# Control of Ship Model with DasyLab

**ORG405 Communication and Cooperation**
Autumn 2005

HØGSKOLEN I AGDER
*Agder University College*
*Faculty of Engineering and Science*

| | |
|---|---|
| **Author(s): Group 3**<br>Brit Furnes-Wilkens, Raymond Koteng<br>Ronny Guttormsen, Pål Nilsen<br>Harald Unander | **Supervisor(s):**<br>Hans Jørgen Mørch<br>Michael Snaprud |
| **Version**: 1.0<br>**Status: final** | **Pages:** 13 (including this page)<br>**Modified date:** 2005-11-17 |

**Keywords:**

ship model, radio control, DasyLab

**Abstract:**

Hans Jørgen Mørch has a ship model (3 meters long) equipped with motors and steering. This is operated by a regular RF remote control. He wants to replace this basic remote control with an improved communication and software solution, where the boat is controlled using a computer. The solution should also integrate with a GPS system, transferring coordinates and speed back to the operator.

After an initial analysis together with Mørch we have agreed to limit the practical work to controlling and monitoring the boat from the DasyLab tool. DasyLab is a graphical data acquisition and control package that Mørch uses for different purposes for remote control, data collection and presentation.

Refer to the group web page for a more detailed project plan at:
http://org405.sub-sonic.net

**License:**

# 1 Version Control

| ***Version***[1] | ***Status***[2] | ***Date***[3] | ***Change***[4] | ***Author***[5] |
|---|---|---|---|---|
| 0.1 | DRAFT | 2005-11-01 | Pilot report adapted to template | Harald |
| 0.2 | DRAFT | 2005-11-13 | Completed background and design chapters | Harald |
| 0.3 | BETA | 2005-11-16 | Filled out missing links | Raymond |
| 1.0 | FINAL | 2005-11-17 | Final edition | Raymond, Harald, Ronny |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

1 **Version** indicates the version number starting at 0.1 for the first draft and 1.0 for the first version uploaded for review. Version numbers are maintained by the project group.
2 **Status** is DRAFT, REVIEW or FINAL
3 **Date** is given in ISO format: yyyy-mm-dd
4 **Change** describes the changes carried out since the previous version
5 **Author** your name(s)

# 1. Table of Contents

# 2 Introduction

A PC with the DasyLab software is connected via a serial port to the PicoPic unit, which controls motors and the rudder. The task is to control the boat with the appropriate handles in the DasyLab tool, while observing the boats movements visually. Optionally, a GPS receiver or other monitoring devices can be added to give automatic position, speed and direction feedback from the boat.

# 3 Background (Review of literature)

## 3.1 Boat control theory [1]

In an ideal case, navigating a vehicle to reach a known point can be fairly simple: compute the heading to a target point; compare that with the heading of the vehicle, and steer based on the difference. It can get much more complicated (obstacles, dynamics, noisy data, etc.) but this basic scheme works well for a boat.

The first task is to determine where the target point is relative to the boat. A vector to the target can be calculated by subtracting the boat's position from the target's position. One of the most common representations for a GPS position is latitude and longitude given in degrees, minutes, and thousandths of minutes.

The boat's heading can be calculated by taking the difference between its positions at two times in order to get a direction vector and then calculating the heading with the arctangent operation. This sounds easy, but in practice there are some problems. The boat is very slow, cruising at less than a foot per second. If the time between positions updates is small, the boat's motion may be legible compared to the precision of the GPS position measurement. Then it is nearly impossible to get a good heading measurement. GPS position measurements also drift slowly over time. Since the boat is also moving slowly, it may move at nearly the same rate as the drift. Then the measurements will be nearly useless. These problems were demonstrated in a test run where the boat meandered all over a lake. The time between samples was only 1 second. Changing the time between samples to 10 seconds greatly improved performance. This allowed enough travel between points to provide a reasonably good heading estimation.

Some GPS units provide velocity and/or heading information, but the GPS must be moving in order to get this data. If the GPS is moving too slowly, these features may not be usable. This was the case with the boat and its GPS unit. Increasing the boat's speed would probably make things work better. The program would be smaller since the boat heading calculations could be replaced with code that simply reads the data from the GPS. The data from the GPS would probably be of much better quality also.

## 3.2 Control Theory [2]

In order to establish a sound basis for the boat control another approach is to consider this as a regular control/feedback loop. In this case a PID controller is useful.

A Proportional-Integral-Derivative controller or PID controller is a common feedback loop component in industrial control applications (see also control theory). The controller compares a measured value from a process with a reference set point value. The difference or "error" signal is then processed to calculate a new value for a manipulated process input, which new value then brings the process measured value back to its desired setpoint. Unlike simpler control algorithms, the PID controller can adjust process inputs based on the history and rate of change of the error signal, which gives more accurate and stable control. It can be shown mathematically that a PID loop will produce accurate stable control in cases where other control algorithms would either have a steady-state error or would cause the process to oscillate.

## 3.3 DasyLab

DasyLab is a powerful but easy-to-use tool for data acquisition and analysis. It's designed for users who want to start work right away, without any programming effort. The software performs data acquisition, visualization, open and closed loop control, and documentation. With DasyLab you can perform all these tasks interactively on the screen. Symbols represent functions, and you can link them in worksheets according to the task at hand. DasyLab also offers an extensive range of drivers for hardware control.

Higher-level communication
Mørch uses DasyLab 5.60.00 which is the same for version we used in our assignment. The Network modules witch handles remote control is available only with DasyLab-Net. DasyLab-Net is able to communicate with other DasyLab programs and to control other Net-Net programs running on connected machines. A typical configuration is a central computer that controls all functions (a master) and many measuring points that are controlled (slaves). The Remote Control Dialog Box allows you to define DasyLab-Net as Client or Server. With Server mode there are no further settings. With Client mode you can define a list of Servers that can be controlled. The settings of each of these Servers can be defined using the Option button or the servers can be ordered to execute immediate instructions, including Start, Stop or Load.

DasyLab-Net features
Data exchange at rates up to 100 000 samples per second.
Remotely start, stop and load experiments, including simultaneous starts of several measurements.

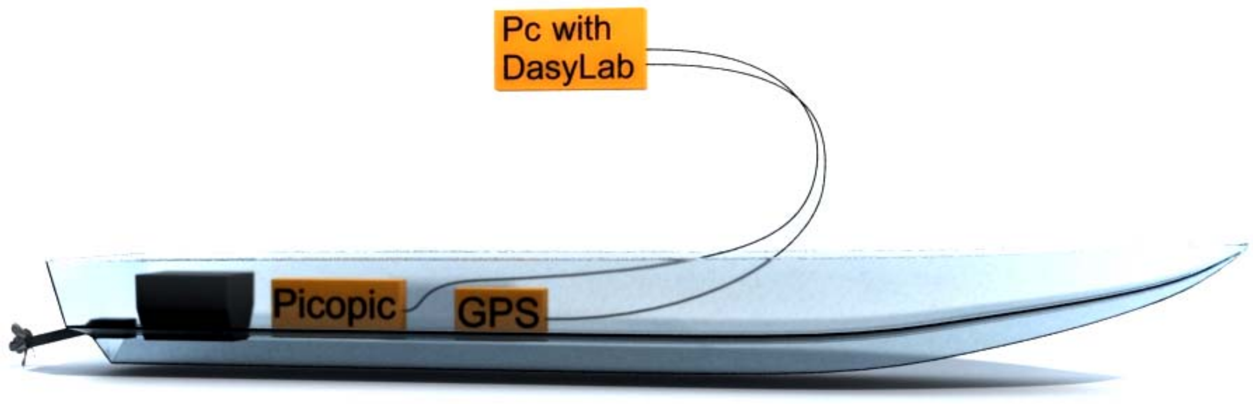
DasyLab-Net systems
Link to running measurements.
Uses the TCP/IP Protocol.
Use existing networks (Microsoft Network, Novell, DEC, etc.).
Runs under Win 95/98 and Windows 2000/NT.

Network - module group
Net Input/Output: Receive or send data to one or more DasyLab Net via a TCP/IP network.
Message Input/Output: Receives or Sends messages from or to other DasyLab Net via a TCP/IP network.
DataSocket Import/Export: Exchange data with OPC server via network.

Installation
The TCP/IP protocol is the basis of DasyLab-Net’s functions. To use the DasyLab-Net functions you need to install a TCP/IP Stack, which includes a WINSOCK DLL file. Installation of MS TCP/IP-32, (which is free of charge by Microsoft), is started using the network set-up from the Network Group. First you install the Microsoft Windows Network with an installed Ethernet board. Then install the TCP/IP-32 as an additional protocol using the driver options.

# 4 Problem description

Establish the mechanisms in DasyLab needed to control the servos in the boat, including server calibration, and possibly an automatic feedback mechanism. In case it is not possible to control the servos directly from DasyLab, a piece of software may be needed between DasyLab and the servo controller.



# 5 Requirements specification

## 5.1 Requirements for the practical part

- It shall be possible to control the boat motors and steering from DasyLab.
- It shall be possible to read back GPS information from the boat.
- A detailed log showing the DasyLab user operations aligned in time with the read back GPS information from the boat shall be available.
- Look at different kind of wireless technologies to make a IP connection.

## 5.2 Requirements for the theoretical

1. Description of how DasyLab can be integrated in an IP network.
2. Description of suitable communication technologies for use between DasyLab and wireless devices for remote control and data acquisition.

# 6 Design – activities

## 6.1 Precondition

Check that the boat motors and rudder can be controlled from last year’s software. This includes check of hardware, electronics, connectors and cabling.

Mørch had a similar ship model project last year. For our help we got a paper copy of last years project report. We also got the source code and executables of the software used. We managed to control the boats servos, with this software.
.

The DasyLab tool can be found here:
http://www.dasylab.com/

The servo controller used in last year's project is PicoPic. This is a multi channel servo controller with a serial interface (RS232). Sending simple ASCII control sequences controls the servos. This unit is used to control both the motors and servos of the boat on a computer.

Documentation of the PicoPic servo controller used in last years project:
http://www.picobotics.com/PicoPic.html

## 6.2 Establish boat control form DasyLab

The boat speed, directions and possibly waypoints are entered in DasyLab with slide controls and text input fields. The data is formatted according to the PicoPic protocol and is then transmitted to the serial port witch PicoPic is connected to, which in turn sets the corresponding voltages and/or pulses for the motors and servos.

## 6.3 Calibration

In order to adjust the scale for the speed slide control and to set the upper and lower limits for the control, the servos must be connected and actual speed must be measured for different output values. When the correspondence has been found a table with slide control positions and the appropriate control values must be stored in DasyLab. A similar procedure must be applied for the rudder control, so that a middle slide position gives puts the rudders in the centre position and so the they are not turned beyond their physical limits.

Please note that this calibration will not result in an exact boat speed or direction given by a certain slide position. It is merely a rough adjustment so that the different slide positions give reasonable outputs to the motors, within their physical and electrical limits, and so that their range can be fully utilised.

The exact direction and speed control will be possible when the GPS information is used in the control loop as defined in the next chapter.

## 6.4 Reading of GPS information

DasyLab can read position information directly from any GPS receiver through a standard serial port. The data from the GPS unit is sent as a sequence of ASCII characters and coded according the NMEA specification. This is easily handled in DasyLab by setting up the appropriate match patterns in the input stream. The read data values can then be used in the feedback control mechanism, according to the control theory in chapter 4.

DasyLab has a PID module and the Two-Point Control module providing standard control logic for open and closed-loop controls. This can be used to implement the steering and velocity feedback loop.

## 6.5 Integration in IP network

DasyLab-Net gives the possibility for a group of DasyLab clients to cooperate over a TCP/IP network. However, it is not clear whether DasyLab can control a remote sensor or control unit without a corresponding remote DasyLab installation. Our original intention was to control the servos with the PicoPic servo controller connected to remote GPRS unit. For this to work DasyLab must be able to make a simple TCP/IP socket connection and messages to the PicoPic and from the GPS unit can then be routed over the Internet.

## 6.6 Wireless communication

Do a survey of available wireless communication technologies for boat control and/or general mobile device communication. Consider range, security, data rate, coverage etc.

This part has been left out due to the scope reduction.

# 7 Implementation

## 7.1 Last years project

After we received the boat and found out how to connect the wires, we had to translate the java code from a Linux interface to a Windows interface. It took a lot of time to find all the devices we needed, but after some time we manage to compile and communicate with the microprocessor. And we were able to control the boat with this microprocessor.

## 7.2 Cell phone and IP

For an IP connection we started to look at different way to use a cell phone to send info from the boat to a server over an IP connection. After some research we learned that we could use java micro edition (j2me) together with a cell phone to set up this connection. We learned that almost every new cell phone could use socket connection trough IP. We made a little application for this based on java wireless toolkit and manage to set up a socket connection with a cell phone. In the appendix you can an extract of the source code. But after a meeting with Mørch we agreed to focus on DasyLab so we didn't make a complete solution of this.

## 7.3 DasyLab

DasyLab is a graphical data acquisition and control package that Mørch uses for different purposes for remote control, data collection and presentation. It has a basic drag and drop system to all components. No one in our group had ever tried it so it was very challenging to understand and use.

# 8 Evaluation and testing

## 8.1 Starting phase

We started to control the boat with the java application. We were able to control the motor, rudder and calibration of these equipments. We started to analyse what kind of signals that was sent to the microprocessor.
The boat is connected trough com port. It's controlled with values that are sending over this com port to the microprocessor. The microprocessor, PicoPic, uses these values to adjust the current to the different type of equipment on the boat.

The signal that is send has following structure:

1. address
2. pin on microchip
3. 2 fist values
4. 2 last values
5. terminate index

## 8.2 Rudder

The rudder is controlled by 2 servos. These servos have to move at same angel in order to control the rudder and therefore have the will get the same signal for the server.
Use pin 19 and 20 on the microcontroller and the values goes in step of 50.

The values look like this:

When centred, value: 1500.
01:13:05:dc:00

01:14:05:dc:00

Max port, value: 1900
01:13:07:6c:00
01:14:07:6c:00

Max starboard, value: 1100
01:13:04:4c:00
01:14:04:4c:00

## 8.3 Motor

The boat has 2 motors and both are controlled at the same time.
Uses pin 1 and 2 on the microcontroller and values go in step of 50.

When centred (idling), value: 1500
01:01:05:dc:00
01:02:05:dc:00

Slowest speed, value: 1000
01:01:03:e8:00
01:02:03:e8:00

Max speed. Value: 2000
01:01:07:d0:00
01:02:07:d0:00

## 8.4 DasyLab

The next step was to use a PC with DasyLab software connected via a serial port to the PicoPic unit which controls motors and the rudder. The task is to control the boat with the appropriate handles in the DasyLab tool while observing the boats movements visually.  If we manage to do this a GPS receiver or other monitoring devices can be added to give automatic position, speed and direction feedback from the boat.
We have also looked at the possibility of using DasyLab to establish an IP connection but settings, at least for the version that we have, that this isn’t possible to connect to other devices. As far as we have learned DasyLab can only connect to other DasyLab witch means that we won't be able to connect to the PicoPic using a cell phone that runs with java micro edition.

# 9 Discussion

## 9.1 Project outcomes

In this project we manage to reach many of our goals.

- connected all wires right on the boat
- translate last years project to windows
- use the application to connect to the PicoPic
- got all the information we needed about the signals that is send to the PicoPic
- made a j2me application for IP connection
- came a long way with our DasyLab project

But we didn't manage to make a complete working control system for DasyLab

## 9.2 Evaluation of the overall results

As you can see from the outcomes topic we manage to reach many of our goals. We had a continuous work on this project and have always felt that we holding our scheme, but the integration with DasyLab were a bigger challenge than expected.

# 10 Conclusion

We started out very ambitious with plans to implement remote control via GPRS to the boat, but it ended up with a much smaller scope that was about controlling the boat by wire from DasyLab. The original scope was really inspired from the original task description where there was no reference to DasyLab or other limitations, and our motivation was really good. After a number of meetings with Mørch it was clear that what he really wanted was the possibility to control the boat from the DasyLab tool. The communication aspect was not important after all. We wasted quite some effort on something that was thrown away. This change, in combination with other demanding tasks being given to the team, was a kind of motivation killer.
When looking back we now see that we should have used less time on defining the task, and it would have been helpful if the task description had been closer to what our supervisor really wanted.

# Appendix

## A1 Glossary & abbreviations [3]

Servo
In its most general definition, a servomechanism, usually shortened to just servo, is a device used to provide mechanical control at a distance. For example, a servo can be used at a remote location to proportionally follow the angular position of a control knob. The connection between the two is not mechanical, but electrical or wireless, for example.

Calibration
Calibration refers to the process of setting the magnitude of the output (or response) of a measuring instrument to the magnitude of the input property or attribute within specified accuracy and precision.

NMEA [4]
NMEA 0183 (or NMEA for short) is a combined electrical and data specification for communication between marine electronics and also, more generally, GPS receivers. The NMEA 0183 protocol is a means by which marine instruments and also most GPS receivers can communicate with each other. It has been defined by, and is controlled by; the US based National Marine Electronics Association. The 0183 standard use a simple serial protocol transmitting a "sentence" from one "talker" to one or more "listeners".

GPRS
General Packet Radio Service (GPRS) is a mobile data service available to users of GSM mobile phones. It is often described as "2.5G", that is, a technology between the second (2G) and third (3G) generations of mobile telephony. It provides moderate speed data transfer, by using unused TDMA channels in the GSM network. Originally there was some thought to extend GPRS to cover other standards, but instead those networks are being converted to use the GSM standard, so that is the only kind of network where GPRS is in use. GPRS is integrated into GSM standards releases starting with Release 97 and onwards. First it was standardised by ETSI but now that effort has been handed onto the 3GPP.

GPS
The Global Positioning System, usually called GPS is a satellite navigation system used for determining one's precise location and providing a highly accurate time reference almost anywhere on Earth or in Earth orbit. It uses an intermediate circular orbit (ICO) satellite constellation of at least 24 satellites.

Java micro edition
The Java 2 Platform, Micro Edition (J2ME) provides a robust, flexible environment for applications running on consumer devices, such as mobile phones, PDAs, and TV set-top boxes, as well as a broad range of embedded devices. Like its counterparts for the enterprise (J2EE), desktop (J2SE) and smart card (Java Card) environments, J2ME includes Java virtual machines and a set of standard Java APIs defined through the Java Community Process, by expert groups whose members include leading device manufacturers, software vendors, and service providers. The following page shows some of the code we made for the j2me application.

```
package mobiltest;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;

public class HelloCell extends MIDlet implements CommandListener {
    private Command exitCommand;
    private TextBox tbox;

    public HelloCell() {
        exitCommand = new Command("exit", Command.EXIT, 1);
        tbox = new TextBox("Klient", "leser serial og sender...", 25, 0);
        tbox.addCommand(exitCommand);
        tbox.setCommandListener(this);


     String connectString = "socket://127.0.0.1:20000";
     OutputConnection sc = null;
     System.out.print("DETTE FUNKER");
      OutputStream os = null;



      try {
          sc = (OutputConnection) Connector.open(connectString, Connector.WRITE);
          os = sc.openDataOutputStream();

          byte[] data = "Hello from a socket!".getBytes();
          os.write(data);

          System.out.print("kom inn på server");
      }

      catch (IOException e) {
          System.out.println("IOException caught:" + e);
      } finally {

          try {
              if (os != null)

                  os.close();
          } catch (IOException ignored) {}

            // free up the socket connection after use
            try {
                if (sc != null) sc.close();
            } catch (IOException ignored) {}
        }

    }
```

## A2 References

| | |
|---|---|
| Homepage DasyLab | http://www.dasylab.com/ |
| Homepage PicoPic | http://www.picobotics.com/PicoPic.html |
| Javax.comm info | http://java.sun.com/products/javacomm/index.jsp |
| Javax.comm info | http://www.javaworld.com/javaworld/jw-05-1998/jw-05-javadev.html |
| Java micro edition | http://java.sun.com/j2me/ |

[1] http://www.acroname.com/brainstem/examples/gpsboat/gpsboat.html
[2] http://en.wikipedia.org/wiki/PID_controller
[3] http://en.wikipedia.org/wiki/
[4] http://gpsd.berlios.de/NMEA.txt