the address is already known. Otherwise an inquiry is made first, followed by the paging procedure. At any time Bluetooth devices are in one of the three major states or in one of seven substates. Figure 4.4 shows the state model. The device that establishes the connection is refered to as the master.

STANDBY

The default state of a unit is the standby state in which the unit stays after it is switched on. In this state, the device is in a low power mode and only the native clock is running. This state is left for the use of an inquiry or page scan.

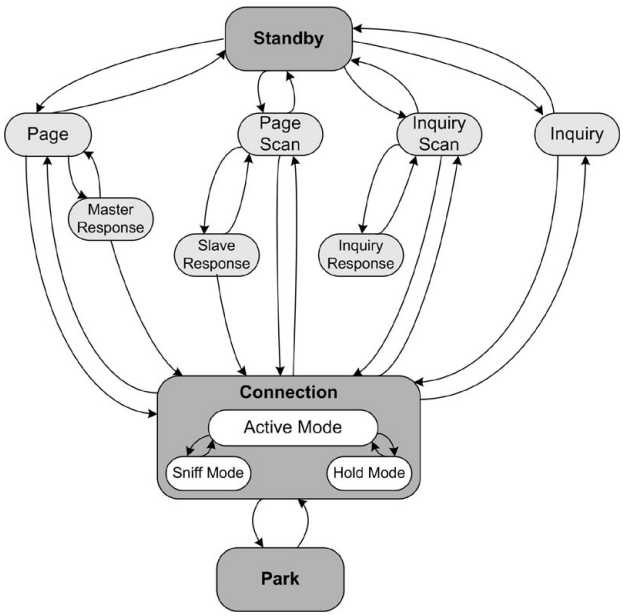


Figure 4.4: Connection States

PAGESCAN

In this substate the device listens for a specific time, the scan window, for the transmission of its own device access code (DAC). Listening to the scan window is repeated and forms the scan interval. There are three different interval modes. R0 represents a continous scan where the scan interval equals the scan window. A maximum of 1.28 seconds and a maximum of 2.56 seconds between two scan windows are defined in mode R1 and R2. The device listens at one hop frequency during each scan window. The scan frequency is selected according to the page hopping sequence, determined by the device address. The page hopping sequence consists of 32 unique frequencies within the 79 MHz. A scan window should be long enough to cover 16 page frequencies. If the device finds its own DAC, its state changes into the slave response state.

PAGE

When a master wants to connect to another bluetooth device, it repeatedly sends a paging message, containing the slave's DAC. Because the master's and the slave's clocks are not synchronized yet, the master does not know when the slave will wake up and on which hop frequency. Thus, it sends a train of identical paging messages at different hop

frequencies and listens for responses between the transmit intervals. The page hopping sequence of 32 frequencies is determined by the BD_ADDR of the slave. The master knows the BD_ADDR of the slave through the device inquiry (explained later), hence, both master and slave know the page hopping sequence. Different from the sequence, the master does not know the phase of the frequency and has to estimate the slave's clock. The page hopping sequence is divided into two trains, containing 16 frequencies each (16 * 625 microseconds = 10 ms). If the master does not get a response of the slave within the first 16 frequencies (Train A) the second Train (B) is used. If the slave does not respond, the trains have to be repeated since the master does not know when the slave will enter the pagescan mode. On a slave response the master switches to the master response substate.

PAGERESPONSE

After the slave receives a page message from the master, it transmits a page response message that carries the slave's DAC (slave response substate). This message is sent on the same frequency as the received page message. Receiving the slave response switches the master to the master response substate, as mentioned above. The master now sends a FHS message on the next frequency in the page hopping sequence. The FHS message contains information about the master's clock, the BD_ADDR, the BCH parity bits, and the class of device. With this information the channel access code can be generated. After acknowledgement of the FHS by the slave, the transmission parameters of both devices are changed to use the master's channel access code and clock. The master's BD_ADDR is from now on used to generate the basic frequency hopping sequence. Finally, both devices reach the connect state and the master takes over the control. The master now sends its first packet, a POLL packet and the slave has to respond to it with any kind of packet, i.e. a data packet. If interruptions occur during the acknowledgements of the FHS or POLL packet, both devices return to the page/pagescan state after a specific timeout.

INQUIRY

To discover unknown units, an inquiry message, containing the general inquiry access code (GIAC) or the device inquiry access code (DIAC), is sent repeatedly at different hop frequencies. The inquiry hopping sequence is derived from the GIAC. In between the inquiry transmission, the discovering device listens for responding FHS packets, which

carries information about the identity and the system clock. The master does not acknowledge responses, but keeps on transmitting the inquiry message. As in the page state, the 32 frequencies are split into two trains, A and B. A train is repeated 256 times before another train is used. Three train switches are necessary to collect all responses. With a length of 10ms of each train, the inquiry procedure can last for 10.24 seconds, but might be aborted earlier when enough answers are collected.

INQUIRY SCAN

This substate is similar to the page scan substate. The unit looks for the inquiry access code (GIAC of DIAC). As in the page procedure the device listens for a scan window of 16 frequencies in an interval of less or equal to 2.56 seconds. During each scan window the device listens to one hop frequency. On a receiving IAC the device switches to the INQUIRY RESPONSE substate.

INQUIRY RESPONSE

The discovered unit will respond with a FHS message to the inquiry message, transmitting its BD_ADDR. To avoid collision with other responding devices, each device will wait for a random time until it answers.

CONNECTION

Once both devices have reached the connection mode, they can send packets. Within this state there are three different modes.

ACTIVE MODE

In the active mode the channel is used by the master as well as by the slave. It is the master's rule to schedule the transmission of packets to up to seven active slaves. The active slaves themselves listen in the master-to-slave slots for packets.

To reduce power consumption, devices can jump into one of the following modes or the PARK state.

SNIFF MODE

The duty life cycle of the slave's activity is reduced. Therefore, it does not listen to all master-to-slave slots like in active mode. The master uses only specific time slots to send packets. These time slots occur after a sniff interval. When the slave detects a message for itself, it continues listening.

HOLD MODE

In the hold mode the ACL is temporarily not supported by the slave. The device keeps its logical transport address. After a timer value, known by master and slave, has been reached, the data transfer restarts.

PARK

In the Park mode a bluetooth device remains synchronised but does not participate in the traffic on the piconet channel. In this state the slave's logical transport address is replaced by two new addresses, the Parked Member Address (PM_ADDR) and the Access Request Address (AR_ADDR). Both are used to unpark the slave either initiated by the master or the slave. The slave wakes up in regular intervals to resynchronize and to listen for broadcast messages. Beside the low power consumption, the PARK state has another advantage. It can be used to keep more than seven slaves in a piconet even if only seven can be in the CONNECTED state. By exchanging connected and parked slaves, the number of slaves can be extended.

4.1.6 Protocol stack

This section deals with the Bluetooth protocols and its protocol stack. As shown in Figure 4.5, the protocol architecture has a layered structure, but does not follow the OSI model. However, some layers correspond with those in OSI and IEEE 802.15 modifies Bluetooth to fit it even more neat into the OSI model. There are two different protocol types; the Bluetooth specific and the non-bluetooth specific protocols. Above the specific protocols either new manufacture-specific or existing protocols can be placed. This structure makes it easier to use already established protocols, like HTTP or UDP, and opens the door to the Bluetooth technology for existing applications. Within applications the core protocols

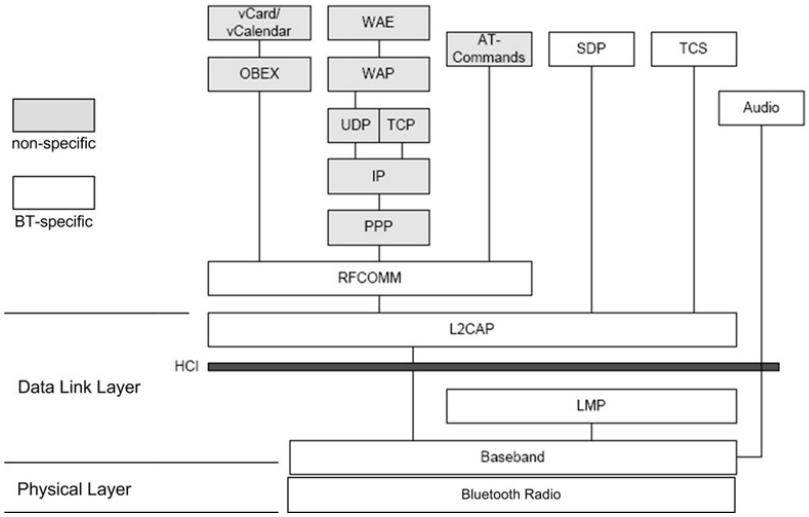


Figure 4.5: Protocol Stack

are always required, whereas the other protocols are implemented when needed. Thus, the protocol stack differs between applications.

Core Protocols

- **Bluetooth Radio:**
  Bluetooth Radio defines the modulation and frequency characteristics.

- **Baseband/Link Control:**
  The baseband enables the physical RF link between Bluetooth units. It controls the synchronisation, frequency hopping sequence and time slot management. The layer also manages the two different link types, Synchronous Connection Oriented (SCO) and Asynchronous Connectionless (ACL)

  - **Audio:**
    Audio applications use the baseband directly. The transmission does not have to go through the L2CAP layer.

- **Link Manager protocol (LMP):**
  The link manager protocol deals with link set-ups, controls the packet sizes, handles power modes and the state of the unit. Furthermore, it is responsible for security

aspects, such like authentication and encryption. The LMP can be integrated into the L2CAP.

- **Host Controller Interface (HCI):**
  The HCI allows access to the Bluetooth hardware capabilities. It provides a command interface to the Baseband controller, the link manager, the hardware status, and the control/event registers.

- **Logical Link Control and Adaptation Protocol (L2CAP):**
  L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation, and group abstractions. Information on the Quality of Service (QoS) are handled as well. Operating over the L2CAP, higher level protocols can use data packets of a size up to 64kbytes.

- **Service Discovery Protocol (SDP):** As the name suggests the SDP describes how applications can discover available services of other Bluetooth devices and the characteristics of those services. This covers also:

  – discovery of available services when new devices enter the client's proximity or new services becomes available on a device in range.

  – discovery of services without knowledge of the available services based on their characteristics

  – determination of services that become unavailable.

Cable Replacement Protocol

- **Radio Frequency virtual Communications (RFCOMM):** The RFCOMM protocol provides the emulation of RS232 serial ports. It is based on the ETSI standard TS 07.10. A subset of the standard and additional extensions form the RFCOMM protocol. The RFCOMM protocol provides transport capabilities for higher level protocols, like OBEX.

Telephony Control Protocol

- **Telephony Control - Binary (TCS Binary):** The TCS is a bit-oriented protocol. It defines the rules for the establishment and release of calls for speech or data exchange between Bluetooth devices. It also controls the transmission of call control signals. The TCS is derived from the ITU-T Recommendation Q.931.
- **Telephony Control - AT (TCS AT):** This protocol supports a set of the AT-commands for telephony control.

Adopted Protocols

- **Point-to-Point Protocol (PPP):** Originally the PPP is a standard to send IP packets over a serial point-to-point link. Therefore, it is a packet-oriented protocol. In Bluetooth it uses the RFCOMM for the point-to-point connection and is responsible for the conversion of the packet data stream into a serial data stream.
- **TCP/UDP/IP:** These protocols have been adopted by Bluetooth to open the door to WANs and LANs. Therefore, Bluetooth devices are able to communicate with devices connected to the Internet.
- **OBEX (Object Exchange):** OBEX or IrOBEX was first designed for infrared communication and was later adopted by Bluetooth. It provides the exchange of binary objects, such like vCard or vCalender, between devices. OBEX works on a client-server structure and is independent of the transport mechanism, so that it can use RFCOMM as a transport mechanism in Bluetooth.
- **Wireless Application Protocol (WAP):** WAP was developed for a variety of wireless WAN technologies to bring the Internet to mobile devices. Within the WAP client/server model, Bluetooth serves as the bearer of the data transport.

4.1.7 Profiles

The SIG released a number of profile specifications to ensure interoperability between devices from different manufacturers. A profile defines a selection of messages and procedures from the Bluetooth specifications and gives a description of the air interface for

specified services and use cases (in short: It defines which options of each protocol are mandatory for the profile). Two connected devices exchange their profiles and determine which services are available, and which data and commands are required for them. The number of profiles is still increasing and the following three only show the relevant profiles for EyeNet.

- **GAP (Generic Access Profile)** defines procedures for device discovery and connection establishment. It further defines operations that can be used by profiles referring to GAP. It also focuses on standby and connecting states, multi-profile operation, and security procedures. GAP ensures that any Bluetooth device, regardless of manufacturer and application, can exchange information in order to discover what type of applications other units support. Therefore GAP is included in every Bluetooth device.

- **SDAP (Service Discovery Application Profile)** defines how a Bluetooth device searches for a specific service or makes a general service search.

- **SPP (Serial Port Profile)** allows the creation of a virtual serial port connection between two devices. The emulation for the serial ports is based on RFCOMM.

Further profiles are GOEP(Generic Object Exchange Profile), FTP(File Transfer Protocol), LAP(LAN Access Protocol), CTP (Cordless Telephony Profile), IntP(Intercom Profile), HSP(Headset Profile), DUN(Dial-Up Networking Profile), OPP(Object Push Profile), SP(Synchronisation Profile) and many more.

4.1.8 Topology

Bluetooth does not have a fixed infrastructure and so it works in an ad hoc mode. When any two or more Bluetooth devices connect together they form a piconet. The devices share the same physical channel, which defines the piconet. Within a ‘Piconet’ there can be up to 8 active units, although the number of units that share the channel and remain in the PARK mode is unlimited. One of the active devices controls the communication and is called ‘Master’, whereas the other devices are called ‘Slaves’. There is no difference in the hardware between a slave and a master, so they are interchangeable and the master is

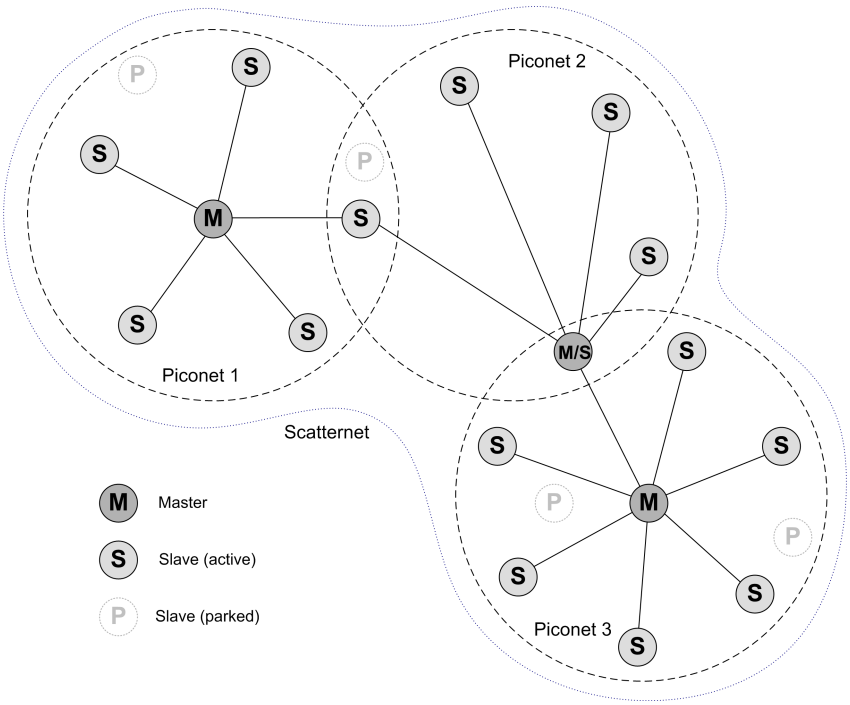


Figure 4.6: Three piconets forming one scatternet

not fixed. The logical network topology of a piconet is a star, with a two point connection between the master and each slave. There are no connections between the slaves.

To extend the number of devices, a so called 'Scatternet' can be created. A Scatternet is a connection of two or more Piconets. Thus, the network topology grows to a tree structure. The connection between the piconets are managed by a shared Bluetooth device, that uses time-division multiplexing to switch between the piconets. This device can either be the master of a piconet or a slave. Each piconet has only one master and therefore a master of one piconet might act as a slave for another piconet. The whole scatternet shares the frequency range of 79MHz, but each piconet has its own hop sequence, determined by the master's clock and device address. As long as the piconets do not share the same hop frequency at any given time, each piconet has the maximum bandwith. Increasing the number of piconets above 10 will lead to saturation. Figure 4.6 shows three piconets forming a common scatternet.

4.1.9 Security

Within wireless communication, security is an important aspect. Therefore, authentication and encryption are implemented in the Bluetooth technology. Authentication verifies who wants to get access to data and secures the message originator. The authentication is based on a link key or the pairing procedure. An encryption key between 8 to 128 bits protects the communication from being eavesdropped. The devices have to agree whether encryption is used or not and about the key size. Beside these two techniques, the high frequency hopping and the short range of Bluetooth increase the security. For more detailed information on this huge aspect, the reader is refered to the Bluetooth specifications.

4.2 Used hardware

The Bluetooth solution of the EyeNet uses two different Bluetooth devices. These were purchased from the company called 'HandyWave'. The network will also work with devices from other manufacturers, whereas the setup programs depend on the HandyWave's hardware units.

4.2.1 HPS-120 HandyPort-Serial

Figure 4.7 shows the HPS-120 HandyPort-Serial, a device developed to bring the Bluetooth technology to any device, that provides a RS-232 serial port. The HandyPort's RS-232 interface is a 9 pin female DCE adapter and allows data transmission with a baud rate up to 115kbps. The pin number 9 differs from the official RS-232 standard and is used for the power supply of the HandyPort-Serial. The small current consumption of a maximum of 110mA can be handled by the EyeBot's microcontroller power supply unit, whose capacity is 500mA. Furthermore, the HPS-120 offers a connection coverage of up to 100 metres when used in line of sight. It supports the SPP and GAP (explained earlier). Due to its single purpose of replacing a serial cable, the HPS-120 supports only one-to-one connections. These Bluetooth modules are the radio units for each of the EyeBots in the EyeNet network. They will always keep the role of the slaves, whereas the HPU-100 Bluetooth device, introduced in the next section, will keep the role of the master.

Figure 4.7: HPS-120 HandyPort-Serial (left) and HPU-100 HandyPort-USB (right)

4.2.2 HPU-100 HandyPort-USB

The HPU-100 HandyPort-USB (aka. MSPP) can be seen in Figure 4.7 As the name suggests the HPU-100 is a USB device and can be plugged into any PC's USB port. It provides point-to-point as well as point-to-multipoint connections in a range of 100 metres (line of sight). SPP and GAP are also supported.

4.2.3 Setup

Before the HPS-120 can be used within the network some of its configuration values have to be changed. In particular, the values for both the EyeBot's and the HandyPort's serial interface have to concur in baud rate and flow control (aka 'handshake'). The default value of the EyeBot's serial port are:

- Speed: 115200bps
- Handshake: NONE

There are two ways to configure the HandyPort-Serial. One is to use a terminal emulator such as 'HyperTerminal' or 'Minicom', the other is to use the setup menu written for the

EyeBot controller. For details on the former the reader is refered to the appendices or the HPS-120 manual; the latter is explained below.

Program Structure

The setup program, placed in the demo menu on the EyeBot, emulates the keystrokes normally entered by a user using one of the mentioned terminal emulators. The setup program is a static version of the dynamic setup through a terminal emulator that provides the user with visual information after each keystroke. It is static in such a way that it keeps track of the values that are entered through the four keys of the EyeBot, but does not provide visual response from the device. Due to the need of only three parameters for the EyeBot/EyeNet settings, only the baud rate, flow control and connection mode are displayed and editable.

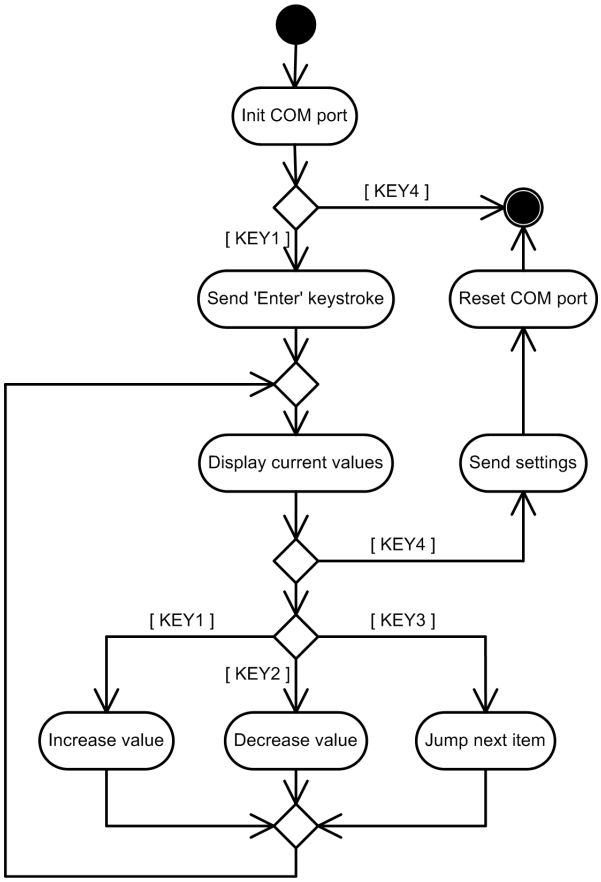


Figure 4.8: EyeBot Config Menu

Figure 4.8 illustrates the sequence of the program. After the HPS-120 has been put into setup mode (flashing green LED) the hexadecimal code for the 'Enter' key is transmitted on keystroke of the EyeBot's 'ETR' button to reach the setup menu. In the following loop the program waits for further key input. Depending on which key is pressed and the actual cursor position either the values of the settings are edited or the cursor jumps to the next position. On the exit of the menu (KEY4 / END) the values that were chosen and displayed at last will be transmitted with the according commands to the HPS-120. This includes the command for saving and rebooting the device.

EyeBot Config Menu

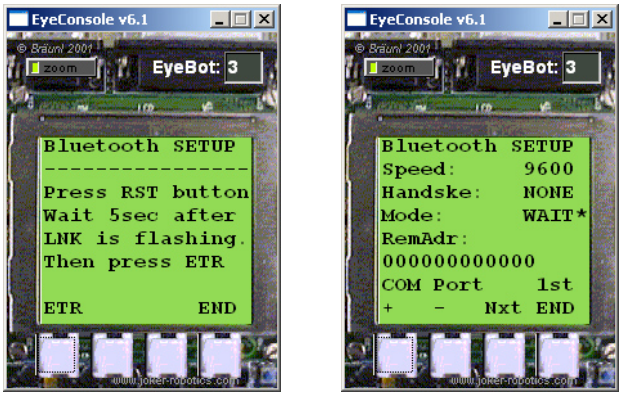
The configuration menu for the HandyPort-Serial is included in the demo programs and according to them also written in C. It can be found on the EyeBot under:

$$\text{Demo} \Rightarrow \text{Network} \Rightarrow \text{BLT}.$$

The start screen and the menu are presented in Figure 4.9(a) and Figure 4.9(b). The selection of the different menu items is provided through the 4 keys on the EyeBot. Only the items which are relevant to the mobile robot network system are implemented. These items are:

- **Speed:**
  The HPS-120's serial port baud rate settings. Possible speeds are: 9600bps, 19200bps, 38400bps, 57600bps, 115200bps
- **Handshake:**
  The flow control settings. Either 'NONE' or 'RTS/CTS'.
- **Mode:**
  Distinguishes between the connecting modes:
  1. **1:1:**
     This establishes a 1-to-1 connection, i.e. between two HandyPorts.
  2. **WAIT:**
     A HandyPort will wait until another device tries to establish a connection.
  3. **REG_CON:**
     Automatic connection between HPS-120 and HPU-100/MSPP.

- **RemAdr:** The hardware address of the remote device.
- **COM Port:** Distinguishes to which virtuell COM port an auto-connection should be established.



(a) Enter menu **(b)** Main menu

Figure 4.9: Bluetooth setup menu

4.3 Network Architecture

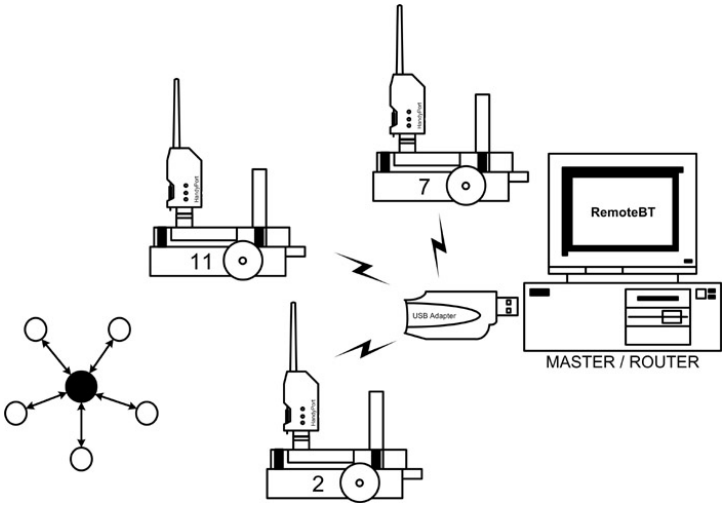


Figure 4.10: Network topology

As mentioned in the first section, the typical Bluetooth network is a piconet in the form of a star structure. So is EyeNet. Each of the robots' HandyPort-Serial modules provides

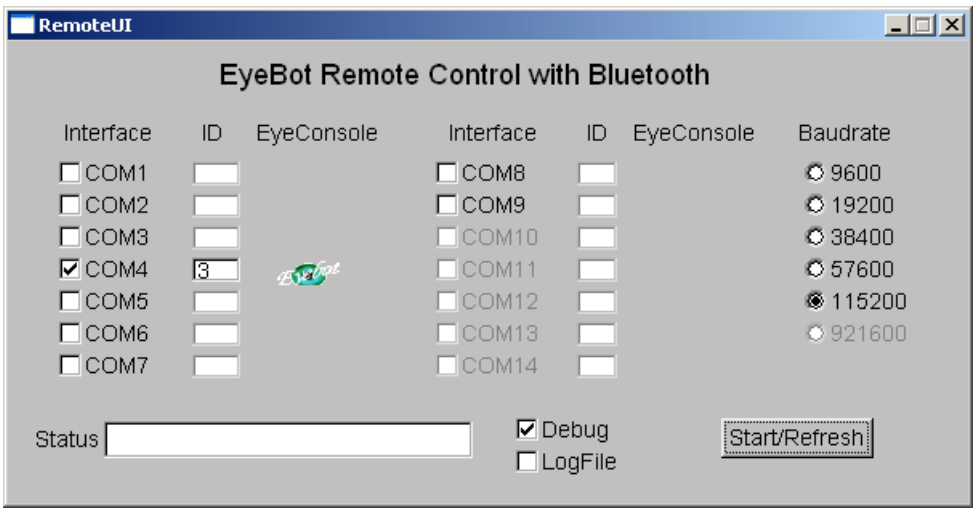


Figure 4.11: GUI Routing Software

a point-to-point connection, emulating an RS232 serial port. They connect to the USB Bluetooth dongle which can establish up to seven connections simultaneously. However, Bluetooth does not support any network routing capability and the communication only takes place between the master and the slaves. To forward messages sent by one EyeBot to another, such a routing function had to be implemented in software.

Figure 4.11 shows the user interface of the routing software. The robot, connected to the selected COM port, can be linked to the network by choosing any one of the corresponding COM tickboxes. The program polls the selected COM ports consecutively for incoming bytes. If incoming bytes are detected, the same routine that runs in the background of RoBIOS is executed. However, after a message is received and buffered it is not queued, rather it is forwarded to its destination (derived from the header). If no bytes are present, the next COM port is chosen.

4.4 BlueSoleil MSPP

Because there was no API provided to control the USB Multi-serial port, the manufacturer's software has to be used to make device and service discovery inquiries as well as connection establishments. Figure 4.12 shows the graphical user interface. When one of the EyeBots' HandyPorts connects to the Multi-serial port, the program will show which emulated COM port on the PC the HanyPort is connected. This COM port then has to be

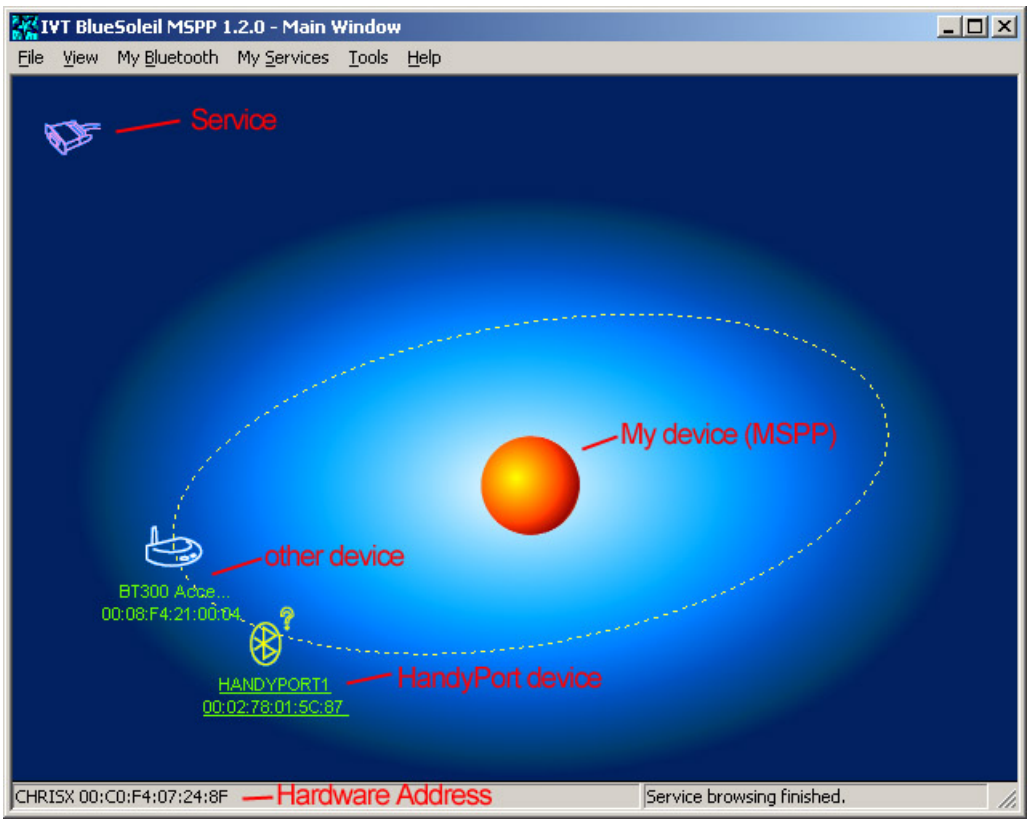


Figure 4.12: BlueSoleil MSPP

chosen in the RemoteBT program. For further information on BlueSoleil MSPP refer to the EyeNet BT manual (cf. Appendix A) and the manufacturer's manual [11].

4.5 RemoteBT

RemoteBT is the final software in the context with the Bluetooth solution. It serves five purposes:

1. Network setup
2. Message routing
3. Remote control for any robot in network
4. Visualization of the actual robots' positions.
5. Message monitoring

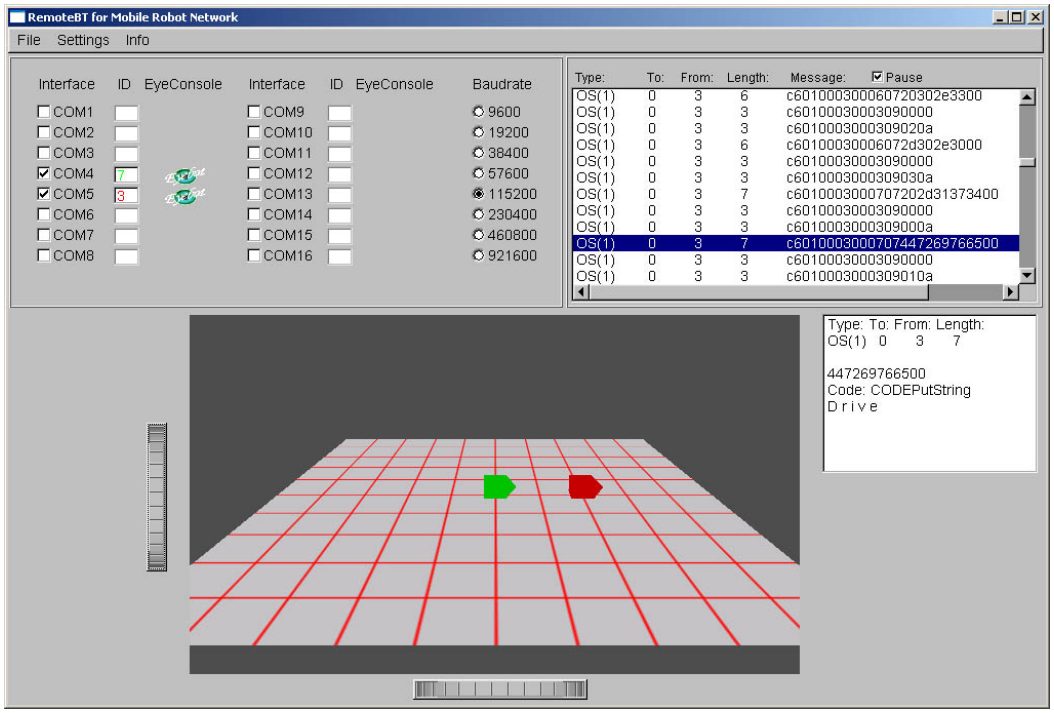


Figure 4.13: RemoteBT GUI

Together with the BlueSoleil MSPP software it sets up the network. BlueSoleil establishes the connections between the Bluetooth devices and provides information about the used COM ports. Robots on these COM ports are added to the network. Their identification number is displayed next to the appropriate COM port tickbox. At any time robots can be included or excluded from the network.

The above explained routing function is integrated into the program. The former GUI was included into the GUI of RemoteBT (Figure 4.13)

As a part of the old Remote Software (see chapter 1), the EyeConsole provides a view of a connected EyeBot controller. The EyeConsole was changed slightly and adapted to the new remote software. After a robot is added to the network a small EyeConsole icon will appear next to the appropriate robot ID field. This icon opens an EyeConsole window and displays the current status of the real controller. Through the buttons on the EyeConsole, commands can be sent to the real controller which will then react as if a real button had been pressed.

A new feature is the implementation of the visualization of the robots' positions. Within an operating system message the actual position (x,y coordinates and rotation

angle) of a robot is transmitted. Distinguished by the `CODEVWPosition` code byte the message content is evaluated and the symbol of the according robot in the 3D environment display is redrawn at the received position. The perspective of the scenery can be changed through the wheels underneath and to the left of the display.

The message display in the upper right corner of the GUI provides information about the traffic through the router and displays every single message along with the first 20 bytes of each message. Each message can be selected and will be analysed in the small window beneath. The message header is resolved and the message content is displayed. In the case of a `CODEPutString` message, the bytes are interpreted into the corresponding ASCII characters. Error codes may occur here too. The message display provides a good way for debugging the network. The messages can also be stored in a logfile.

4.5.1 Software Structure

A multithreaded software structure was chosen to implement RemoteBT, where for each robot in the network a new thread is started. This structure avoids the need for the software caring about switching between the COM ports. This task is now given to the operating system.

The graphical user interface for RemoteBT was designed with a cross-platform C++ GUI toolkit called FLTK, particulary with it's UI builder FLUID. FLTK is event based and supports openGL. Unfortunately, it does not support multi-threading, but by locking the event thread, this disadvantage can be bypassed.

The program starts as usual with the initialization of some variables. The main window is then created and the program enters the FLTK event loop. If one of the COM port tickboxes is selected or deselected a refresh function is started. The refresh function checks the selected COM ports. Robots that are connected to a deselected COM port are destroyed, releasing their appropriate thread. Already, existing Robots are skipped and new COM port selections create a new robot object. As mentioned above, each robot starts a new thread. A pointer to the own instance of the robot is passed as a parameter for the thread function.

Within each thread an instance of the `class Message` and the `class NetworkBT` is created. The NetworkBT object is used to initialize the COM port with the properties of the robot passed by the pointer. As long as the robot exists, messages are received. If

the robot is not authenticated yet, its ID according to the COM port will be stored in a routing table. Furthermore, its ID will be displayed in the GUI as well as a symbol in the environment display. The icon to it's EyeConsole will then also be accessible. A message containing all IDs of active robots is broadcasted to inform all robots about the change. Finally, the received message is either processed or forwarded to its destination. In the instances where no messages are waiting, keystrokes on the EyeConsole are sent to the real controller. Figure 4.14 shows the thread function.

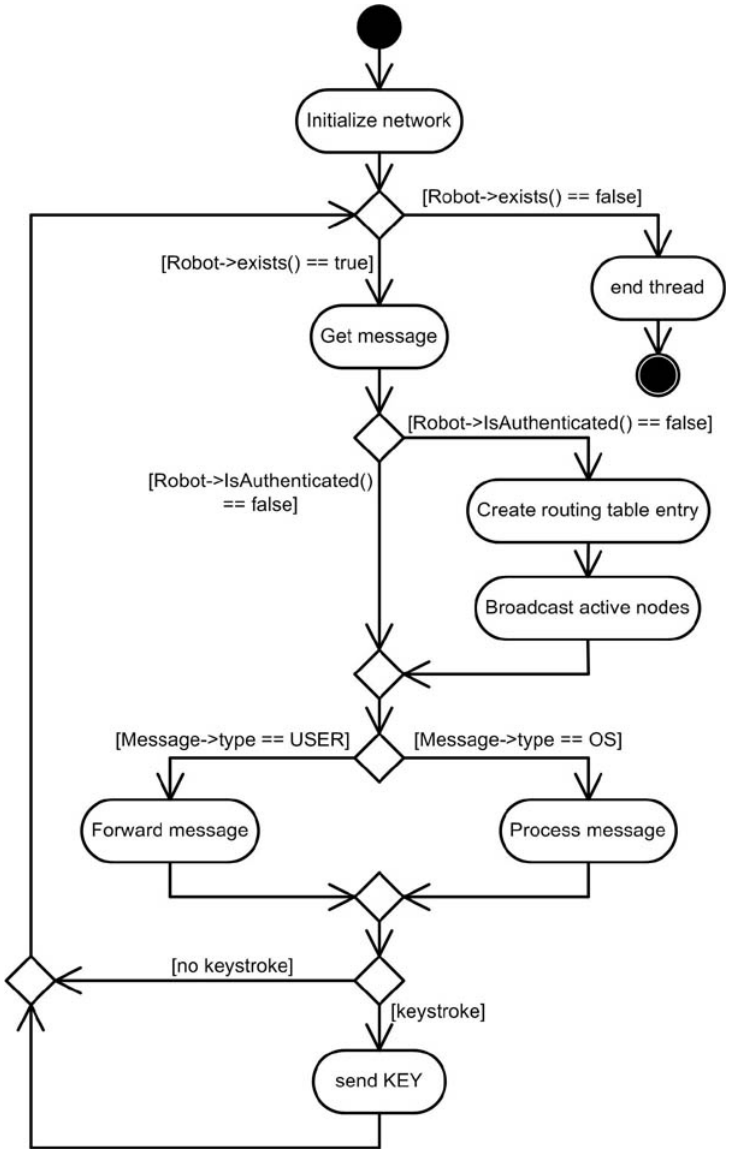


Figure 4.14: Thread function

4.5.2 Implementation

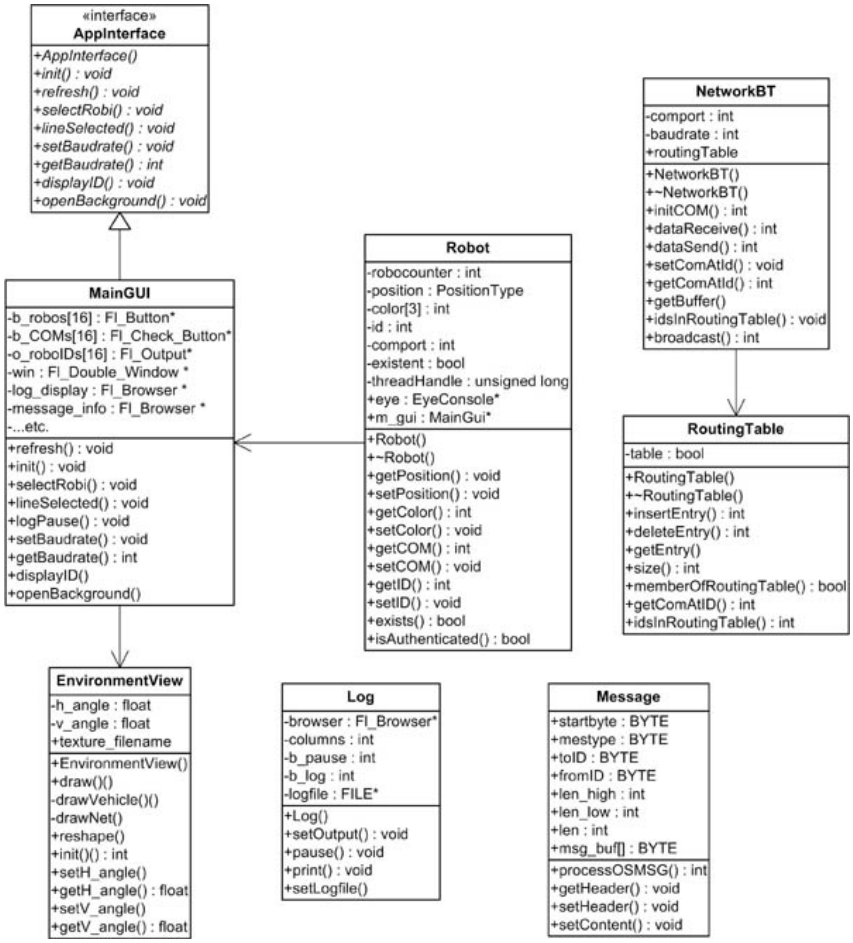


Figure 4.15: Class diagram

RemoteBT was developed as an object-orientated program to reuse the main part of it in the RemoteWLAN program. The source code was written in C/C++. Besides some classes from the previous remote program, such as `class EyeConsole` which were essentially adopted, a few new classes have been implemented. The following will give an overview on these classes to provide a basic understanding.

Interface AppInterface

The `class AppInterface` is the appropriate interface to the MainGUI class. It defines a few pure virtual functions. Furthermore, it contains all variables that do not belong to the GUI but need to be accessible within the MainGUI.

class MainGUI

This class represents, as the name suggests, the main window. It is created with the GUI builder FLUID. It contains all GUI elements (buttons, menubar, tickboxes, openGL window), their callback functions, and further functions. It derives from the interface `class AppInterface` that contains the non-GUI variables. This design was chosen to split the pure GUI relevant code from the rest. This has two advantages. On the one hand the GUI can easily be modified with FLUID without problems as long as the interface functions are overwritten. On the other hand FLUID indeed provides functionality to add code, but a real IDE provides more programming comfort. Therefore, the messy source code, delivered by FLUID, contains only the GUI variables, their callback functions and function declarations, whereas the function definitions are written in a separate file.

class NetworkBT

This class handles network based properties. It contains information about the COM port and the baud rate of an RS232 connection. An object of the RoutingTable class is a class variable of the NetworkBT class. That way there is one common routing table for all instances of the NetworkBT class. Functions to initialize the COM port and to send and receive messages are included here.

class RoutingTable

The routing table keeps track about the relationship between the COM ports and the robots' IDs of current connections. The routing table contains entries from `struct Entry` which itself consists of values for a COM port and an ID. Functions to retrieve information from the routing table are available.

class Robot

`Robot` is the class from which an object is created for each single robot connected to the network. All robot related information is encapsulated here, including a robot's ID, position, symbol color, existence status, the COM port it is connected to, as well as a thread handle. Access to these properties is given through a few different get/set functions. The thread function which is started with the creation of a new `Robot` object is not included

in the class, but can be found in the same source file.

class Message

The purpose of this class is the processing of received messages. It stores the message's header as well as its content. By calling the `processOSMSG()` function, an OS message is evaluated and according to its code byte processed as either a command for the EyeConsole or for the 3D environment display.

class Log

The `Log` class is mainly responsible for the output of error messages and traffic in the display in the upper right corner of the GUI. It provides a function to specify to which display the output should be written. Function overloading is used with two print functions to print either a common string or a resolved message. `Log` also provides the opportunity to pause the output and to write it into a logfile.

class EnvironmentView

The `EnvironmentView` display gives an impression of the actual robot's environment. `EnvironmentView` is derived from the `Fl_Gl_Window` class which supports the use of openGL, but demands a draw function. The `draw()` function gets the positions of the living robots and uses further functions to draw the background as well as symbols for the robots to draw the whole scenery. The viewing angles are kept in the class and can be accessed by the appropriate get/set functions. Furthermore, the class is responsible in loading and creating the background texture.

Chapter 5

EyeNet WLAN Network Implementation

This chapter is organized in the same way as the earlier chapter about Bluetooth. An overview about the wireless LAN standards gives a basic knowledge about the underlaying technology of EyeNet in section 5.1. The used hardware within the WLAN solution and its setup program for the EyeBot controller is then introduced. Furthermore, the topology is shortly presented in section 5.3 The concluding section describes the remote program for WLAN, called RemoteWLAN. It shows how the former Bluetooth version could be used by slightly changing the source code according to the network functions.

5.1 WLAN Technology

Not less large as the Bluetooth specifications are the ones for WLAN[1] (Wireless Local Area Network). A version of the IEEE 802.11 standards, covering wireless LAN, are included in the enclosed CD (see Appendix C). This section introduces the basics to provide a quick understanding of WLAN.

5.1.1 Introduction

The IEEE 802.11 specifications are a family of standards that define a medium access control and a physical layer for wireless networking between fixed and portable devices.

[1] In this context WLAN refers to the IEEE 802.11 standards

IEEE 802.11 standards				
	802.11	802.11a	802.11b	802.11g
Data rates[1] [Mbps]	1, 2	6, 9, 12, 18, 24, 36, 48, 54	1, 2, 5.5, 11	1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54
Frequency [GHz]	2.4, IR	5	2.4	2.4
Transmission technique	FHSS, DSSS, Infrared	OFDM	HR-DSSS	OFDM
+/-	- low data rate + unlicensed ISM band	+ very high data rate + no interference with Bluetooth - incompatible to 802.11b - lower range	+ high data rate + unlicensed ISM band	+ very high data rate + unlicensed ISM band + compatible to 802.11b

Table 5.1: Part of the IEEE 802.11 standards

It was first released in 1997. In 1999 the former WECA (Wireless Ethernet Compatibility Alliance) was founded and is nowadays known as the Wi-Fi (Wireless Fidelity) Alliance. Their task is to certify and confirm the interoperability of IEEE 802.11 devices. Only those products are allowed to carry the Wi-Fi label.

WLAN differs from WPAN(IEEE 802.15, i.e. Bluetooth) in higher transfer rates (1 - 54 Mbps), higher transmission power (100mW) and larger range (300m). As seen before, WPAN has its own specific protocols for connection establishment and higher services, whereas WLAN specifies only the two lowest layers of the OSI model.

5.1.2 Standards & Protocol Stack

The original 802.11 standard defines the MAC layer protocol and three possible techniques for the physical layer. Two of them are based on a spread spectrum technique, the other one is infrared, which never grew importance. Both spread spectrum techniques work in the unlicensed 2.4GHz ISM band (cf. section 4.1.3). Unfortunately, all three of the techniques have limited transmission rates of up to 2Mbps. To be competitive to 10Mbit/s Ethernet the data throughput had to be increased. Two further spread spectrum techniques were introduced to IEEE 802.11, providing higher bandwith of 54Mbps and 11Mbps. These different techniques led to the different substandards of IEEE 802.11,

[1] Gross data rate

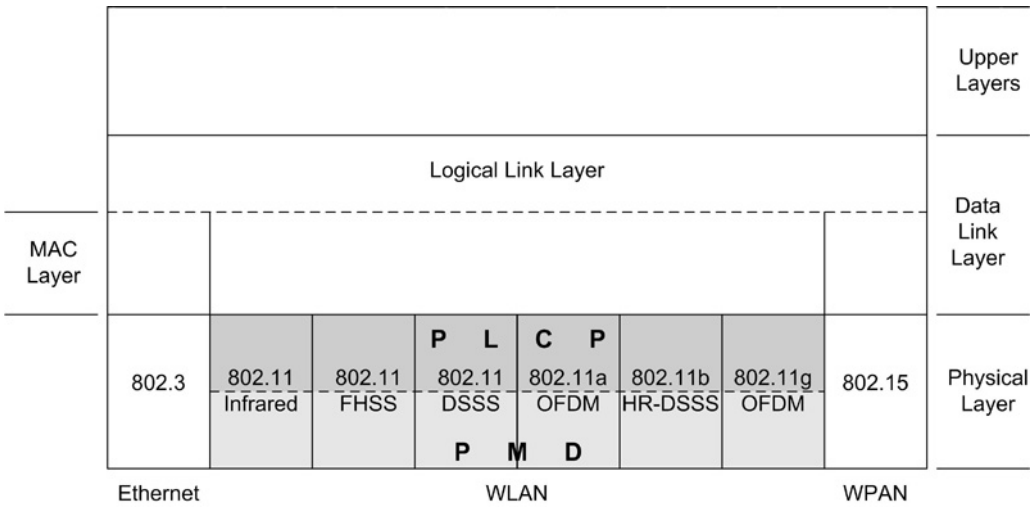


Figure 5.1: Part of the 802 protocol stack

seen in Table 5.1. Further details on the spread spectrum techniques are explained in section 5.1.3 (Physical Layer).

Besides the IEEE 802.11a, b, and g there also exist e, d, f, h, i, etc. extensions for the standard. These are amendments for IEEE 802.11 and contain topics such as security enhancement and Quality of Service requirements.

Figure 5.1 shows a part of the 802 protocol stack. As can be seen, the Data Link Layer of the OSI model is split into the MAC and the LLC sublayer. The LLC is the same in all the standards of IEEE 802, providing a common interface to the next layer. Therefore, higher protocols such as TCP/IP can work independently from the MAC and PHY layer. Thus, the higher protocols can use 802.11 protocols as well as others like IEEE 802.3 (Ethernet).

5.1.3 Physical layer

Four of the five transmission techniques of today's WLAN are based on the spread spectrum technique. The fifth is infrared, which never grew importance due to the restriction of direct line of sight connections and is only mentioned for the sake of completeness. The two transmission techniques of the original IEEE 802.11 are Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS). The spread spectrum technique converts a narrowband signal into a wideband signal to make it less vulnerable

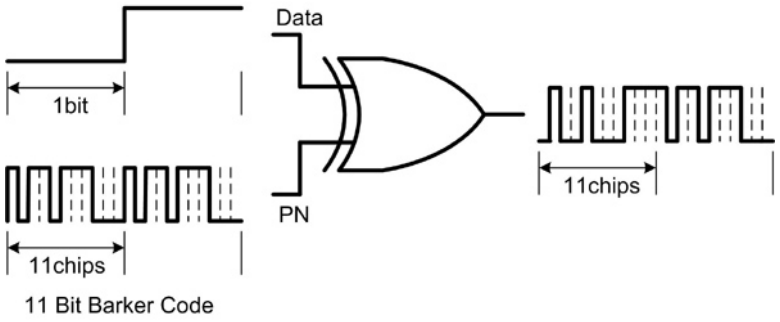


Figure 5.2: DSSS technique

to narrowband interferences. The energy of the signal keeps the same. FHSS is also used in Bluetooth technology and is explained in chapter 3. Unlike Bluetooth, in IEEE 802.11, the time spent at each frequency can be up to 400ms (Bluetooth 625μs). DSSS uses a pseudo random numerical sequence (PN) and XORs it with a bit of the data stream. The PN consists of several intervals called chips and the combination with the data leads to the spreading of the narrowband data signal. In IEEE 802.11, the PN is the Barker code, a special 11 bit sequence with good autocorrelation properties. As illustrated in Figure 5.2 one data bit is encoded by 11 chips, forming a symbol. Finally, the new 'chipped' data stream is either modulated with Binary Phase Shift Keying (BPSK) or Quadrature Phase Shift Keying (QPSK) to accomplish 1Mbps or 2Mbps rates. A decoder in the receiver decodes the chips and creates the original data stream. In the reverse procedure, the wideband signal becomes strong and narrowband again. A narrowband interference is spread and becomes wideband with a low intensity.

The bandwith of the spread signal is 11chips$\times$2Mbps=22MHz. Therefore, there can be 3 not-overlapping DSSS channels, each 25MHz away in one location. The whole ISM band is separated into 14 overlapping channels, but only 3 of them are used in one location.

The high-speed WLANs use Orthogonal Frequency Division Multiplexing (OFDM) or High Rate Direct Sequence Spread Spectrum (HR-DSSS). 802.11a and 802.11g are representatives of OFDM[2]. OFDM splits the data signals and transmits them on multiple frequencies in parallel at the same time. Each data signal uses one sub-carrier. Unlike plain FDM, the sub-carriers are closer together and overlap in OFDM. They can overlap

[2]802.11a works on 5GHz, 802.11 works on 2.4GHz

because their frequencies are orthogonal in a mathematical sense and their spectra do not interfere. In 802.11a,g, each 20MHz wide channel (carrier) is broken into 52 sub-carriers, each approximately 300kHz wide. 48 sub-carriers are used for data transmission and 4 for synchronization. Depending on the desired transmission rate the signal is modulated by either BPSK, QPSK or QAM (Quadrature Amplitude Modulation).

HR-DSSS is used in 802.11b. To achieve higher data rates up to 11Mbps, 64 different chipping sequences known as Complimentary Code Keying (CCK) are used as PN. Eight data bits are grouped together. The first 6 bits select one of the 64 ($=2^6$) unique code words, each 8 chips long. The remaining 2 bits determine the QPSK modulation for the symbol. Therefore, one symbol consists of 8 modulated chips, which encode 8 data bits. For reasons of compatibility the symbol rate is 1,375 MSps to provide the 11Mcps of 802.11.

Figure 5.1 illustrates the division of the physical layer into the Physical Medium Dependant (PMD) and the Physical Layer Convergence Protocol (PLCP) sublayers. The PMD deals with modulation and coding. The PLCP is independently from the medium an interface for the MAC layer and provides the Clear Channel Assignment (CCA) signal.

5.1.4 MAC layer

The MAC layer uses two access methods called Distributed Coordination Function (DCF) and Point Coordination Function (PCF). The implementation of DCF is mandatory, whereas PCF is optional. DCF is based on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism. The term CSMA means that several stations use the same channel and monitor its state. Unlike on a wired medium a collision detection is not possible on a wireless medium, hence the mechanism tries to avoid them. Each station has to listen before it can talk (LBT). If the medium is free for a special time known as Distributed Interframe Space (DIFS), a station can begin transmission. If it is busy the station waits until the medium is idle plus the DIFS time plus an additionally pseudo random time. This avoids several stations beginning transmission immediately after the channel is idle again. Even so, a collision can occur and an acknowledgment (ACK) for each frame is needed. The ACK frame is sent after a short IFS (SIFS) by the recipient. The shorter an IFS is, the earlier transmission starts and therefore, the higher its priority. It may occur that even the ACK frame is jammed. If so, the sending station increases the

wait time and tries to retransmit the frame until it receives the ACK. Another problem known as the Hidden-Station is treated with a Ready To Send / Clear To Send (RTS/CTS) handshake mechanism. A hidden station situation occurs when not all stations can talk to each other. In such a case a station senses the medium as idle, whereas it is used by another station out of its own range. To prevent this, the sender transmits a RTS frame and waits for the acknowledgment with a CTS frame. These frames are sent after a SIFS so they have a higher priority than data transmission. Both frames also carry information about the total time for the data frame transmission. Either a RTS or CTS triggers the Net Allocation Vector timer of all the other stations that receive the RTS/CTS frame. The NAT counts down the total time and during this period a station does not try to transmit. This method decreases the probability of collisions.

The above mentioned optional PCF access method uses a point coordinator (PC) to determine which station gets the transmission permission. The PC is an access point and therefore PCF is only available in infrastructure mode (cf. section Topology). PCF is based on a polling system with the PC as polling master.

Furthermore, the packet fragmentation/defragmentation and the Frame Check Sequence, a CRC checksum are managed in the MAC layer.

5.1.5 PLCP Packet Formats

Figure 5.3 illustrates the two different packet formats for FHSS and DSSS/OFDM. Both start with a preamble used for synchronization and are followed by the PLCP header. The header carries length and CRC information, and in the case of FHSS 4 bits code the transmission speed. In the case of DSSS/OFDM the transmission speed is represented by 8 bits and an additional 8 bits are reserved for future use. The variable payload concludes the packet. Due to the header informations and handshake messages the net data rate is around 40% less than the gross rate.

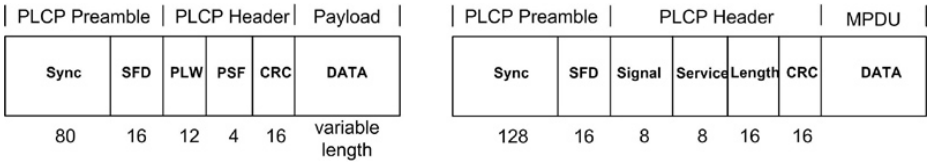


Figure 5.3: Frame Format for FHSS (left) and DSSS/OFDM (right)

5.1.6 Topology

The IEEE 802.11 standard specifies the two following operating modes.

- Ad-hoc mode
- Infrastructure mode

In the ad-hoc mode wireless endpoints communicate directly with each other over a peer-to-peer network. The official IEEE name for this kind of network is Independent Basic Service Sets (IBSS). It allows an easy, quick and cheap setup for a small range network, but does not provide a connection to a distribution system (DS), i.e. a wired network. In a network with infrastructure mode the communication takes place via an access point(AP). The end stations connect to one AP, which also forms a bridge to the wired network (DS). Such a network is called a Basic Service Set (BSS). Two or more BBSs and a DS form an Extended Service Set (ESS). Each end station belongs to one BSS and use the AP's transmission channel, but may change their affiliation when moved to the next BSS.

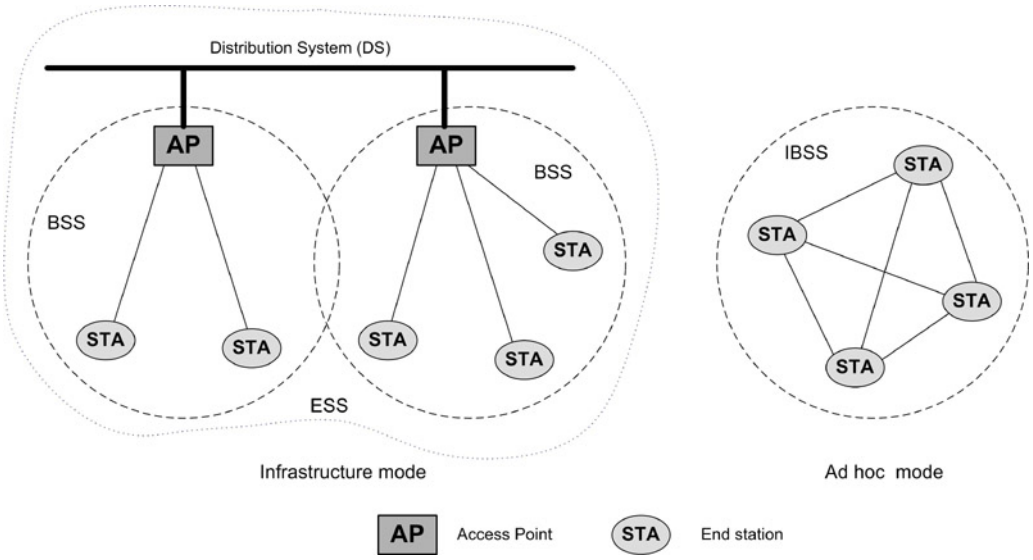


Figure 5.4: WLAN modes

5.1.7 Security

Security is an important issue in networks, especially in wireless networks where the data stream is not limited to a cable but spread over radio signals. Nevertheless it is difficult to eavesdrop the signal due to the spread spectrum techniques. Up to now there are three major security concepts implemented in WLAN.

1. Service Set Identifier (SSID)
2. Media Access Control (MAC) address filtering
3. Wired Equivalent Privacy (WEP)

The SSID can be compared with a network name. The SSID must be configured in each device to gain access to the wireless network. It is possible to enable a broadcast for the SSID, eliminating the security aspect of SSID.

The MAC address filtering can only be used in an infrastructure network. The AP(s) keep(s) a list of MAC addresses of the stations that are allowed to connect to the network. However, a MAC address can be changed[3] and in larger networks the administration effort rises since every AP needs an up-to-date list.

WEP is an encryption mechanism to secure the transmitted data. Either a 40 bit or 104 bit secret key, which is distributed to every station in the network, is concatenated with a 24 bit initialization vector. These 64 or 128 bits are given to a RC4 random number generator and the output is XORed with the data. Unfortunately studies showed that the WEP mechanism is not sufficient.

The correct configuration of the three concepts provide a reasonably level of security and further work concerning security is done in the IEEE 802.11i work group.

[3] Although meant to be a unique identifier tools exist to change the MAC address

5.2 Used Hardware

To integrate the EyeBot controller into the world of WLAN a so called WiPort was purchased. With a PC running the remote control program a standard IEEE 802.11b network card can be used to hook into EyeNet.

5.2.1 Lantronix WiPort

Figure 5.5: Lantronix WiPort

The WiPort is an embedded WLAN device server and allows the connection of a serial device such as the EyeBot to a wireless network. It works on the IEEE 802.11b standard and supports several network protocols, including TCP, UDP, Telnet,and many others. Two serial interfaces with a baud rate of up to 921kbps as well as a 10/100 Ethernet interface are provided. The WiPort requires a 3.3V DC power supply provided through a pin on the EyeBot. Unfortunately, the peak current consumption reaches 460mA which is close to the 500mA limitation of the microcontroller's power supply controller. The high power consumption leads also to a short battery life time. Detailed information on the WiPort can be found in [23] [24] [22] [21].

The server, a power control unit, status LEDs and the physical interfaces (1× serial port, 1× 10/100 Ethernet port) are placed on a small circuit board and together with an

antenna form the WLAN module. A male adapter was chosen for the serial port to connect the module directly into the EyeBot's COM port.

5.2.2 Setup

Like the HandyPort in the Bluetooth solution, the WiPort requires some configurations first. These can be done through:

1. DeviceInstaller software - requires a PC & network card
2. WebManager - requires a PC, network card & Webbrowser
3. Telnet - requires PC & network card
4. Serial Port using terminal program - requires PC & serial cable
5. Serial Port using EyeBot config menu - requires EyeBot

The last option was created to be able to configure the hardware directly on the EyeBot. This avoids annoying plugging between the EyeBot and a PC, which is not even required. 'How-to's for the other options can be found in the WiPort user manual ([24]).

Program Structure

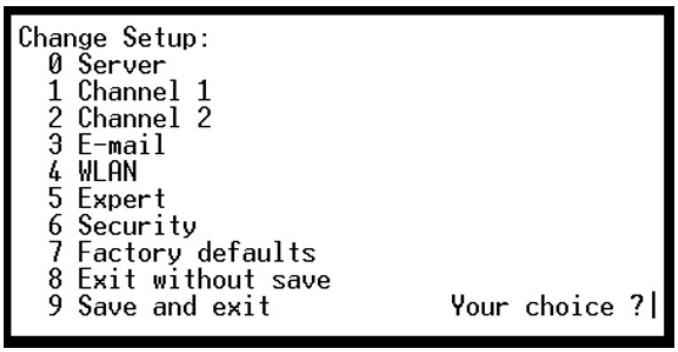


Figure 5.6: WiPort Configuration menu

The configuration menu program for the EyeBot adopts the behavior of the configuration menu program for a terminal program. Figure 5.6 shows the main menu displayed in

a terminal program. Because the features of the WiPort cover more than is used in EyeNet only the relevant menu items were necessary to implement. These are:

0 Server
Basic device settings for WiPort, i.e. IP address, netmask.

1 Channel1
Settings regarding to the first serial port, i.e. baud rate, flow control.

4 WLAN
Basic network settings, i.e. SSID.

8 Exit without save

9 Save and exit

To enter the main menu of WiPort the program sends `0x78` in ASCII code for a 'X' until the device responds. This emulates the 'X'-keystroke in the terminal program. The WiPort responds with its MAC address and with an emulated Enter keystroke the menu is reached. Besides the above displayed main menu, the WiPort also sends all its configuration values which are buffered. Figure 5.8(b) shows the main menu for the WiPort's configuration menu on the EyeBot. Each menu item has two appropriate submenus, one to display their values and one to change them. On entering a menu item the program emulates the keystroke which would have been sent in the terminal program. If the `INF`ormation submenu is chosen, the program sends an 'Enter' keystroke to receive the actual values. Some implemented string compare functions are executed to find the right values, which are finally displayed. Entering the `SET`up submenu displays a new menu with further menu items. Changes to these menu items will be stored and on exit of the submenu they will be transmitted (emulating the according keystrokes again). The final storage of all current values in the WiPort is achieved when the menu program is exited. Figure 5.7 illustrates the workflow.

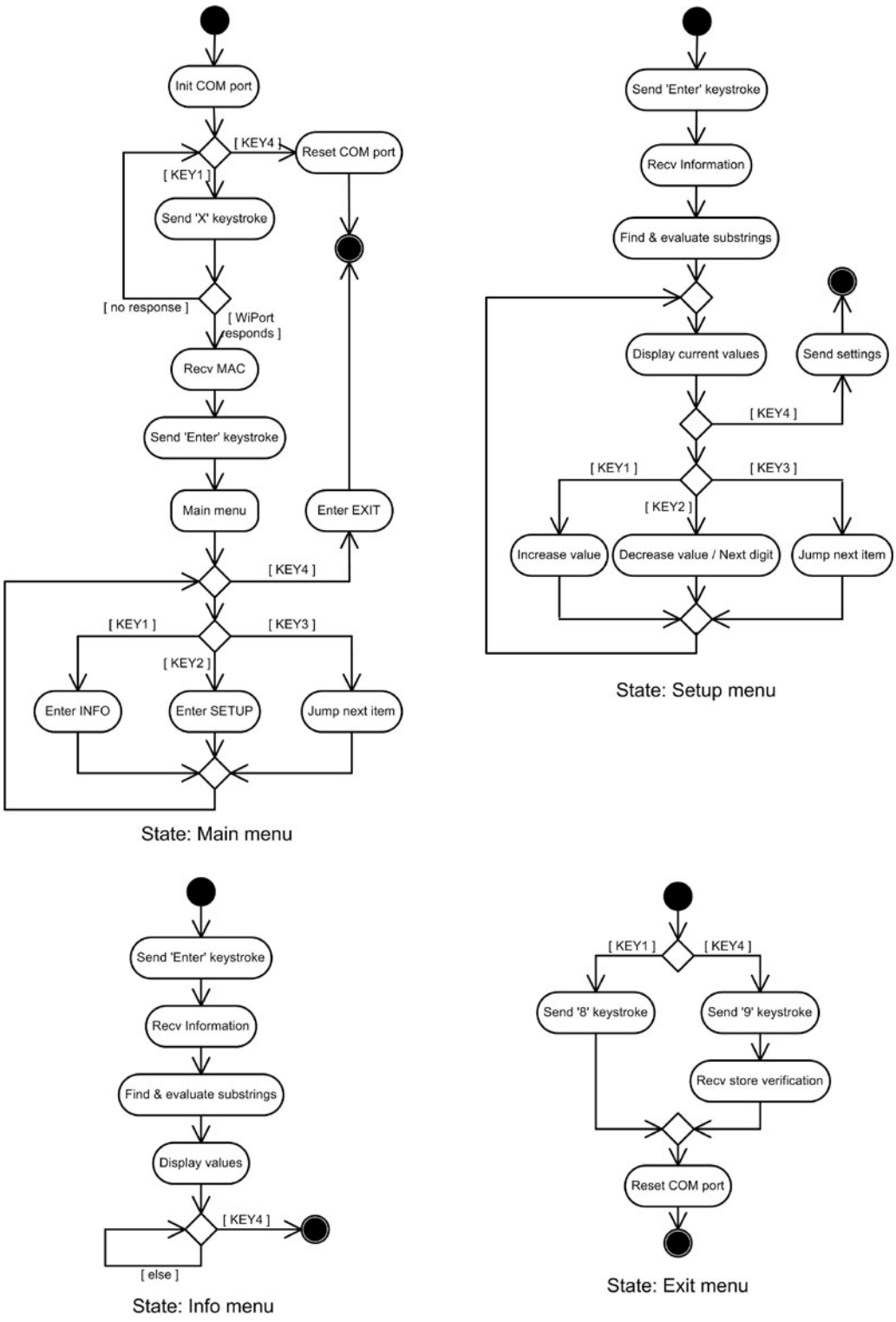


Figure 5.7: Program structure

EyeBot Config Menu

The configuration menu for the WiPort is included in the demo programs and according to them also written in C. It can be found on the EyeBot under:

Demo ⇒ Network ⇒ WLAN.

Figures 5.8(a) up to 5.8(i) present the different menus.Until the main menu on the WiPort is reached, the LCD displays '`connecting...`'. The 4 keys on the EyeBot can be used to navigate through the menus.

Server menu:
Available items are:

- **MAC:**
  The WiPort's hardware address (unchangeable).
- **IP address:**
  The WiPort's IP address can be configured here.
- **Mask:**
  The number of host bits of the netmask are set here.

The '+' key increases the numbers, the '⇒' key jumps to the next digit.

Channel 1 menu:
Available items are:

- **Speed:**
  Available baud rates are 9600, 19200, 38400, 57600, 115200, 921600 bps.
- **I/F Mode:**
  The bit combination distinguishes the parity and stop bit settings[4].
- **Flow:**
  Sets the flow control settings.
- **PortNo:**
  The Port No setting represents the source port number in TCP connections.

[4]Refer to the EyeNet WLAN or WiPort manual

- **ConnectMode:**
  Distinguishes between TCP and UDP connections.

- **DatagramType:**
  Only with UDP. Sets UDP datagramm to directed or non-directed.

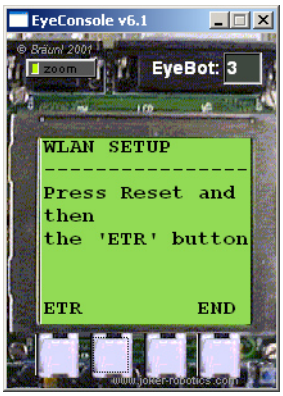
WLAN menu:

Only `INF`ormation menu available. Settings here can remain unchanged.

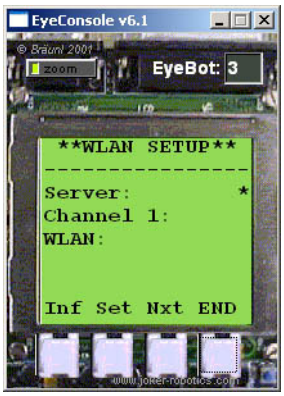
- **WLAN:**
  Displays if WLAN is enabled or not.

- **SSID:**
  Shows the SSID of the network.

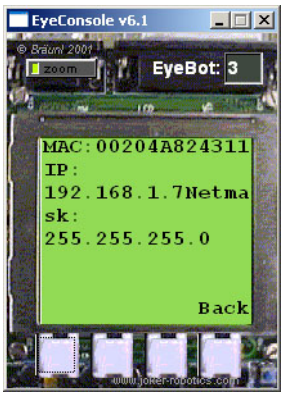
- **WEP:**
  Displays the WEP settings



(a) Enter menu

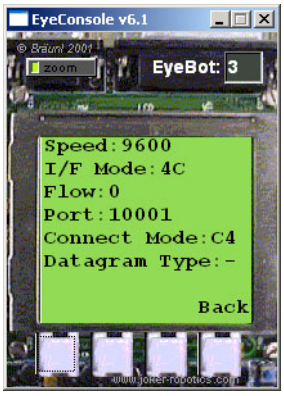


(b) Main menu

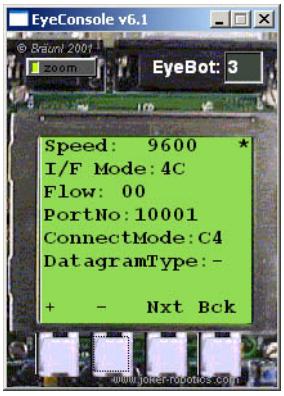


(c) Server Info menu

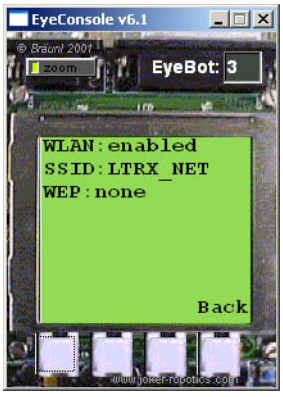
(d) Server Setup menu



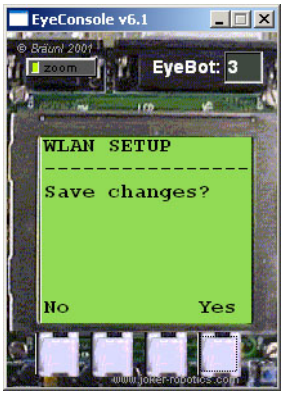
(e) Channel 1 Info menu



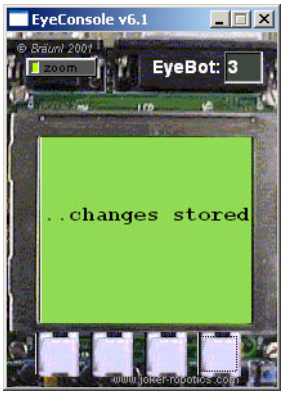
(f) Channel 1 Setup menu



(g) WLAN Info menu



(h) Exit menu



(i) Save verification

Figure 5.8: WiPort Config Menu for EyeBot

5.3 Network Architecture

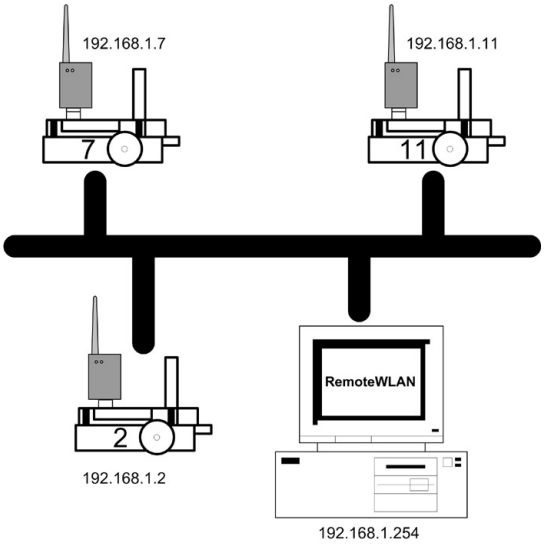


Figure 5.9: Logical topology

As seen in an earlier section a WLAN can work in either Ad-Hoc or infrastructure mode. EyeNet is meant to be an easy and quick configurable network and therefore an ad hoc network, but it should also work in an infrastructure network. As an ad hoc network, all EyeBots are linked together forming a fully connected network in a physical manner. The logical topology is a bus.

EyeNet is implemented as a small private class C network. Its IP range is from

IP range: 192.168.1.1 - 192.168.1.253

with a subnetmask of

Subnetmask: 255.255.255.0

Like in every network the ‘0’ and ‘255’ for the host are reserved as broadcast addresses. In EyeNet the ‘254’ is also a reserved number. For every EyeBot its IP address is equivalent to its ID number (i.e. ID = 7 $\Rightarrow$ IP = 192.168.1.7). All the operating system messages for the remote control program are send to the ID zero. Because the according IP address would be a broadcast address, the ID zero is assigned to the ‘254’. Therefore, the network card of the PC running the RemoteWLAN program has to be configured to the 192.168.1.254 address.

5.4 Connection-Oriented or Connection-Less

In the beginning of the implementation of the WLAN version a question was raised concerning the transport layer protocol. The WiPort's network protocol stack contains both TCP and UDP. The advantage of the connection-oriented TCP is its reliability due to a handshake mechanism. UDP is connectionless and data is sent without any acknowledgments of reception from the destination node. One might say that TCP is then the better choice, but there is a need for a closer look at the conditions of EyeNet.

The reliability of TCP is an advantage as long as small messages are sent. If a message is lost, TCP takes care of its retransmission. This might be valuable for important position or command messages for the EyeBots, but poses a problem with the EyeBot's camera image transmission. If such an image is lost it will be retransmitted until the PC[5] acknowledges its error free delivery. Due to the fact that the retransmission might need several tries the image is already old and worthless and furthermore it blocks the next image. To discard the image would be the best solution in this case.

UDP instead does not care whether a message is received or not, and is therefore widely used for multimedia transmission. In the case of image transmission in EyeNet it is more adequate than TCP. The advantage of reliable transmission of important messages is indeed not available with UDP.

The final decision of which transport layer protocol to use was decided by something else: Bugs. During the project and the work on the relatively young WiPort some problems emerged within the use of the UDP mode. The following two bugs have been reported to and were verified by Lantronix.

1. The Most Significant Byte (MSB) of the length value in the message header for UDP is not evaluated. Thus the maximal data length can be 255 bytes.

2. Between each UDP message WiPort requires a pause >10ms. Otherwise messages are randomly lost.

With these significant problems, the UDP mode was not useful. Especially in the case of image transmission, UDP was insufficient, since an image of 14400 bytes had to be split into 57 parts with >10ms pause in between. The smallest sleep time step of RoBIOS

[5]Image transmission in EyeNet is explicit between EyeBots and a host PC

is 10ms. Therefore, the pause between two messages were 20ms (>10ms) resulting in more than 1 second wait time. The work was continued using TCP as the transport layer protocol.

5.5 RemoteWLAN

RemoteWLAN is the comparable program to RemoteBT for the WLAN solution. It serves the same purposes except that there is no need for a routing mechanism. Routing is covered through the IP layer.

Figure 5.10 shows the GUI for RemoteWLAN. It may be remarkable that there is not a big difference with RemoteBT. Neither is there a big difference in the functionality and implementation. The network functionality merely had to be altered.

In the upper left corner of the GUI the former COM port selection field was replaced by a network overview field. This display shows all currently connected EyeBots and their IP addresses. The EyeBot symbols serve as buttons to open the according EyeConsole window. The 'Refresh' button refreshes the overview field.

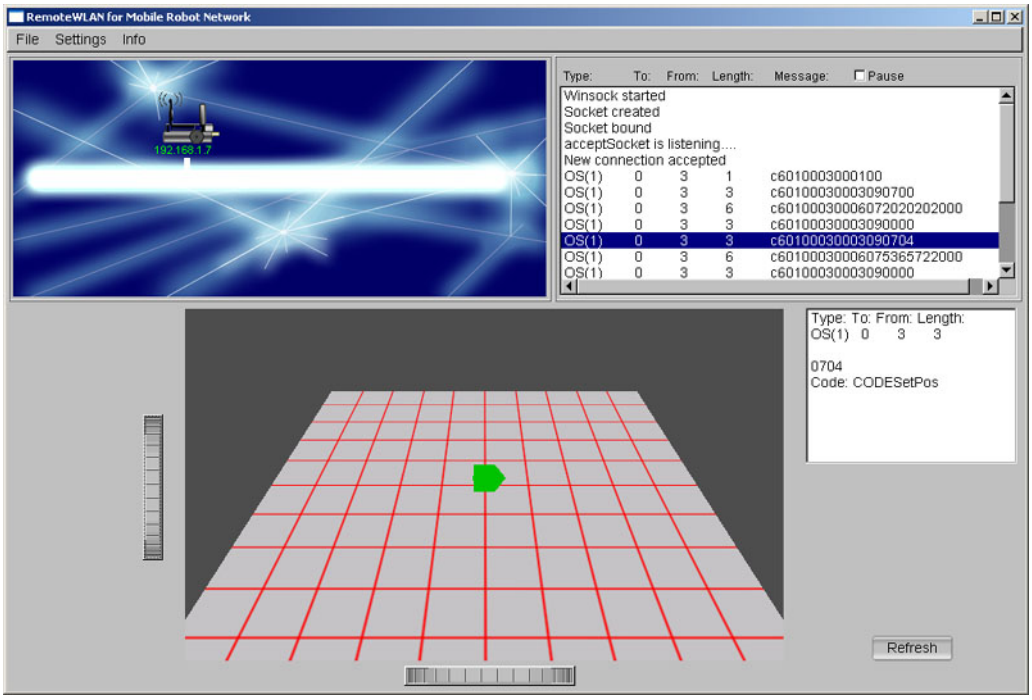


Figure 5.10: RemoteWLAN GUI

5.5.1 Program Structure

The program structure is basically the same as in RemoteBT (cf. section 4.5) and only the changes will be discussed here.

After program execution the 'Refresh' button can be used to broadcasst an ICMP ping message. The numbers of retries for the ping can be set. For each responding EyeBot a new instance of the Robot class will be created and a new thread started. If a robot is already existent in the network this will be skipped. Robots that are no longer in the network are destroyed. Within each thread an instance of the Network class is created. Then a server socket is created that provides sequenced, reliable, two-way, connection-based byte streams used for a TCP connection. The socket is bound to a port and listens for incoming connection requests. As long as an EyeBot does not connect[6] to the socket the thread does not claim any CPU resources due to its blocking mode. If the listening socket finds a request it will accept it and the thread function is executed.

5.5.2 Implementation

The view changes in the software required an implementation of two further classes, dealing with the access to the network.

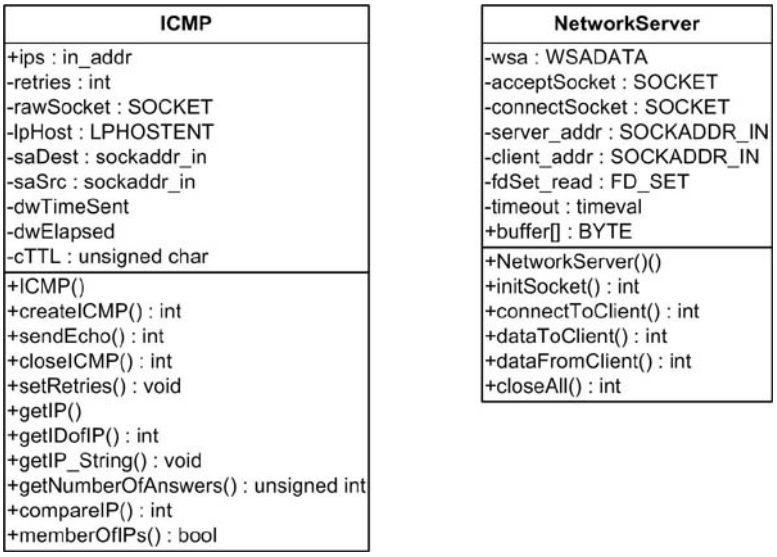


Figure 5.11: The two additional classes

[6] connects by `RADIOInit()`

ICMP class

As the name suggests, this class deals with the Internet Control Message Protocol [9]. It is used to create a broadcast ping with a raw socket [28]. The required data structures to create the ping message are defined in the same source file. The approach of the raw socket was preferred to the use of Microsoft's ICMP.dll [32], which is not an official part of the Win32 API and will not be supported in future. The raw sockets also ease the conversion of the source code for a Linux version.

The class has a vector variable that stores the IPs of all responding network nodes. Furthermore, a few functions provide access and other operations on these IPs.

NetworkServer class

This class is the equivalent to the NetworkBT class. Unlike the NetworkBT, the NetworkServer class is based on sockets rather than COM port connections. But the other features are almost the same. This includes the opportunity to receive and send messages. Information on socket programming was gained from [26] and [34].

Chapter 6

Comparison of EyeNet Versions

The two previous chapters dealt with the solution for the EyeNet based on the Bluetooth and WLAN technology. This chapter now compares both versions and exposes their disadvantages. This chapter also dwells on outstanding advantages of both.

6.1 Comparison of the basic Technologies

First it has to be mentioned that comparing Bluetooth and WLAN is like comparing apples and oranges. Both technologies are meant to cover different roles. Bluetooth is designed to be a low power and cheap solution for easy short-range networks like WPANs. The IEEE 802.11 standards indeed are designed for WLANs with higher data rates, larger range and transmission power. Nevertheless, this section shows the main differences between both.

	Bluetooth	**WLAN**
Data rate	1 Mbps	54 Mbps
Range	10 - 100 m	≈300 m
Max. current	100 mA	420 mA
Transmission power	1 - 100 mW	100 mW
Active nodes	8	unlimited
Number of parallel systems	theoretical: 80 use case: 10	3
Frequency	2.4 GHz	2.4, 5 GHz

Table 6.1: Characteristics of Bluetooth and WLAN

Data rate

Bluetooth provides 1Mbps (721kbps net) and 2Mbps are planned for future versions. WLAN provide different levels of data rates (1, 2, 5.5, 6, 9, 11, 12, 18, 24, 36, 48, 54). The net data rate is around 60%.

Range & Transmission power

Bluetooth is able to range 10m in the low power mode and up to 100m (outdoor) in the high power mode (100mW). WLAN transmission reaches up to 300m (outdoor) with a transmission power of 100mW.

Number of active nodes

Eight Bluetooth devices can be active in a piconet. The number of nodes in WLAN is theoretically unlimited, but restricted due to the fact that the bandwith is decreased the more nodes are active.

Number of parallel systems

Theoretically, up to 80 piconets (there are 80 1MHz frequency slices) can exist, but an empirical number is 10. WLAN is restricted to 3 different networks, because only 3 channels are allowed at the same place (cf. section 5.1.3)

Frequency

Both work in the 2.4 GHz ISM band and additionally a frequency range around 5GHz is used in WLAN. Due to the fact that both technologies use the same frequency range, they interfere with each other. Bluetooth causes more trouble for WLAN than vice versa. Bluetooth hops faster between the frequencies and therefore it hops out of the interference faster.

6.2 Comparison of EyeNet BT and EyeNet WLAN

This section gives a direct comparison between the two EyeNet versions. Some differences are based on the technologies, some are based on the used hardware and some are based on the implementation.

	Bluetooth	**WLAN**
Performance	115200bps	115200bps[1]
Range[2]	≈8m	≈20m
Max. current	100mA	420mA
Topology	Star, requires routing software on PC	Bus
Active Nodes	8 (7 EyeBots, 1 Routing&Remote Controll PC)	254 (253 EyeBots or PC, 1 Remote Controll PC)
Price:	≈$80	≈$150

Table 6.2: Comparison EyeNet with Bluetooth and WLAN

Performance

Although the data rates of Bluetooth and WLAN are by far higher, the data rates for EyeNet are limited by the bottleneck formed by the serial port. Unfortunately, the EyeBot's serial port is limited to 115200bps so that the advantage of the WiPort's higher baud rate is futile at the moment. The next version of the EyeBot called Mark V will support higher baud rates.

Image type	Resolution *(bytes)*	**Bluetooth**	**WLAN**	**BiM-418**
		frames per second (seconds per image)		
Colour	high (14400)	0.6 (1.51s)	0.6 (1.51s)	<0.1 (68.59s)
	low (1800)	2.5 (0.4s)	2.5 (0.4s)	0.1 (9.59s)
Grey	high (4800)	1.5 (0.64s)	1.5 (0.64s)	<0.1 (24.91s)
	low (600)	3.1 (0.32s)	3.1 (0.32s)	0.2 (4.72s)

Table 6.3: Comparison of performance

Table 6.3 shows the results of a performance test with image transmission. As expected are the results for Bluetooth and WLAN identical which confirms that the serial port is the bottleneck of the system. However, the performance of the new EyeNet versions are by far higher than the performance was before. An acceleration with a factor 3 is based on the higher baud rate of the Bluetooth and WLAN modules (115200bps vs. 38400bps). A further improvement is the direct data transmission. In the former version

[1]WiPort supports 921600bps, but EyeBot is still limited. The future version of the EyeBot will support higher baud rates.

[2]Worst case:Indoor and heavy traffic load

of EyeNet, messages were queued and a background routine was called only every 1/100 second to send a message. Now messages are sent as they occur.

Range

Other than the given literature values the range for the EyeNet Bluetooth and WLAN are:

	Bluetooth		**WLAN**	
	indoor	outdoor	indoor	outdoor
Ping[1]	≈20m	≈120m	-	-
Image transmission (reduced resolution)	≈20m	≈100m	≈20m	>120m
Image transmission (full resolution)	≈8m	≈25m	≈20m	>120

Table 6.4: Comparison of range

The tests were accomplished with three different kinds of traffic load. The first row shows the experienced values when running a ping program on two EyeBots. The ping program subsequently sends a ping (=1byte message) between the EyeBots. Sending images resulted in a heavier load. Full colour images with a size of 14402 bytes and reduced resolution images with $1/8$ of the original image size were used to cause the traffic. Within the shown ranges the systems worked without malfunctions. When the robot got close to the borders the transmission became delayed. The outdoor measurement took place on the UWA cricket field (approximately 120×70 metres) with direct line of sight. The indoor measurement took place on one level of the Electrical Engineering building of UWA and represents a standard office environment.

Current consumption

The used Bluetooth modules have the advantage of less power consumption which is an important criteria for a battery powered vehicle.

Network structure

The outstanding advantage of the WLAN solution is its ability to manage 253 active EyeBots and a monitoring PC. The EyeNet with Bluetooth is restricted to 7 active EyeBots and the network master (PC running the routing procedure). Furthermore, a PC is required

[1] A second EyeBot with WLAN adapter was not available

to form the Bluetooth network, whereas a PC is optional for the WLAN network.

Price

The used HandyPort Bluetooth modules cost around 80 USD. The WiPort WLAN server device including the appropriate circuit board costs almost twice as much. This difference can be traced back to the fact that nowadays Bluetooth circuit boards are a mass product and cost less then 5 USD. The WiPort is a niche product, covering a small application range.

It is hard to say which of both solutions is more adequate. It depends on the requirements on the network. For small groups of robots (less than 7) the Bluetooth solution provides a cheap and handy network, but requires always a PC and the BlueSoleil software. The WLAN solution provides the opportunity for larger networks, but is more expensive, and the WiPort with its huge range of features might be a little bit of an overkill. But in the end both EyeNet versions fulfill their tasks.

Chapter 7

Conclusion and further work

This chapter concludes this report and will finally give an overview of the latest conditions and further work.

The realization of two new versions for EyeNet using both Bluetooth and WLAN technologies were presented in this thesis. Based on the former version of EyeNet the system library routines had to be re-implemented with the requirement of backward compatibility. Besides the radio routines, a further function for transmitting the EyeBot's camera images was designed new to improve their transmission performance. All the changes were placed in a single new version of RoBIOS able to work on a Bluetooth or WLAN base. The new version is now RoBIOS 6.2

Due to the missing routing ability within Bluetooth, such a functionality was successfully implemented. New to EyeNet is a visual feedback of the EyeBots' positions. All components were integrated into one object-orientated C/C++ program called RemoteBT. The software was currently compiled under Microsoft operating systems but the creation of a Unix/Linux version can be achieved with only a few changes. The software was compiled with MSVC++ 6.0 and MinGW[1].

In a second step the WLAN network was created. A small private class C network was chosen in which robots are identified through their IP, which itself is derived from the robot's ID number. Based on the work for the RemoteBT software the RemoteWLAN software could be programmed with slight changes.

Further work should be done on testing the WLAN network. Because there was only one WLAN module throughout the project, decent testing could not be achieved. Testing

[1] Minimal Gnu for Windows

should include inter-robot communication as well as the correctness of RemoteWLAN. Overall for both versions a strenuous test should be undertaken to see the network behavior under high traffic.

Furthermore, a refinement of the 3D environment view window would be another task. The current primitive robot symbols could be replaced with 3D models such as in the EyeSim[2] simulator. An adoption of the EyeSim's source code should be possible.

Another improvement would be the implementation of a fast image compression algorithm to increase the frame rate and to decrease the network traffic. Short research on this subject was done by integration of a S3TC[3] compression algorithm. Unfortunately, the CPU of the Mark IV EyeBot is not sufficient enough to perform the algorithm fast enough. The implementation in the radio routines showed that encoding an image took as much time as it did to send it. The advantage of less traffic was nullified by the loss of S3TC. Therefore, the algorithm was not used. Either another faster compression algorithm or a faster CPU in the next EyeBot version might improve this issue.

[2]Simulator software for the EyeBot

[3]Compression algorithm for textures. Invented by S3 Inc.

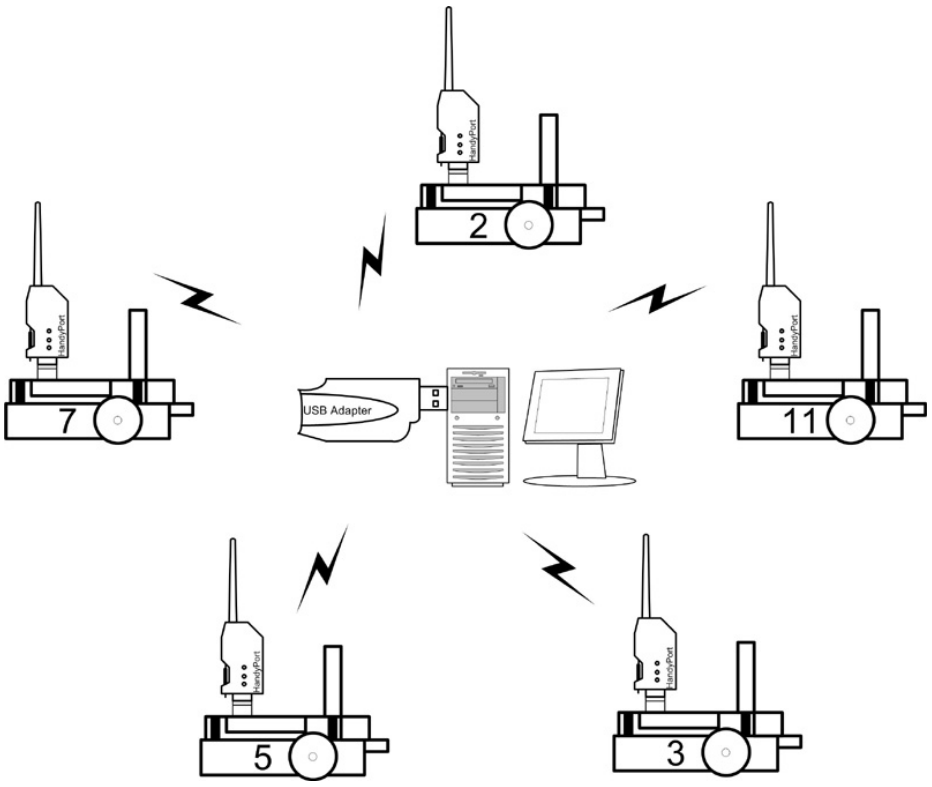
Appendix A

EyeNet BT Manual

The following pages are the technical documentation for the EyeNet with Bluetooth technology. The manual describes all needed configuration for the hardware and the procedures to run the network.

EyeNet

Wireless Network for Mobile Robots
using Bluetooth™ Technology



November 2004

1 Introduction

This manual introduces you to the wireless network solution for mobile robots using Bluetooth technology. The EyeNet gives you the opportunity for communication between mobile robots as well as the remote controll of any robot connected to the network. The next pages will show how to setup, configure and run the small Bluetooth network and its devices.
Make sure you will have the following items:

- Bluetooth Wireless Serial Multi-Port
  (Handywave HPU-100)
- Bluetooth Wireless RS-232 Transceiver
  (Handywave HPS-120)
- IVT BlueSoil MSPP 1.2.0 Software (or higher)
- RemoteUI Software
- EyeBot with RoBios 6.2 (or higher)

2 Quick User Guide

This section gives you a quick step-by-step guide how to run the EyeNet. This guide assumes, that the multi serial port (MSPP) and the IVT BlueSoleil software is installed. Refer to the MSPP manual to see how to install it.

1. Connect the multi serial port to the USB port of your PC.
2. Connect the HandyPort with a Null-Modem adapter to the EyeBot.
3. Switch on the EyeBot and configure the HandyPort as described in section 3.
4. Run the IVT BlueSoleil software.
5. Double click on the red sun. The available Bluetooth devices will appear.
6. Double click on a HandyPort symbol. The MSPP checks the available services for this device. After a short while the serial cable symbol should turn purple.
7. Double click the purple serial cable symbol. A connection to the highlighted HandyPort should be established. A pop-up window shows you the appropriate COM port.
8. Run the RemoteBT program.
9. Tick the COM port boxes to which the HandyPort(s) is(are) connected.
10. Run your EyeBot program that might use the RADIO functions or use the remote functions.
11. The EyeBots should authenticate by transmitting their IDs, which are displayed next to the COM tick box.
12. On authentication an ‘EyeConsole’ symbol appears. Click on the symbol to show the EyeConsole.

3 Configuration

To configure the HandyWave Bluetooth HandyPort modules you can either use a Terminal emulator like Hyperterm or Minicom, or you can use the configuration menu on the EyeBot.

3.1 EyeBot Config Menu

Connect the HandyPort HPU-120 with a Null-Modem cable to the serial port 1 of the EyeBot. Switch on the EyeBot and the red operation (OPR) LED should light. Go to the configuration menu for the HandyPort, which can be found under:
Demo $\Rightarrow$ Network $\Rightarrow$ BLT.
Take a peaked thing, i.e. a pen, and press the reset button (RST) at the HandyPort. The green link (LNK) LED should flash now. Wait 5 seconds after the LED is flashing and then press ETR. You will now see the setup menu. It contains only the items which are relevant to the mobile robot network system.
These items are:

- **Speed:**
  The baud rate of the serial port of the HandyPort can be changed here. Possible speeds are: 9600bps, 19200bps, 38400bps, 57600bps, 115200bps
- **Handshake:**
  The Handshake can either be set to ‘NONE’ or ‘RTS/CTS’.
- **Mode:**
  There are three modes available:
  1. **1:1:**
     This establishes a 1-to-1 connection between the HandyPort and one other Bluetooth device that supports the same mode (i.e. another HandyPort) and has the appropriate settings. The address of the remote device must be specified under ‘RemAdr’. As soon as the remote device is available, the HandyPort will connect to it.

2. **WAIT:**
   In this mode the HandyPort will wait until another device tries establish a connection. The remote address will not be evaluated.
3. **REG_CON:**
   The 'Register & Connect' mode is used with the BlueSoil software. Within this mode the HandyPort will automatically connect to the remote device and the Multi-Port will take the COM port, which was specified within the BlueSoil software and the HandyPort. Choose the COM port under 'COM Port'. This way you can control to which COM port the HandyPort will be connected.

- **RemAdr:** Specify the hardware address of the remote device. Use the '+' button to increase the hexadecimal number. Use the '→' to jump to the next number.
- **COM Port:** Specify to which COM port of the Multi-Serial port device the HandyPort should connect. Choose '1st' to connect to the first available COM port that was specified as 'auto-start' in the BlueSoleil software.

To confirm the displayed values press 'END' and wait until the green link (LNK) LED stops flashing and the red operation (OPR) LED switchs on. The HandyPort is configured now. Figure 1 shows the screenshots of the setup menu.

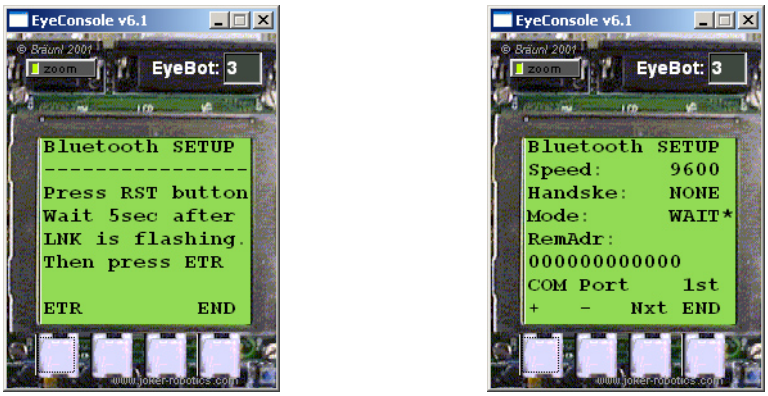


Figure 1: Bluetooth setup menu

3.2 Terminal Program

To get access to the whole configuration menu of the HandyPort use a terminal program such as ‘Hyperterm’ or ‘Minicom’. Connect the HandyPort HPU-120 to the serial port of your PC and apply power supply through a USB cable. The following instructions will show how to access the HandyPort menu by using Hyperterm.

1. Run Hypertrm

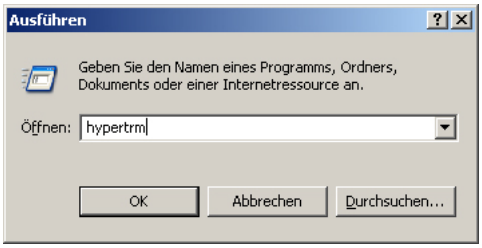


Figure 2: Run Hyperterm

2. Go to File ⇒ Properties Select the appropriate COM port under ‘Connect using’ and click ‘Configure’.

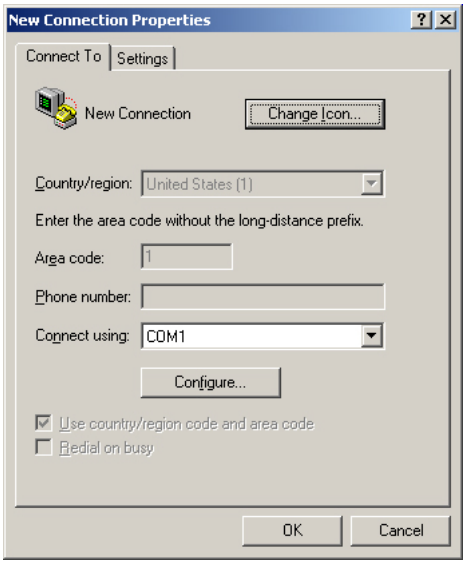


Figure 3: Properties

3. Change the settings to the values that are shown in Figure 4.

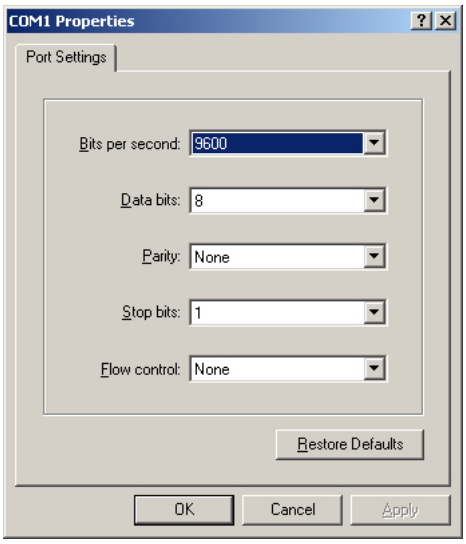


Figure 4: Port settings

4. Press the reset (RST) button on the HandyPort and wait 5 seconds after the green LED is flashing. Then press 'Enter' and the menu should appear.

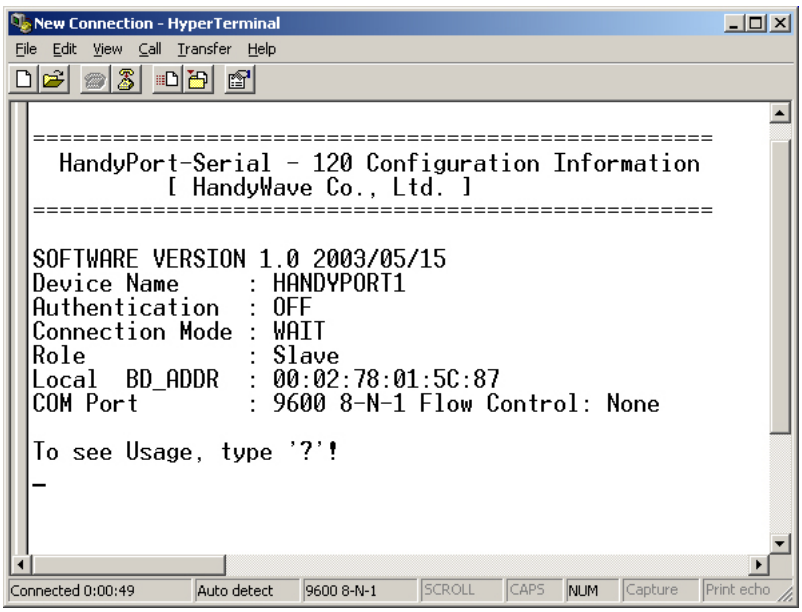


Figure 5: HanyPort Menu

Further details on the HandyPort and its configuration can be found in the HPS-120 manual.

3.3 Settings

Make sure that the speed and the flow control settings are always the same on the EyeBot and the HandyPort to prevent errors. The default values for the RADIO functions on the EyeBot are:

- Speed: 115200bps
- Handshake: NONE

4 BlueSoleil Software

Please refer to the MSPP manual to install the IVT BlueSoleil MSPP software.
After the installation, plug in the multi-serial port to an USB port and run the software. Figure 6 shows the graphical user interface.

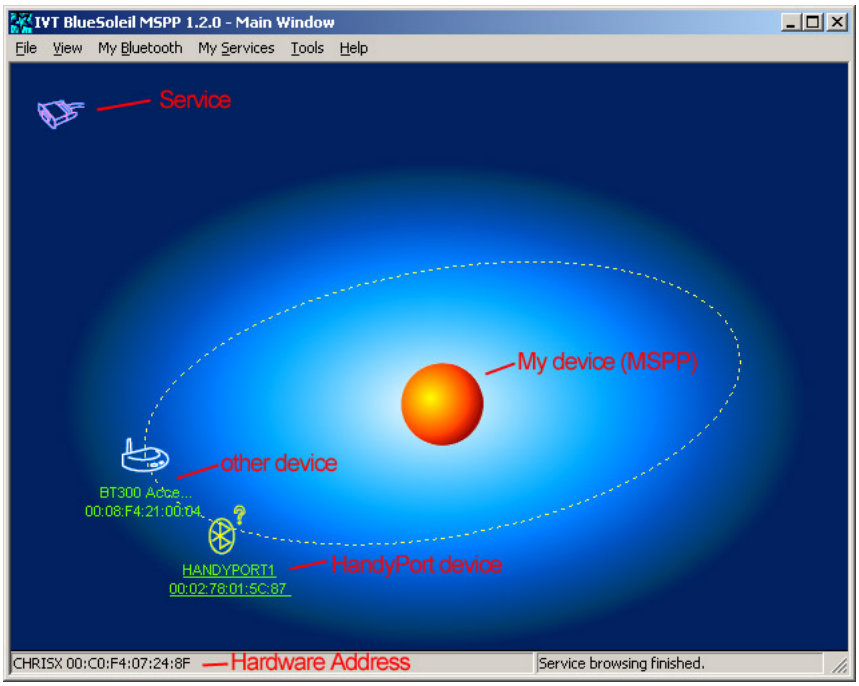


Figure 6: BlueSoleil GUI

- On startup the software searchs for available Bluetooth devices. It will show your available HandyPorts (HPS-120) and might show other devices in range.
- The sun represents the HPU-100 and its hardware address is shown in the lower left corner.
- The serial cable symbol represents the available service of the highlighted Bluetooth device. Double click on a device to update these services.
- Double click on the service symbol, i.e. the serial cable, to establish a connection.

- Depending on the mode settings of the HPS-120 and the MSPP software, the devices will connect automatically (see Chapter 2).
- Either a small pop-up window in the lower right corner of your screen or the properties window will tell you to which COM port a device is connected to your computer.
- On right click on one of the devices a menu will open with all relevant items.
- The MSPP software in combination with the HPU-100 is able to connect up to seven Bluetooth devices simultaneously.

5 RemoteBT

This section deals with the RemoteBT software. This software is the center part of the network and is responsible for the operation of the network. Figure 7 shows the graphical user interface of the program.

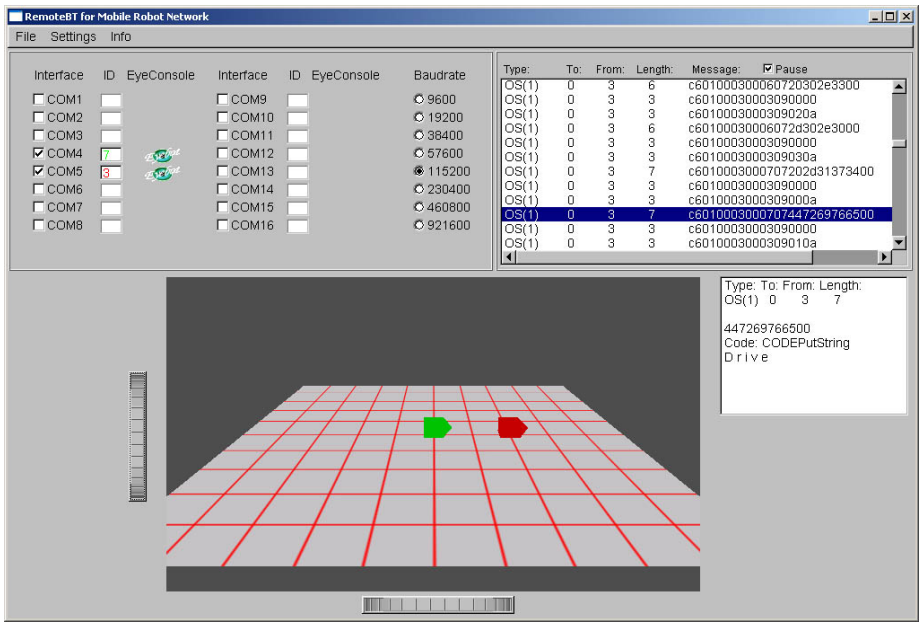


Figure 7: RemoteBT GUI

- After the HandyPorts are connected to the HPU-100, tick the right COM port boxes.
- You can now either run your program on the EyeBot, that uses the RADIO functions, or enable the remote function.
- The EyeBot will authenticate and its ID will appear beside the COM port tick box.
- A small EyeConsole icon will also appear in the same row. Use this icon to view the EyeConsole. The EyeConsole, shown in Figure 8, shows you the current display of the real EyeBot. Through the buttons of the EyeConsole you have full controll of your EyeBots.

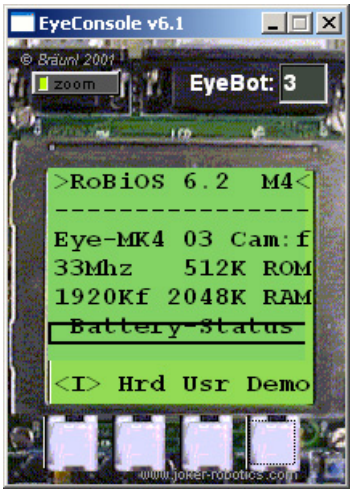


Figure 8: EyeConsole

- The traffic through the HPU-100 is displayed in the upper right corner of the GUI. Press the ‘Pause’ button to pause the display. Choosing one message will display its content in the small display below.
- Choose the ‘Logfile’ box under ‘Settings’ to create a file called ‘log-file.txt’. All incoming messages will be listed there (NOTE: every byte will be written to the file which might lead to a huge file)
- If the remote functions on the EyeBot is enabled, its driving positions will be displayed on the field. The initial point with the coordinates x=0; y=0 is the center of the grid. The initial direction points to the right (rad=0).

6 Troubleshooting

This section shows you how to handle problems and error messages.

Problems and Error Messages

Problem/Message	Reason	Solution
The devices are connected, but no transmission is done.	The HandyPort(HPS-120) and the serial port do not have the same settings.	Change settings of the HandyPort to the same of the EyeBot's serial port.
	The HandyPort and the chosen baud rate of the RemoteBT program do not match.	Change the baud rate either on the EyeBot+HandyPort or in the RemoteBT program.
	The chosen COM port in the RemoteBT and the COM port to which the HPS-120 is connected, do not match.	Get the COM port by a right click on the device in the BlueSoleil software and tick the appropriate COM tick box in the RemoteBT software.
	There might be more than one EyeBot with the same ID.	At no time is is allowed to have two or more EyeBots with an identical ID in the network. The robot's ID is a unique specifier.
The MSPP/HPU-100 and the HPS-120 cannot connect.	The HPS-120 is in configuration mode. (flashing green LED)	Press the RST button to leave configuration mode.
'Error while initializing COM'	The chosen COM port is not registered with the operation system.	Make sure that the COM port is available under the operation system.

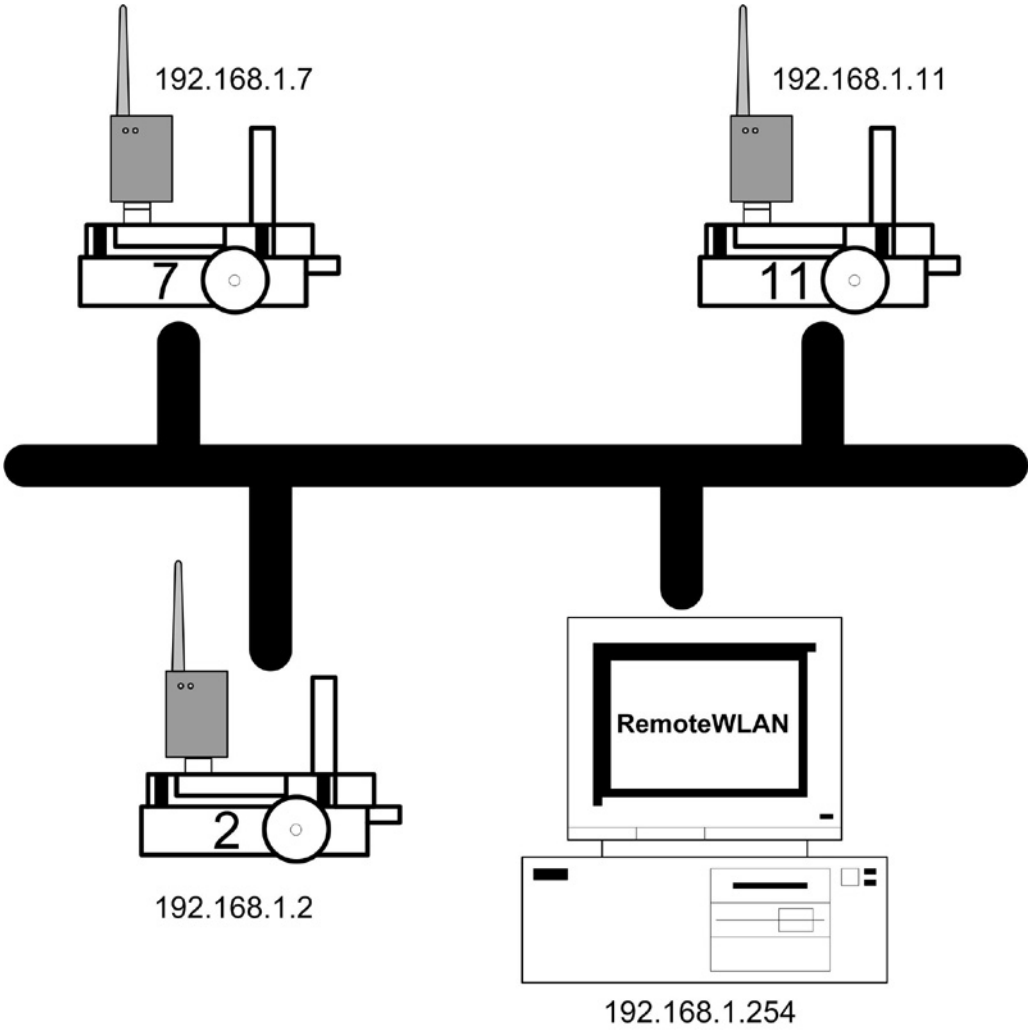
Appendix B

EyeNet WLAN Manual

The following pages are the technical documentation for the EyeNet with WLAN technology. The manual describes all needed configuration for the hardware and the procedures to run the network.

EyeNet

Wireless Network for Mobile Robots
using WLAN Technology



January 2005

1 Introduction

This manual introduces you to the wireless network solution for mobile robots using WLAN (IEEE 802.11b) technology. The EyeNet gives you the opportunity for communication between mobile robots as well as the remote controll of any robot connected to the network. The next pages will show how to setup, configure and run the small class C network and its devices.

Make sure you will have the following items:

- Serial wireless LAN adapter
  (Lantronix WiPort on circuit board)
- RemoteWLAN Software
- EyeBot with RoBios 6.2 (or higher)

2 Quick User Guide

This section gives you a quick step-by-step guide how to run the EyeNet.

1. Connect the wireless LAN adapter to the serial port of the EyeBot.
2. Switch on the EyeBot and configure the WiPort (cf. section 3) with the following values:
   - IP: 192.168.1.EyeBotID (i.e. EyeBot ID = 7 → IP = 192.168.1.7)
   - Netmask: 255.255.255.0
   - Speed: 115200bps
   - Handshake: NONE
   - I/F mode: 4C
   - Port no.: 10001
   - Connect mode: C4

   Make also sure that all the adapters that should form one network use the same SSID.
3. Wait a short moment until all adapters are connected to the network.
4. Run your EyeBot program that uses the RADIO functions.

Optional:

5. Plug in an IEEE 802.11b,g wireless network adapter into a PC.
6. Set the network adapter to:
   - IP: 192.168.1.254
   - Netmask: 255.255.255.0
   - SSID: The same as on the WiPorts.
7. Wait until the PC is connected to the network.
8. Run the RemoteWLAN program on a host PC.
9. Enable the remote functionality on the EyeBot.

10. Press the refresh button and wait for responding EyeBots.
11. The EyeBots and their IP address are displayed. Click on the robot symbol to show the EyeConsole.

3 Configuration

To configure the Lantronix WiPort module you can use a Terminal emulator like Hyperterm or Minicom, the Lantronix DeviceInstaller software, the WebManager, a Telnet connection, or you can use the configuration menu on the EyeBot.

3.1 EyeBot Config Menu

Connect the serial WLAN adapter directly to the serial port 1 of the EyeBot. Switch on the EyeBot and a red operation LED should light. Go to the configuration menu for the WiPort, which can be found under:
Demo ⇒ Network ⇒ WLS.
Take a peaked thing, i.e. a pen, and press the reset button on the WLAN adapter or switch it off and on again. Then press `ETR`. You will now see the setup main menu. It contains three submenu items. For each item there are two modes available. Press the `Inf` key to enter the appropriate information mode displaying the current settings. Press the `Set` key if you want to change the settings. The three available submenus are:

- **Server:**
  All relevant settings for the network device.
  - MAC:
    The unique MAC address of the network adapter.
  - IP:
    Displays the current IP address of the network adapter.
  - Netmask:
    Shows the used netmask. In the set-up mode it shows the number of host bits of the netmask.
- **Channel:**
  Submenu for the settings of the WiPort's first serial port.
  - Speed:
    The baud rate of the serial port of the WiPort can be changed here. Possible speeds are: 9600bps, 19200bps, 38400bps, 57600bps, 115200bps, 921600bps.

- I/F Mode:
  The bit combination distinguishes the parity and stop bit settings[1]. The two most common modes are implemented and available:

  4C = 8-bit, No Parity, 1 stop bit

  78 = 7-bit, Even Parity, 1 stop bit

- Flow:
  Sets the flow control settings.

  0 = No flow control

  1 = XON/XOFF flow control

  2 = Hardware handshake with RTS/CTS lines

  5 = XON/XOFF pass characters to host

- PortNo:
  The Port No setting represents the source port number in TCP connections. The port for EyeNet is **10001**.

- ConnectMode:
  Distinguishes between TCP and UDP connections.

  CC = Allways accept incoming connections. Directed UDP mode.

  C4 = Allways accept incoming connections. Manual connection. TCP mode.

  C0 = Allways accept incoming connections. Remote connection. TCP mode.

  For EyeNet this value has to be **C4**.

- DatagramType:
  Only with UDP. Sets UDP datagramm to directed (1) or non-directed (0).

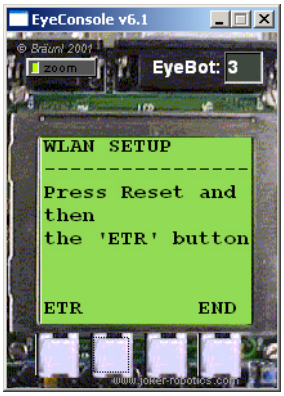
- **WLAN:**
  Displays information on the wireless LAN settings.

  - WLAN:
    Shows whether WLAN is enabled or disabled.

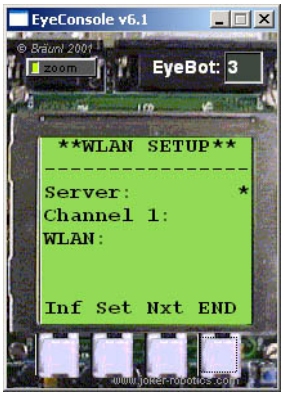
[1] see also WiPort manual

- SSID:
  Displays the SSID. The SSID is arbitrary but must be the same with all the network adapters that want to form a network.
- WEP:
  Security information

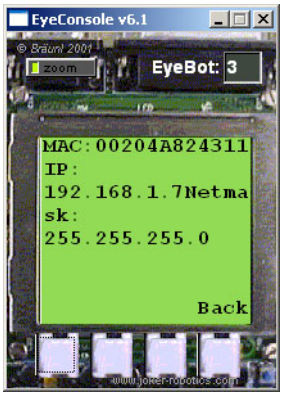
The 4 keys on the EyeBot can be used to navigate through the menus. To exit the configuration menu press `END`. Then choose whether you want to save the current settings (`YES`) or not (`NO`).
Figures 1(a) up to Figure 1(i) show screenshots of the setup menu.



(a) Enter menu

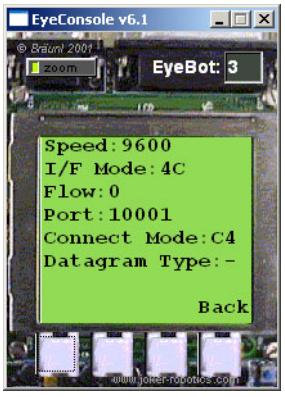


(b) Main menu

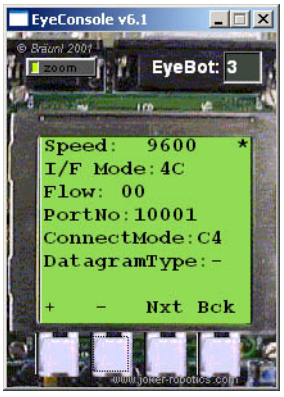


(c) Server Info menu

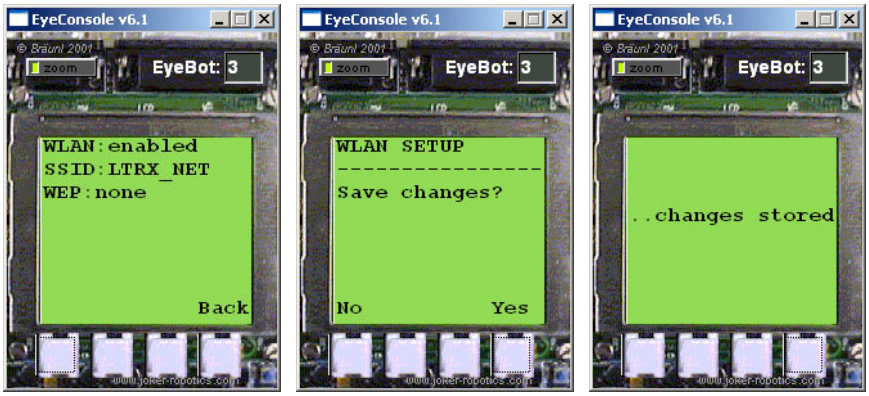
(d) Server Setup menu



(e) Channel 1 Info menu



(f) Channel 1 Setup menu



(g) WLAN Info menu (h) Exit menu (i) Save verification

Figure 1: WiPort Config Menu for EyeBot

3.2 Terminal Program

To get access to the whole configuration menu of the WiPort use a terminal program such as ‘Hyperterm’ or ‘Minicom’. Connect the WiPort with a Null-Modem cable to the serial port of your PC and apply power supply through a USB cable. The following instructions will show how to access the configuration menu by using Hyperterm.

1. Run Hypertrm

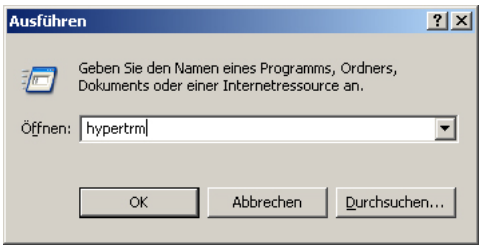


Figure 2: Run Hyperterm

2. Go to File ⇒ Properties Select the appropriate COM port under ‘Connect using’ and click ‘Configure’.

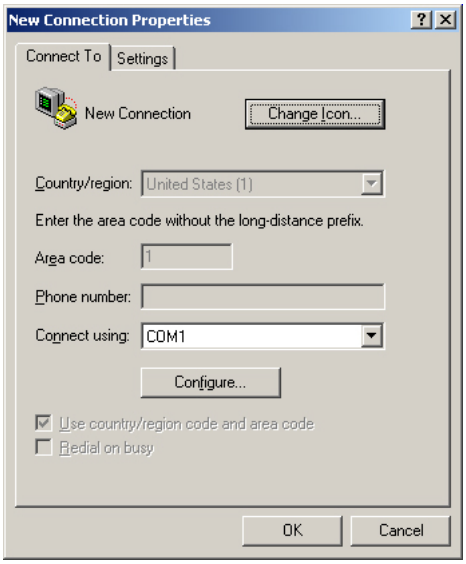


Figure 3: Properties

3. Change the settings to the values that are shown in Figure 4.

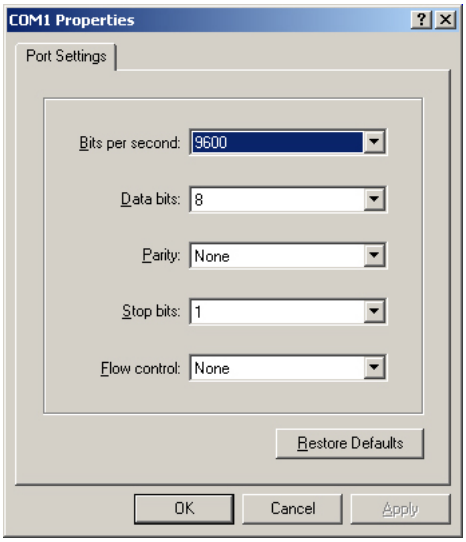


Figure 4: Port settings

4. Press the reset button on the WiPort or switch it off and on. Then hold down the ‘x’ key until the WiPort responds by transmitting its MAC address. Then press ‘Enter’ and the menu should appear.

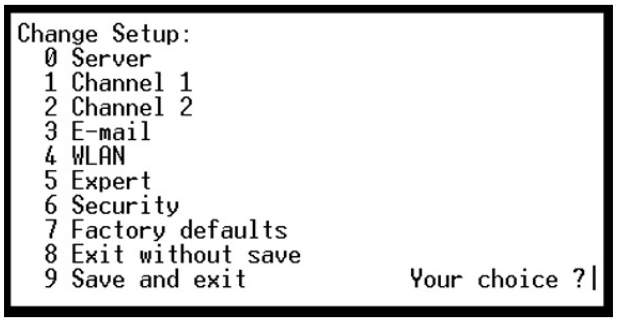


Figure 5: WiPort Menu

Further details on the WiPort and its configuration can be found in the WiPort manual.

3.3 Settings

Make sure that the speed and the flow control settings are always the same on the EyeBot and the WiPort to prevent errors. The default values for the RADIO functions on the EyeBot are:

- Speed: 115200bps
- Handshake: NONE

4 RemoteWLAN

This section deals with the RemoteWLAN software. This software is an optional part of the network and can be used to monitor and remote control robots. Figure 6 shows the graphical user interface of the program.

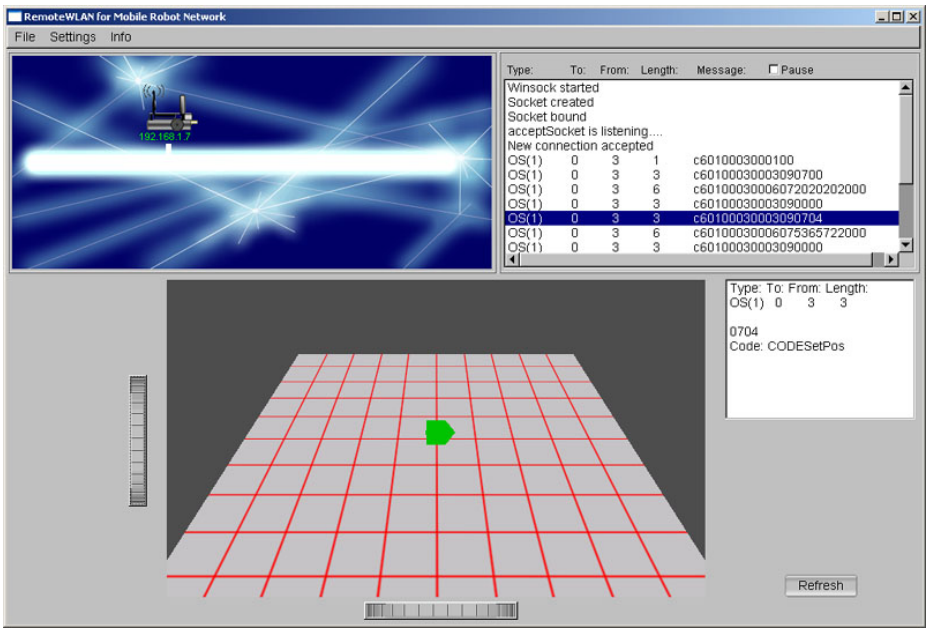


Figure 6: RemoteWLAN GUI

- After the PC is connected to the network press the ‘Refresh’ button.
- Each responding EyeBot and its according IP address will be displayed.
- Now enable the remote function on the EyeBot.
- Use the robot symbol to view the EyeConsole. The EyeConsole, shown in Figure 7, shows you the current display of the real EyeBot. Through the buttons of the EyeConsole you have full controll of your EyeBots.
- The messages sent between the remote control program and an EyeBot appear in the display on the upper right corner of the GUI. Press

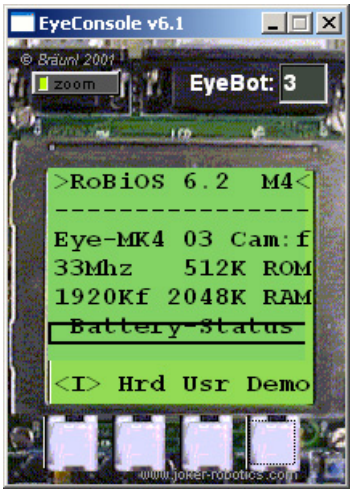


Figure 7: EyeConsole

the ‘Pause’ button to pause the display. Choosing one message will display its content in the small display below.

- Choose the ‘Logfile’ box under ‘Settings’ to create a file called ‘logfile.txt’. All incoming messages will be listed there (NOTE: every byte will be written to the file which might lead to a huge file)
- To increase the number of respond replies for the EyeBots go to ‘Settings’ and increase the number of retries.
- If the remote functions on the EyeBot is enabled, its driving positions will be displayed on the field. The initial point with the coordinates x=0; y=0 is the center of the grid. The initial direction points to the right (rad=0).

5 Troubleshooting

This section shows you how to handle problems and error messages.

Problems and Error Messages

Problem/Message	Reason	Solution
The devices are connected, but no transmission is done.	The WiPort and the EyeBot's serial port do not have the same settings.	Change settings of the WiPort to the same of the EyeBot's serial port.
	The device is not configured to use port 10001.	Change the settings to PortNo: 10001.
	The IP of a WLAN adapter and the ID of a robot do not match.	Configure the last octet of the IP address to the corresponding EyeBot's ID.
A network adapter cannot connect to the network.	The WiPort is in configuration mode.	Switch the WiPort off and on to leave configuration mode.
	The network adapter is not configured to the same SSID.	Change the SSID according to the network SSID.
The RemoteWLAN program does not work even if the PC is connected.	There might be collisions with a firewall running on the PC.	Configure the firewall to allow the use of port 10001.
	The network card of the PC is not configured to the IP 192.168.1.254	Change its IP address to this value.

Appendix C

CD contents

This thesis includes a CD-ROM, which can be found on the last page. It contains the whole source code, a part of the Bluetooth as well as the IEEE 802.11 specifications, papers, hardware manuals, EyeNet manuals, grafics and the thesis in an electronic format itself.

Source

The directory *Source* contains the directories *RemoteBT*, *RemoteWLAN*, *TestApplications*, *Sys-Demo* and *RoBIOS6.2*. There are VC++ 6.0 project files as well as a Makefiles available for the RemoteBT and RemoteWLAN software. The directory *TestApplications* contains the applications used for testing the new RADIO functions. The configuration menus for the used network devices (BT and WLAN) are included in the demo program for the EyeBot. The source code for the new version can be found in the *Sys-Demo* directory. The new RADIO functions and further related changes to the RoBIOS are included in the *RoBIOS6.2* directory.

Bluetooth Specifications

In this directory the Bluetooth Core, the GAP and SPP specifications are kept. Furthermore the IEEE 802.15 specifications for WPAN can be found here.

IEEE 802.11 - WLAN

As the name suggests, this directory includes the IEEE 802.11 standard and some of their extensions.

Papers

This directory stores nearly all the papers which were used by the author and are written down in the references. They cover the topics Bluetooth, WLAN, Robot networking, and networking.

Hardware Manuals

The manuals for the Bluetooth devices HPU-100 and HPS-120 as well as the manual for the WLAN adapter WiPort can be found here.

EyeNet Manuals

This directory has two subdirectories *EyeNetBT* and *EyeNetWLAN*, each containing the manual's .tex and .pdf files.

Graphics

All graphics used in the thesis are included in the *graphics* folder.

Report

The .tex files and the whole report are located here.

References

[1] IPv6 - Information page.
available at: http://www.ipv6.org.

[2] RoboCup - Webpage.
available at: http//:www.robocup.org.

[3] AU-System. *Bluetooth Whitepaper*, 1.1 edition, 2000.
available at: http://www.ausystem.com.

[4] Frank Außerlechner. *Skriptum zur Vorlesung Netzwerkpraktikum*.

[5] Bluetooth SIG. *Generic Access Profile*.
available at: http://www.bluetooth.org.

[6] Bluetooth SIG. *Specification of the Bluetooth System*, 2003.
available at: http://www.bluetooth.org.

[7] T. Bräunl and Peter Wilke. Flexible Wireless Communication Network for Mobile Robot Agents . *Industrial Robot*, 28:220–232, 2001.

[8] Thomas Bräunl. *Embedded Robotics*. Springer, 2003.

[9] Douglas E. Comer. *Computer Networks and Internet*. Prentice Hall, 3 edition, 2001.

[10] HandyWave Co., Ltd. *HandyPort Usage - Auto-connecting between HPSs and MSPP - Application Note*.
available at: http://www.handywave.com.

[11] HandyWave Co., Ltd. *HandyPort-USB Multi Serial Port Profile (MSPP) Wireless Serial Multi Port User's Manual*, 2.0 edition.
available at: http://www.handywave.com.

[12] HandyWave Co., Ltd. *HandyPort/HandyCore - Extended Command Set - User's Manual*.
available at: http://www.handywave.com.

[13] HandyWave Co., Ltd. *HPS-120 HandyPort-Serial User's Manual*, 2.0 edition.
available at: http://www.handywave.com.

[14] Wolfgang Hascher. Bluetooth - Der Kabel-Killer. *tecCHANNEL*, 2000.
available at: http://www.tecchannel.com.

[15] IEEE. *802 IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture*, IEEE Std 802-2001 edition, 2002.
available at: http://www.ieee.org.

[16] IEEE. *Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)*, IEEE Std 802.15.1-2002) edition, 2002.
available at: http://www.ieee.org.

[17] IEEE. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11, 1999 (R2003) edition, 2003.
available at: http://www.ieee.org.

[18] IEEE. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band*, IEEE Std 802.11g-2003) edition, 2003.
available at: http://www.ieee.org.

[19] IEEE. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-speed Physical Layer in the 5 GHz Band*, IEEE Std 802.11a-1999(R2003) edition, 2003.
available at: http://www.ieee.org.

[20] IEEE. *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band*, IEEE Std 802.11b-1999(R2003) edition, 2003.
available at: http://www.ieee.org.

[21] Lantronix Co. *DeviceInstaller User Guide*, 2004.
available at: http://www.lantronix.com.

[22] Lantronix Co. *WiPort Development Kit Quick Start Guide*, 2004.
available at: http://www.lantronix.com.

[23] Lantronix Co. *WiPort Integration Guide*, 2004.
available at: http://www.lantronix.com.

[24] Lantronix Co. *WiPort User Guide*, 2004.
available at: http://www.lantronix.com.

[25] Tim Lüth and Thomas Längle. Multi-Agenten-Systeme in der Robotik und Artificial-Life. Technical report, University Karlsruhe, Institut für Prozeßrechentechnik und Robotik, 1995.

[26] G. Arnold D. Treadwell H. Sanders M. Hall, M. Towfiq. Windows Sockets - An Open Interface for Network Programming under Microsoft Windows.

[27] David Morton. Understanding IPv6. *PC Network Advisor*, 83:17 – 22, 1997.

[28] Lewis Napper. *WinSock 2.0*. Hungry Minds Inc., 1997.

[29] M. Fujita R. Rojas M. Veloso P. Lima, T. Balch and H. Yanco. RoboCup 2001 . *IEEE Robotics & Automation Magazine*, pages 20 – 30, June 2002.
available at: http://www.ieeeexplore.ieee.org.

[30] NeHe Productions. OpenGL Tutorial. http://nehe.gamedev.net.

[31] Radiometrix Ltd. *BiM, Low Power UHF Data Transceiver Module*, 2004.
available at: http://www.radiometrix.co.uk.

[32] Microsoft Support. Implementing Internet Pings Using Icmp.dll, 2000.
http://support.microsoft.com/kb/q170591.

[33] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, 4 edition, 2003.

[34] Heinz Unkelbach. *Skriptum zur Vorlesung Softwaremethoden.*

[35] Monika Weiner. Die Agenten kommen. *Frauenhofer Magazin*, 3:30 – 31, 2001.
available at: http://www.frauenhofer.de.

[36] C.R. Weisbin and G. Rodriguez. NASA Robotics Research for Planetary Surface Exploration . *IEEE Robotics & Automation Magazine*, pages 25 – 34, December 2000.
available at: http://www.ieeeexplore.ieee.org.

[37] Gerhard Willms. *C++ Grundlagen Buch*. Data Becker, 2 edition, 2001.

[38] Alex S. Fukunaga Y. Uny Cao and Andrew B. Kahng. Cooperative Mobile Robotics: Antecedents and Directions . *Autonomous Robots*.