# A Digital Interface for Imagery and Control of a Navico/Lowrance Broadband Radar

**Final preprint version - The original publication is available at www.springerlink.com**

Adrian Dabrowski, Sebastian Busch and Roland Stelzer

**Abstract** The paper describes a method to establish compatibility between an autonomous surface vessel control system and a Navico Broadband Radar BR24. The solution obtains radar imagery and control of the antenna unit over its standard Ethernet interface, making the proprietary controller unit optional. It presents devices, software and methods used for empirical protocol analysis and documents the findings. Protocol details for the following functions have been identified: Operation, zoom level, various filter settings, scan speed and keep alive. An open source implementation with basic operational functionality has been made available. It features a live network mode and a replay mode using captured network traffic. In live mode, controlling radar operation as well as zoom level is possible. In both modes the radar imagery stream is rendered and displayed.

## 1 Introduction

Radar is an important sensor in nautical navigation, as it is also for autonomous robots. Typically a radar systems vendor delivers an integrated solution: A radar transceiver together with a display unit. Radar provides

Adrian Dabrowski
Faculty member of Höhere Technische Bundeslehr- und Versuchsanstalt Wien V - HTL Spengergasse, e-mail: dabrowski@spengergasse.at and Austrian Society for Innovative Computer Sciences, e-mail: adrian@innoc.at

Sebastian Busch
Austrian Society for Innovative Computer Sciences,
e-mail: mail@sebastianbusch.at

Roland Stelzer
Head of Austrian Society for Innovative Computer Sciences,
e-mail: roland@innoc.at

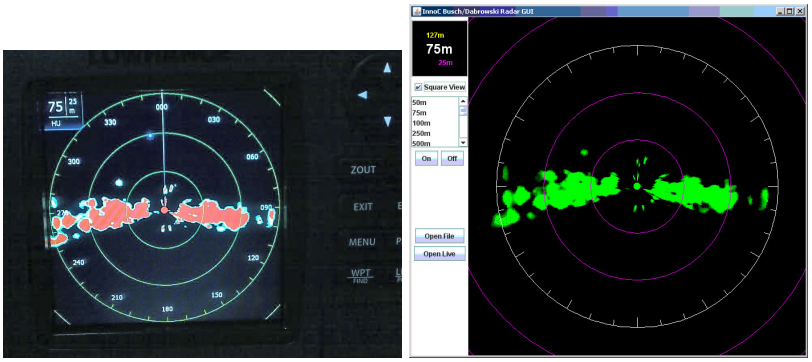


**Fig. 1** A photograph of the display unit and a screenshot of the JAVA program showing the same data.

perception of the surroundings with a birds-eye view, accurate range measurement and navigability under poor visual conditions.

## 2 Motivation

The ASV Roboat[1] needs a radar solution which provides the control algorithms with dynamic information about the surrounding real world in order to implement obstacle avoidance and possibly to handle other navigational and mapping tasks. The solution should preferably deliver its data directly in digital form.

The developers of project ROAZ II[1] as well as INNOC[2] independently contacted several vendors of small radar systems but had no success in getting interface specifications, even when offered to sign an NDA[3]. Some vendors have their own PC/Windows software on sale, others presumably want to keep that option for the future.

Russel Technologies Inc. (former Xenex Innovations Ltd) sells a commercial PC/Windows based solution [2] including an SDK for Furuno Radars. The System consists of an embedded PC which digitizes the proprietary Furuno video signal, as well as a USB-to-RS422 bridge to control the antenna unit. This system is used by the SSC San Diego USV team [3].

Within the project ROAZ II [4] a Furuno NavNet Vx2 radar based solution has been evaluated. As a result, the authors were unsatisfied that among other things the radar needs gain adjustments, which they could only do manually.

[1] LSA - Laboratório de Sistemas Autónomos, Instituto Superior de Engenharia do Porto

[2] Austrian Society for Innovative Computer Sciences

[3] Non Disclosure Agreement

For the ASV Roboat the team decided for a Navico/Lowrance BR24 Broadband Radar [5]. This radar uses *Frequency Modulated Continuous Wave* (FMCW) radar technology. The decision was based on the low power consumption compared to conventional pulse radars, and the fact that Ethernet is used for communication in this system.

Pulse radars are typically rated in the kilowatt output power range, whereas this one has a peak power output of 100 mW nominal and a total power usage of about 20 W (at 12 V) in operation. Pulse radars measure the time of flight of a reflected electromagnetic signal. In contrast, FMCW radars send out signals of linearly increasing frequency and compare the frequency of an incoming reflection to the currently sent frequency. Knowing the rate of frequency increase allows the calculation of distance. Despite its low power consumption, FMCW radars can be turned on and off almost instantaneously without a significant warm up time - needing less than two seconds for spinning up. This feature will further help to reduce power consumption on long-term missions.

Already having an Ethernet network in place on the ASV Roboat leverages the integration of the radar into the control system. It proofed of much help in the empirical protocol analysis, that the lower transmission layers comply to an industry standard.

## 3 Structure of a LowRance Installation

The Lowrance installation (Figure 2) allows multiple sensors and consumers to be connected. For example additional sensors can be transducers or NMEA heading sensors. A *Radar Interface Box* is used to connect a power supply for the antenna unit and has a standard Ethernet port on the controller/display side.

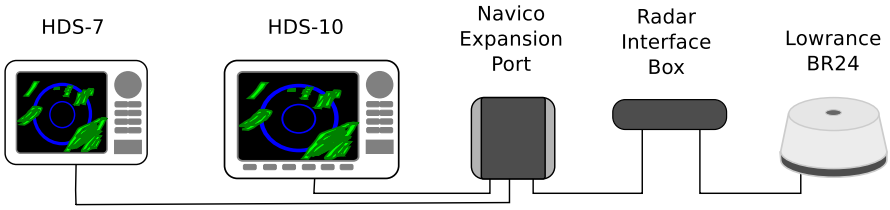


**Fig. 2** A typical installation as provided by the manufacturer [5]. The optional *Navico Expansion Port* is a common Ethernet switch.

## 4 Methods for Empirical Protocol Analysis

All described methods and research had been done solely to establish compatibility between our autonomous surface vessel control system and the radar system mentioned above. Neither the data nor the protocol incorporated any recognizable protection scheme.

To be able to record the data exchanged between antenna unit and controller unit we decided to use an Ethernet hub (a switch with a mirror port would work as well) which we placed in between *Radar Interface Box* and the *Display and Control Unit*. We connected a notebook to the hub and used Wireshark[4] to record and analyse the network traffic[5].

The same setup has been used to test our implementation of the display and control functions as we could simultaneously use the proprietary controller and display and our application to compare the results.

To test which data streams are essential and which not, a computer with two Ethernet interfaces and a software bridging solution[6] has been used. It allowed us to filter out specific traffic based on layer 3 and 4 attributes without breaking the broadcast domain.

## 5 Network Communication

During boot-up the radar antenna transceiver and the display unit automatically configure their IP addresses. If no DHCP server is available, both devices select an IPv4 *Zeroconf* link local address as described in RFC3927[6]. All communication is done using UDP multicast datagrams. All units emit *Internet Group Management Protocol* (IGMPv3[7]) subscription announcements. This approach has two advantages. (1) It enables the system to handle multiple subscribers (i.e. controller/display units) and (2) The IP addresses of the devices are irrelevant, since they solely communicate through fixed multicast groups.

## 6 Control and Data Streams

Figure 3 shows the 11 data streams we identified on both devices. A *source arrow* indicates a device that actually sends data to that multicast group. A

[4] Wireshark is a network protocol analyser. Project website `http://www.wireshark.org/`

[5] As it turned out any IGMP unaware switch will also work. These switches will handle multicast traffic like broadcasts and therefore retransmit it on all active ports.

[6] The bridge module (`http://sourceforge.net/projects/bridge/`) is available in the official linux kernel distribution

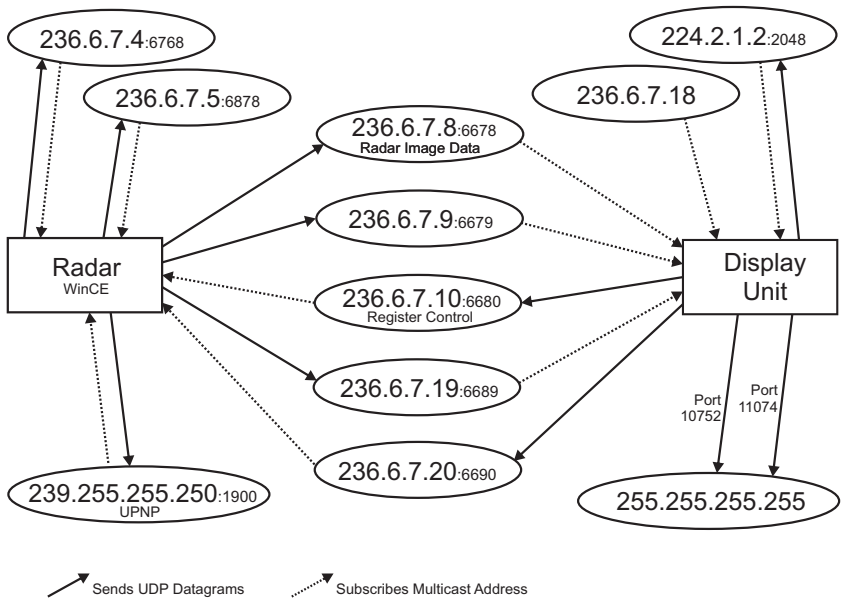


**Fig. 3** Reconstructed multicast streams of a typical operation session. Port numbers are derived from actual sent data, because IGMP multicast subscriptions do not carry port information.

*sink arrow* indicates a device that uses IGMP to subscribe to that multicast group. Data streams that have no sink are probably used to communicate with extensions we did not install. Streams where the same device class is source as well as sink, are presumably for internal coordination between multiple devices of the same class(e.g. multiple display units).

From the five multicast streams that connects the radar and the display unit in one or the other direction, we identified two operation critical ones and tried to name them: The *Image Data Stream* and the *Control Register Access*.

## *6.1 Radar Image Data Stream*

This stream provides radar scanline data as soon as the radar is turned on by setting the appropriate control registers 0x01 and 0x02. One UDP frame containing 32 scanlines is 17160 bytes long, which get fragmented by Ethernet into 1500 byte pieces (default Ethernet MTU). For payload data structure see table 1. Multi byte fields are transmitted in little endian.

**Table 1** Image data packet data structure

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
|---|---|---|---|---|---|---|---|---|
| 01 | 00 | 00 | 00 | 00 | 20 ns | 00 ss | 02 | Frame Header |
| 18 l | 02 st | 73 rc | 0B | 00 | 44 | 0D | 0E | Scanline Header |
| 1D a | 07 | 34 | 92 | 0C scale | 00 | 00 | 01 | |
| 00 | 44 | 59 u1 | 13 | 00 | 40 | 00 | CB u2 | |
| 512 polar pixels in 8 bit grayscale ⋮ | | | | | | | | Scanline |

× 32

| | |
|---|---|
| ns | Number of scan lines, always 32d (untested) |
| ss | Size of scan line, always 512d (untested) |
| l | length of scan line header, always 24d (untested) |
| st | unknown, probably status: 02 valid data; 18 spin up |
| rc | raw scan line counter 0..4095 |
| a | absolute angle of scan line 0..4095 = 0..360$^o$ |
| scale | scan radius in meters = $scale * 10/\sqrt{2}$ |
| u$x$ | variable, but unknown |

The resolution of a full 360$^o$ scan (see figure 4) is 512 pixels by approx. 2048 scanlines as field $a$ is incremented by two. The radar sometimes skips single or multiple scanlines presumable when the automatic interference rejection discards them or the unit needs to re-sync mechanically.

Decoding the imagery is stateless, as all necessary information is included in the scan line frame. See Alg. 1 on how the variables are used.

### 6.1.1 Wireless Network Transmission Issues

Although the data stream uses just around 1 MBit of net bandwidth, it is not suitable for standard WiFi (e.g. IEEE 802.11 b/g) because of the

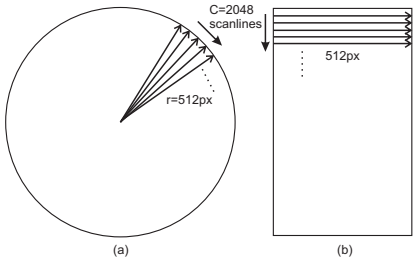


**Fig. 4** Radar imagery in Cartesian form (a) and in its original polar form (b).

**Algorithm 1** Map scanline to Cartesian pixels ($a$, $polarpixels[512]$)

**Require:** $Output.width = Output.height$
  $cosa \leftarrow cos(a \cdot 360/4096)$
  $sina \leftarrow sin(a \cdot 360/4096)$
  $oscale \leftarrow Output.width/1024$
  $omid \leftarrow Output.width/2$
  **for** $r \leftarrow 0..511$ **do**
    $Output.pixels2D[omid + cosa \cdot r \cdot oscale, omid + sina \cdot r \cdot oscale] \leftarrow polarpixels[r]$
  **end for**

way it is send out. Access-points negotiate a separate data rate for each client depending on radio conditions. Broadcasts and multicasts however are transmitted using the lowest possible speed, so that all station in range are able to receive it. That speed is usually set to 1 or 2 MBit (gross speed) by the manufacturer and can be overwritten only in some products. The achievable net speed under good conditions is often approximated by dividing the gross speed by half. In either case the image data will use almost all airtime or even more. Lost data is inevitable under such conditions. Raising the multicast speed (where possible) on the other hand will prevent clients from associating with that access-point in greater distance or bad radio conditions.

As a consequence for the ASV Roboat, we either need an access-point that filters multicast or we need to split up the network into an internal and an external one. The latter to be accessible via WiFi for monitoring purposes.

## *6.2 Register Control*

Multicast address 236.6.7.10 port 6680 is used to set and read control registers in the radar. Return messages seem to be sent via 236.6.7.9 port 6679 but as they are not needed for the basic operation, we have not investigated the return channel so far. An example can be found in table 2.

Some operations require to set multiple registers, in which case multiple UDP-datagrams are sent.

**Table 2** Example of packet payload data for setting a register

| 0 | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| 03 | C1 | 10 | 27 | 00 | 00 | Set range to 1 km |
| reg | cmd | data | | | | |

| | |
|---|---|
| reg | Register number. It describes the function to access. |
| cmd | Always 0xC1 for *write* commands. |
| data | variable length payload to be written, at least one byte. Multi-byte fields are encoded *little endian*. |

The names of the registers below are arbitrary. We tried to match them with their apparent function.

### 6.2.1 Register 01/02: Radar Operation

Payload length: 1 byte

To turn the radar on, register 0x00 and 0x01 need to be set to 1. This starts the *image data stream*. Likewise to turn the radar off, both registers have to be set to 0.

Sometimes the first scan lines after a power on may contain wrong angle information. This resolves within one full turn of the transmitter.

### 6.2.2 Register 03: Zoom Level

Payload length: 4 bytes (1 dword)

The register 0x03 controls the range of radar operation. The scan radius is encoded as dword in decimetres. The format suggests that any value within a valid range can be set. We suggest to stick to the predefined ranges (see table 3) as used by the manufacturers display unit, as we can not confirm if other values produce accurate data.

A change is reflected in the *scan line header* in the radar image data stream in typically under a second.

### 6.2.3 Register 06: Filters and Preprocessing

Payload length: 1 byte selector + 8 byte data = 9 bytes

Register 0x06 offers access to a wide range of filters and preprocessing functions, such as *Sea Clutter Compensation*, *Rain Clutter* and *Gain control*. The first byte of the payload acts as selector, a comprehensive description can be found in table 4.

**Table 3** Register values for zoom ranges

| Range | Hex Data |
|---|---|
| 50 m | 0xf4,0x01,0,0 |
| 75 m | 0xee,0x02,0,0 |
| 100 m | 0xee,0x03,0,0 |
| 250 m | 0xc4,0x09,0,0 |
| 500 m | 0x88,0x13,0,0 |
| 750 m | 0x4c,0x1d,0,0 |
| 1 km | 0x10,0x27,0,0 |
| 1.5 km | 0x98,0x3a,0,0 |

| Range | Hex Data |
|---|---|
| 2 km | 0x20,0x4e,0,0 |
| 3 km | 0x30,0x75,0,0 |
| 4 km | 0x40,0x9c,0,0 |
| 6 km | 0x60,0xea,0,0 |
| 8 km | 0x80,0x38,1,0 |
| 12 km | 0xc0,0xd4,1,0 |
| 16 km | 0x00,0x71,2,0 |
| 24 km | 0x80,0xa9,3,0 |

**Table 4** Register values for filters and preprocessing

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Automatic Gain | 06 | C1 | 00 | 00 | 00 | 00 | 01 | 00 | 00 | 00 | A1 |
| | reg | cmd | sel | | | | | | | | |
| Manual Gain | 06 | C1 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | x1 |
| | reg | cmd | sel | | | | | | | | val |
| Rain Clutter Filter | 06 | C1 | 04 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | x2 |
| | reg | cmd | sel | | | | | | | | val |
| Sea Clutter Harbour | 06 | C1 | 02 | 00 | 00 | 00 | 01 | 00 | 00 | 00 | D3 |
| Automatic | reg | cmd | sel | | | | | | | | |
| Sea Clutter Manual | 06 | C1 | 02 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | x3 |
| | reg | cmd | sel | | | | | | | | val |

Note: There is no *automatic rain clutter* adjustment: $x2$ is in the range from 0x01 to 0x50, with 0x4D being the default value.

### 6.2.4 Register 08: Interference Rejection

Payload length: 1 byte

This option at register 0x08 accepts four values, where 0=*off*, 1=*low*, 2=*medium* and 3=*high*.

### 6.2.5 Register 0A: Target Boost

Payload length: 1 byte

The *Target Boost* setting at register 0x0A accepts three values, where 0=*off*, 1=*low* and 2=*high*.

### 6.2.6 Register 0E: Local Interference Filter

Payload length: 1 byte

This option has four values, where 0=*off*, 1=*low*, 2=*medium* and 3=*high*.

### 6.2.7 Register 0F: Scan Speed

Payload length: 1 byte

Setting 1 increases the scanning speed, while 0 resets to normal speed.

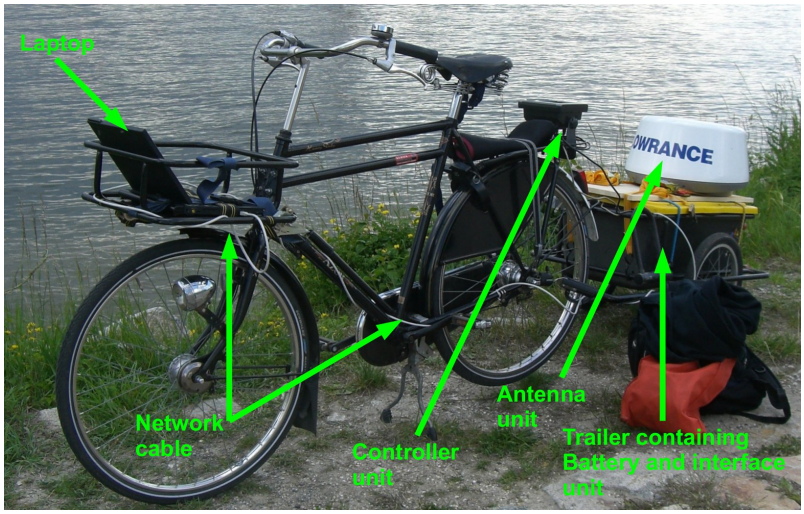


**Fig. 5** The sample data acquisition setup

#### 6.2.8 Register A0: Keep Alive

Payload length: 1 byte

The radar turns off automatically after 20-60 seconds when it loses contact to the display unit. To prevent this a keep alive timer has to be reset regularly. Register 0xA0 has to be set to value of 2 every approx. 10 seconds.

## 7 Sample Data Acquisition Setup

By May 2011, the mounting brackets for the ASV Roboat are still in construction. To acquire real-world sample data in a dry test, we chose to mount a test setup of the system on a bike trailer. The setup (see figure 5) consists of radar antenna, radar interface box, a 12 V lead battery, radar controller unit and a laptop. For ease of use the laptop has been mounted on the front carrier of the bike and connected to the other components on the trailer. On the laptop is run: (1) the authors' JAVA implementation to control the radar and check the output and (2) Wireshark to capture network traffic for later analysis. To acquire positions, a smart phone running a GPS tracking software has been mounted alongside the radar antenna.

This setup allowed the authors to record sample data for dynamic values of speed and direction. For further analysis radar data will be mapped to the recorded GPS data and fed into the mapping module of the Roboat software to be used as sample data for steering and obstacle avoidance algorithms. In

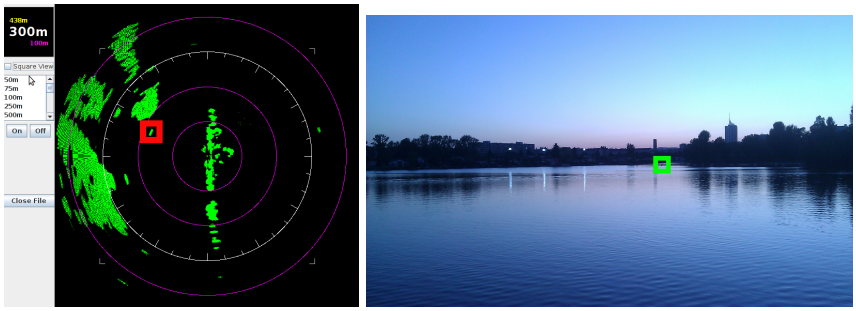


**Fig. 6** Radar data/photo comparison "Alte Donau" with canoeist (radar port side)

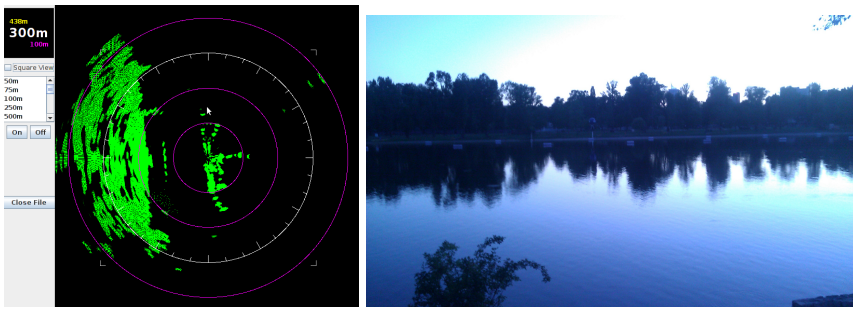


**Fig. 7** Radar data/photo comparison "Alte Donau" with buoys (radar port side)

figures 6, 7 comparisons of photo and radar data give an idea of how well the setup might detect objects in fine weather conditions.

## 8 Power Consumption

The electronics in the radar antenna unit has no power switch. It gets supplied and boots up as soon as there is power at the radar interface box, but does not turn on the transmitter. In this *standby* mode the unit consumes 0.15 A at 12 V. In full operation the unit consumes about 1.4 A. The display unit has a power switch and consumes about 0.7 A when turned on.

Values may vary depending on mode of operation or settings.

## 9 Sample Implementation

Simultaneously with the paper an open source sample implementation in JAVA is released and available at `http://www.roboat.at/technologie/radar/` It provides the following features:

| | |
|---|---|
| Source selection | operate on live radar network connection or on a pre-recorded PCAP-File[7] |
| Activation | turn the transceiver on and off |
| Range selection | set the zoom level which is also reflected in acquisition parameters |
| Image decode | image data is decoded and displayed in Cartesian form. |
| Filter settings | may be adjusted over the network. |

## 10 Data Usage & Future Work

Test data suggests that a strong preprocessing is necessary as radar artefacts, interference and reflections can disturb the imagery.

As discussed, image data from the radar is presented in polar pixels relative to the vessel. This makes it easy use it in a vessel steering algorithms which use the same coordinate system.

The reactive approach by Sauze and Neal [8] can be applied directly on the data, as each radar scan line can be used instead of a *raycast*.

For the radar to be used in the sweep line algorithm by Stelzer et.al. [9] radar data first has to be converted into objects representing the outline of identified obstacles (i.e. polygons in a Cartesian-like map). Together with the ship's sailing wind capabilities, transponder obstacles (e.g. AIS) and other map data this is used to generate a polar diagram of preferable directions to go [9, Fig.7]. In that diagram every obstacle creates a dent, as it makes that direction less attractive. It is the base for decisions of the *Short Course Routing* layer.

To facilitate this, the mapping module is currently extended to accept and deliver data in different formats and coordinate systems transparently hiding the conversion process. Formats are Cartesian coordinates relative to map origin and polar coordinates relative to the vessels true heading and position. That way the head-up radar data is transformed north-up. As position and heading information is only available at much larger intervals than the radar scan lines, position and heading will have to be extrapolated based on current course over ground and turn rate.

[7] The native packet capture file format of libpcap and WinPcap, which Wireshark and other tools are built upon.

Instead of extracting and converting obstacles from polar radar to a Cartesian map, that again produces polar data, a hybrid algorithm may be used. Map and radar may each produce a preferable directions polar diagram of their own and merge them in a final step. This approach involves less complexity but does not allow to use dynamics of obstacles and therefore calculations like the time and distance of closest approach as needed for some COLREG decisions [10, section 4.2].

## 11 Criticism and Conclusion

The empirical found protocol described in this paper allows the direct digital access to radar imagery on a low level from a *Navico Broadband Radar* via standard Ethernet and an UDP/IP stack. This is an easily available interface for mid-range embedded systems. This paper does not deal with any of the other issues involved in operating a radar and reading its imagery. Especially it does not deal with automatic adjusting(e.g.. gain), preprocessing (eg. removing radar artefacts) or analysing the data and extracting obstacles. All topics we will have to face in the next months. We have also not tested special cases like [3] where the boat turn rate approximates that of the radar.

Furthermore on our class of boats (a 4 meter sailing vessel) the radar will be mounted at quite a low height of about 1 m and will need a heeling compensation either mechanically or algorithmically (e.g. removing/reconstructing bad parts of scan). This may require further test with different radar antenna angels relative to reference plane(ground/sea). Therefore we expect a fairly limited range in comparison with the technically possible range.

## References

1. Austrian Association for Innovative Computer Science - INNOC, "ASV Roboat." `http://www.roboat.at`.
2. Russel Technologies Inc., "RTI Radar Installation Maintenance Manual XIR3000C," 2009. available at `http://www.russelltechnologies.ca/downloads/RTIInstallationManualXIR3000CwithFurunoRadars.pdf`.
3. J. Larson, M. Bruch, and J. Ebken, "Autonomous navigation and obstacle avoidance for unmanned surface vehicles," in *SPIE Unmanned Systems Technology VIII, Orlando FL*, April 2006.
4. C. Almeida, T. Franco, H. Ferreira, A. Martins, R. Santos, J. M. Almeida, J. Carvalho, and E. Silva, "Radar based collision detection developments on USV ROAZ II," in *OCEANS 2009 EUROPE*, pp. 1–6, IEEE, 2009. ISBN 978-1-4244-2522-8.
5. Navico, "Broadband Radar TM - The Essential Guide," 2009.
6. S. Cheshire, B. Aboba, and E. Guttman, "RFC 3927 - Dynamic Configuration of IPv4 Link-Local Addresses," May 2005.

7. H. Holbrook, B. Cain, and B. Haberman, "RFC 4604 - Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast," August 2006.
8. C. Sauze and M. Neal, "A raycast appoach to collision avoidance in sailing robots," in *International Robotic Sailing Conference 2010 Proceedings*, pp. 26–33, June 2010.
9. R. Stelzer, K. Jafarmadar, H. Hassler, and R. Charwot, "A reactive approach to obstacle avoidance in autonomous sailing," in *International Robotic Sailing Conference 2010 Proceedings*, pp. 34–40, June 2010.
10. M. R. Benjamin, J. J. Leonard, J. A. Curcio, and P. M. Newman, "A method for protocol-based collision avoidance between autonomous marine surface craft," *Journal of Field Robotics*, 2006.