Users Manual
AB Mail Robot

Version 1.0

Author: Martin Melin
Date: December 16, 2010

Course name:	Control Project	E-mail:	danba185@student.liu.se
Project group:	AB MailMen	Document responsible:	Martin Melin
Course code:	TSRT10	Author's E-mail:	marme287@student.liu.se
Project:	AB Mail Robot	Document name:	UsersManual_v10.pdf

Status

Reviewed	Karl Granström André Carvalho Bittencourt	2010-12-15
Approved	Karl Granström	2010-12-15

Project Identity

Group E-mail:	danba185@student.liu.se
Homepage:	http://www.isy.liu.se/edu/projekt/tsrt10/2010/postrobot-2010/
Orderer:	Karl Granström, Linköping University **Phone:** +46 13 281333, **E-mail:** karl@isy.liu.se
Customer:	Marcus Pettersson, ABB Västerås **Phone:** +46 21 345194, **E-mail:** marcus.pettersson@se.abb.com
Course Responsible:	David Törnqvist, Linköping University **Phone:** +46 13 281882, **E-mail:** tornqvist@isy.liu.se
Project Manager:	Daniel Barac
Advisors:	André Carvalho Bittencourt, Linköping University **Phone:** +46 13 282622 , **E-mail:** andrecb@isy.liu.se

Group Members

Name	Responsibility	Phone	E-mail (@student.liu.se)
Daniel Barac (DB)	Project Leader	+46 702641857	danba185@student.liu.se
Martin Melin (MM)	Document responsible	+46 702887793	marme287@student.liu.se
Erik Erkstam (EE)	Software responsible	+46 737266812	erier982@student.liu.se
Hugo Kinner (HK)	Test responsible	+46 733688378	hugki634@student.liu.se
Manfred Hallström (MH)		+46 703679569	manha932@student.liu.se
Simon Hugosson (SH)	Control responsible	+46 708778471	simhu610@student.liu.se
Niklas Carlsson (NC)	Information responsible	+46 709601672	nikca291@student.liu.se
Nicklas Forslöw (NF)	SLAM responsible	+46 739516430	nicfo307@student.liu.se

Document History

Version	Date	Changes made	Sign	Reviewer
0.1	2010-11-23	First draft	MM	MM, DB
0.2	2010-12-02	Second draft	MM	DB
0.3	2010-12-03	Third draft	MM NF	DB
0.4	2010-12-08	Fourth draft	EE	EE DB
0.5	2010-12-15	Fifth draft	EE	EE DB
1.0	2010-10-11	First Version	DB	DB

Contents

1 Introduction

Fido-Dido (Figure 1) is a wheeled mobile robot manufactured by MobileRobots Inc. Its most important features are a SICK laser range finder, 8 forward-facing ultrasonic array sonars, odometers and a Micro-Controller. The robot shall be able to autonomously navigate in an office-landscape using a pre-defined map which it also shall be able to update when changes are detected. It shall also be able to find its way from point A to point B.

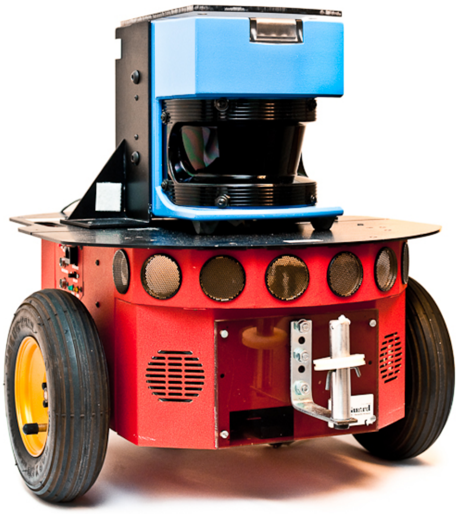

Figure 1: Fido-Dido

This document provides a detailed description of how the software and hardware can be used to achieve our goals, specified in section 1.2.

1.1 Parties

Client of the project is Karl Granström, PhD-student in Automatic Control, Department of Electrical Engineering at Linköping University. Supervisor of the project is André Carvalho Bittencourt, PhD-student in Automatic Control, Department of Electrical Engineering at Linköping University. Customer of the project is Marcus Pettersson at ABB Cooperate Research Center (ABB CRC). The development is performed by the project group ABB MailMen.

1.2 Purpose and goal

The purpose of the project is to develop software and then implement it on a robot so that the robot can navigate in an office environment. The project group will evaluate future possibilities to make this robot able to deliver objects, e.g. mail, which is covered in the technical documentation. The goal of the project is to develop a system for an autonomous robot which shall be able to navigate from point A to point B in an office space, specifically at ABB CRC. The long term goal of the product is to extend the robot with a robotic arm to enable the delivery of objects.

1.3 Usage

The product can be used as a platform for further research at ABB CRC or at Linköping University.

1.4 Background information

This project is a part of the CDIO-course TSRT10 at Linköping University and is performed in collaboration with ABB CRC. The group consists of eight students with expertise within automatic control and sensor fusion.

1.5 Scope

The robot is limited to operate in an office environment. The robot cannot be expected to be able to open doors or move objects. The floor on which the robot will operate must be free from obstacles which the robot cannot detect.

2 Overview of the System

The system consists of a MobileRobots P2-DX robot with a HP ProBook 6450b laptop mounted on it. The laptop runs the application that controls the robot and could also be used as an interface if a user wants to manually control the robot or view the robot's map. The robot and the laptop communicate via a RS232 serial port. The different parts of the system and how they communicate can be viewed in Figure 2. The laser and sonars send range measurements and the odometers send wheel position and speed estimates. The micro controller (MC) decodes the data from the sonars and odometers and passes it on the program. The MC also translates the motion commands from the program to control signals for the motors. When collecting data, the user drives the robot with the laptop's arrow keys. When mapping is done the user can tell the program where the robot should go. The output from the program to the user is a map and the robot's current position in the map.

2.1 Subsytems

This section includes a short description of the subsystems.

2.1.1 Robot

The robot is a wheeled MobileRobots P2-DX robot equipped with odometers, a laser, sonars and a micro controller (MC). The micro controller receives data from the sonars and odometers. It also sends control signals, calculated by the software on the laptop, to the servos.

2.1.2 Laptop

The laptop receives sensor data from the laser and the robot's micro controller. It is also responsible for the software which handles SLAM, trajectory planning and calculation of the control signals.

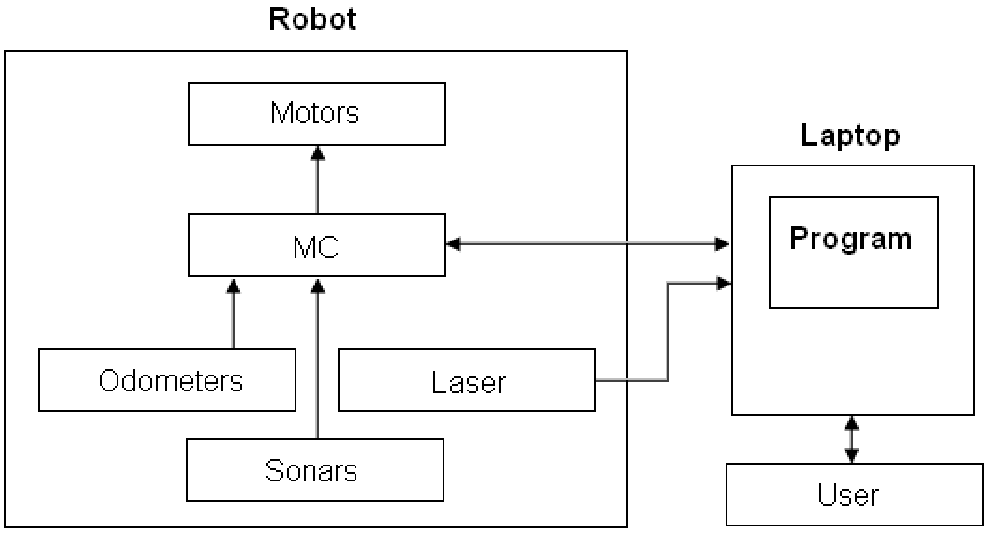

Figure 2: Overview of the system.

The application is written in C++ and uses the ActivMedia Robotics Interface for Application (ARIA) library for all communication with the robot. ARIA has utilities for both receiving data from the sensors and controlling the motors. The program can be divided into three subsystems with specific tasks according to Figure 3. The SLAM system receives range measurements from the laser and sonars. The control system receives a pose estimate from the odometers and pass it on to the SLAM system together with a calculated covariance matrix. The SLAM system sends a grid map and an improved pose estimation to the trajectory planner which plans a trajectory to the goal and sends the next position to go to, to the control system. It, in turn, sends the desired wheel velocities to the motors.

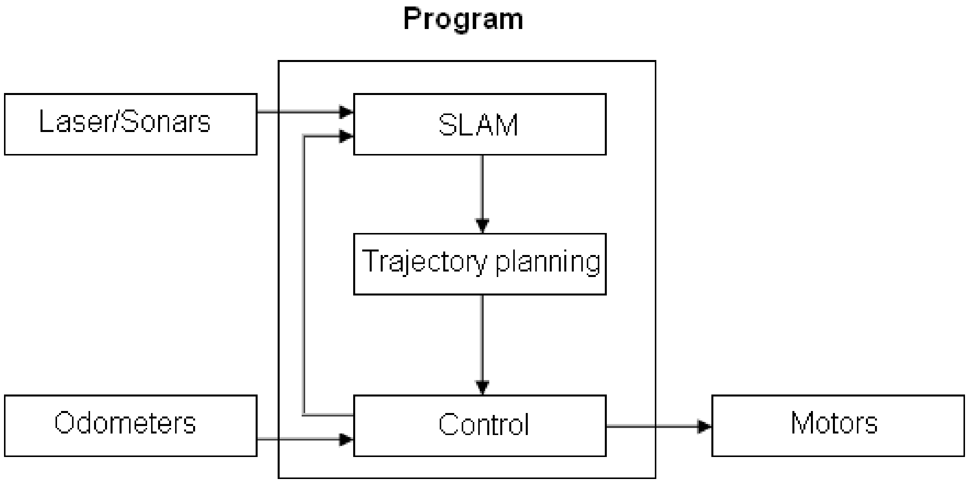

Figure 3: Program structure.

3 Using the programs

In this section it will be explained how to create a map by using the software and how to transport the robot from point A to point B within the created map.

3.1 Getting Started

In order to use the software the Robot should be connected to a laptop in the following way.

1. Connect your Laptop to the serial usb adapter, located on the robot, see Figure 4.
2. Turn on the robot by switching the switch on the Robot's right side to on, see Figure 5.

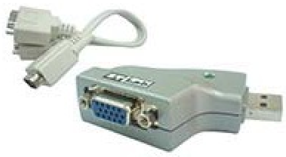

Figure 4: Serial Usb adapter to be connected to your laptop.

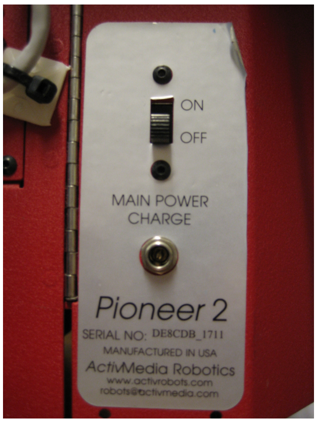

Figure 5: Switch to on to start the robot.

3.2 Creating a map

1. Open the directory ABMailMen/Debug.
2. Run the program by double clicking ABMailMen_MapMaker.exe.
3. Use the arrow keys to move the robot around the area of interest.
4. The file "scan.2d" will be saved in the current directory including measurements from the recorded session.

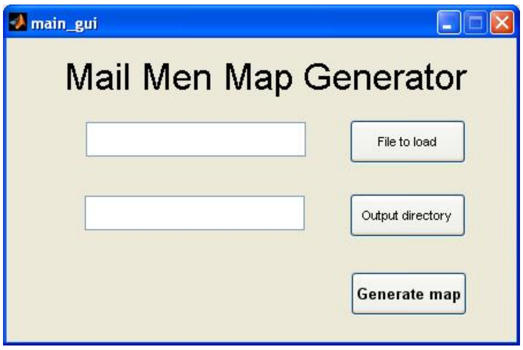

Figure 6: Map-Generator.

5. Start the Map Generator program in MATLAB by running the file main_gui.
6. Specify where the program can load the scan.2d file.
7. Select the output directory to the location of ABMailMen.exe.
8. Run the program by pushing the Generate map button. Wait until the text ”Complete: Map generated” is shown.
9. Close main_gui and get ready to take the robot for a ride.

3.3 Driving the Robot from point A to point B.

1. Start the program ABMailMen.exe. A grid map and a line map of your explored area should appear, see Figure 7.

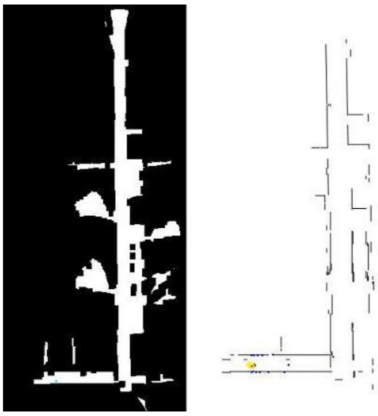

Figure 7: Grid map and line map.

2. Right click in the line map to set your starting position.
3. To set the initial direction, right click on the starting position and drag the cursor towards the direction the robot is facing.
4. Set the goal by left clicking somewhere within the line map.
5. Lean back and enjoy.

4 Important Parameters

Certain parameters are relevant to the program usage. They can be found in Mainprogram.cpp and edited to improve your results.

maxDistance [m] is the radius of the area in which the laser measurements are used, with the robot in the origin. Higher values would give the robot the possibility to update objects far away in the map, but would also decrease the speed of the program. 10 meter has been used throughout the project but the distance can be larger if the pose uncertainty is low and increased computations is not an issue.

gridsize [m] (used in the OccupancyGrid constructor) defines the size of each grid.

The robot always defines origin as the position where the robot begins when the data acquisition for the offline-slam algorithm starts. Negative robot positions will then occur if the robot returns to its start position and goes in the opposite direction relative to the initial heading. This will cause problems in the occupancy gridmapping algorithm since the map is created by multidimensional vectors where negative elements does not exist, therefore an offset but also a large enough map is needed.

Offset X and offset Y [m] defines an offset to guarantee that the robot never falls outside the grid map.

numOfRowsInMap and numOfColsInMap defines the size of the map and depends mainly on the gridsize.

number_of_particles sets the number of particles. Increasing the number of particles gives the robot a better pose estimate at any given time. On the other hand it is more computationally demanding meaning that the SLAM algorithm cannot estimate the robot pose as often. Increasing the number of particles can particularly be a good idea if the robot operates in a small environment.

segment_size decides how many of the 361 laser values that should be considered in the calculations. Increasing this parameter yields a worse localization of the robot but decreases computation of the SLAM algorithm. As a rule of thumb the segment_size should not be larger than 40 or smaller than 20.

uncertaintyThreshold is the variable for the likelihood that the map will be updated when a new object is detected, where 1 corresponds to update every detection and 0 to not updating at all. Increasing this parameter gives the robot a better chance of having a good map locally and detect new objects even when it is badly localized although it can make the global map differ more from reality. If it is desirable to detect new objects faster, increase this parameter. Decreasing this parameter is likely to make the robot drive more smoothly, however, since it does not update the map as frequently, it is more likely to get stuck if the environment is dynamic.