Lab Manual

TI2726-B Embedded Software

2016/2017 Robotics part

Delft University of Technology

W.F. Heida and K.G. Langendoen

TU Delft Delft University of Technology

Challenge the future

TI2726-B Embedded Software V1.2 (Thursday 20th April, 2017)

This manual is typesetted with LaTeX in Nimbus Roman 10 pt. The schematic figures were produced with Dia.

Preface

When redesigning the practicals for the Embedded Software course we realized that robots are fun, and smartphones are everywhere. Thus, it did not take long to arrive at the conclusion that we could combine both into an entertaining programming lab. The first idea was to use the smartphone both as a sensing platform and as the computing heart of a robot, but as Android hacking was not a learning objective of the course we looked for an alternative. On the suggestion of Martijn Wisse and Gijs vd Hoorn we moved to a setup where students could do normal Linux-based programming on a laptop with the ROS middleware making the connection to the phone and robot base. After a rough first edition of the lab in 2014 (robots were assembled during the course), we are now ready to try out a polished version with the expectation that most teams should be able to successfully complete the test tracks at the final demonstration session.

Delft, April 2017
Koen Langendoen

The manual

In this manual there will a number of text boxes with different icons. The information in this boxes is not needed for the lab, but the might contain useful information if you run into problems.

Be aware!
This type of box warns for frequently occurring mistakes and problems.

Hint!
This type of box contains a hint that might be useful when solving an exercise.

Background information!
This type of box gives more information about the technologies that are used in this exercise.

Contents

Chapter 1

Introduction

The purpose of this lab is to become familiar with programming in an Embedded Systems environment, which comes with certain limitations. Furthermore, you will have to work with the Robot Operating System (ROS) [1], which is a middleware that facilitates connecting different systems to each other without having to worry too much about the differences in technologies and protocols between these systems. The system that will be built is a line follower that uses a camera instead of light or IR sensors. This system consists of three components, each with a different role:

- **Smartphone:** the camera of the smartphone that is mounted on the robot makes images of the floor in front of the robot where the line should be detected.
- **Laptop:** the laptop will run the ROS core and perform line detection on the images of the smartphone.
- **Robot:** the robot has to follow the line on the ground.

In Fig. 1.1 a block diagram of the system is shown along with the communication techniques that will be used for communication between the different components. These communication techniques will be hidden from the user by the publish-subscribe model that is used by ROS.

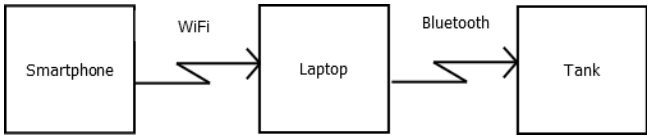

Figure 1.1: Block diagram of the system.

1.1 Timeline

This lab consists of several assignments, which may or may not include a deliverable.

The reports should include the group number and the names and student numbers of the group members on the front page. Furthermore, the report should be in **PDF** format. Reports in any other format will be rejected. All submitted source code files

Table 1.1: The time line for the second part of the lab.

Week	Assignment	Deliverable
4	Requirements & Set up	Report of 1 A4
5	ROS tutorials	none
6	Arduino	none
7	Line follower	none
8	Integration	Final report of 3-4 A4

have to include the header in Listing 1.1. Furthermore, the source code should include comments that indicate the functionality of the code.

Listing 1.1: The header for all source files.

```
/**
 * Group number: x
 * Student 1:
 * name, student number
 * Student 2:
 * name, student number
 */
```

1.2 Passing Criteria

The criteria for passing the lab are listed below. Every criterion has a letter attached to it which is the category it belongs to. For passing the lab you will have to meet the criteria in category C. If you meet category B and/or A you will receive a higher grade.

C The robot can drive around based on `geometry_msgs/Twist` messages published on the topic `cmd_vel`.

C The robot will stop when an object of considerable size is placed in front of it.

B The robot is able to follow the line in test tracks 1-4 (see Fig. 1.2).

A The timers in the Arduino sketch are set via the AVR timing registers and not with the Arduino libraries.

A The speed is adjusted based on the distance to the robot in front when several robots drive on the same test track.

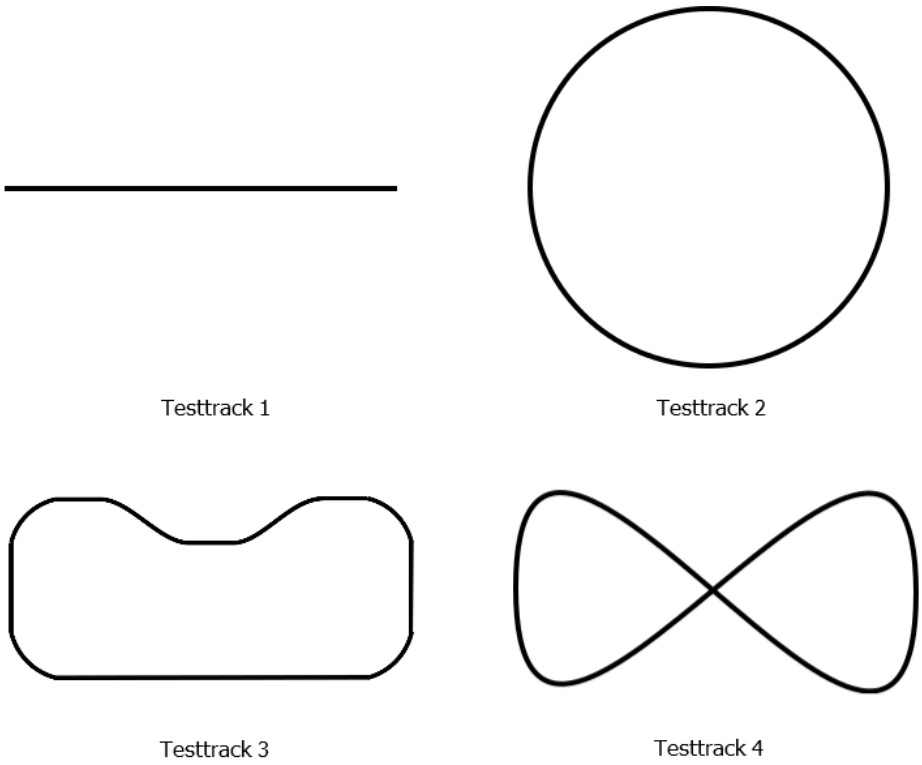

Figure 1.2: The test tracks

Chapter 2

Set up of environment

During this lab we will use ROS, which is only available on Linux. Ubuntu is the Linux distribution that has the most support by the ROS community and for which binary packages are available. It is therefore recommended that you use Ubuntu during this lab. If you are already have another Linux distribution installed on your computer, you can try to install ROS from source, but support for Linux related problems by the teaching assistants will be limited. If you use another version of Linux than Ubuntu you can continue to Section 2.3.

It is recommend that you run the lab on your own laptop, because we will not use the TU network for the WiFi connection, so it will be easier to set up a connection with the lab's network from your own laptop. There are two options for running Ubuntu on your laptop: dual boot (preferred) or via a Virtual Machine (VM). Information about setting up dual boot on your laptop can be found in Section 2.1. Information about the VM can be found in Section 2.2.

Concurrency
Setting up the environment for the lab will include a lot of waiting time. In order to save time, start with setting up the environment and work on the requirements while you are waiting (see Section 3.1).

2.1 Ubuntu Dual Boot

The preferred option for running Ubuntu is the dual boot option, because this will not cause performance problems and you will be able to use all hardware modules of your laptop properly. Installing Ubuntu on a machine that already runs Windows or Mac is relatively simple. You only need to make sure that you have about 50 GB of free disk space on your laptop. For setting up a dual boot, follow the steps below.

Ubuntu version
During this lab we will use the indigo distribution of ROS, which runs on Ubuntu 14.04 LTS. Do not use another version of Ubuntu or ROS, because this might cause problems.

1. Create a back-up on an external drive of your original OS and data.

2. Check if you have to remove or resize a partition in order to be able to install Ubuntu [1]. Make sure that you have a backup of this partition.

3. Free space on the disk for the Ubuntu installation. This can also be done in the Ubuntu installer later on. In Windows the Windows Disk Management can be used for shrinking partitions. If you have to shrink your Windows boot partition, this might cause problems, because of unmoveable files. Solutions for moving these files can be found here.

4. Download the right image of Ubuntu 14.04 LTS from `http://www.ubuntu.com/download/desktop`

5. Create a bootable USB drive with the downloaded image. You can use Linux Live USB creator under Windows for this.

6. Boot your laptop from the USB drive. If the laptop uses HP UEFI, you might have to open and close the BIOS options, before you see the USB boot option.

7. Go through the installation procedure until you are asked "How do you want to partition the disk?". Use the option "'Install Ubuntu alongside Windows" if you have Windows on your laptop.

8. Partition the Ubuntu installation according to the scheme in Table 2.1.

9. Finish the installation.

10. Reboot your laptop to your original OS.

11. Use EasyBCD to create a second entry in your original bootloader.

Table 2.1: The partition scheme for the Ubuntu installation.

Partition	Size
/boot	512 MB
/root	30 GB
/home	10 GB
/swap	8 GB (should be equal to amount of RAM)

Boot & Swap partition
If you install Ubuntu on the same disk as your original OS, place the boot partition and swap partition at the end of your disk if you want to continue to use Ubuntu after this lab. This will give you the possibility of increasing your Ubuntu installation if needed without having to move the boot partition.

Once you have finished installing Ubuntu you can continue to Section 2.3 for the installation of the ROS distribution and packages.

[1] See `http://www.maketecheasier.com/differences-between-mbr-and-gpt/` for more information on partition schemes

2.2 Virtual Machine

If you choose to use a VM be aware that it can have performance issues that can prevent you from running specific software. This will most likely be the case when you try to use graphical intensive applications, but we will not use them very regularly during this lab. It might also occur that you run into problems using hardware components from your laptop. If you want to use a VM, please make sure that your laptop has enough disk space (>20 GB) and enough RAM (>4 GB, preferably 8 GB). Instructions for setting up a Virtual Machine in Windows with VMWare Player can be found here.

2.3 Robot Operating System (ROS)

If you use Ubuntu, you can follow the steps below to install the ROS distribution and packages that are needed for this lab.

ROS version
ROS comes in several versions, called distributions, that can differ quite a lot depending on the version. For this lab we will use the Indigo distribution. Do not use any other distribution, because that might cause compatibility problems with the used packages.

1. Follow the ROS installation guide until Section 1.7. You can use the mirror of the TU Delft instead of the general repository[2].
2. Execute the following command on the command line for installation of the remaining packages:

```
sudo apt-get install \
  ros-indigo-rosserial-arduino \
  ros-indigo-rosserial \
  ros-indigo-teleop-twist-keyboard
```

If you do a source installation (not recommended), you will have to install the following[3]:

1. The ROS Indigo distribution
2. ROS Serial including the Arduino libraries
3. Teleop twist keyboard

[2] See `http://wiki.ros.org/ROS/Installation/UbuntuMirrors`.
[3] See `http://wiki.ros.org/indigo/Installation/Source` for the installation manual.

2.4 Additional tools

Along with the ROS distribution and packages, you will need to install some additional tools that will be used during the lab.

1. Install the Arduino IDE with the command:

   ```
   sudo apt-get update && sudo apt-get install arduino
       arduino-core
   ```

   Perform Section 3.2 of the ROSserial Arduino IDE Setup

2. Install the Bluetooth manager by executing the command below on the command line:

   ```
   sudo apt-get install blueman
   ```

3. (Optional) If you prefer your own editor over the Arduino IDE, you can use the Ino toolkit to compile and upload your sketches to the Arduino.

Chapter 3

Assignments

3.1 Assignment 1: Requirements

When you want to build a system you will have to draw up requirements that the system should comply with. For this assignment we will provide the initial requirements for the system.

1. The robot should be able to follow a line on the ground.
2. The robot should be able to drive at different speeds.
3. The robot should stop when an obstacle is in front of it, so it doesn't collide with the obstacle.
4. For safety, the robot must stop when no new commands have been received from the laptop for some time.
5. The ROS line-follow node should be independent of the robot implementation.

These requirements are a little fuzzy and need to be detailed, even quantified when units like time and lengths are involved. The first assignment is to add quantities to the requirements and justify these numbers. Write the quantified requirements and their justification down on at most one A4. Additional requirements that you deem necessary, can be included too. (The report should be written in English.) Submit your report, labeled with group number and student names, in PDF format on CPM.

3.2 Assignment 2: Introduction to ROS

Once all the software needed for the lab has been installed you can start using ROS. The ROS community provides a nice set of tutorials that introduces all the basic tools needed for working with ROS. Furthermore, it shows how you can create your own publishers, subscribers, services, and clients. Go through all the tutorials at Section 1.1. Make sure that you use the **catkin** version of the tutorials, because that is the build system that ROS Indigo uses. You only have to do the C++ version, because this will be the language that we will use in ROS.

Copy/Paste
You can go through the tutorials by just copy/pasting everything. But, you will need these skills later on during the lab, so it is strongly recommended to read the tutorials carefully and try to understand what you are doing.

3.2.1 ROS workspace

Wstool can be used to maintain this workspace, but for this lab it is not necessary to do so. It advised to use some Version Control System during this lab. Furthermore, it is advised to adhere to the best practice for building up a workspace, which is shown in Fig. 3.1. This tree is set up for use with Git, but you can also use another version control system like SVN. The workspace can also include other non catkin packages, like the Arduino projects. As long as these packages do not contain the file `package.xml` catkin will leave these packages alone.

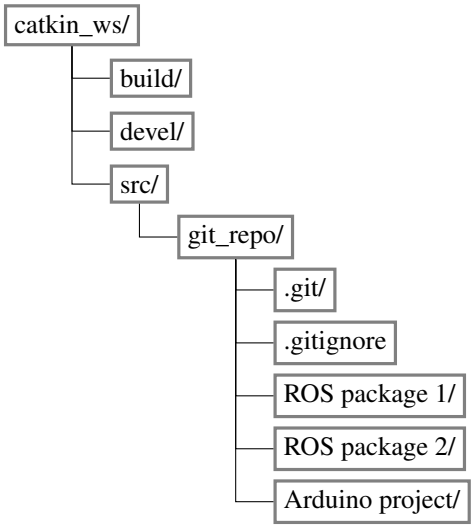

Figure 3.1: The best practice for the structure of a ROS workspace.

3.3 Assignment 3: Using the Arduino with ROS

In this assignment you will start working with the robot and the Arduino Mega ADK.

3.3.1 Robot

The robot that will be used consists of several components:

1. Chassis
2. Tamiya 70097 twin motor gearbox
3. Power Boost 1000 DC/DC boost converter [2]

4. Mini LiPo charger [3]
5. Li-Ion battery pack [4]
6. LCHB-100 H-bridge [5]
7. Arduino Mega ADK [6]
8. HC-05 bluetooth dongle
9. HC-SR04 ultrasonic sensor [7]

Components 3 till 5 are there for providing the power that is needed to drive the robot and Arduino. The battery can be charged by plugging in the adapter to the power plug. There is a switch that can be used to turn the robot on and off. In Table 3.1 the Arduino ports to which the H-bridge, bluetooth dongle and ultrasonic sensor are connected, are shown.

Table 3.1: Arduino ports to which the external components are connected.

Component	Port	Arduino Pin
LCHB-100	1REV	7
	1EN	24
	1FWD	6
	2REV	3
	2EN	25
	2FWD	2
HC-05	TX	18
	RX	19
HC-SR04	Trigger	23
	Echo	22
Yellow LED	-	13

3.3.2 Arduino

The robot is equipped with a micro-controller, the Arduino MEGA ADK, which can be used for controlling the robot. You can use tutorials on how to program the Arduino and tutorials on how to use the Arduino with ROS to get started.

The sketch, i.e. the Arduino program, needs to perform the following functions:

- Control the motors using PWM signals and timers.
- Listen to `geometry_msgs/Twist` messages on the topic `cmd_vel`, and translate them into movements.
- Measure if an object is in front of the robot using the ultrasonic sensor and stop if it is too close.
- Stop when no `Twist` messages are received for some time.

You can follow the steps below for building the sketch, but if you have another idea about how to build up the sketch you are free to do that as well.

- Test if the H-bridge is active low or active high.
- Control the motors via the Serial port with the keys W, A, S and D.
- Add a timer to the sketch to make the motor control interrupt based.
- Add a ROS subscription to the topic `cmd_vel` so you can use `Twist` messages to control the motors.

For the final deliverable the Arduino sketch should be called `robotbase_x.ino` where x is your group number.

PWM
Some timers of the ATMega 2560 also control the PWM outputs. Check the Mega ADK documentation and pin mapping to make sure that you don't introduce conflicts in your sketch.

Arduino libraries
The timers of the ATmega 2560 can be controlled in two ways: via an Arduino library or by directly using the timing registers. For category C you are allowed to use the Arduino libraries. For category A you are only allowed to use the ATmega 2560 timing registers. More information about the timing registers can be found in the data sheet of the ATmega 2560.

Bluetooth serial port
The connection between the laptop and the Arduino can be made via 2 mediums: the USB cable or the bluetooth dongle. These mediums are connected to different Serial ports of the Arduino (USB is on Serial, bluetooth is on Serial1). You have to set the right Serial port in your sketch and can use the example code below. The code overloads the standard settings that are used by `ros_lib` to use the Serial port and baud rate that you specify.

```
class NewHardware : public ArduinoHardware {
  public: NewHardware():ArduinoHardware(&Serial1,
      57600){};
}; ros::NodeHandle_<NewHardware> nh;
```

Note that when you use ROS on the Arduino to connect to the laptop, you cannot use the Serial (or Serial1 if you use the bluetooth) port for anything else in your sketch.

Bluetooth settings
The settings of the bluetooth are 57600 baud, 8n1. The pin code for the Bluetooth is 1234 or 0000.

You can test your final design with the package `ros-indigo-teleop-twist-keyboard`, which enables you to publish `Twist` message on `cmd_vel`. You can launch this application by executing:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

3.4 Assignment 4: Building the line follower

In Section 3.3 the Arduino side of the system was built, so now it is time to build the ROS side on the laptop. The ROS side of the system should perform the following tasks:

- Receive the images from the camera tutorial application.
- Perform line detection on the images from the smartphone.
- Publish `geometry_msgs/Twist` messages on the topic `cmd_vel` based on the detected line.

For the final deliverable the ROS node should be called `linefollower_x.cpp` where x is your group number.

Image processing
For line detection you can use the opencv2 and cv_bridge packages. In ROS indigo opencv is not available by default. Executing "sudo apt-get install ros-indigo-cv-bridge" should install opencv on your system.

Camera tutorial app

For loading the camera tutorial application on a smartphone you need to perform the following steps:

1. Ensure your android phone has internet an go to the course website.
2. Download from the robot lab page the camera app apk file.
3. Install this app on your phone and run it.
4. Use `ifconfig` on your Ubuntu PC to get the IP address for the app. Ensure that the correct IP address and port are used in the app.

IP address
Before the smartphone can be connected to your laptop, the ROS core on your laptop needs to know its own IP address. The environment variable `ROS_IP` should be set to your IP address (*not* your hostname) and the variable `ROS_MASTER_URI` should be set to `http://$ROS_IP:11311/`

If you use a VM you can use two methods to make the roscore available on eduroam:

- Port 11311 should be forwarded from the host to the guest. In Virtual Box this is the preferred method.
- The VM should use a bridged network adapter (only select bluetooth and the wireless adapter). In VMware player this is the preferred method, because port forwarding is not included in VMware Player.

3.5 Assignment 5: System Integration

In the previous assignments the individual components of the system have been developed. These components need to be integrated into one system in this final assignment. The integration might expose some unexpected problems in the implementation of the individual components. Furthermore, you might want to spend some time on optimizing the components to get a better overall performance in the tests.

3.5.1 Final report

The final report (3-4 pages, properly labeled, PDF format) should include the following information:

- The requirements that were submitted earlier (1 page).
- Description of the test to determine if the the H-bridge is active low or high.
- Description of the used timer settings on the Arduino.
- Description of the Twist-message to motor-interface mapping.
- Description of the line detection algorithm.
- Description of the algorithm for generating the Twist messages.

The source files should be deliver in a zip file. The Arduino sketch should be called `robotbase_x.ino` and the ROS node should be called `linefollower_x.cpp` where x is your group number.

Bibliography

[1] Open Source Robotics Foundation. (2014, Oct.) Robot Operating System. [Online]. Available: http://www.ros.org/

[2] Adafruit Industries. (2014, Oct.) PowerBoost 1000 Basic - 5V USB Boost @ 1000mA from 1.8V+. [Online]. Available: http://www.adafruit.com/products/2030

[3] ——. (2014, Oct.) Adafruit Mini Lipo w/Mini-B USB Jack - USB LiIon/LiPoly charger - v1. [Online]. Available: http://www.adafruit.com/products/1905

[4] ——. (2014, Oct.) Lithium Ion Battery Pack - 3.7V 6600mAh. [Online]. Available: http://www.adafruit.com/products/353

[5] Wikipedia. (2014, Oct.) H-bridge. [Online]. Available: http://en.wikipedia.org/wiki/H_bridge

[6] Arduino. (2014, Oct.) Arduino MEGA ADK. [Online]. Available: http://arduino.cc/en/Main/ArduinoBoardMegaADK

[7] Thread Studio. (2014, Oct.) Ultrasonic ranging module: HC-SR04. [Online]. Available: http://www.electroschematics.com/wp-content/uploads/2013/07/HC-SR04-datasheet-version-2.pdf