Bionics course project work 2008

General

The goal of the assignment is to use bio-inspired whiskers as robot sensors. The assignment will be done using the J2B2-robot as the platform. J2B2 is a tower-like robot platform which can move with two wheels, see figure 1. The final idea of the project work is to use data from the whisker sensors and make the robot follow a path marked in a building's floor. The robot can be thought as a guidance dog that is helping a blind person to walk. The assignment requires some programming skills (only Matlab & Simulink), and knowledge of some data-analysis methods to efficiently use the data from the whisker sensors.

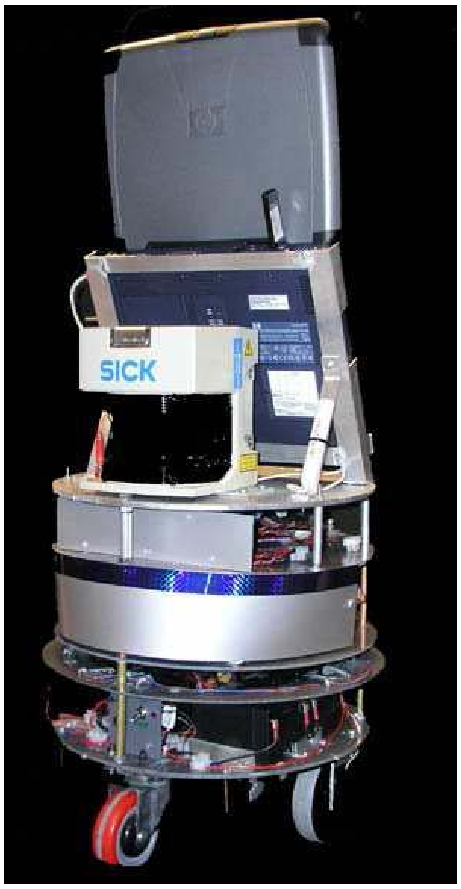


Figure 1 - J2B2 Robot

In the assignment your group needs to gather some data from the whiskers with a given test environment (works fully in Matlab) and then create a Simulink model that is compiled to a C file (so no C coding needed) and then execute your model with a given user interface done with C++. Detailed instructions for using the test environment, model creation & compiling and the C++ interface are found in this document. Each group has an own folder in the desktop where are all the necessary files for the assignment. If you want to be sure that others will not see what you are doing, copy the contents of your folder to somewhere else. In any case, no group should be looking at other groups' files!

Hardware and signal details

In addition to the robot, the following hardware is included:

- Four whisker sensors (attached to the robot)
- Sensor amplifier and filter board
    - o Performs low-pass filtering, amplifying and 8-bit A/D conversion
- Serial port (RS232) connection to a PC
- Power supply for the board

Sensor amplifier and filter board

Before analog to digital conversion sensor output must be amplified and high frequencies must be filtered out to avoid aliasing. There is a combined amplifier and low-pass filter board in the robot where the sensors are attached.

Both amplification and low-pass filtering are done using linear operational amplifiers with capacitors to filter high frequencies. The board has the following design.

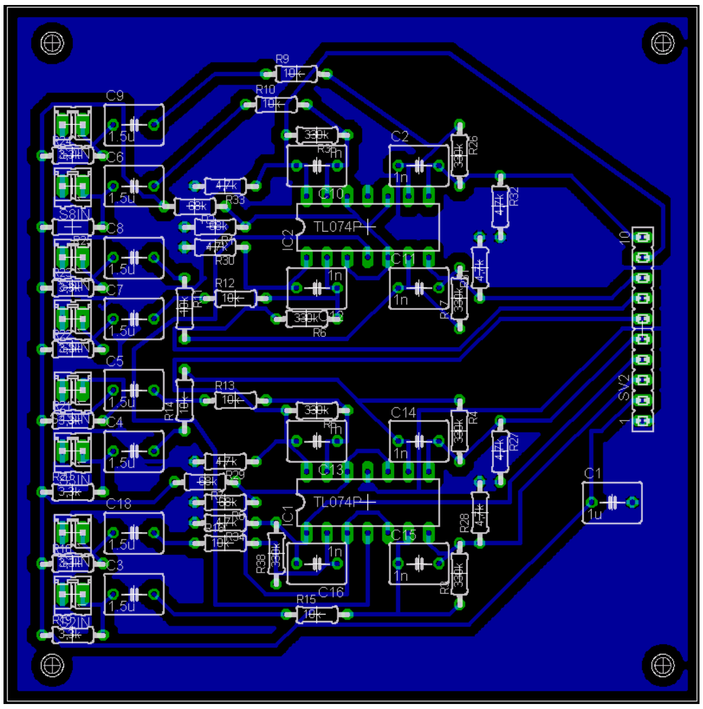


Figure 2. Board design

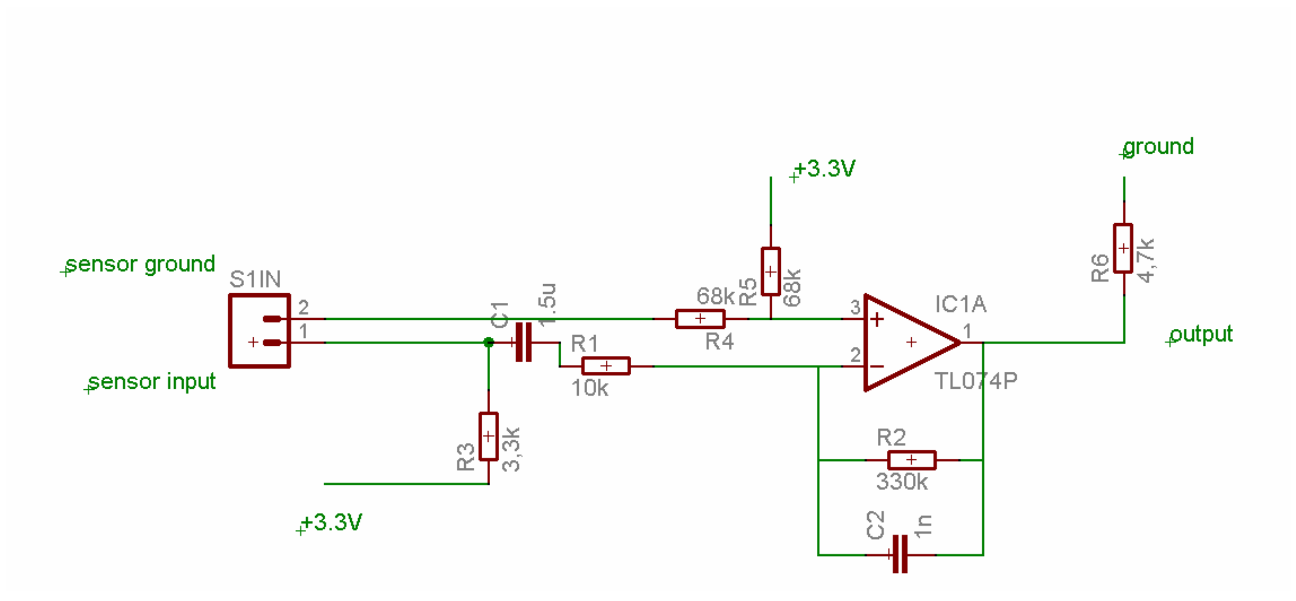


Figure 3. Amplifier and low-pass filter schematic

The board includes amplification and low-pass filtering for 8 whiskers. The schematic of the board can be seen from figure 3. The board amplification and low-pass cutoff frequency depend on the values of the capacitor C2 and resistors R1 and R2. The 3dB low-pass cutoff frequency is calculated with the following formula:

$$w_{cut} = \frac{1}{2\pi R_2 C_2}$$

and the amplification from the relation of the resistors:

$$A = \frac{R_2}{R_1}$$

With these values this gives the whiskers 33 times amplification and 3dB cutoff frequency 480Hz. The capacitor acts as a first degree low-pass filter, so the amplification drop is 20dB/decade. The amplifier output is then sampled with 8kHz frequency which gives approximately 20dB drop for the Nyquist frequency (4kHz) which should be enough to avoid aliasing with the A/D conversion.

Software and communication

The robot is controlled with a PC that is connected to robot server. All the sensor data can be examined through a test interface that will be given. Giving commands to the robot and this way testing your achievements is possible through a user interface. Details of the interface will be given later in chapters “Test environment” and “C-interface”. **In the assignment, the 4 whiskers are connected to channels 1,2,3 and 5** (channel 4 has a weak signal for some reason, so it is not used). **Therefore all signals from other channels can be totally ignored.**

The communication of the whisker-sensors does not happen through the robot interface but through a serial port. The microcontroller responsible for the A/D conversion sends constantly whisker data to serial port at 250Hz frequency. At the start of each packet there is a byte (value is 0X44 in hexadecimal) to identify the start of sequence. After this byte follows 8 bytes which give the current state of the whisker sensor as 8 bit number. A tenth byte in the packet identifies the number of the packet. So each packet contains start byte (0x44) followed by 8 bytes of whisker data. Serial port communication is set to 57600 bps. Example data from a whisker and 2 hits can be seen in the next figure. From the test data you can figure out that the nominal value (the whisker stays stable, no motion) of the output data is 128 and when the whisker is moving, the value gets closer to 0 and 255.

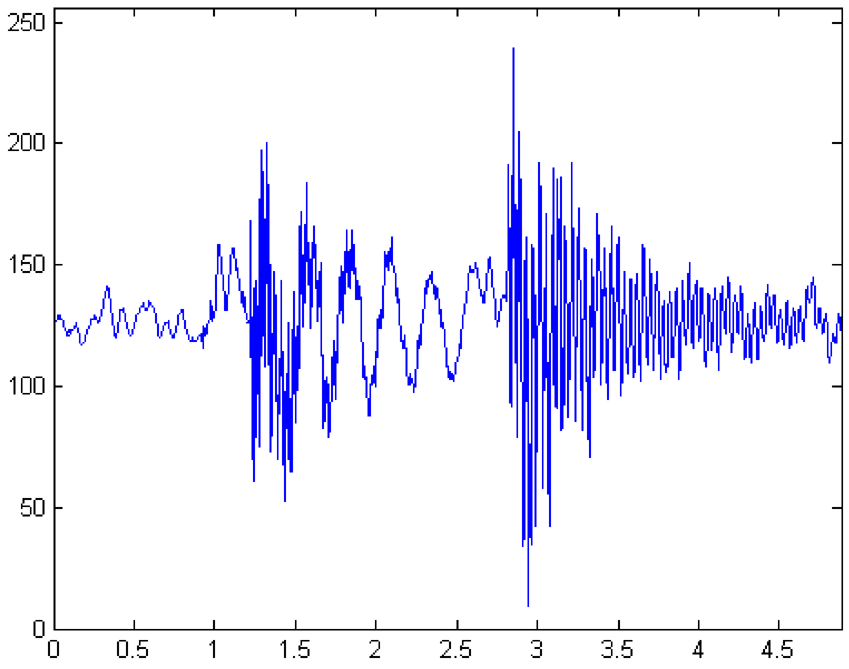


Figure 4 .Test data from 2 hits

Test environment

The test environment works fully in Windows XP and Matlab and it is easy to gather data from the whiskers and later analyze it by yourself freely. In the Windows desktop there is a folder “Bionics project 2008” which includes the “Test_environment” folder. Open Matlab R2007a (shortcut is in the desktop) and change the current directory to the test environment folder. That folder contains 2 Matlab functions called “DataAnalyze” and “ExamineHits”.

The first one is given ready for you, so do not edit in unless it is allowed with a comment in the code. It includes a loop that reads 500 bytes from the serial port at once and forms a vector of each channel. It returns a matrix of all the gathered data, so you can analyze it freely after the execution of DataAnalyze. Also the output data is plotted. During its execution, make the whiskers go through situations that you want to gather data from. For more details, read the comments in that file.

The second one you can implement to detect a hit, and you can freely edit it’s input parameters. However, the function should return 1 when a hit is detected and 0 otherwise.

In the assignment you should use this environment to what kind of driving commands you should use in different situations. The driving commands must include a value for direct and angular speeds. The directional speed should be less than 0.3 (m/s). When giving angular speeds, a positive value makes the robot turn left and vice versa. Angular speed unit is rad/s.

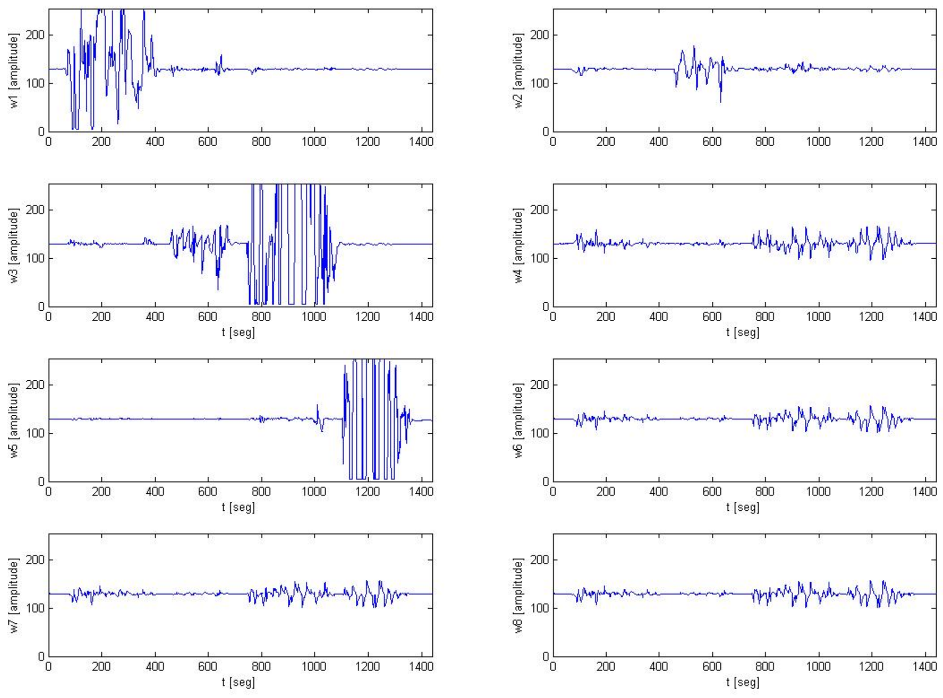


Figure 5. Amplitudes from whiskers 1, 2, 3 & 5

Building your Simulink-model and executing it in the robot

In order to make your own work to go in the robot and test your work in real environment you need to modify a given Simulink model and compile it to C/C++ code. The Simulink model is in Driving Environment directory and it is called “WhiskerAnalyzer.mdl”. Open the model with Matlab, and you get the view below.

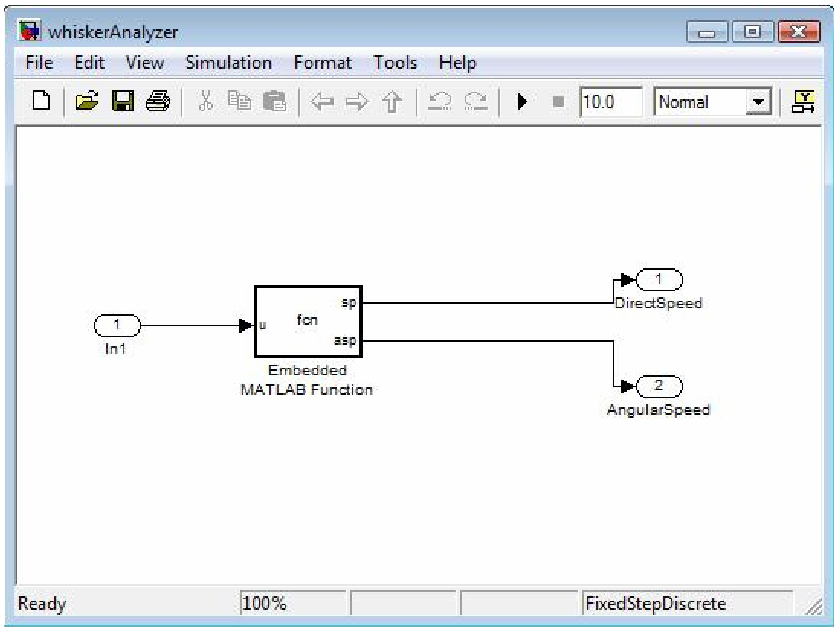
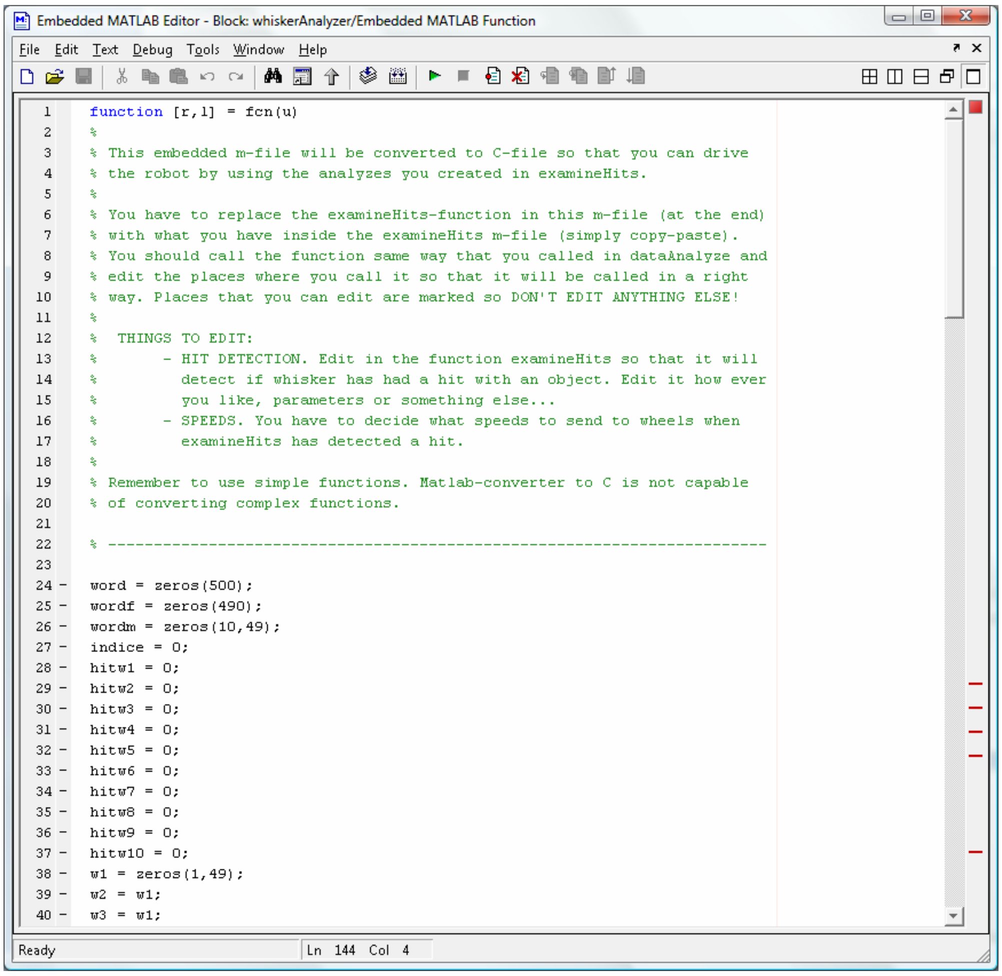


Figure 6. The Simulink model

The input of the model is 500 bytes of whisker data and the two outputs are directional and angular speeds. Next you need to modify the embedded Matlab function –block. Open the block simply by double-clicking it. In the text editor (Figure 7) that opens you can edit the file. See detailed instructions in the comments.


Embedded MATLAB Editor - Block: whiskerAnalyzer/Embedded MATLAB Function
File Edit Text Debug Tools Window Help
Ready
Ln 144 Col 4

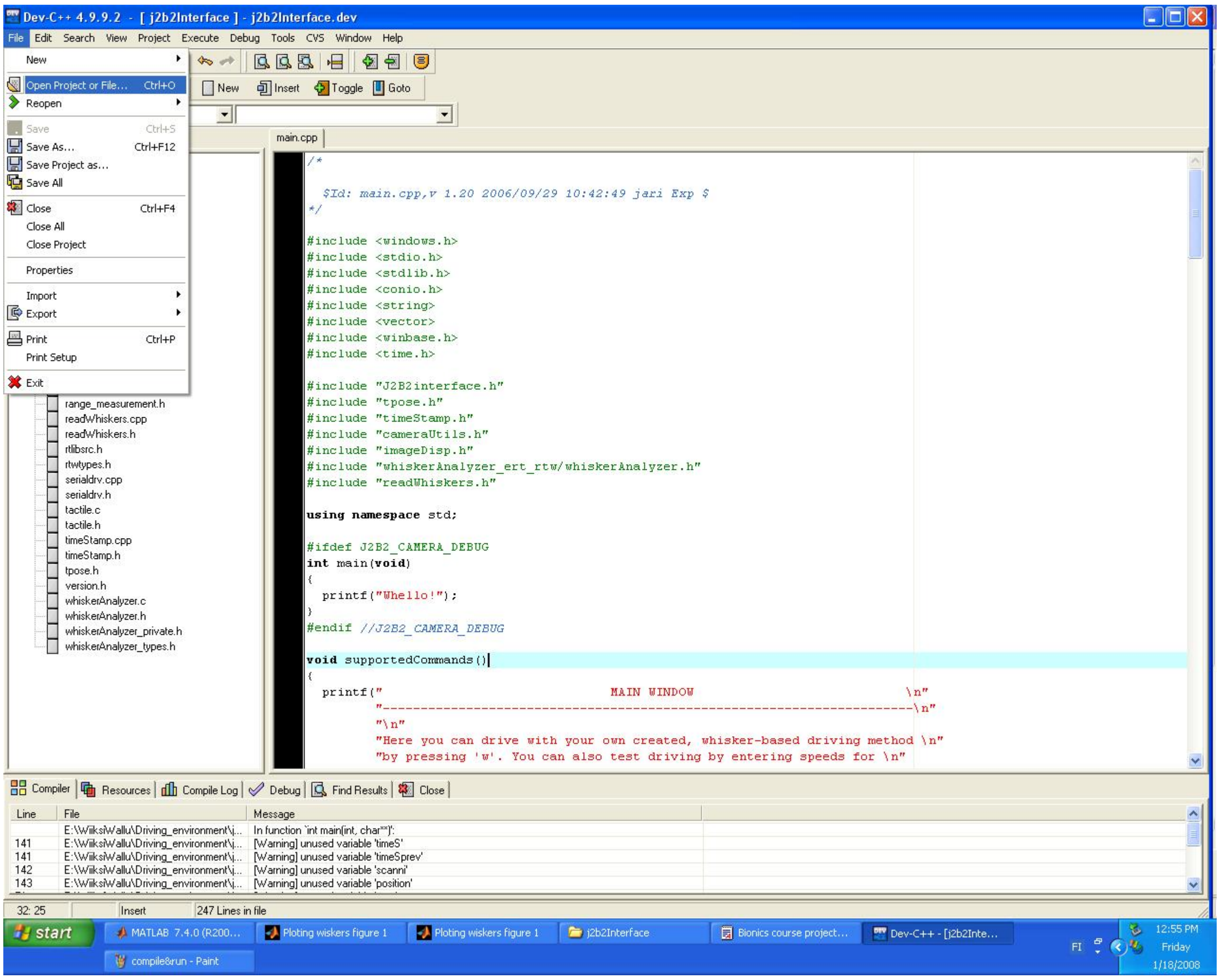
```
function [r,l] = fcn(u)
%
% This embedded m-file will be converted to C-file so that you can drive
% the robot by using the analyzes you created in examineHits.
%
% You have to replace the examineHits-function in this m-file (at the end)
% with what you have inside the examineHits m-file (simply copy-paste).
% You should call the function same way that you called in dataAnalyze and
% edit the places where you call it so that it will be called in a right
% way. Places that you can edit are marked so DON'T EDIT ANYTHING ELSE!
%
%  THINGS TO EDIT:
%       - HIT DETECTION. Edit in the function examineHits so that it will
%         detect if whisker has had a hit with an object. Edit it how ever
%         you like, parameters or something else...
%       - SPEEDS. You have to decide what speeds to send to wheels when
%         examineHits has detected a hit.
%
% Remember to use simple functions. Matlab-converter to C is not capable
% of converting complex functions.

% -------------------------------------------------------------------------

word = zeros(500);
wordf = zeros(490);
wordm = zeros(10,49);
indice = 0;
hitw1 = 0;
hitw2 = 0;
hitw3 = 0;
hitw4 = 0;
hitw5 = 0;
hitw6 = 0;
hitw7 = 0;
hitw8 = 0;
hitw9 = 0;
hitw10 = 0;
w1 = zeros(1,49);
w2 = w1;
w3 = w1;
```

Figure 7.Embedded function

Compilation

Compiling the code From Matlab to C/C++ is really easy: just press ctrl+b or select Tools -> Real-time workshop -> Build model, which automatically compiles the model and puts the generated c-files into the right folder. Note that building the model is possible also in the editor (Figure 7) but it works better when done in the model window (Figure 6). Also note when compiling that the compiled code goes to MATLAB's current directory so make sure that it is /Driving_environment when compiling.

Now you have converted the Matlab-files to C-code. You still have to create EXE-file, which happens with DevCpp. Open DevCpp from the desktop. There is a DevCpp-project file ready for which automatically uses the files you compiled from Matlab to C. Open the project from the FILE- menu (see figure 8). You find the right project in folder: \Driving_environment\FSR07.dev.

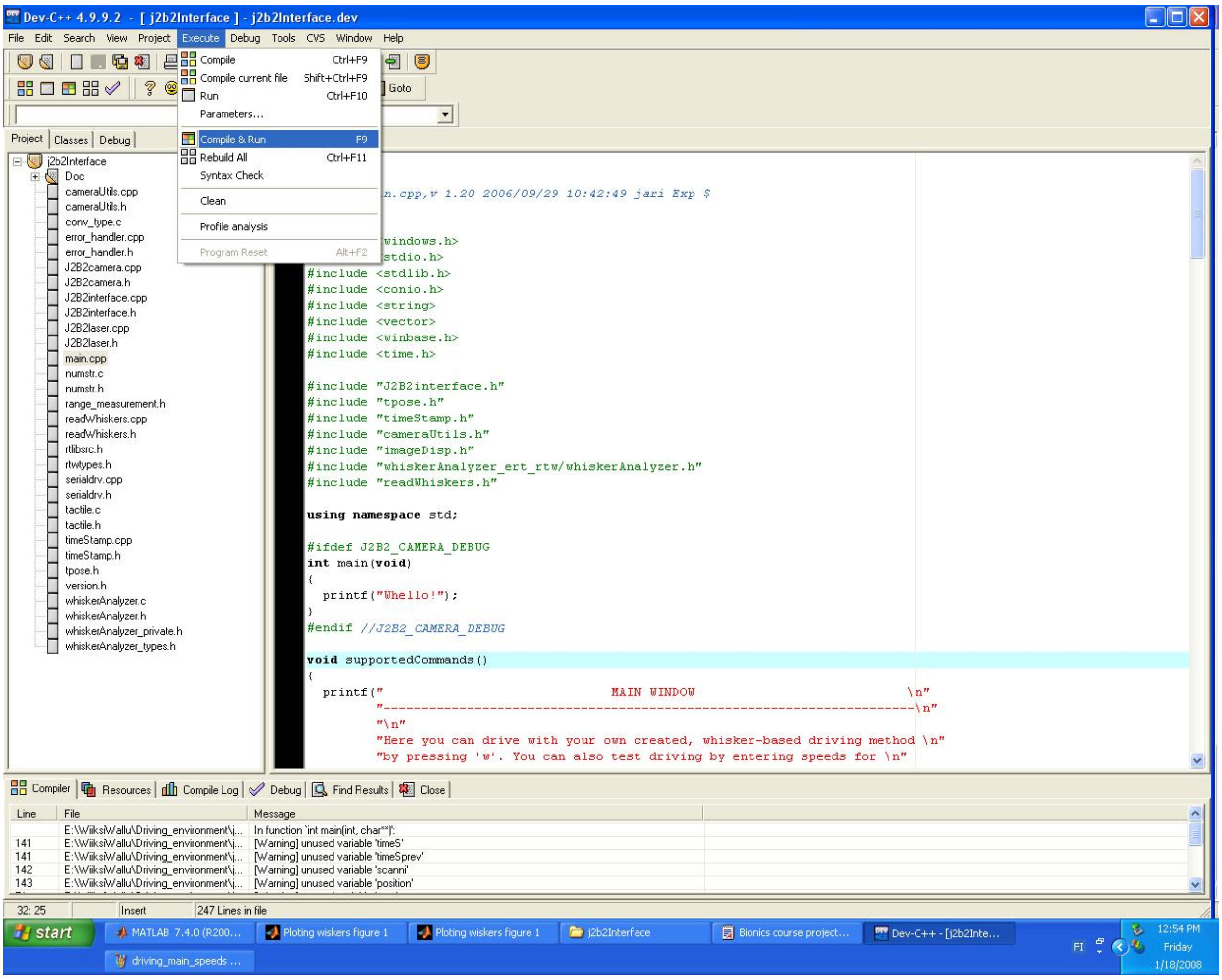


```
/*

  $Id: main.cpp,v 1.20 2006/09/29 10:42:49 jari Exp $
*/

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string>
#include <vector>
#include <winbase.h>
#include <time.h>

#include "J2B2interface.h"
#include "tpose.h"
#include "timeStamp.h"
#include "cameraUtils.h"
#include "imageDisp.h"
#include "whiskerAnalyzer_ert_rtw/whiskerAnalyzer.h"
#include "readWhiskers.h"

using namespace std;

#ifdef J2B2_CAMERA_DEBUG
int main(void)
{
  printf("Whello!");
}
#endif //J2B2_CAMERA_DEBUG

void supportedCommands()
{
  printf("                              MAIN WINDOW                              \n"
         "----------------------------------------------------------------------\n"
         "\n"
         "Here you can drive with your own created, whisker-based driving method \n"
         "by pressing 'w'. You can also test driving by entering speeds for \n"
```

```
Line  File                                   Message
      E:\WiiksiWallu\Driving_environment\j...  In function `int main(int, char**)':
141   E:\WiiksiWallu\Driving_environment\j...  [Warning] unused variable 'timeS'
141   E:\WiiksiWallu\Driving_environment\j...  [Warning] unused variable 'timeSprev'
142   E:\WiiksiWallu\Driving_environment\j...  [Warning] unused variable 'scanni'
143   E:\WiiksiWallu\Driving_environment\j...  [Warning] unused variable 'position'
```

Figure 8. Opening the project

Now the project has been opened. You should not modify any of the C-files included in the project, and all you have to do is compile the project to an executable and run it. That happens by pressing F9 or selecting Compile & Run from the EXECUTE-menu (see figure 9).



```
n.cpp,v 1.20 2006/09/29 10:42:49 jari Exp $

windows.h>
stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string>
#include <vector>
#include <winbase.h>
#include <time.h>

#include "J2B2interface.h"
#include "tpose.h"
#include "timeStamp.h"
#include "cameraUtils.h"
#include "imageDisp.h"
#include "whiskerAnalyzer_ert_rtw/whiskerAnalyzer.h"
#include "readWhiskers.h"

using namespace std;

#ifdef J2B2_CAMERA_DEBUG
int main(void)
{
  printf("Whello!");
}
#endif //J2B2_CAMERA_DEBUG

void supportedCommands()
{
  printf("                              MAIN WINDOW                            \n"
         "----------------------------------------------------------------------\n"
         "\n"
         "Here you can drive with your own created, whisker-based driving method \n"
         "by pressing 'w'. You can also test driving by entering speeds for \n"
```

```
Line  File                                    Message
      E:\WiiksiWallu\Driving_environment\j...  In function `int main(int, char**)':
141   E:\WiiksiWallu\Driving_environment\j...  [Warning] unused variable 'timeS'
141   E:\WiiksiWallu\Driving_environment\j...  [Warning] unused variable 'timeSprev'
142   E:\WiiksiWallu\Driving_environment\j...  [Warning] unused variable 'scanni'
143   E:\WiiksiWallu\Driving_environment\j...  [Warning] unused variable 'position'
```

Figure 9. Compiling and running the project

The program runs until you manually shut it down so observe the robot well to avoid collisions in the laboratory. It takes a couple of seconds until the robot starts to move so be patient.

Connecting to robot

To establish a connection to robot, you need to have an Ethernet – cable connected between the laptop and robot. Also the internet connection settings at the laptop need to be set up the following way:

IP: 192.168.10.2

Mask: 255.255.255.0

Default gateway: 192.168.10.1

If you want to connect to internet, you need to change the settings to:

IP: 130.233.120.81

Default gateway:130.233.120.254

Robot control interface

There is a simple control interface that is already implemented to robot. You can drive the robot around using keys: ‘a ,s ,w and d’. You can terminate the control thread (for whisker control) by pressing ‘t’, this can come handy at testing phase. The robot also checks the bumpers when its moving, so it should stop when the bumpers hit something. You can implement a bit more advanced usage of bumpers in the project.

Requirements for the assignment

In the laboratory’s room there is a marked path in the floor. Your group’s task is to program the robot follow the path as accurately and quickly as possible using the whiskers and robot bumpers!

Project work report

In addition to the competition groups should also make a report of the subject. The report should answer at least to following questions:

- How could whisker sensors be used in robotics? In what kind of situations are they suited well / badly?
- Search information about use of bionic sensors in general. What are the main advantages / disadvantages? What kind of bionic sensor applications could we see in the future?
- Analyze the data from whisker sensors in different situations (robot moving, not moving, whiskers hit something). Are there any correlations between the whisker sensors? Could the sensors be used as a sensor group (data fusion)? What kind of methods could we use for the analysis?
- Comments and suggestions about the project work.

FAQS

The simulink code doesn’t compile, what do I do?

Read the instructions from above, before compiling make sure youre Matlab working directory is Driving_environment. Also more complicated matlab functions might not compile.

The devcpp project doesn’t compile, what do I do?

Make sure you compile are compiling to right directory (Driving_environment).

How do I use the robot?

Read instructions about using the robot from FSR project work page: http://automation.tkk.fi/AS-84-3145-ProjectWork/Important. Also read the code from main.cpp to see how it works.

I compiled the robot code but it doesn't run.

Make sure you the laptop is connected to robot with Ethernet cable. Also make sure the connection settings are OK (see above).

I cant get any data from whiskers

Make sure the serial port is connected to DPS-board and to laptop. Also make sure the green light in DSP board is blinking. If not, press the reset button on the DSP to restart the processor.

Also make sure that you have connected the USB-serial wire to right COM port. You can check the COM port used from Control Panel->System->Hardware->Device Manger->Ports.

For other problems you might have, contact project work assistant pekka.autere@tkk.fi.