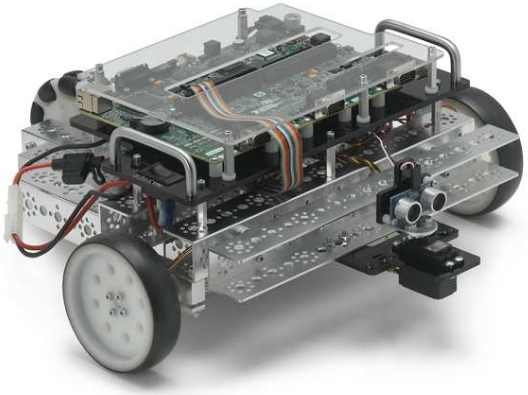
BME Department of Control Engineering and Information Technology – www.iit.bme.hu

Measurement Syllabus

CONTROL OF A MOBILE PLATFORM IN LABVIEW ENVIRONMENT



Author: Bálint Kiss bkiss@iit.bme.hu
Eszter Virágh viragh.eszter@gmail.com

2014 February (translation and extension)
2021 February (on-line add-ons)

Embedded and ambient systems laboratory (VIMIAC09 - EN)

Contents

Chapter 1

Introduction

The objective of this practical measurement session is to study the control of simple mobile platforms. The study includes the understanding of signal processing methods, their algorithmic implementation and measurement based verification of the resulting behavior.

Mobile platforms used during this measurement session are indoor devices equipped with (differential) drives and incremental sensors, colour sensors and ultrasonic distance measurement sensors. One of the platforms is also equipped with a digital compass. On-board control algorithms run on a microcontroller and on an FPGA unit. The control programs are downloaded to the platform using a TCP/IP link and their functioning can be monitored, signals can be saved and visualized on the host PC using the same connection.

The development of the control algorithms happen in a high level, graphical (diagram based) environment on a host PC. The code run on the mobile platform is automatically generated from the high level description. The development environment controls the code generation procedure, its download to the platforms and the monitoring of the running code. Such tools and development environments are also referred to as rapid control prototyping systems. The environment we use during the measurement is National Instruments LabVIEW, which is also widely used in the industry.

The remaining part of this manual is organized as follows. Chapter 2 describes some important features of LabVIEW and Chapter 3 describes the robot platform to be controlled. The main signal processing problem is odometry, that is the determination of the position and orientation of the moving platform based on measured quantities. This problem is described in Chapter 4. The measurement task, the test questions and the requirements for the report to be submitted are specified in Chapter 5.

Students are expected to arrive prepared to the measurement session in order to execute the tasks smoothly and as scheduled. Preparedness can be checked by quizzes or by oral questions at the beginning of the session. Unprepared students may be denied to participate. Students can test using the test questions.

Chapter 2

A short description of LabVIEW

2.1 Introduction

LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) is a complex, integrated development environment. Its first version appeared in 1986 and its primary objective was to support seamless development of data acquisition and instrument control applications.

Since that time, the application fields of the product were considerably widened and they include today instrumentation, manufacturing automation, robotics, etc. Despite the impressive growth of functionalities, the concept remains the same; the basis of development is the so-called virtual instrument (VI). Its user interface is called the front panel and operations on signals (i.e. data) are defined by a block diagram. This natural separation of data manipulation and visualization supports the object oriented and event-driven programming paradigms. VIs can be customized or created by the user, and they can be reused in other VIs. These sub-VIs are analogous to functions in traditional programming languages.

The flow of operations and the operations themselves are defined in a graphical way by creating a block diagram. The resulting programming language (referred to as the G language) is considered as a VHLL[1] dataflow language.

The frontpanel (depicted in Figure 2.1) hosts controls and indicators. Controls (switches, knobs, dialogs, etc.) are signal sources whereas indicators (LEDs, gauges, meters, plots, etc.) serve as signal sinks on the block diagram. Consequently, the VI can be seen as a function whose input parameters are controls and output parameters are indicators. The block diagram also defines the order of execution of operations in the VI. The

[1] Very High Level Language

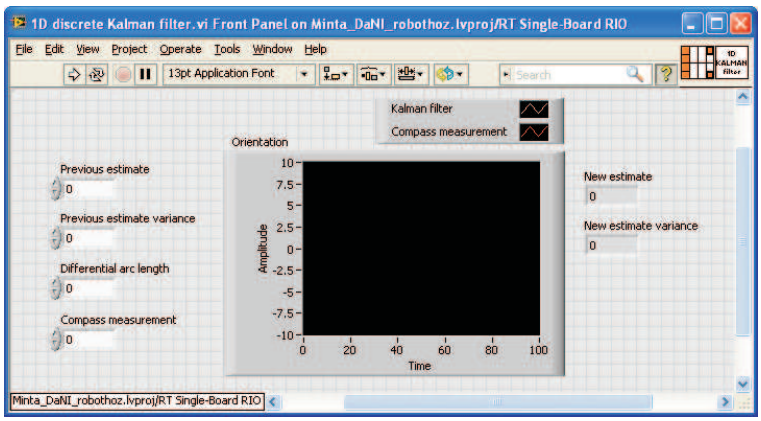


Figure 2.1: Front panel of a VI

execution of any item (which can be a sub-VI too) starts if all its inputs are available and VIs generate their output parameters simultaneously.

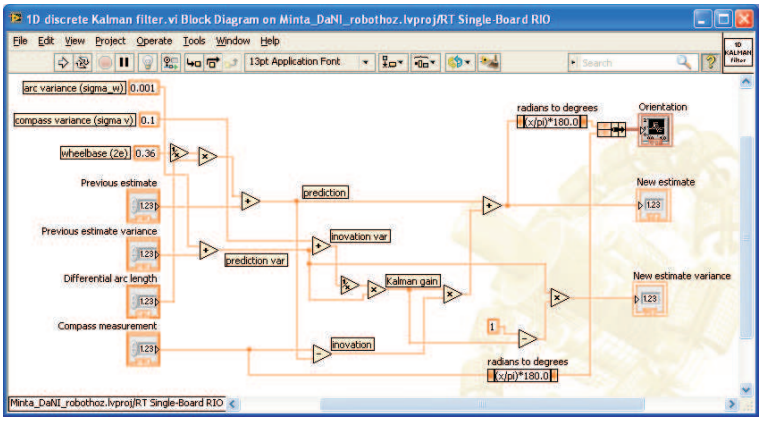


Figure 2.2: VI block diagram

Available items to be placed on the front panel or in a block diagram are organized on separate palettes (see Figure 2.3), which can be opened by a right click.

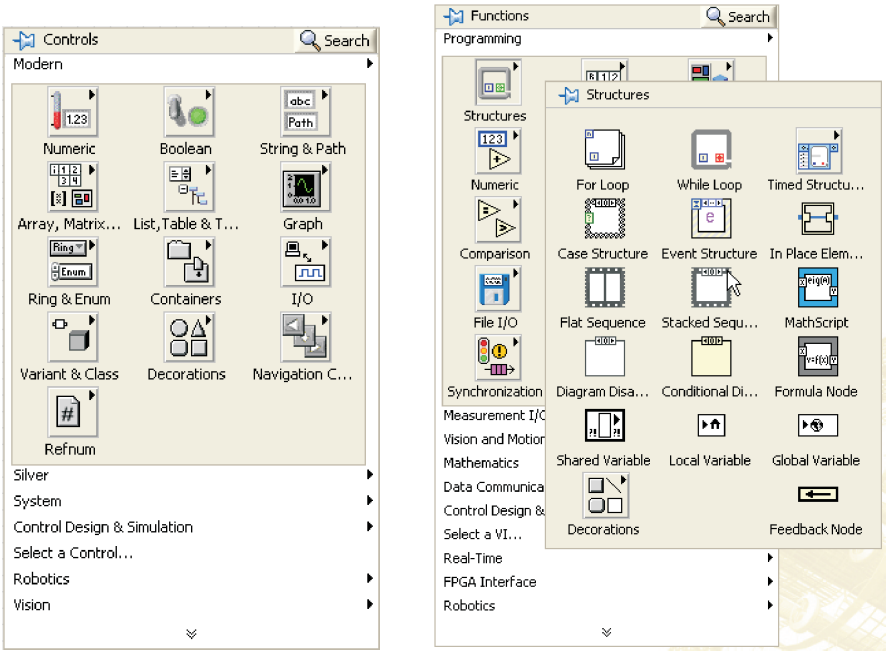


Figure 2.3: *Control* and *Function* palettes for the front panels and block diagrams, respectively

Items on a diagram can be connected using the wiring tool. Connections can be created between compatible terminals, so it is possible to connect an output (indicator) terminal to an input (control) terminal. It is not possible to create connections for the highlighted VI. The datatype for a given connection is specified by the connection colour and wire style. Automatic conversion is available to some extent.

The VIs can be organized in folders or palettes. It is possible to purchase additional palettes to the plain LabVIEW version from National Instruments. The robotics module is such a function library extension.

The execution (or run) of the VI can be initiated from the toolbar, or more precisely with the Play button

of both editing windows. Debugging resources are also available and it is advised to use the help extensively if error message occurs instead of normal run.

2.2 Help, tutorials, examples, on-line resources

LabVIEW has a detailed and extensive help. A very useful tool is the context based help, which can be accessed using Ctrl + H and provides the description of the item in focus. Two other shortcuts are also useful: Ctrl + E toggles the front panel and the block diagram and Ctrl + B removes automatically all broken wires. The environment is quite intuitive otherwise, and it is easy to get acquainted with it.

The product web page (www.ni.com) is rich in tutorials, case studies, and examples. Some resources are available only upon registration. The scope of this syllabus is obviously limited, hence not all features can be detailed. Links included in the document may serve as departure points for more information.

The solution of a lot of tasks is possible through the modification of examples. This is the reason why LabVIEW contains an example browser capable to search a huge on-line solution database. In is important to create copies of the examples first, before the modifications, otherwise one may modify the originals of the examples and damage them.

2.3 Loops

An execution structure from the programming palette will have a special importance, namely the loop and its timed version. The basis of real-time operation of the code is a periodic and deterministic execution of piece of code, where the sensor signals from the outside world are processed and a decision is made to change the signals sent to the actuators.

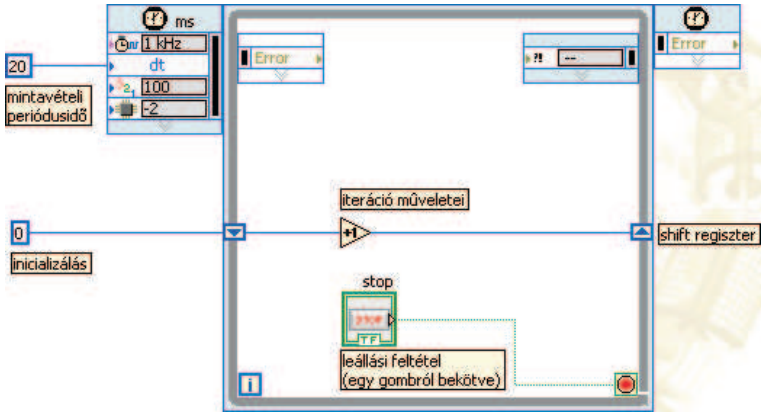


Figure 2.4: Timed loop

Figure 2.4 shows a timed loop. The kernel of the loop is limited by a grey frame, all operations inside the frame are iterated. A special sink inside the loop is responsible for stopping the iteration and it is placed by default in the right bottom corner of the frame. Any expression can provide input to this sink, including a control push button on the front panel. If we use a timed loop, the execution of the kernel depends on time, hence a periodicity can be defined. An additional data structure appears around such a loop, where timing parameters can be set. The period of execution for the loop in Figure 2.4 is 1ms. This means that the kernel of the loop is re-executed every 1ms (even if the actual execution time of the kernel operation is much quicker). Iteration stops if the terminal condition becomes true.

It is possible to use results of the previous iteration during the current execution of the loop. This is made possible by the so-called shift registers. A shift register is a pair of a sink and a source placed at the same height

on the loop frame on its right and left side, respectively. Initial values for such registers (i.e. their values for the first iteration when there is no previous one) should be set on the left side and from outside the loop. It can be easily seen that the loop in Figure 2.4 actually keeps the number of loop execution in the shift register, initialized to 0.

2.4 The LabVIEW project

The LabVIEW program behind a more complex task requires the implementation of several VIs and sub-VIs. Different VIs may run on different processing units of the architecture. It is desirable to organize all these components into a so-called project as shown in Figure 2.5. The project specifies not only the VIs and processing resources where the VIs are executed, but any other hardver elements providing access to physical signals to VIs .

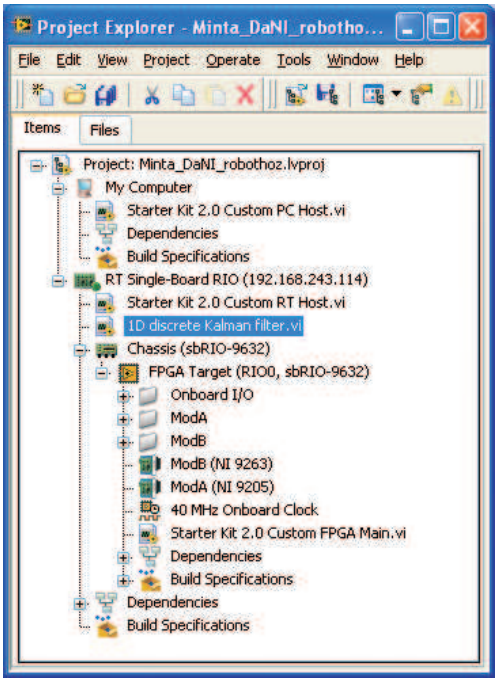


Figure 2.5: LabVIEW project in the project explorer

Students will manipulate and complement some items of such a project during the measurement session. It is important to understand that the computer where the LabVIEW code is developed is different from the computer (or processing resources) on which the code is actually executed. The development of the VIs happen in the LabVIEW environment installed under the Windows operating system, but the actual place of execution of a VI is determined by its place in the project tree. The position of the VI also determines the set of physical signals which can be processed or calculated by the VI directly and then eventually sent over to other VIs. It is therefore important to correctly associate a VI with its running environment.

The actual environment defined by the hardver setup can be visualized using the Measurement & Automation eXplorer (NI Max for short). We can browse the available processing resources (processor boards, FPGAs, etc.), the I/O channels and we can also initiate their reset and restart. Some resources may be available remotely (through the net). In most cases all resources are automatically detected.

VIs, where the place of execution is not Mycomputer, cannot be run directly from LabVIEW, they have to be deployed. Appropriate code has to be generated first which is than downloaded to the execution resources. Two such execution devices will be available during the measurement: an FPGA modul and a Real-time (RT)

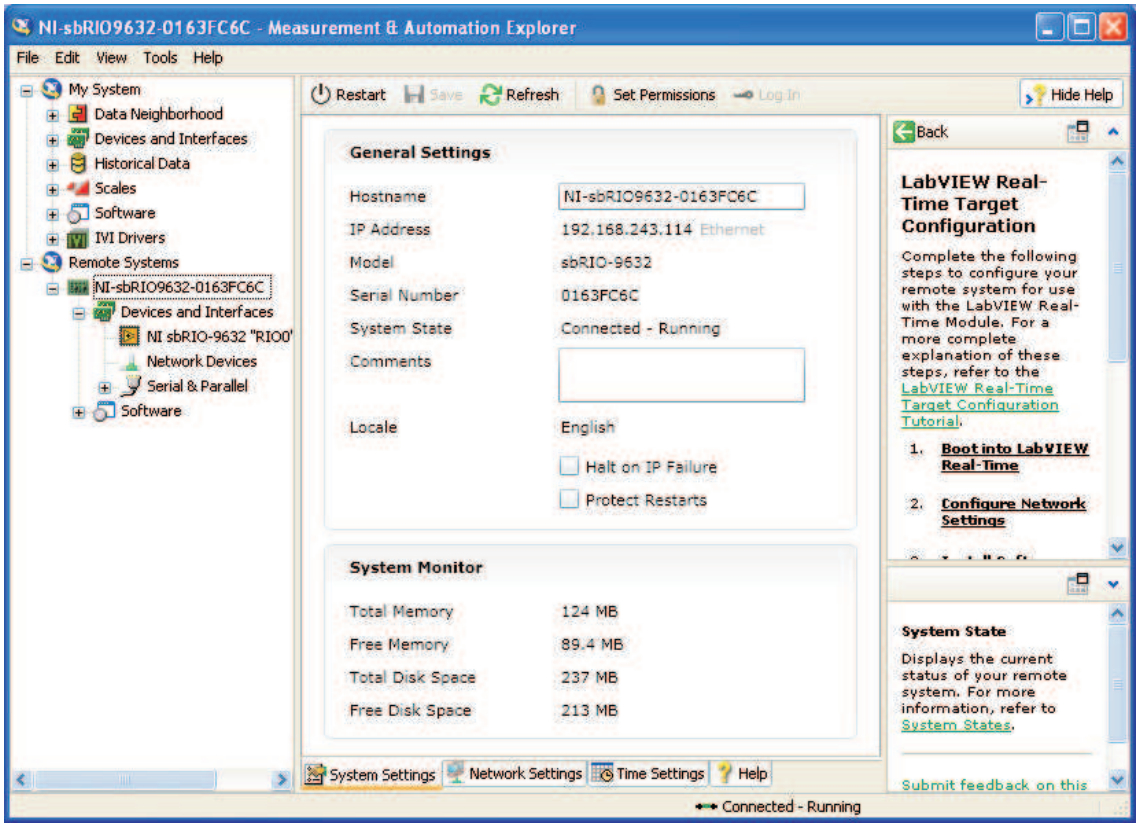


Figure 2.6: Measurement & Automation Explorer

modul. The procedure of code generation and code download are commonly referred to as deployment. The deployment is automatic and generates a highly optimal code. Only one VI can be deployed on the FPGA which must contain all operations to be executed on that resource. The code generation and deployment to FPGA is much slower (typically 30-40 minutes) compared to the RT modul. The price to pay for shorter deployment time is a longer execution time for the same operations, hence the operations on FPGA may be executed with a much higher frequency. The VI deployed on FPGA must not be modified during the measurement session.

Deployment is initiated automatically when a VI is asked to be run. The code is re-generated only if the VI has been modified since the last deployment.

The execution place of a VI restricts the available function palette elements one may use to design that VI. VIs deployed on different processing resources can communicate with each other. Since there is a wide range of options to realize communication between VIs and these options are hardware dependent, these details will be clarified in Chapter 3.

Recall that the front panel (user interface) is separated from the block diagram and the code is deployed for the block diagram only. This means that the front panel of any VI can be still visualized on Mycomputer even if the related block diagram is deployed to another hardware. The connection between the deployed code and the front panel is asynchronous and subject to delays. The breakdown of the communication will not interrupt the execution of the deployed code on the remote hardware.

2.5 Reuse of VIs

Modular programming requires reuse of code. This is realized in most programming languages by the definition of functions. VIs are analogous to functions in the LabVIEW environment. A VI can be reused in the block

diagram of another VI (that is actually a function call) if its input and output parameters are defined. Once this is done, the VI can be instanciated in other diagrams and it can be connected to other VIs. In order to do this, one has to define a terminal panel to the VI where we place input and output terminals of from the front panel.

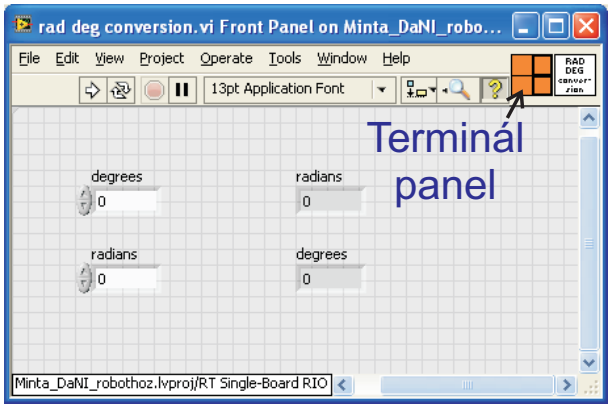


Figure 2.7: Terminal panel in the editing window of front panel

The terminal panel can be edited on the front panel where one can associate its parts to controls and indicators. The number of terminals of the VI can be modified and a mask can be also created. A terminal panel is shown in Figure 2.7.

Chapter 3

The DaNI robot platform

DaNI is an indoor mobile platform developed around National Instruments hardware and software components. Main features of the platform are:

1. Differential drive: two independently actuated wheel and a third omnidirectional wheel (Swedish wheel). The angular velocity of each actuated wheel is controlled by a low level PID controller.
2. Ultrasonic distance measurement sensor (can be rotated around its vertical axis).
3. Incremental encoders on the motor axes (400 counts per revolution).
4. Digital compass (type: Devantech CMPS03, this is an add-on realized by students).
5. On-board camera (not used for this measurement exercise).
6. Single-board embedded computer (National Instruments single board RIO - sbRIO-9632), with a MPC5200 processing unit and a Spartan 3 Xilinx FPGA, digital and analogous I/O lines, and TCP/IP link.

There are two important switches on DaNI and they interrupt the power supply of two different modules. The MASTER switch turns on/off the embedded single board computer (RIO on top). Motors can be turned on separately using the MOTOR switch. It is advised to keep the motors switched off whenever the robot is not expected to move which ensures longer battery cycles.

A simulation environment and a large set of VIs (i.e. built-in functions) support directly the use of the DaNI platform with LabVIEW. The VIs execute low level motion control and sensor signal processing tasks but many of them support higher level operations such as the creation of maps of the environment, navigation, and path planning. Globally, these VIs help smoothly the realization of the sense - think - act loop, leading to autonomous behavior.

The schematic of the connections between the processor board, the power stage and the sensors is given in Figure 3.1. The programming of the platform happens in the LabVIEW environment. Some VIs are executed on the FPGA of the single board RIO device. These VIs are responsible for low level task, they receive the sensor signals. These VIs (always subVIs of a single VI deployed) are referred to as FPGA VIs.

Another set of functions is executed in real time on the MPC5200 microcontroller, we call them RT VIs. Their code is generated and downloaded (deployed) automatically to sbRIO. RT VIs can seamlessly communicate with the FPGA VIs. Front panels of RT and FPGA VIs can be visualized on the PC during execution thanks to the TCP/IP connection between the sbRIO and the PC.

It is important that all control loops requiring real time and deterministic behavior (such as speed control of motor axes) run on FPGA or MPC5200, otherwise variable and undeterministic delays on the TCP/IP link or in the Windows system may destabilize the system.

We will overview in the sequel the sensors and actuators of the robot as well as the RT components of the robot control architecture which will be used during the execution of the measurement tasks.

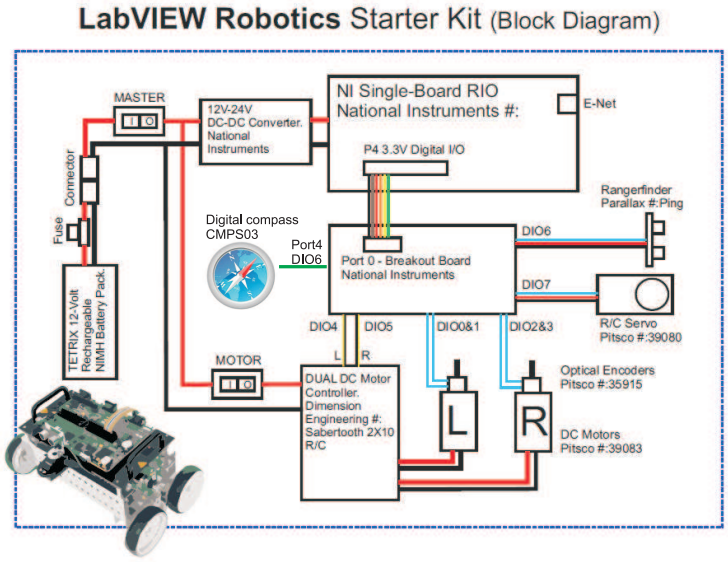


Figure 3.1: Architecture of the DaNI mobile platform with sensors, motors and connections

3.1 Sensors and actuators

The mechanical parts of the robot are manufactured by a company called Pitsco and the default configuration contains most of the sensors and actuators enumerated bellow. The sbRIO card offers a wide range of additional I/O lines (digital and AD/DA channels, serial ports, Ethernet) where more sensors and actuators can be interfaced if desired. Such customization has been already carried out on the platform used during the measurement session as a digital compass and a camera were added to the original configuration.

Figure 3.2: Optical encoders mounted on motor axes (400 counts/revolution)

The rotation angle of both motor axes can be measured with high accuracy, thanks to optical incremental encoders shown in Figure 3.2. Each encoder have two signal channels and it sends two impulse sequences during rotation, dephased to each other. Counters are driven by these signals and they count edges. Direction of rotation can be determined by observing the order of subsequent rasing and falling edges. For the encoders mounted on DaNI, 400 counter increments belongs to a full rotation. Since the diameter of the wheels on DaNI is 0.1 meter, one increment represents less than a 0.8mm ($0,1 \cdot \frac{\pi}{400} = 7,8530 \cdot 10^{-4}$ meter) long displacement of the contact point on the ground or on the surface of the wheel.

Thanks to these incremental encoders, the displacement length of contact points for each wheel can be read

out with a predefined sampling time (if one can suppose that the wheels do not slip). Using these displacement values as inputs, an approximation based on the kinematic model of the robot can be used to update the position and orientation of the of the platform. Related calculations are defined in Chapter 4.

Obstacles in the forward motion direction of the robot can be detected using a vertically rotating ultrasonic distance measurement sensor (see Figure 3.3). Rotation is achieved using a stepper motor drive. The sensor emits signal packets in the ultrasonic range detects reflected packets, and measures the time of flight which is proportional to the distance of the object from where the reflection arrives back. An impulse triggers the dispatch of a 40 kHz packet flying with approx. 350 m/s in the air and (part of) it is reflected to the sensor if an obstacle is present in range. The output of the sensor shows an impulse during the process such that its width is proportional to the time of flight, hence with the distance of the obstacle as its edge rises as the packet is sent and falls when reflection is detected. The range of the sensor is approx. 3 meters, no reflection is detected from obstacles beyond that distance.

Figure 3.3: The Parallax Ping))) ultrasonic distance measurement sensor

The orientation of the distance measurement sensor can be changed w.r.t. the robot using the stepper motor in the $\pm 90^{\circ}$ range. This allows the periodic scan of the area in front of the robot.

A digital compass module is mounted on the robot in order to have an absolute measurement for the orientation of the platform. The senor module is a Devantech CMPS03 device and it is not part of the default setup. The senor detects the magnetic field of the Earth in the plane of the sensor so it is not tilt compensated. This is not an issue as the robot moves on a horizontal flat plane. This sensor also outputs pulses such that the width determines the orientation angle.

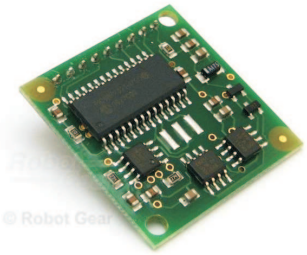


Figure 3.4: Devantech CMPS03 digital compass module

The platform is equipped with 12 Volts DC motors, depicted in Figure 3.5. The power amplifier stage includes current control loops and the on-board computer realizes cascade velocity control on the motor axes thanks to the VI running on the FPGA. The velocity control loops are entirely independent of each other. The feedback velocity signal is determined using the incremental encoders. The maximal nominal angular velocity of the motor axes is 154 rpm. Based on the wheel diameter this is equivalent to a 0,8 $\frac{m}{s}$ maximal nominal speed of the platform.

Figure 3.5: DC motor in the drive

3.2 Real-time controller software

All functions necessary to the real-time control of the robot run on the on-board embedded computer (also referred to as the target). Two processing units are available in this architecture and they communicate with each other: a Xilinx Spartan FPGA circuit and an MPC5200 microcontroller. These items appear in the template project available for the measurement. The architecture is also shown in Figure 3.6.

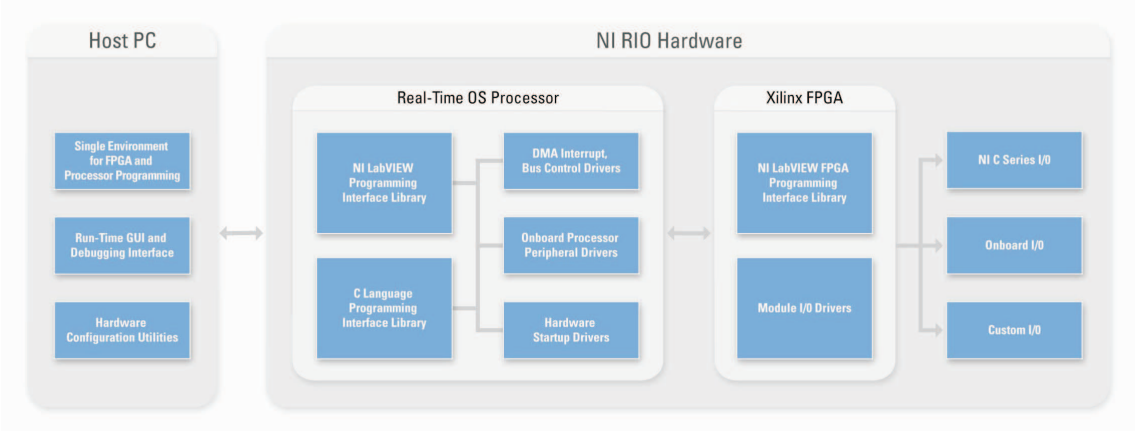


Figure 3.6: Devices of the sbRIO and connections with the PC host running the LabVIEW environment

The VI running on the FPGA processes first the signals of all sensors and it implements both PID velocity control loops of the wheel axes. There is no need to modify this VI during the measurement session. All indicators and controls of this FPGA VI can be accessed by other VIs, the only condition is the existence of a physical communication link between the FPGA and the processing unit where the other VI is executed. The communication is real-time for the VIs deployed on the microcontroller of the sbRIO.

The access to the indicators and controls of the FPGA VI is indirect and happens through a so-called FPGA reference VI. Its data structure allows read and write operations for RT VIs. Separate VIs are used for reading indicators and writing controls and they can included in the timed control loop of the RT VI.

The main loop of the template RT VI, which will be completed according to the tasks during the measurement session, is depicted in Figure 3.8. One can easily identify the FPGA reference VI outside the loop. VIs executing read and write operations inside the loop are also easy to find. In our case, the read VI inside the loops gets the actual value of the compass. The processing of the signals of other sensors and the sending of the reference velocity signals to the motors are realized by higher level VIs. These VIs obviously contain the FPGA read/write VIs, but they also include error handling and signal processing functions. Most of these higher level VIs belong to the Robotics module of LabVIEW.

The *Arc length* VI processes the signals from the incremental encoders. The outputs of this VI are the total length of the displacement of the contact points of the respective wheels. The diagram of this VI is depicted in

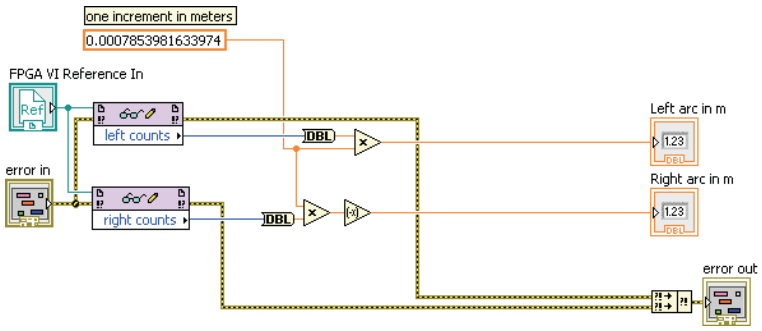


Figure 3.7: Block diagram of the *Arc length* VI

Figure 3.7. If one would like to calculate the displacement during one sample interval, the use of shift registers is advised in the main timed control loop.

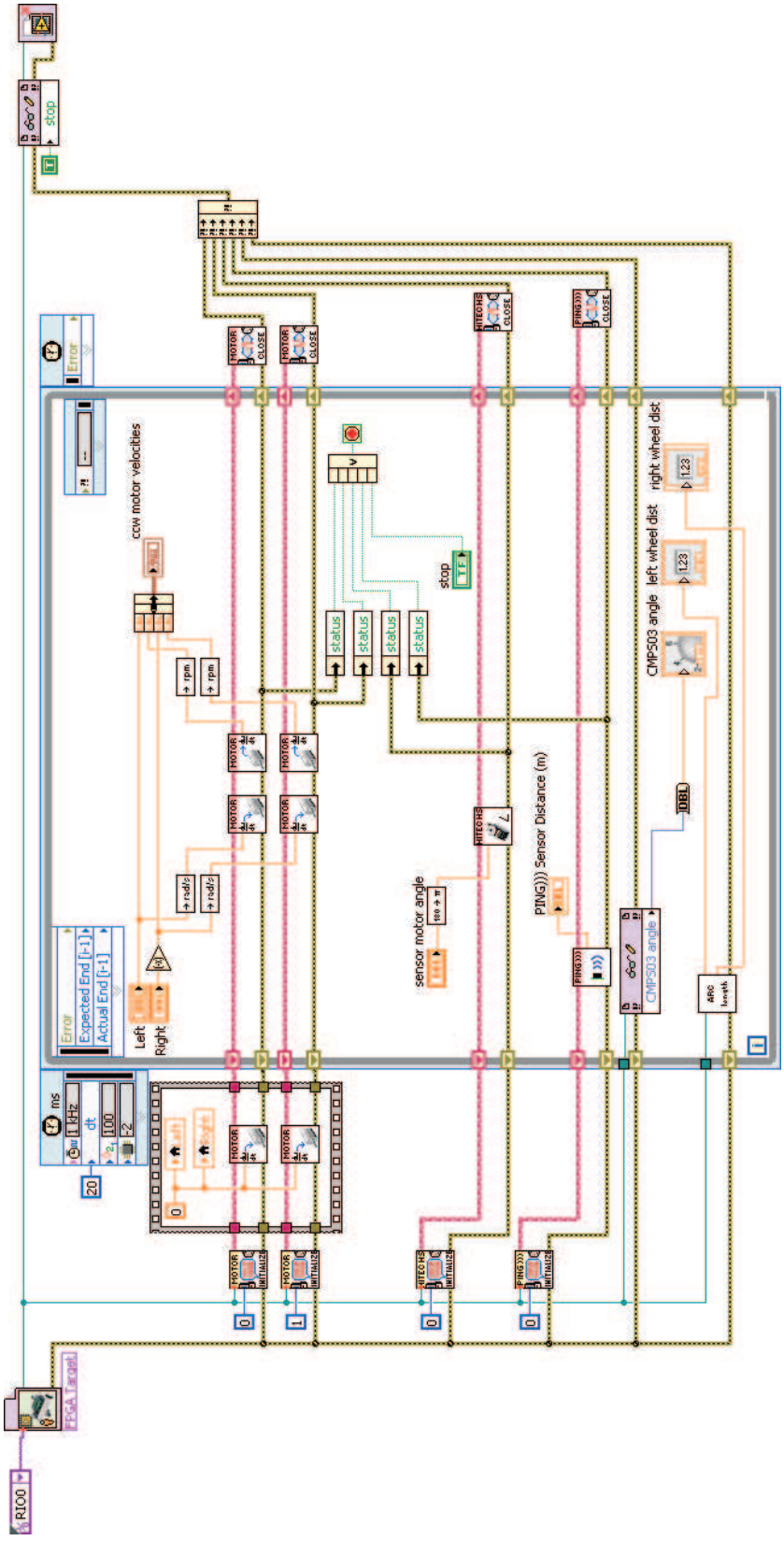


Figure 3.8: Main timed loop of the RT VI

Chapter 4

Odometry and Kalman-filters

Most navigation tasks require the knowledge of the instantaneous position and orientation (or its variation) of the robot platform based on the information provided by the sensors. This measurement problem is commonly referred to as the odometric problem and the methods for its solution are provided by odometry. (The word odometry is a combination of two greek words: *hodos* (travel) and *metron* (measurement).

Another notion connected to odometry is *dead reckoning* - *DR*, where the solution of the odometric problem (i.e. the calculation of the robot position and orientation) is provided incrementally, based on the time elapsed and on the information on the velocity state of the mobile platform. Results provided by DR methods accumulate measurement errors therefore they are used in conjunction with other (less often available) sensor information such GPS, DGPS and digital compass.

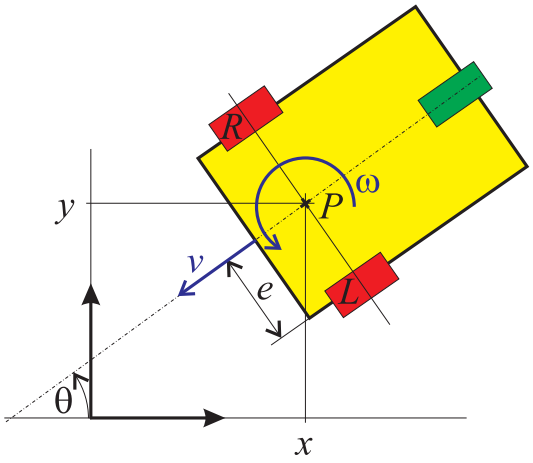


Figure 4.1: Convention used to describe robot position and orientation

A simple solution for the odometry problem is to measure the arc lengths of contact points of the wheels with the ground and reconstruct the change of orientation and position. Calculations behind this method are based on the differential (or kinematic) model of the robot platform.

4.1 Kinematic robot model

The robot is a rigid body with a three dimensional configuration manifold. A configuration on this manifold can be specified by three parameters: x, y gives the coordinates of the axis midpoint and the orientation is specified by the angle θ. These parameters are also explained in Figure 4.1. The model will have one geometric parameter, denoted by e, which represents the distances of the wheels from the central symmetry axis.

Let us denote by v the common mode longitudinal velocity of the two wheels (this velocity has a sign, positive values represent forward motion as in the Figure) and let us denote by ω the (yaw) angular velocity (positive values represent the direction as in the Figure). If one supposes that wheels do not slide sideways, the following kinematic relations describe the motion of the robot:

$$\dot{x} = v\cos\theta \tag{4.1}$$
$$\dot{y} = v\sin\theta \tag{4.2}$$
$$\dot{\theta} = \omega \tag{4.3}$$

Let us denote by v_L (resp. v_R) the velocity of the contact point of the left (resp. right) wheel w.r.t. the vehicle. Again, if one supposes no sliding at the wheels, these contact point velocities allow to calculate v and ω.

$$v = \frac{v_R + v_L}{2} \tag{4.4}$$
$$\omega = \frac{v_R - v_L}{2e} \tag{4.5}$$

Please recall that both v and ω can be determined from the geometric parameter of the robot (e) and the measured angular velocities of the wheels.

4.2 Kinematics based odometry - dead reckoning

Figure 4.2 shows a sample trajectory of the robot. Using the data read out periodically from the counters driven by the incremental encoders, the time functions of the arc lengths of contact points are available. It is easy to see that the robot can perform an instantaneous straight line motion only if the rate of change are the same for both arc lengths and a pure instantaneous rotation happens if the rate of change of the arc lengths differ only in a sign. These elementary motions do not commute, the final position is different if their order is interchanged.

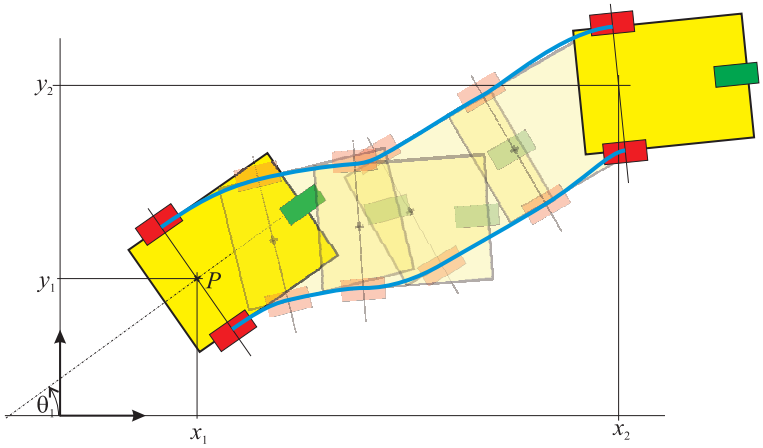


Figure 4.2: Arc lengths of contact points along a trajectory

We can reconstruct the motion of the robot from sequence of elementary (combined straight line and rotation) displacement. If the displacement is sufficiently short, the curvature or radius is considered as constant. Such a small displacement is given in Figure 4.3. Arc lengths of contact points depend on motor velocities which are measured. Denote these arc lengths (right and left, respectively) by δ_R and δ_L. In order to calculate the coordinates of P_{i+1} and the orientation θ_{i+1}, one has to calculate first ΔL and $\Delta\theta$. Let us denote the arc length of the point P by $\delta_P = \frac{\delta_R+\delta_L}{2}$.

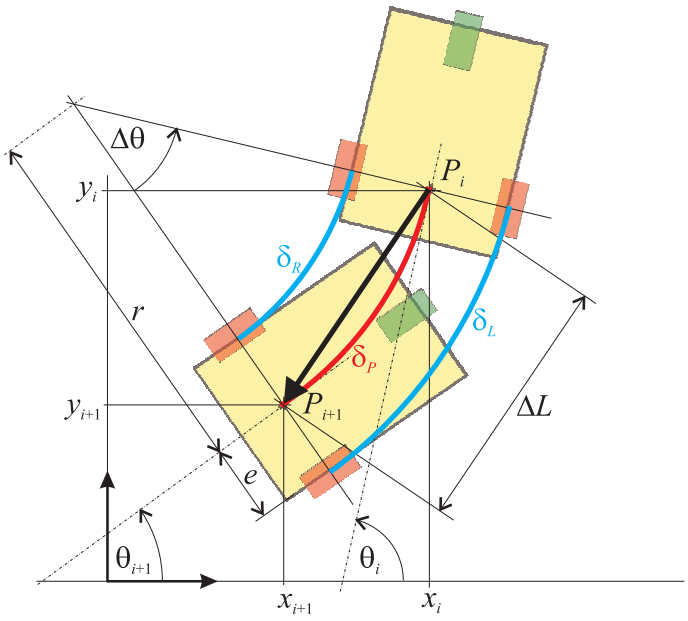


Figure 4.3: Displacement with constant curvature

Since $\delta_L = \Delta\theta(r-e)$ and $\delta_R = \Delta\theta(r+e)$, one can easily obtain

$$\Delta\theta = \frac{\delta_R - \delta_L}{2e}, \tag{4.6}$$

where r is the instantaneous radius of rotation. Denote by $\rho = \frac{1}{r}$ the instantaneous curvature. Their expressions read

$$r = e\frac{\delta_R + \delta_L}{\delta_R - \delta_L} \qquad \rho = \frac{1}{e}\frac{\delta_R - \delta_L}{\delta_R + \delta_L}, \tag{4.7}$$

and one may choose the one which is numerically more stable (bigger denominator). The displacement along the arc can be approximated by the displacement along the string, hence $\Delta L = 2r\sin\frac{\Delta\theta}{2}$. Since the sine of an angle can be approximated by the angle itself for small angles, we get $\sin\alpha \sim \alpha$, hence $\Delta L \sim \delta_P$.

After some elementary further calculations, the above discussion results

$$x_{i+1} = x_i + \Delta L\cos\left(\theta_i + \frac{\Delta\theta}{2}\right) \tag{4.8}$$

$$y_{i+1} = y_i + \Delta L\sin\left(\theta_i + \frac{\Delta\theta}{2}\right) \tag{4.9}$$

$$\theta_{i+1} = \theta_i + \Delta\theta = \theta_i + \frac{\delta_R - \delta_L}{2e}, \tag{4.10}$$

These simple formulae can be implemented in any programming language including LabView.

4.3 Orientation calculation using compass data

Dead reckoning will accumulate measurement errors in incremental calculations of the position and orientation if these computations are never corrected by absolute (eventually even inaccurate) information on position and on orientation. Error accumulation may be caused by inaccurate knowledge of a parameter like e, the sliding of wheels or inaccurate knowledge of wheel diameters which allows calculating arc lengths δ_L and δ_R based on rotation angles of wheel axes.

Table 4.1: Notations in the Kalman-filter

Notation	Explanation
P	covariance matrix of the estimation (variance here)
$\tilde{z}_k$	innovation at step k
S_k	covariance matrix (variance here) of the innovation is step k
K_k	(optimal) Kalman gain

The robot is not equipped with absolute position sensors such as (D)GPS or with external and fix camera system, but the digital compass (Figure 3.4) detecting the magnetic field of the Earth provides absolute orientation information. The accuracy of the measurement of arc length is higher (thanks to the high resolution of the incremental encoders which we use according to (4.8)-(4.10)) than the accuracy of the orientation measurement.

Let us now consider separately the estimation problem of orientation. Let u_k denote the actual differential arc length defined as $u_k = \delta_R - \delta_L$. Let us denote by ζ_k the orientation at the sample k and by z_k a the observation obtained from the digital compass for the same sample. We reuse the odometric relation (4.10) and we take into consideration the uncertainty of both the arc length and the absolute orientation measurements. The resulting system reads

$$\zeta_{i+1} = \zeta_i + \frac{1}{2e}u_k + w_k \tag{4.11}$$

$$z_k = \zeta_k + v_k, \tag{4.12}$$

where w_k and v_k are the realizations of the uncertainty in arc length and in compass measurements, respectively. Based on their resolution, the largest error of incremental encoders is 0.8mm as discussed earlier in Chapter 3. The error distribution can be approximated by a normal distribution so that $3\sigma = 0,8$ mm (a usual practice), hence w_k is the realization of a white noise with zero expected value and $\sigma_w = \frac{0.8}{3}10^{-3}$ standard deviation. Based on the compass module datasheet, the uncertainty of its measurement is $5^\circ = 0.087266$rad, thus following a similar reasoning as for the arc length, v_k is also the realization of a white noise with zero expected value and $\sigma_v = 0.3$ standard deviation, hence

$$w_k \sim N(0, \sigma_w^2) \qquad v_k \sim N(0, \sigma_v^2) \tag{4.13}$$

Our task is to provide estimates of ζ_k based on the measured values of u_k and z_k, subject to uncertainties. The method described bellow can be applied to MIMO system and with other distributions.

The computations we will use are commonly referred to as Kalman-filter[1] which is actually a dynamical system. This filter does not only provide estimates of ζ_k, but it also calculates the uncertainty of these estimates. We introduce $\hat{\zeta}$ for the notation of the estimate of ζ. We will distinguish the *a priori* estimate before the measurement z_k, (denoted by $\hat{\zeta}_{k|k-1}$) and the *a posteriori* estimate, calculated after the measurement z_k (denoted by $\hat{\zeta}_{k|k}$). We use a similar indexing for other quantities too. All quantities involved in the filter calculations are explained in Table 4.1. As the task here is to estimate a scalar quantity (i.e. orientation), the covariance matrices are simply variance values.

The Kalman-filter is unbiased, hence the expected value of the error of both *a priori* and *a posteriori* estimations vanish. The following recursive calculations have to be executed during the operation of the filter so that the form of the uncertain system (4.11) and the parameters (4.13) of the distributions are taken into consideration:

1. Prediction phase (*a priori* computations before the compass measurement)

[1]Kalman Rudolf Emil (1930-): American mathematician and electrical engineer born in Budapest, honorary member of the Hungarian Academy of Sciences

(a) $\hat{\zeta}_{k|k-1} = \hat{\zeta}_{k-1|k-1} + \frac{1}{2e}u_k$ - *a priori* prediction based solely on the arc length measurement.

(b) $P_{k|k-1} = P_{k-1|k-1} + \sigma_w^2$ - uncertainty of the *a priori* prediction based on (4.11) which increases the variance with the uncertainty of the arc length measurement.

2. Update phase (*a posteriori* computation after the compass measurement value becomes available)

(a) $\tilde{z}_k = z_k - \hat{\zeta}_{k|k-1}$ - error of the predication calculated based on the arc length (compared to the compass measurement), also referred to as the innovation

(b) $S_k = P_{k|k-1} + \sigma_v^2$ - uncertainty of the innovation (variance)

(c) $K_k = P_{k|k-1}S_k^{-1}$ - Kalman gain. This will be the weight of the innovation in the calculation of the *a posteriori* estimation.

(d) $\hat{\zeta}_{k|k} = \hat{\zeta}_{k|k-1} + K_k\tilde{z}_k$ - the *a posteriori* orientation estimate is the weighted sum of the *a priori* estimate and the innovation.

(e) $P_{k|k} = (1 - K_k)P_{k|k-1}$ - uncertainty (variance) of the *a posteriori* estimation, it is decreased compared to the uncertainty of the *a priori* estimate.

Initial conditions have to be set for $\hat{\zeta}_{0|0}$ and for its uncertainty $P_{0|0}$ in order to start the iteration correctly. In our case, we may suppose that the inital orientation is perfectly known, so $\hat{\zeta}_{0|0} = P_{0|0} = 0$.

The above iteration can be implemented in any programming language, inlcuding LabVIEW and the result provided can be further used in other odometric calculations such as (4.8)-(4.10).

Chapter 5

Quiz questions, tasks and measurement documentation

5.1 Quiz questions

You may test your knowledge and preparedness to the measurement prior to its start using these question. You may expect similar questions in the entry quiz at the beginning of the measurement session.

1. Describe the main features of the National Instruments DaNI robot platform including its actuators and on-board sensors!
2. Enumerate the processing units of the embedded on-board computer of the DaNI platform. How it is possible to generate real-time operations with VIs developed in the LabVIEW environment on a PC.
3. Give the variables of the configuration manifold of a mobile robot evolving on flat ground and illustrate their meaning.
4. Give the kinematic equations of motions of a mobiel robot evolving on flat ground and explain the meaning of variables and parameters.
5. Illustrate and give the fundamental odometric equations for a differentially driven robot platform if one can measure the arc lengths of contact points of the wheels!
6. Give the system equation that is the basis of the Kalman-filter on the orientation. Define the state, the input and measured quantity.
7. Enumerate the computational steps of the Kalman-filter algorithm.

5.2 Measurement exercises

The following tasks have to be carried out during the measurement session.

1. Study the LabVIEW environment and the items of the template project!
2. Implement the algorithm (4.8)-(4.10) and deploy it to the robot!
3. Measure the error of the position and orientation estimates obtained using dead reckoning after a motion sequence.
4. Increase the diameter of one of the wheels, and measure again the error of position and orientation estimates.

5. Implement and deploy the Kalman-filter for the estimation of the orientation and check if the filter can compensate the wheel diameter errors thanks to the digital compass

6. Use the Kalman-filter and the distance measurement sensor to turn the robot automatically in the direction of the farthermost obstacle.

5.3 Measurement documentation requirements

The measurement documentation handed in by students must contain a title page with

1. title of the measurement,

2. names and Neptun codes of participating students, number of the measurement group,

3. date when the measurement took place.

The measurement documentation describes the solutions for the different measurement tasks, their implementation in LabVIEW, results of the operations and their evaluation.

The measurement documentation should be sent by email to the supervisor of the measurement session in pdf format during the period of 7 days starting with the day of the measurement.

The grade for the measurement is evaluated based on the measurement documentation and on the results of the quiz.