AS-84.3200

Automation- and systems technology
project work

Ceilbot
-
Software

Autumn 2010

Final Report

Supervisor

Tomi Ylkorpi, David Leal Martinez

Study points

2 cr

Author

Klaus Müller

Content

1 Introduction

In our days automation systems are used in every kind of industrial field. During the last years technological progress and decreasing prices of automation components have opened a new field of application: home automation.

Also the ceilbot project takes place in this area. So it aims to simplify people’s everyday life in their homes. Tasks of the ceilbot could be cleaning rooms, helping disabled people with their housework or to bring things from one place to another. All these tasks require a safe locomotion in a continuously changing environment with a lot of obstacles. For ground based service robots this could be a big problem. For these reasons the ceilbot is mounted to the ceiling and moves along a rail on a trolley, which also simplifies the power supply.

During the last semesters several project groups have developed a concept for the ceilbot. The goal of the actual group is to create a prototype of it.

2 Goal of the project

This project is a subproject of the ceilbot project and focuses on the software development and implementation. The first goal is to setup a communication infrastructure between the stationary computer, the trolley computer and the subsystems. The communication infrastructure is compulsory and very important for the other subprojects of the ceilbot project. So it gets the highest priority in our work plan.

Besides the communication, the user-machine-communication is also a very important part of the software project. That’s why the second goal is to implement a basic extendable user interface in order to control the robot and get a visualization of the robots sensor data. Because of the short project time period the user interface will be implemented in a minimal but functional way.

The third goal is to implement a high level control, which translates abstract user commands in a sequence of subsystem commands. This should also include a “watchdog” implementation to avoid accidents caused by wrong user commands or command overlapping.

3 Final results

The following section should give an overview of the work done.

During the first steps of graphical user interface (GUI) implementation the software group figured out that the implementation in a team takes more time than expected. Furthermore the time frame was very short to implement all projected goals in succession. For these reasons the group decided to split the GUI implementation and the upper level software implementation in two parts; the development of these parts went on parallel. Miguel Pérez Cardoso incurred the upper level software part and Klaus Müller the GUI implementation. Because Miguel Pérez Cardoso does not participate the course *AS-84.3200 Automation- and systems technology* the following section only contains the GUI implementation.

3.1 Communication infrastructure

During the first weeks of the project the group studied with the help of some tutorials and examples the structure and functions of *Machine Control Interface* (MaCI) and *Generic Intelligent Machines net* (GIMnet). Furthermore two MaCI (library based on GIMnet) developers (Antti Maula and Matthiew Myrsky) supported the group in this scope. It has to be mentioned that GIMnet and MaCI are very powerful and huge tools/libraries. As a consequence the group members could not get familiar with all the components of this software framework.

3.1.1 GIMnet and MaCI

As mentioned in the upper section the communication structure is based on GIMnet. GIMnet is a communication infrastructure for robot systems and is based on a small lightweight application called *tcpHub.* TcpHub acts as connector between several GIMnet nodes. Several TcpHubs can be connected and build together one big GIMnet, which connects all nodes of the network with each other.

MaCI is a library, which provides a width band of interface to control and to steer robot systems. The interfaces are implemented as client-server-architecture and are used for every kind of sensor or machine.

The service server is registered at the local tcpHub and the client can establish a connection over GIMnet. Because of the modular structure of these interfaces, the different application layers can be developed very independently. If for example the robot hardware is not available, MaCI provides dummy interfaces, which allow the programmer to implement the software without any hardware connection. In addition to that, it is also possible to exchange service server of the same kind in an easy way, which decreases the outage ration.

3.1.2 Communication structure

Finally the group members created the design of the project environment shown in figure 2. This design contains two GIMnet hubs. The first one is placed at the trolley computer and connects all modules - linked to the trolley pc - with each other and with the second GIMnet hub placed at the stationary pc. The additional GIMnet hub at the trolley allows the modules to communicate with each other without any connection to the stationary pc (first hub).

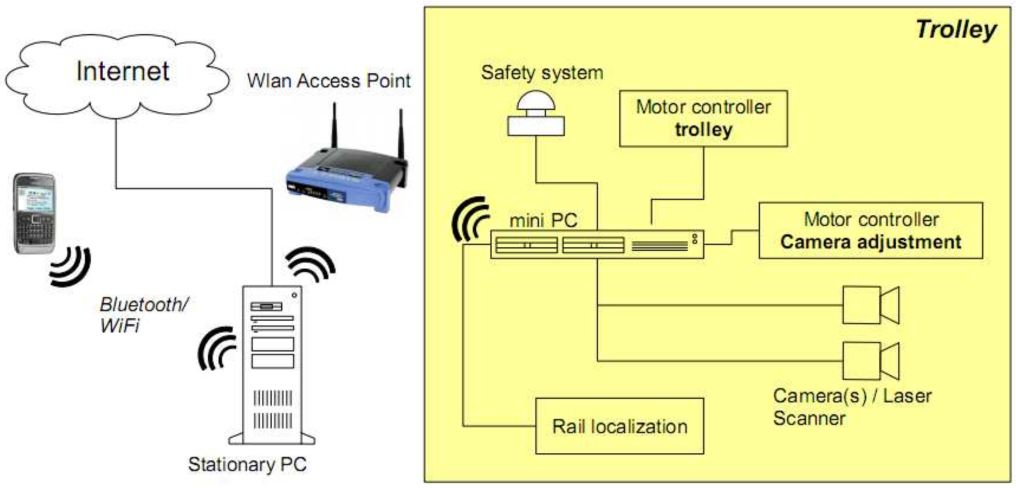

Figure 1 Architecture of the ceilbot

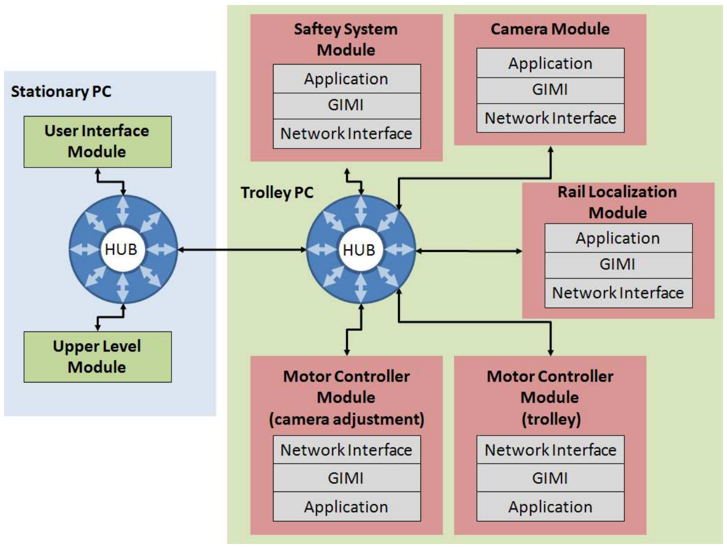

Figure 2 GIMnet design of the ceilbot project

During the first tests the group figured out that the development of the upper level software and the UI is difficult without the real hardware. Unfortunately the first running hardware system (robot and rails) will be completed at the end of the project. So the group searched for a solution and found a robot simulator (FSRsim), which can be used to simulate the robot hardware in a GIMnet environment. The installation of this component and the following tests delayed the timetable for one week.

3.2 Graphical User Interface

During this period the goal of the whole ceilbot project was to create a prototype without manipulator, which is able to move along the rails. Because of this fact the user interface has a limited function volume. The main task was to display the sensor data and to control the robot in an easy way.

3.2.1 Design

As mentioned in the upper section the aim was to provide a simple but functional Graphical User Interface. The GUI design can be found in figure 3.

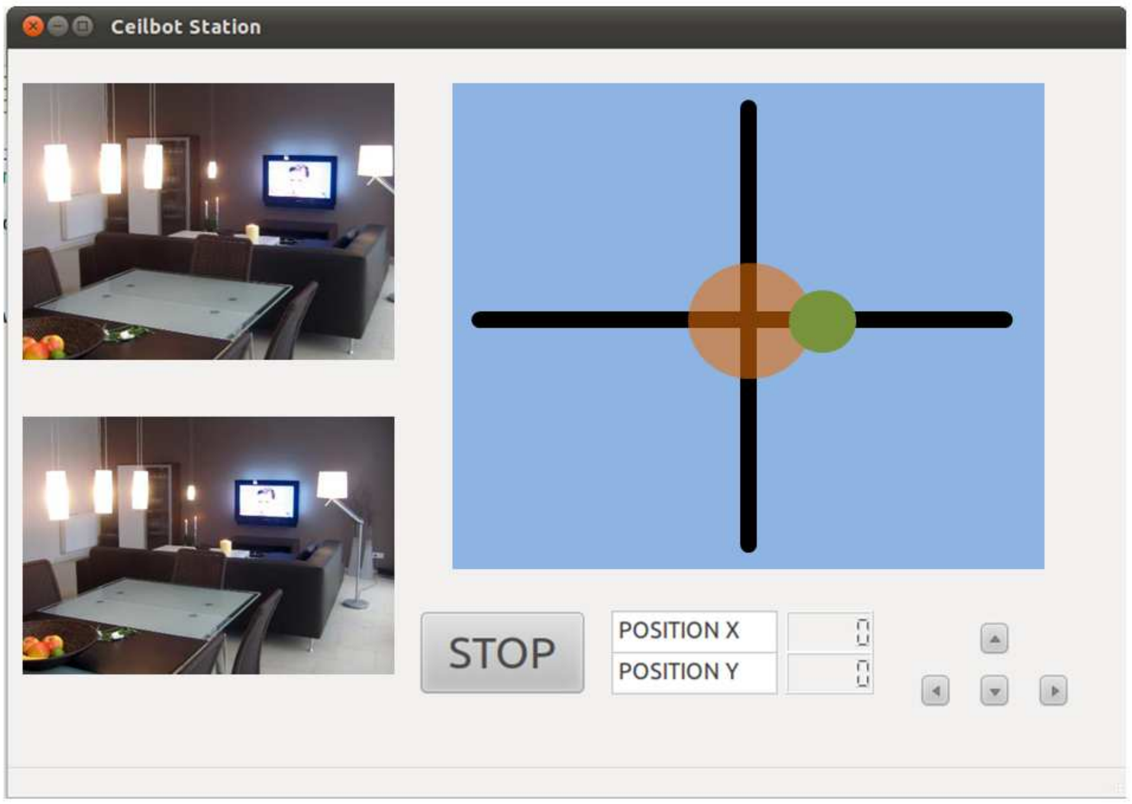

Figure 3 First Ceilbot GUI

On the left side of the window, two video streams of a prospective stereo vision system are displayed. In the next period of the project the stream can easily be redirected to a computer vision application. A map of the rail system is placed on the right side. The green dot represents the robot and the red circle the rail changing system. By clicking on any position of the map the robot moves to the nearest rail position. Some control elements can be found in the lower right corner. With arrow buttons the trolley can be moved in the direction of the rail.

The stop button stops the system immediately. Furthermore two displays represent the robots position on the rails. All components of the GUI can easily be changed or exchanged with other implemented components.

3.2.2 Implementation

The first bigger step of every implementation beginning is to set up the right development environment with all includes and libraries. This is an especial problem in case of combining or including several frameworks to one application. Also this project contents different subsystems. Besides the upper aims for creating this user interface, it is also important to setup a development environment and test the interaction between these frameworks. So it will be easier for following groups to start or continue with the development.
Besides the upper mentions, communication framework GIMnet in combination with the machine interfaces (MaCI) the GUI Framework Qt was chosen to develop the GUI. Qt is a very powerful and cross-platform framework which supports every kind of GUI.
The following sections give an overview of the implementation work.

Development environment

As mentioned in the upper section, the Qt Framework is used to build up the GUI. Qt provides a lot of different designs and layout options, which can directly created in c++ code. The creation of a proper design cost a lot of time in this way and requires experience. For this reason there are a lot of Qt GUI designer tools, which simplify the way of creating a GUI. For this project the Qt creator was chosen. It is a development environment, which provides beside a source code editor and a debugger also a designer.

GUI Framework Qt

Besides the mentioned design and layout options, Qt provides a special language construct, called “slot and signal”. This construct can be used to link different Qt-classes with each other. For this reason the Qt classes can be extended with signals, which send in case of an intern class event a signal to connected slots on the one hand and on the other hand with slots, which can be activated by signals. Signals can also commit parameters to the slots. For example a button object has a signal “pressed” and is connected to a counter object which has a slot “count”. If signal “pressed” activates slot “count”, the counter object will increase the counter.

In this case two signals and five slots were used. The signals inform the slots about new events of the MaCI clients. The first signal is thrown when a new position and the second when a new image is available.
The slots react to these signals and refresh the connected Qt widgets.

FSRsim simulator

The FSRsim simulator was the basis for all tests which have been done during the development. It provides an interface for different kinds of maps and vehicles and simulates all sensor data like imaging, position and movement control. For the ceilbot an own environment was created, which provides position data and movement control. Because of the fact, that most of the ceilbot sensors and equipment has not been chosen yet, the ceilbot simulator module has a very limited functionality and should be extended after the hardware will be chosen. For this reason another simulator module was used, which provides all kind of needed sensor data. The used module can easily be changed to the final ceilbot simulator module.

MaCI Integration

For the upper mentioned goals three MaCI clients are necessary: SpeedControlClient, PositionClient and ImageClient. All this clients have to be initialized with a gimi object. A gimi object represents the connection over the GIMnet to the GIMnet hub and has to be initialized with address and port of the hub. Additionally the clients need the name of service server.
For reasons of simplifying the use of the listed clients, wrapper classes are used to connect the application with the MaCI interfaces which is shown in figure 5. These wrapper classes are located in the common folder path, so also the higher level software can use these classes. The wrappers initialize the clients and return the client object to the calling class.

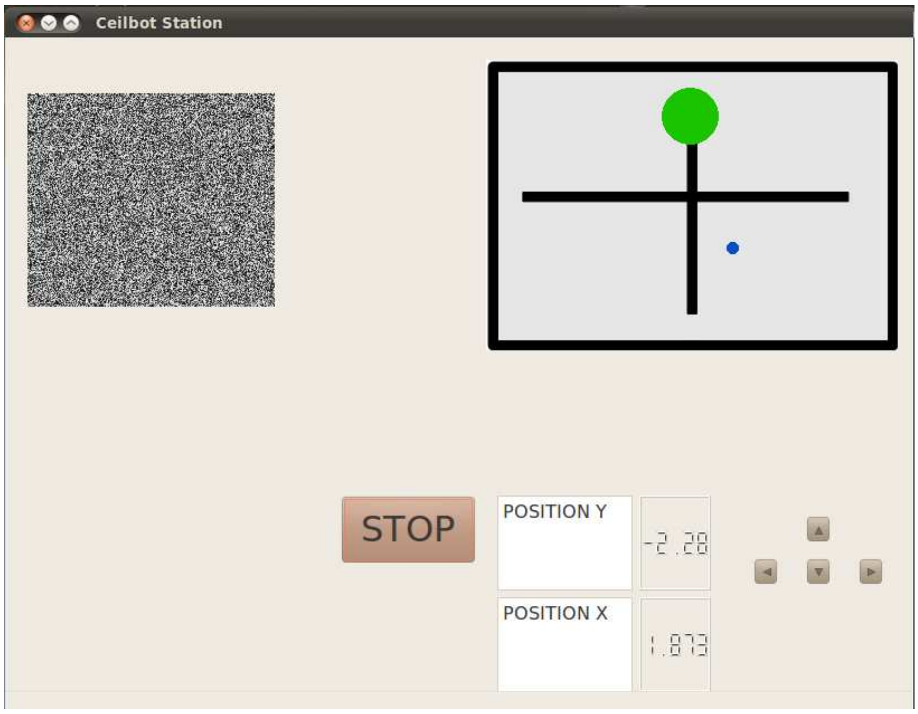

Figure 4 The GUI with all supported interfaces. On the left side the video stream of the simulator (the simulator just transfers random dot pictures, a second stream can easily be embedded), on the bottom the steering elements to move the robot and the position displays. The green dot on the map represents the robot and the blue dot the user target click.

If different service servers are located at the same gimi object, they should be initialized with the same gimi object. For these reasons the wrapper classes are implemented as singleton pattern. If for example the gimi wrapper class is called, the wrapper class will check if a gimi object with the commited parameters already exists. In case of an already existing gimi object, the class will return this. Otherwise the gimi object will be created and returned.

This singleton pattern is also used for the other wrapper classes. If another client is required than the one already created, the class will create a new client object and will hold the previous one.

This architecture can easily be transferred to further wrapper classes in the future.

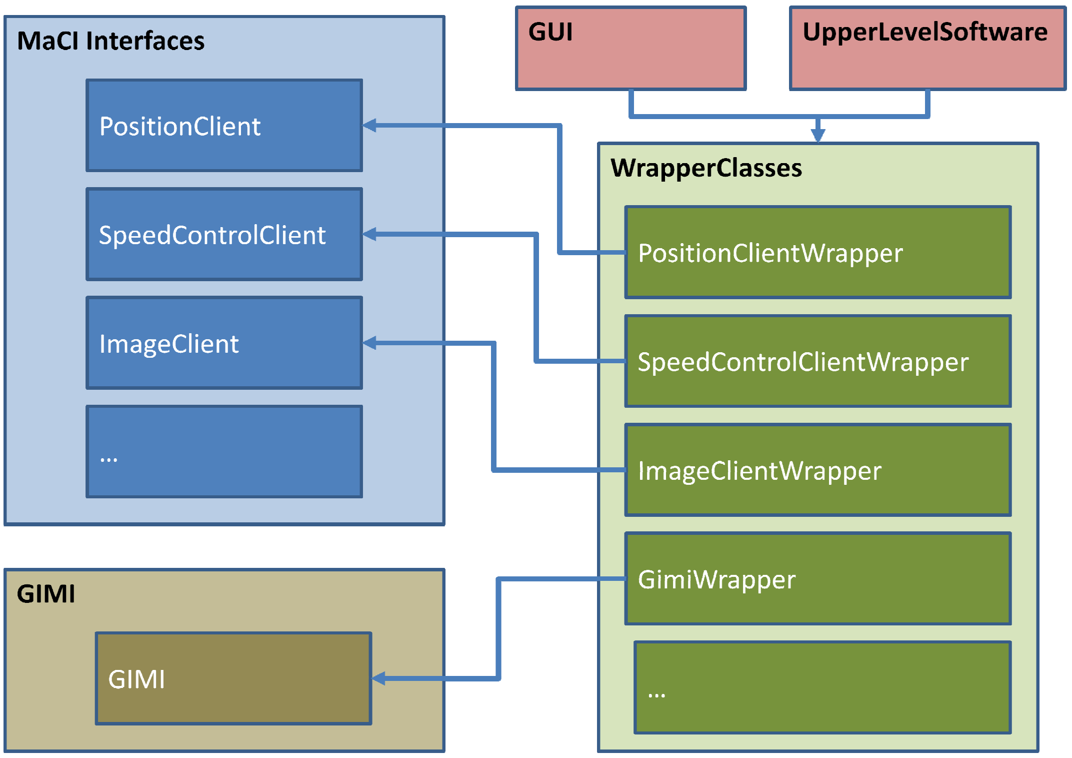

Figure 5 Diagram of the wrapper classes

3.2.3 Configuration of the application

For the fact, that this application can be used with different MaCI based robots and simulators, it could be necessary to change the configuration. The whole source code can be found in the projects repository (*https://jemma.hut.fi/svn/Ceilbot*). The following section describes, which changes have to be done.

MainWindowImpl.cpp

As in figure 4 shown, the GUI provides a map of the current robot world. If the robot world is changed, the map should also be changed. The easiest way to do this is to generate a PNG-graphic of the map and save this in the projects */bin* folder under the name *map.png*. Additionally the range of robot world's position coordinates should be updated, because the application need these coordinates to calculate the real robot position, in case an user clicked a target position on the map to move the robot to this point. These coordinates can be set on top of the *MainWindowImpl.cpp* (variables: *robotWorldMinX*, *robotWorldMaxX*, *robotWorldMinY*, *robotWorldMaxY*). This four variables describe the position range in x and y direction of the robots measurement system.

GimiWrapper.cpp

The GimiWrapper class has the function to initialize the gimi-object, which represent the connection between the application and the tcpHUB. For these reasons the address and the port of the tcpHUB application have to be defined. This can be done at the top of the *GimiWrapper.cpp* file. The GIMnet access point address is represented by variable *gimnetAP* (default value: *"localhost"*) and the GIMnet access point port by *gimnetAPPort* (default value: *50000*).

PositionClient - ImageClient - SpeedControlClient

For the connection to the service servers the clients need the data source names of the servers. These can be committed by calling the constructor of the services' wrapper classes.

4 Time report

The whole project takes more time than expected. Especially the period of implementation has extended the working time drastically, because of searching and fixing errors. The whole time is listed below.

Week	*Details*	*Time (h)*
39	• group meeting • project plan	6
40	• studying GIMnet • project plan • group meeting	6
41	• tests with GIMnet • group meeting	10
42	• GUI design • test with MaCI and simulator • group meeting	10
43	• intermediate report	6
44	• intermediate presentation • GUI implementation • group meeting	30
45	• GUI implementation • group meeting	35
46	• GUI implementation and test • final report	20
47	• last tests • final report	15

48	• final report • application adjustment	7
		145

Table 1 Table of hours worked

5 Conclusion

Beside the fact that the project has taken more time than expected, all projected goals have been fulfilled. The communication system has been set up and tested. After all components of the ceilbot will have been finalized, the next step will be to connect these components to the infrastructure network and replace the simulator with them.

The GUI has been designed and implemented to the desired purpose and the first step for the final control software has been done. The following groups can go on with the development and implement the functions which are necessary for the final use of the robot.

The upper level software was incurred by Miguel Pérez Cardoso and is not described in this report precisely.

Personally the project work was a great experience for me. Not only the confrontation with the ordinary problems and troubles during a project has broadened my horizon, but also the work within an international team has fulfilled my expectations of a semester abroad. Additionally I have gained an insight into the communication infrastructure of a robot system and the interaction of two frameworks, Qt and MaCI.

6 References

[1] Ceilbot. The ceilbot project. *http://autsys.tkk./en/ceilbot* [08.10.2010]

[2] Generic Intelligent Machines. GIMnet. *http://gim.tkk.fi/GIMnet* [08.10.2010]

[3] Klaus Müller, Miguel Perez. Ceilbot – Software – project plan autumn 2010, *http://autsys.tkk.fi/intranet/as-0.3200/attach/A10-17/Ceilbot-software%20-%20Project%20Plan%20-%20Documentation.pdf* , 2010.

[4] Qt Reference Documentation. *http://doc.qt.nokia.com/4.7/index.html* [14.11.2010]

[5] Mark Townsend. Exploring the Singleton Design Pattern, *http://msdn.microsoft.com/en-us/library/Ee817670%28pandp.10%29.aspx* [18.11.2010]