Two-wheeled Unstable Transporter

User's Manual

www.inteco.com.pl

NOTES

SAFETY OF THE EQUIPMENT

The equipment, when used in accordance with the supplied instructions, within the parameter set for its mechanical and electrical performance, should not cause any danger to health or safety if normal engineering applications are observed.

If, in specific cases, circumstances exist in which a potential hazard may be brought about by careless or improper use, these will be pointed out and the necessary precautions emphasised.

Some National Directives require to indicate on our equipment certain warnings that require attention by the user. These have been indicated in the specified way by labels. The meaning of any labels that may be fixed to the equipment instrument are explained in this manual.

Risk of electric shock

PRODUCT IMPROVEMENTS

The Producer reserves a right to improve design and performance of the product without prior notice.

All major changes are incorporated into up-dated editions of manuals and this manual is believed to be correct at the time of printing. However, some product changes which do not affect the capability of the equipment, may not be included until it is necessary to incorporate other significant changes.

ELECTROMAGNETIC COMPABILITY

This equipment, when operated in accordance with the supplied documentation, does not cause electromagnetic disturbance outside its immediate electromagnetic environment.

COPYRIGHT NOTICE

ACKNOWLEDGEMENTS

Inteco Sp. z o. o. acknowledges all trademarks.

MICROSOFT, WINDOWS are registered trademarks of Microsoft Corporation.

MATLAB and Simulink are registered trademarks of MathWorks Inc.

CONTENTS

1. INTRODUCTION AND GENERAL DESCRIPTION

Fig. 1.1 Two-Wheeled Unstable Transporter

Two-Wheeled Unstable Transporter (**UnTrans**) shown in Fig. 1.1 is a self-balancing mobile system. From a control theory point of view UnTrans is an unstable, nonlinear system to some extent similar to the Inverted Pendulum. To stabilise the system in an upright position two DC motors drives a system wheels forward and backward. The appropriate direction and torque produced by the DC motors is calculated by a single board computer running in Real-Time control algorithm. The controller output

is based on measurement performed by an Inertial Measurement Unit (IMU), consisting of accelerometers and gyroscope, and encoders.

The UnTrans is the battery powered, autonomous system (i.e. no an external controller is needed).

The system serves for testing and verifying in practice linear and nonlinear control algorithms. The control algorithms are designed and prepared in the real-time MATLAB/Simulink environment and then sent wirelessly to the UnTrans system.

This manual describes:

- the system,
- the mathematical models and theory related to control experiments,
- the real-time experiments,
- how to use the library of ready-to-use real-time controllers,
- how to design and apply step-by-step an own controller in the MATLAB/Simulink/Linux environment.

It is assumed that a user has got an experience with MATLAB and Simulink from MathWorks Inc.

1.1 SYSTEM COMPONENTS

To use the **UnTrans** system the following software and hardware components are required:

Hardware

- **UnTrans** - the autonomous unit shown in Fig. 1.1,
- A battery charger,
- A Pentium or AMD based personal computer equipped with the WLAN interface.

Software

- Microsoft Windows W7 and MATLAB R2011, Simulink and RTW (Simulink Coder) toolboxes (not included),
- Linaro cross-compiler,

and

- CD-ROM including the UnTrans software and the e-manuals.

MATLAB cannot be installed in the "Program Files" directory (name of the directory cannot includes space).

Manuals:

- *Installation Manual*
- *User's Manual*

The experiments and corresponding measurements have been conducted by the use of the standard INTECO systems. Every new system manufactured and developed by INTECO can be slightly different to those standard devices. It explains why a user can obtain results that are not identical to those in the manual.

1.2 SYSTEM OVERVIEW

The schematic diagram of the UnTrans system is shown in Fig. 1.2.
The transporter is equipped with two independently controlled wheels. The center of mass of the vehicle is above the axis of rotation of the wheels. Such a construction makes UnTrans an unstable system. To maintain upright position, the angle of the transporter roll-bar must be continuously monitored and all deviations from the vertical position must be corrected by the wheels motions. The advantage of this structure is a great maneuverability using only two DC drives.

UnTrans consists of the following elements:

- The PandaBoard single board computer platform equipped with a wireless card, the processor ARM Cortex-A9, set of GPIOs and slot for an SD card. The board is responsible for communication with an external computer from where algorithms are loaded and where measurements are visualised. The operating system is stored on the SD card.
- The interface board containing a CPLD (*complex programmable logic device*) chip that can be reprogrammed to introduce a desired functionality of digital inputs/outputs without any hardware modification. In the CPLD chip the dedicated configuration (logic) is stored. This element reads encoders and generates the PWM signals for the DC motors. It also provides the power for PandaBoard. The special protocol of Inteco ensures the information exchange between the interface board and PandaBoard.

- Inertial Measurement Unit (IMU) equipped with accelerometers and gyroscope sensors. The accelerometers are mounted at an angle of 45 degrees. Data from sensors are transmitted to the interface board.
- Two DC motors are battery powered, PWM controlled and equipped with gearboxes (ratio 1:10).
- Two incremental encoders (400 imp/rpm quadrature) mounted at the axles of the motors measure the rotation angles of the wheels.
- There are two 12V, 2.3 Ah batteries.

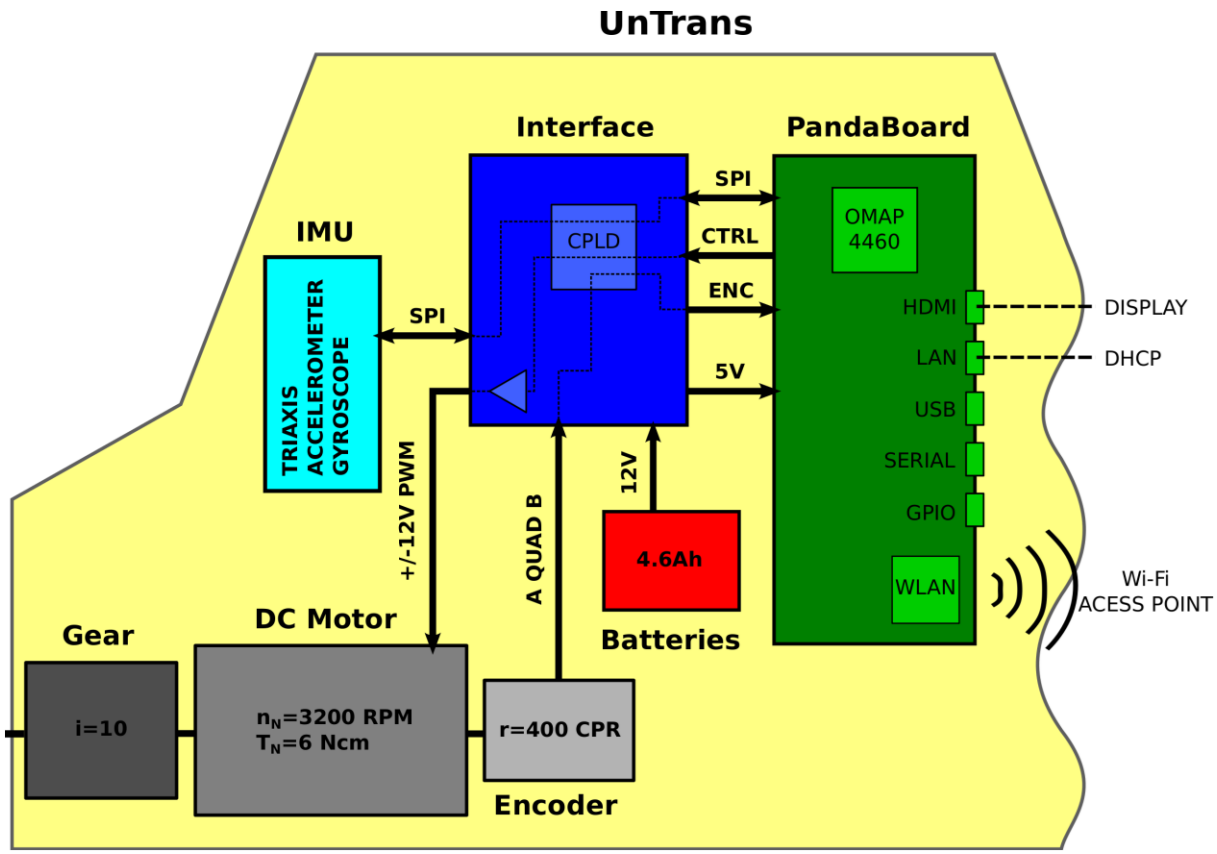


Fig. 1.2 UnTrans system block diagram

1.3 SOFTWARE INSTALLATION

Insert the installation CD and follow the displayed commands.
The procedure takes some time because simultaneously with the installation of the UnTrans toolbox is installed the Linaro cross-compiler.

2. STARTING AND TESTING PROCEDURES

2.1 STARTING PROCEDURE

It is assumed that UnTrans toolbox installation was successful.

Invoke MATLAB by double clicking on the MATLAB icon. The MATLAB command window opens. Then type:

UnTrans

MATLAB brings up the *UnTrans Control Window* (see Fig. 2.1). The user has an instant access to all basic functions of the Transporter control and simulation systems from the *UnTrans Control Window*.

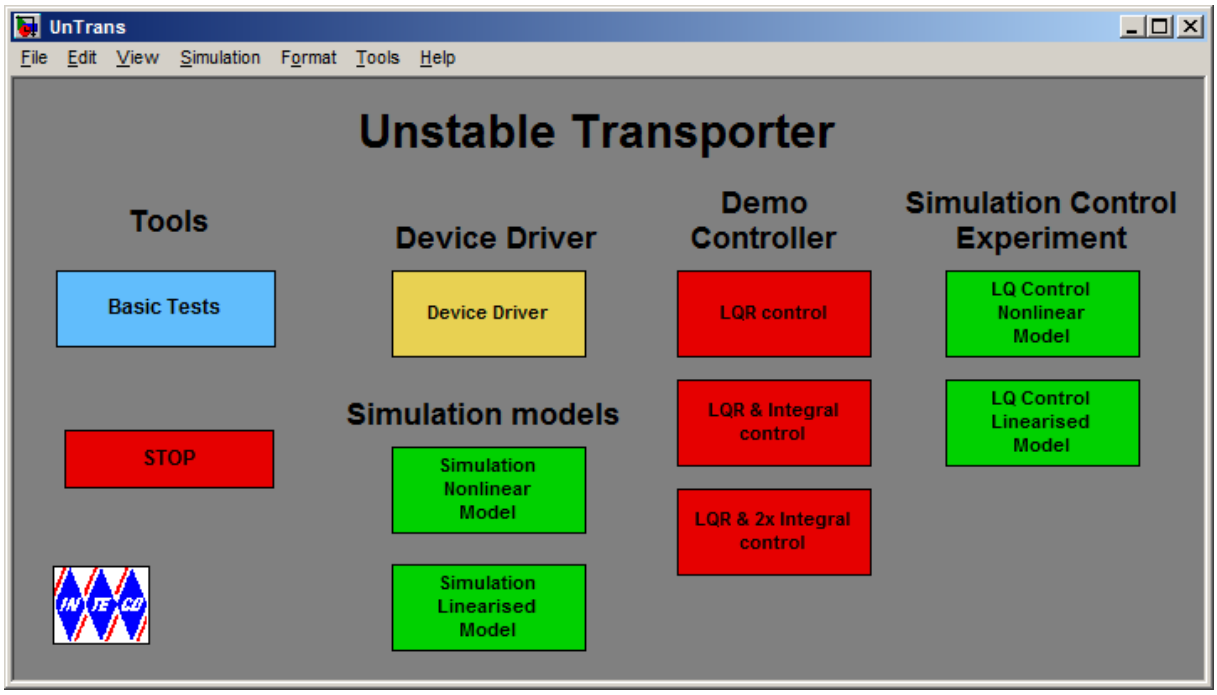


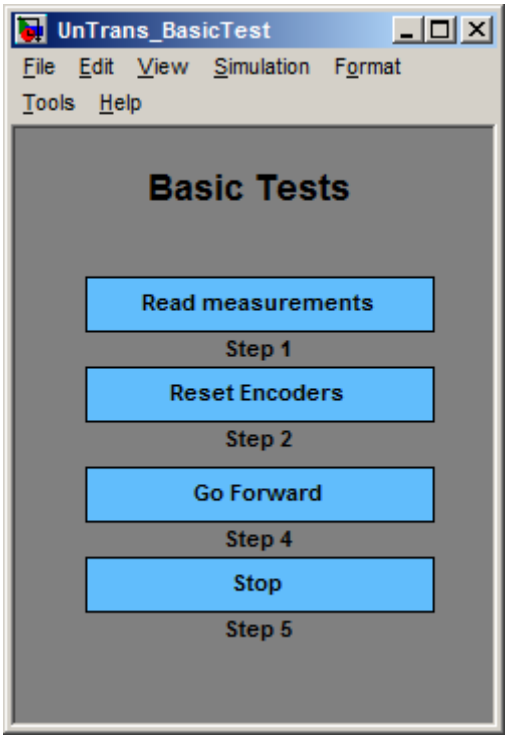
Fig. 2.1 Control Window of the *UnTrans* system

The *UnTrans Control Window* contains testing tools, drivers, models and demo applications. See section 3 for the detailed description.

3. UNTRANS CONTROL WINDOW

3.1 BASIC TESTS

This section explains how to perform the basic tests. The tests have to be performed obligatorily before start the real-time experiments. They are also necessary if any incorrect operation of the system was detected. The tests have been designed to validate the existence and sequence of measurements and controls. They do not relate to accuracy of the signals.



1. *Read mesasurements* button returns the values of sensors: Enc1, Enc2, AcceX, AcceY and Gyro.
2. *Reset Encoders* button

   resets encoders of the left and right wheels.
3. *Go Forward* button

   moves the transporter forward without stabilisation! *Go Forward* test is preferably carried out by lifting up the carrier, which allows to observe rotation of the wheels.
4. *Stop* button

   turns off the controls for two wheels.

Fig. 3.1 *Basic Tests* window

3.2 DEVICE DRIVER

The driver integrates the MATLAB/Simulink environment and UnTrans transforming and transmitting the measurement and control signals from/to the UnTrans system. To build a new application it is recommended to save a copy of the model with device driver (to preserve proper configuration). After clicking the UnTrans *Device Driver* button the window shown in Fig. 3.2 opens.

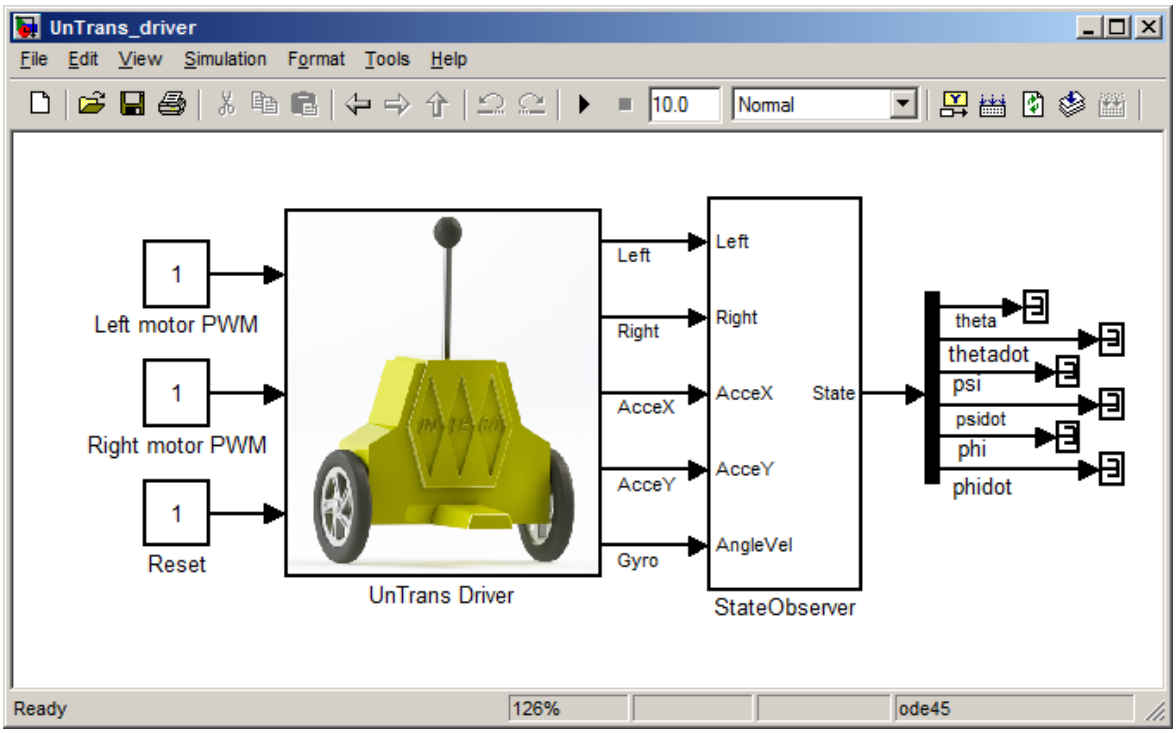


Fig. 3.2 *UnTrans Device Driver*

The driver has two PWM inputs for controlling the DC motors of the transporter. The first input controls the DC motor at the left. The second input controls the DC motor at the right. These control signals are the PWM type. The third input resets the incremental encoders which measure the angles of the wheels. The outputs are as follows: the angular position θ_l of the left wheel and the angular position θ_r of the right wheel, the accelerations in X and Y directions and the gyroscope signal. The *State Observer* block recalculates the signals of the sensors to the state variables $\theta, \dot{\theta}, \psi, \dot{\psi}, \varphi, \dot{\varphi}$.

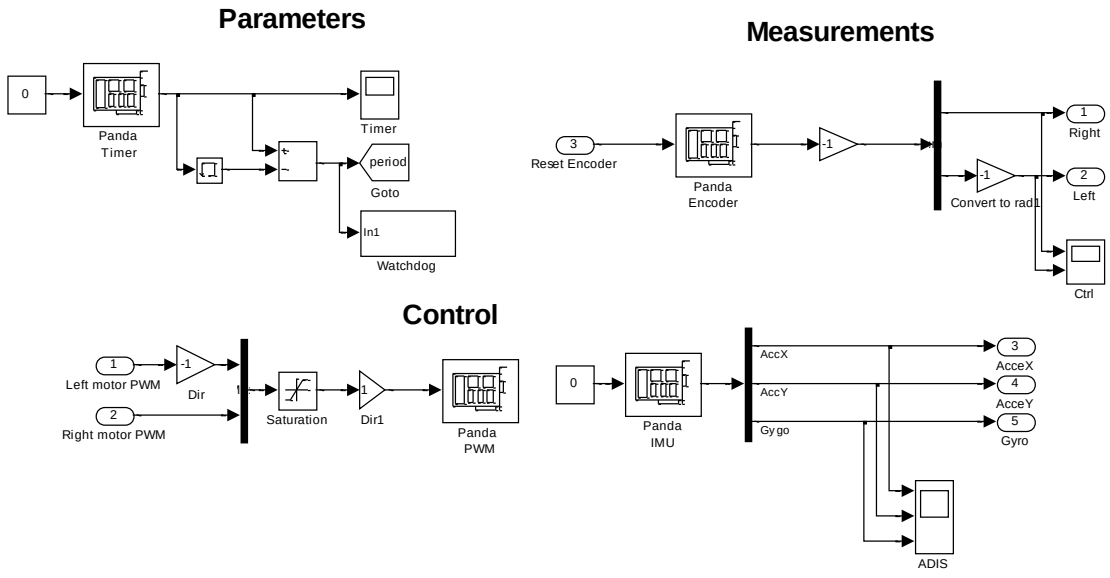


Fig. 3.3 Interior of the *UnTrans Device Driver*

A large deviation from the set sampling time period results in a loss of the system stability. Please notice the *Watchdog* block. The block sets the watchdog flag if the sampling time exceeds the specified value. The watchdog flag switches off the control signals. The reset signal (from subsytem *Reset*) can be used to reset the watchdog flag. The *Watchdog* block protects the transporter against damage.

Do not introduce any changes inside the original driver. They should be done only inside its copy.

The C source code of the all components of the driver is included in the *DevDriv* directory.

3.3 SIMULATION MODELS

Two simulation models are introduced. They are used to familiarize a user to the UnTrans system operation and templates for developing and testing the user-defined control algorithms. In Fig. 3.4 and Fig. 3.5 the nonlinear model and linearized model of the unstable transporter are shown.

Both models have two input PWM signals $u = [u_l \; u_r]^T$ for the left and right motors. The output signals in both models are ordered in the same way as the motors. These outputs are simply the state variables $x = [\theta_1 \; \theta_2 \; \psi_1 \; \psi_2 \; \varphi_1 \; \varphi_2]^T$ described in the detail in section 4.

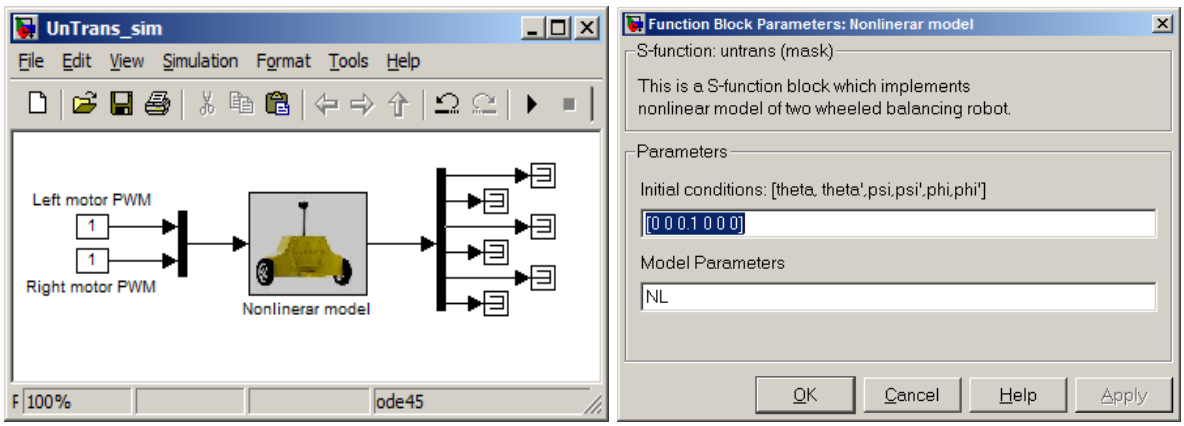


Fig. 3.4 Nonlinear simulation model and its mask

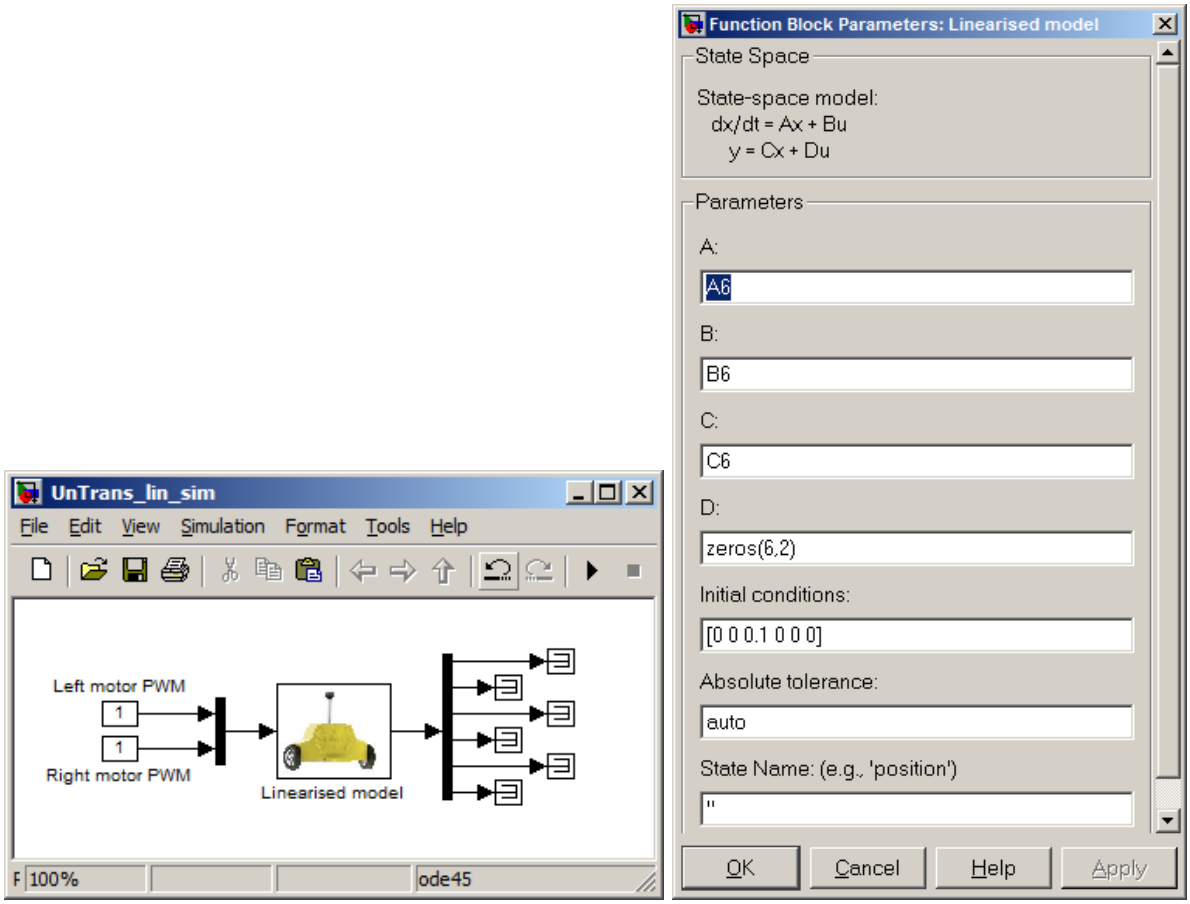


Fig. 3.5 Linearized simulation model and its mask

Remember the UnTrans system is structurally unstable and must run with a controller to get a stable behaviour.

Choose *Fixed step* solver options and set *Fixed-step size* equal to 0.01.

3.4 DEMO CONTROLLERS

There are three demos which allow performing real-time experiments. All the Simulink models shown in this section contain a number of controllers obtained by solving the LQ problem.

All the real-time Simulink models allow you to accomplish the task of stabilization and basic movements of the vehicle. Basic movements are translational motion and rotation.

All three models outwardly look similarly (see Fig. 3.6). The differences are only in the controller blocks where the different controllers are applied. The controllers are described later in this section and are shown in Fig. 3.7, Fig. 3.8 and Fig. 3.9.

The way of operation is the same. A detailed method of use is given in Section 5.3.

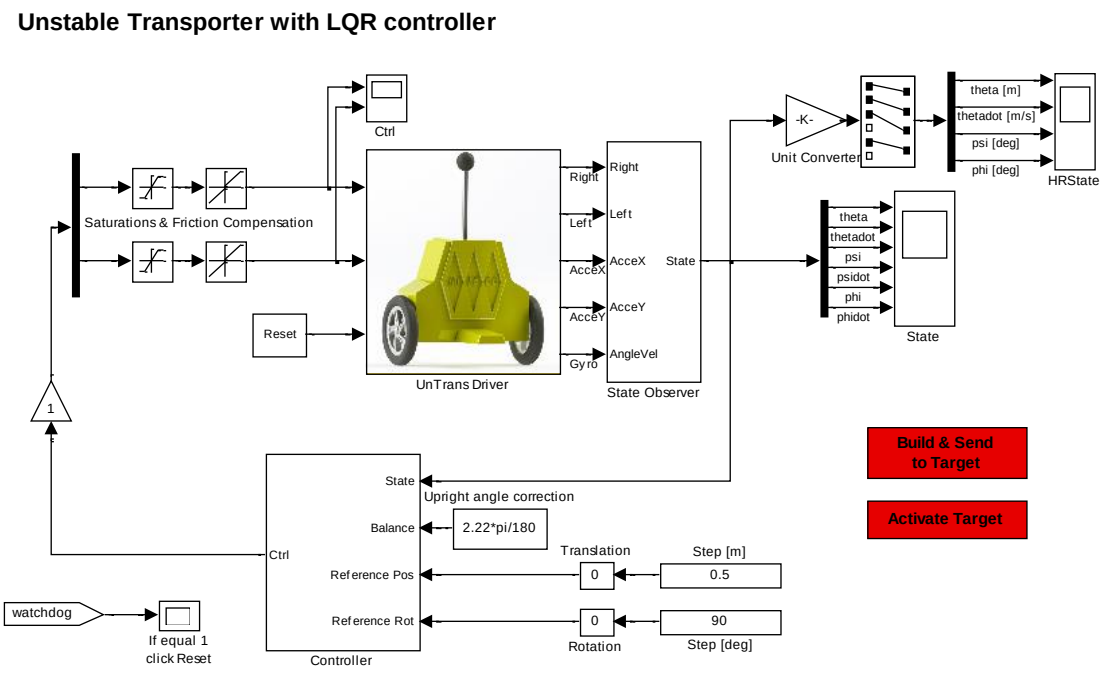


Fig. 3.6 Real-time demo model

There are three control models:

- **LQ controller –** Simulink model with the state feedback. Fig. 3.7 shows the applied controller.

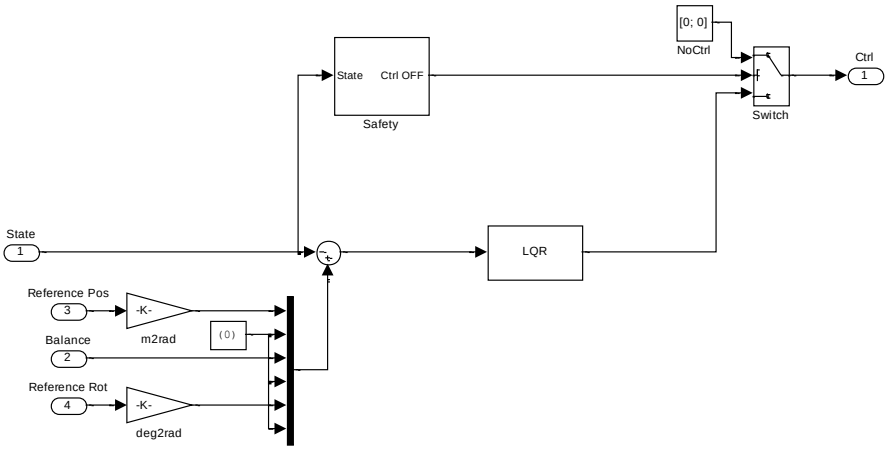


Fig. 3.7 LQ controller

- **LQ & integral controller** – Simulink model with the state feedback where state is expanded with one integral variable I_psi. Fig. 3.8 shows the controller applied in this case.

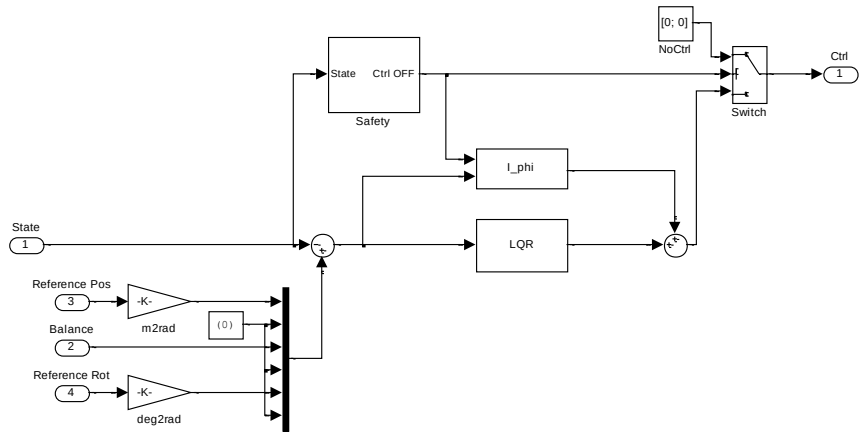


Fig. 3.8 LQ + integral controller

- **LQ & 2xintegrals controller -** Simulink model with the state feedback where state is expanded with two integral variables I_theta and I_psi. Fig. 3.9 shows the controller applied in this case.

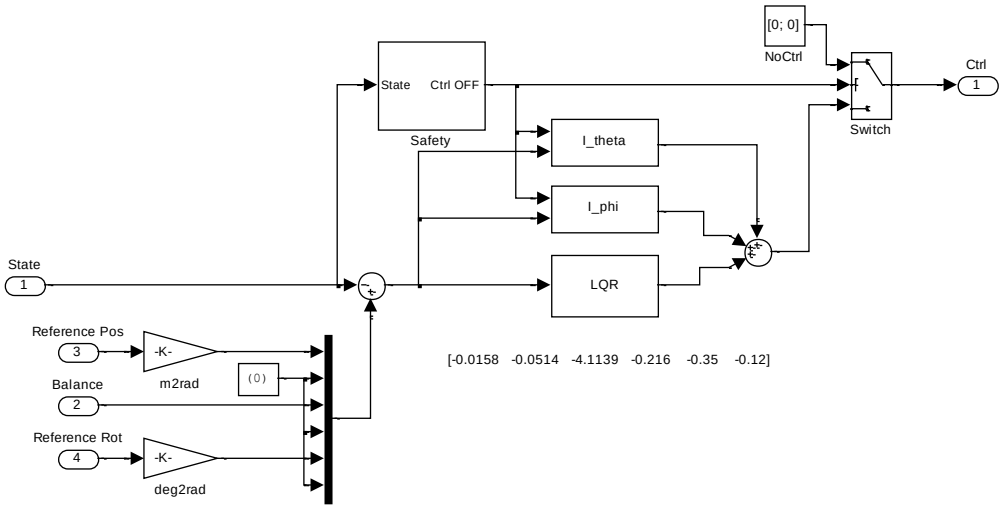


Fig. 3.9 LQ + 2 integrals controller

3.5 SIMULATION CONTROL EXPERIMENTS

Two examples of simulation models are introduced. These models are used to familiarize a user with the UnTrans system operation and to show the templates for developing and testing the user-defined control algorithms.

There are two models:

- **LQ Control Nonlinear Model** - where the nonlinear model is used

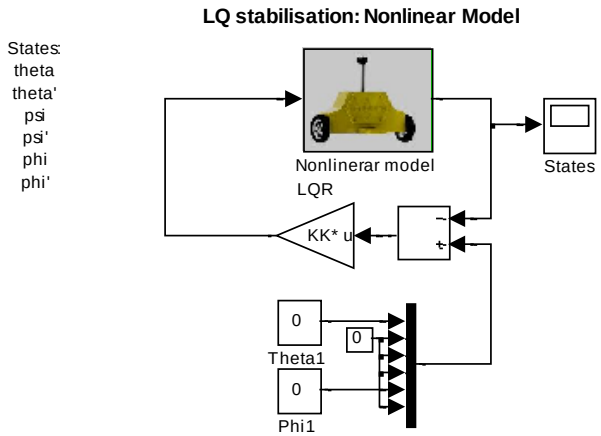


Fig. 3.10 LQ control: nonlinear simulation model

The block *Nonlinear model LQ* contains the S-function to solve equations 4.33, 4.34 and 4.35. During the model initialization the *UnTransParams* m-file is read and then the parameters are calculated in the *UnTransNonlinearModel* m-file.

- **LQ Control Linearized Model** - where the linearized model is used.

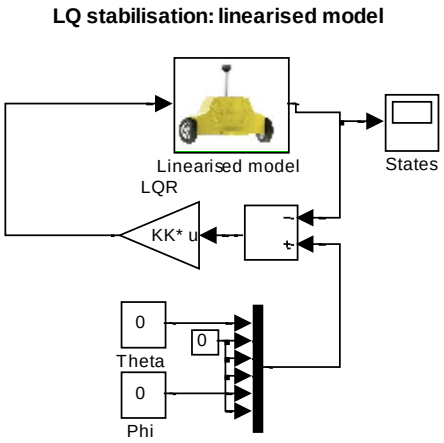


Fig. 3.11 LQ control: linearized simulation model

In this case also before simulation the *UnTransParams* m-file is read and then the parameters are calculated in *UnTransLTIModel6* m-file.

Both models contain the LQ state feedback controllers and allow to realize the task of stabilization and basic movements of the transporter.

4. MATHEMATICAL MODEL OF THE TWO-WHEELED UNSTABLE TRANSPORTER

Modern methods of design of advanced controllers usually require high quality models of the process. The classical procedure of a model development consists of the following steps:

- development of the mathematical model based on physics of the process,
- simplification of the model and/or its transformation into a standard form,
- development of a simulation model,
- tuning of the model parameters (identification),
- practical verification of the model.

Measured or calculated parameters of the model:

Parameter	Units	Description
$m = 0{,}32$	kg	weight of the wheel
$2R = 0{,}15$	m	diameter of the wheel
$J_w = mR^2$	kgm^2	Moment of inertia of the wheel
$J_w = 0.0013$	kgm^2	Identified moment of inertia of the wheel
$M = 5.41$	kg	weight of the vehicle
$W = 0{,}4$	m	width of the vehicle
L= 0,102	m	height of the mass center of vehicle
$J_\psi = ML^2/3$	kgm^2	estimated moment of inertia of the tilt axis vehicle
$J_\psi = 0,\ 104$	kgm^2	Identified moment of inertia of the tilt axis vehicle
$J_\varphi = 0,\ 0484$	kgm^2	moment of inertia of the vehicle related to the axis of rotation
$J_m = 0.00119$	kgm^2	Identified moment of inertia of the DC motor and gearbox taking into account gearbox ratio
$R_{DC} = 1$	Ω	Resistance of the winding of the DC motor
$K_t = 0{,}025$	Nm/A	Torque constant of the DC motor

K_e = 0,025	Vs/rad	Voltage constant of the DC motor
f_m= 0,00024		Identified friction coefficient between the vehicle and DC motor

The schematic diagram of the vehicle is shown in Fig. 4.1.

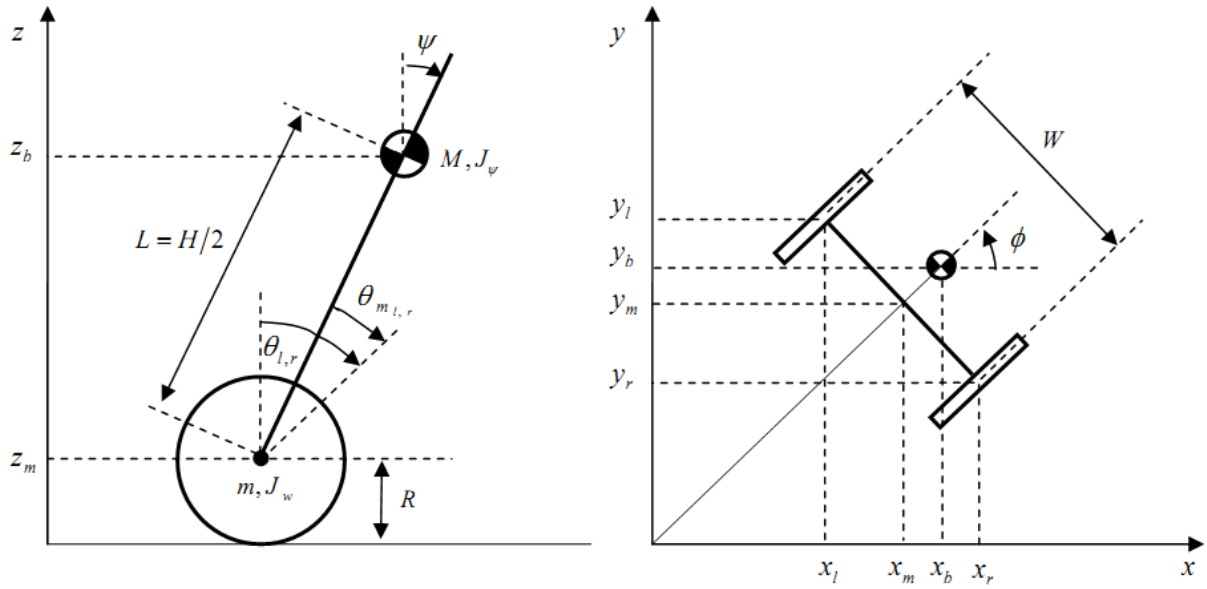


Fig. 4.1 Unstable transporter system: coordinates and forces

Dynamic of the unstable transporter is obtained from the Euler-Lagrange equations. As generalized coordinates of the system are adopted variables θ - average rotational velocity of the wheel, ψ - yaw angle from the vertical axis and φ - the angle of rotation around the vertical axis. There is another method of selecting generalized coordinate the wheel velocity and the angle of deviation from the vertical (θ_r, θ_l, ψ) but a review of the literature showed that the chosen selection simplifies the equations without loss of generality of the model.

4.1 NONLINEAR MODEL

Relations between variables illustrated in Fig. 4.1 can be described by the following equations:

$$\theta = \frac{1}{2}(\theta_r + \theta_l) \tag{4.1}$$

$$\varphi = \frac{R}{W}(\theta_r - \theta_l) \tag{4.2}$$

$$\dot{x} = R\dot{\theta}\cos\varphi \tag{4.3}$$

$$\dot{y} = R\dot{\theta}\sin\varphi \tag{4.4}$$

$$(\theta_r, \theta_l) = (\theta + \frac{W}{R}\varphi, \theta - \frac{W}{2R}\varphi) \tag{4.5}$$

$$(x_r, y_r) = (x + \frac{W}{2}\sin\varphi, y - \frac{W}{2}\cos\varphi) \tag{4.6}$$

$$(x_l, y_l) = (x - \frac{W}{2}\sin\varphi, y + \frac{W}{2}\cos\varphi) \tag{4.7}$$

$$(x_b, y_b, z_b) = (x + L\sin\varphi\cos\varphi, y + L\sin^2\varphi, L\cos\varphi) \tag{4.8}$$

Energy in a linear and rotary motion and potential energy are given in equations (from 4.9 to 4.11)

$$T_1 = \frac{1}{2}m(\dot{x}_r^2 + \dot{y}_r^2) + \frac{1}{2}m(\dot{x}_l^2 + \dot{y}_l^2) + \frac{1}{2}M(\dot{x}_b^2 + \dot{y}_b^2 + \dot{z}_b^2) \tag{4.9}$$

$$T_2 = \frac{1}{2}J_w(\dot{\theta}_r^2 + \dot{\theta}_l^2) + \frac{1}{2}J_\psi\dot{\psi}^2 + \frac{1}{2}J_\varphi\dot{\varphi}^2 + \frac{1}{2}J_m(\dot{\theta}_r^2 - \dot{\psi})^2 + \frac{1}{2}J_m(\dot{\theta}_l^2 - \dot{\psi})^2 \tag{4.10}$$

$$U = Mgz_b \tag{4.11}$$

The Lagrange equation has a form

$$L = T_1 + T_2 - U \tag{4.12}$$

To obtain a dynamic we use the Lagrange-Euler equations

$$\frac{d}{dt}(\frac{\partial L}{\partial\theta}) - \frac{\partial L}{\partial\theta} = F_\theta, \quad \frac{d}{dt}(\frac{\partial L}{\partial\dot{\psi}}) - \frac{\partial L}{\partial\psi} = F_\psi, \quad \frac{d}{dt}(\frac{\partial L}{\partial\dot{\varphi}}) - \frac{\partial L}{\partial\varphi} = F_\varphi$$

where F_θ, F_ψ, F_φ are generalized forces acting parallel to the respective generalized variables.

Calculating derivatives of coordinates and partial sums of Lagrangian we obtain

$$\dot{\theta}_r = \dot{\theta} + \frac{W}{2R}\dot{\varphi} \tag{4.13}$$

$$\dot{\theta}_l = \dot{\theta} - \frac{W}{2R}\dot{\varphi} \tag{4.14}$$

$$\dot{\theta}_r^2 + \dot{\theta}_l^2 = 2\dot{\theta}^2 + \frac{W^2}{2R^2}\dot{\varphi}^2 \tag{4.15}$$

$$\dot{x}_r = R\dot{\theta}\cos\varphi + \frac{W}{2}\dot{\varphi}\cos\varphi \tag{4.16}$$

$$\dot{x}_l = R\dot{\theta}\sin\varphi - \frac{W}{2}\dot{\varphi}\sin\varphi \tag{4.17}$$

$$\dot{y}_r = R\dot{\theta}\cos\varphi + \frac{W}{2}\dot{\varphi}\cos\varphi \tag{4.18}$$

$$\dot{y}_l = R\dot{\theta}\sin\varphi - \frac{W}{2}\dot{\varphi}\sin\varphi \tag{4.19}$$

$$\dot{x}_b = R\dot{\theta}\cos\varphi + L\dot{\psi}\cos\psi\cos\varphi - L\dot{\varphi}\sin\psi\sin\varphi \tag{4.20}$$

$$\dot{y}_b = R\dot{\theta}\sin\varphi + L\dot{\psi}\cos\psi\sin\varphi - L\dot{\varphi}\sin\psi\cos\varphi \tag{4.21}$$

$$\dot{z}_b = -L\dot{\psi}\sin\psi \tag{4.22}$$

$$x_b^2 + y_b^2 + z_b^2 = R^2\dot{\theta}^2 + L^2\dot{\psi}^2 + L^2\dot{\varphi}^2\sin^2\psi + 2RL\dot{\theta}\dot{\psi}\cos\psi \tag{4.23}$$

$$\dot{x}_r^2 + \dot{x}_l^2 + \dot{y}_r^2 + \dot{y}_l^2 = 2R^2\dot{\theta}^2 + \frac{W^2}{2}\dot{\varphi}^2 \tag{4.24}$$

Substituting equations (4.13)…(4.24) to equation (4.12) and using Euler-Lagrange equation we obtain

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\theta}}\right) - \frac{\partial L}{\partial\theta} = (2m+M)R^2 + 2J_w + 2J_m)\ddot{\theta} + (MRL\cos\psi - 2J_m)\ddot{\psi} - MRL\dot{\psi}^2\sin\psi \tag{4.25}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\psi}}\right) - \frac{\partial L}{\partial\psi} = (ML^2 + J_\psi + 2J_m)\ddot{\psi} + (MRL\cos\psi - 2J_m)\ddot{\theta} - ML^2\dot{\varphi}^2\sin\psi\cos\psi - MGl\sin\psi \tag{4.26}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\varphi}}\right) - \frac{\partial L}{\partial\varphi} = (\frac{1}{2}mW^2 + ML^2\sin^2\psi + \frac{W^2}{2R^2}(J_w + J_m) + J_\varphi)\ddot{\varphi} + 2ML^2\dot{\varphi}\dot{\psi}\sin\psi\cos\psi \tag{4.27}$$

Making use of equations 4.25-4.27 and the definition of the generalized forces we arrive at equations describing the system dynamics

$$((2m+M)R^2 + 2J_w + 2J_m)\ddot{\theta} + (MRL\cos\psi - 2J_m)\ddot{\psi} - MRL\dot{\psi}^2\sin\psi = F_\theta \tag{4.28}$$

$$(ML^2 + J_\psi + 2J_m)\ddot{\psi} + (MRL\cos\psi - 2J_m)\ddot{\theta} - ML^2\dot{\varphi}^2\sin\psi\cos\psi - Mgl\sin\psi = F_\psi \tag{4.29}$$

$$(\frac{1}{2}mW^2 + ML^2\sin^2\psi + \frac{W^2}{2R^2}(J_w + J_m) + J_\varphi)\ddot{\varphi} + 2ML^2\dot{\varphi}\dot{\psi}\sin\psi\cos\psi = F_\varphi \tag{4.30}$$

There are second-order derivatives of two variables in the equations (4.28) and (4.29). The equations in this form cannot be solved numerically because of the algebraic loop. The set of equations must be unraveled before the further development. To this aim, the formulas of Cramer will be applied:

$$c_1\ddot{\theta} + f_1(\psi)\ddot{\psi} = F_\theta + f_2(\psi,\dot{\psi}) \tag{4.31}$$

$$f_1(\psi)\ddot{\theta} + c\ddot{\psi} = F_\psi + f_3 \tag{4.32}$$

where $c_1 = ((2m+M)R^2 + 2J_w + 2J_m)$, $f_1(\psi) = (MRL\cos\psi - 2J_m)$, $c_2 = (ML^2 + J_\psi + 2J_m)$, $f_2(\psi,\dot{\psi}) = MRL\dot{\psi}^2\sin\psi$, $f_3(\psi,\dot{\varphi}) = ML^2\dot{\varphi}^2\sin\psi\cos\psi + MgL\sin\psi$.

After applying the formulas of Cramer we obtain a set of equations in the form:

$$\ddot{\theta} = \frac{c_2(F_\theta + f_2) - f_1(F_\psi + f_3)}{c_1c_2 - f_1^2} \tag{4.33}$$

$$\ddot{\psi} = \frac{c_1(F_\psi + f_3) - f_1(F_\theta + f_2)}{c_1c_2 - f_1^2} \tag{4.34}$$

$$\ddot{\varphi} = \frac{F_\varphi - 2ML^2\dot{\varphi}\dot{\psi}\sin\psi\cos\psi}{(\frac{1}{2}mW^2 + ML^2\sin^2\psi + \frac{W^2}{2R^2}(J_w + J_m) + J_\varphi)} \tag{4.35}$$

The generalized forces are defined

$$F_\theta = d_1(u_r + u_l) + 2d_2(\dot{\psi} - \dot{\theta}) \tag{4.36}$$

$$F_\psi = -d_1(u_r + u_l) + 2d_2(\dot{\theta} - \dot{\psi}) \tag{4.37}$$

$$F_\varphi = -d_1\frac{W}{2R}(u_r - u_l) - \frac{W^2}{2R^2}d_2\dot{\varphi} \tag{4.38}$$

$$d_1 = \frac{K_l}{R_m}, d_2 = \frac{K_lK_b}{R_m} + f_m \tag{4.39}$$

4.2 LINEARIZED MODEL

The linearized model is obtained by the Taylor expansion of (33-35) around the unstable equilibrium state in the upper location $x_0 = (0,0,0,0,0,0)^T, u_0 = (0,0)^T$.

Before linearization we perform the following substitution

$$\begin{aligned}
&\theta_1 = \theta, \theta_2 = \dot{\theta}_1, \dot{\theta}_2 = \ddot{\theta} \\
&\psi_1 = \psi, \psi_2 = \dot{\psi}_1, \dot{\psi}_2 = \ddot{\psi} \\
&\varphi_1 = \varphi, \varphi_2 = \dot{\varphi}_1, \dot{\varphi}_2 = \ddot{\varphi}
\end{aligned} \tag{4.40}$$

$$
\begin{aligned}
\dot{\theta}_1 &= \theta_2 \\
\dot{\theta}_2 &= \frac{c(F_0 + f_2) - f_1(F_\psi + f_3)}{dc - f_1^2} \\
\dot{\psi}_1 &= \psi_2 \\
\dot{\psi}_2 &= \frac{d(F_\psi + f_3) - f_1(F_\theta + f_2)}{dc - f_1^2} \\
\dot{\varphi}_1 &= \varphi_2 \\
\dot{\varphi}_2 &= \frac{F_\varphi - 2ML^2\dot{\varphi}\dot{\psi}\sin\psi\cos\psi}{(\frac{1}{2}mW^2 + ML^2\sin^2\psi + \frac{W^2}{2R^2}(J_w + J_m) + J_\varphi)}
\end{aligned}
\tag{4.41}
$$

The linearized model has a form

$$
\frac{d(\Delta x)}{dt} = A\Delta x + B\Delta u \tag{4.42}
$$

where: $x = [\theta_1\ \theta_2\ \psi_1\ \psi_2\ \varphi_1\ \varphi_2]^T$, $u = [u_l\ u_r]^T$,

$\Delta x = x - x_0$ is the modified state vector (deviation from the equilibrium state point x_0),

$\Delta u = u - u_0$ is deviation of the control, relative to u_0,

Matrices A and B are Jacobians of the equations (4.41) :

$$
A = [\frac{\partial f(x,u)}{\partial x}(x_0, u_0)],\ \ B = [\frac{\partial f(x,u)}{\partial u}(x_0, u_0)],
$$

During linearisation the set of linear equations decomposes into two subsystems:

$$
A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & a_{2,2} & a_{2,3} & a_{2,4} \\ 0 & 0 & 0 & 1 \\ 0 & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix},\ \ B_1 = \begin{bmatrix} 0 & 0 \\ b_{2,1} & b_{2,2} \\ 0 & 0 \\ b_{4,1} & b_{4,2} \end{bmatrix} \tag{4.43}
$$

and

$$
A_2 = \begin{bmatrix} 0 & 1 \\ 0 & a_{6,6} \end{bmatrix},\ \ B_2 = \begin{bmatrix} 0 & 0 \\ b_{6,1} & b_{6,2} \end{bmatrix} \tag{4.44}
$$

Where:

$$
a_{2,2} = \frac{-2\beta(c + f_1(0))}{dc - f_1^2(0)},\ a_{2,3} = \frac{-MgLf_1(0)}{dc - f_1^2(0)},\ a_{2,4} = \frac{2\beta(c + f_1(0))}{dc - f_1^2(0)},\ a_{4,2} = \frac{2\beta(d + f_1(0))}{dc - f_1^2(0)},
$$

$$a_{4,3} = \frac{dMgL}{dc - f_1^2(0)}, a_{4,4} = \frac{-2\beta(d + f_1(0))}{dc - f_1^2(0)}, b_{2,1} = b_{2,2} = \frac{\alpha(c + f_1(0))}{dc - f_1^2(0)},$$

$$b_{4,1} = b_{4,2} = \frac{-\alpha(d + f_1(0))}{dc - f_1^2(0)}$$

$$a_{6,6} = \frac{-\frac{W^2}{2R^2}\beta}{(\frac{1}{2}mW^2 + \frac{W^2}{2R^2}(J_w + J_m) + J_\varphi)}, \; b_{6,1} = b_{6,2} = \frac{-\frac{W}{2R}\alpha}{(\frac{1}{2}mW^2 + \frac{W^2}{2R^2}(J_w + J_m) + J_\varphi)} \quad (4.45)$$

5. REAL-TIME CONTROL EXPERIMENTS DESIGN AND IMPLEMENTATION

In the following section a control experiment is described. The experiment consists of three phases: the design phase, simulation phase and real-time implementation phase.

5.1 DESIGN OF LINEAR CONTROLLER

The UnTrans system demo presented in this section deals with the control task: *stabilize the transporter in the upper position* (an unstable steady-state point) and *move by a predetermined distance.*

The design of the continuous LQ controller is shown below. For a very small value of the sampling time the response of the discrete system converges to the response of the corresponding continuous system. It is just our case.

The linearized dynamical model of the unstable transporter is described by the linear differential equations (4.42–4.44) at the unstable equilibrium point defined as: $x_0 = (0,0,0,0,0,0)^T$, $u_0 = (0,0)^T$:

$$\dot{x} = Ax + Bu \text{ , where } A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & a_{2,2} & a_{2,3} & a_{2,4} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & a_{4,2} & a_{4,3} & a_{4,4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & a_{6,6} \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ b_{2,1} & b_{2,2} \\ 0 & 0 \\ b_{4,1} & b_{4,2} \\ 0 & 0 \\ b_{6,1} & b_{6,2} \end{bmatrix},$$

$x = [\theta_1\ \theta_2\ \psi_1\ \psi_2\ \varphi_1\ \varphi_2]^T$, $u = [u_l\ u_r]^T$, a_i and b_i are defined in (4.45).

A quadratic cost function has the form

$$J(u) = \frac{1}{2}\int_0^\infty [x^T(\tau)Qx(\tau) + u^T(\tau)Ru(\tau)]d\tau \,, \tag{5.1}$$

where : Q is a nonnegative definite matrix $Q \geq 0, Q = Q^T$,
R is a positive definite matrix $R > 0, R = R^T$,
and the pair (A, B) is controllable.

The weighting matrices Q and R are selected by a designer but they must satisfy the above conditions. This is most easily accomplished by choosing Q to be diagonal with all diagonal elements positive or zero.

The LQ optimal control u^* is given then by:

$$u^* = -K^* x \tag{5.2}$$

the optimal state feedback matrix.

The optimal control problem is now defined as follows: find the gain K^* such that the feedback law (5.2) minimizes the cost function (6.2) subject to the state equation (6.1). The optimal feedback gain can be obtained by iterative solution of the associated matrix Riccati equation:

To solve the LQ controller problem the *lqr* function can be used from the Matlab Control System Toolbox. The synopsis of *lqr* is: *[K,S,E] = lqr(A,B,C,D,Q,R)*.

In our case the state equation matrices are as follows:

$$A=\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -0.4733 & -29.1464 & 0.4733 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0.1401 & 29.1651 & -0.1401 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -2.6349 \end{bmatrix},\quad B=\begin{bmatrix} 0 & 0 \\ 109.3664 & 109.3664 \\ 0 & 0 \\ -32.3752 & -32.3752 \\ 0 & 0 \\ -0.5055 & 0.5055 \end{bmatrix}$$

matrix C is identity and D is zero matrix.

The weighting matrices *Q* and *R* have the following forms

$$Q=\begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 750 \end{bmatrix},\quad R=\begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix},$$

After command *[K,S,E] = lqr(A,B,C,D,Q,R)* the optimal gain matrix *K* for the considered parameters is calculated as

$$K=\begin{bmatrix} -0.0158 & -0.0514 & -4.1139 & -0.5304 & -0.1000 & -0.1077 \\ -0.0158 & -0.0514 & -4.1139 & -0.5310 & 0.1000 & 0.1007 \end{bmatrix},$$

Because of vibration caused by noisy angular velocity signal from the gyroscope the regarding gain was manually reduced:

$$K=\begin{bmatrix} -0.0158 & -0.0514 & -4.1139 & -0.2160 & -0.1000 & -0.1077 \\ -0.0158 & -0.0514 & -4.1139 & -0.2160 & 0.1000 & 0.1007 \end{bmatrix},$$

Recall now our task control:

Stabilize the transporter in the upper position (unstable equilibrium point)* and *move by a predetermined distance.

The LQ control simulation experiments are performed for the following data:

- Setpoint: $x_r = (0.5,0,0,0,0,0)^T$
- and starting point: $x_0 = (0,0,0,0,0,0)^T$,

5.2 SIMULATION

Using the nonlinear model shown in Fig. 5.1 the simulation experiment with the LQ controller is performed.

The *Nonlinear model LQ* block contains the S-function to solve equations 4.33, 4.34 and 4.35. During the model initialization the *UnTransParams* m-file is read and then the parameters are calculated by the *UnTransNonlinearModel* m-file.

The model is slightly modified by adding new blocks: the converter and scope . The converter recalculates the radians to meters for the distance and the radians to degrees for the angle. State variables stored in the new units are displayed in the HRState scope.

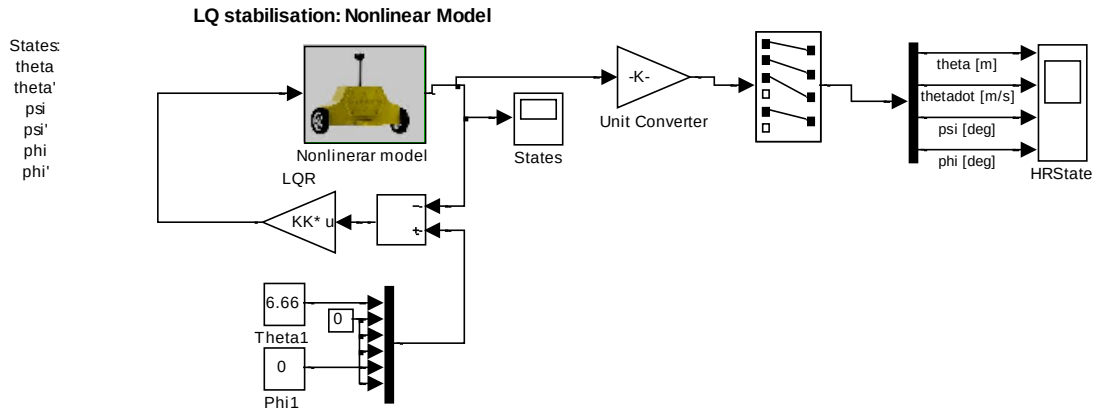


Fig. 5.1 LQ control – the modified nonlinear model

The initial state of the vehicle is equal to zero. The results of the experiment are shown in Fig. 5.3.

The identical stabilization experiment with an LQ controller is performed using linearized model shown in Fig. 5.2. In this case also before simulation the

UnTransParams m-file is read and then the controller parameters are calculated by the *UnTransLTIModel6* m-file. The results are illustrated in Fig. 5.4.

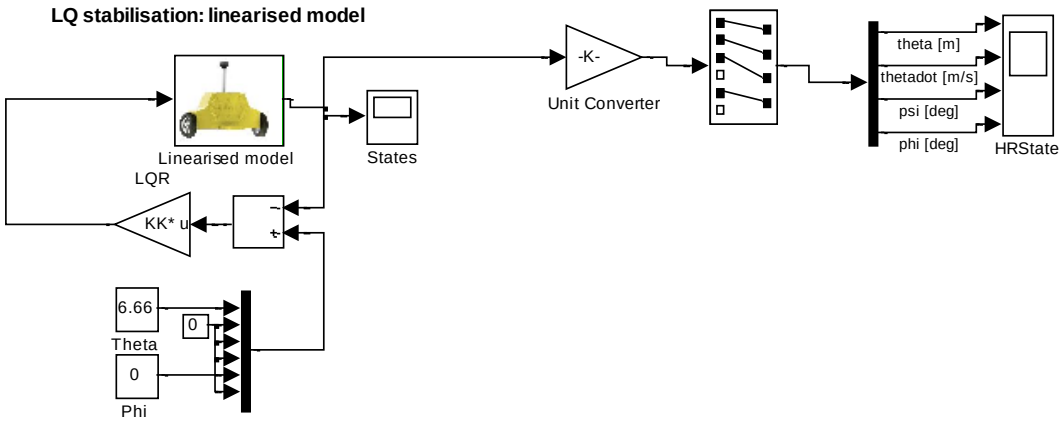


Fig. 5.2 LQ control – the modified linearized model

The LQ simulation is shown in section 3.4. The identical experiments are performed with nonlinear and linearized models. The apropriate results are shown in Fig. 5.3 and Fig. 5.4.

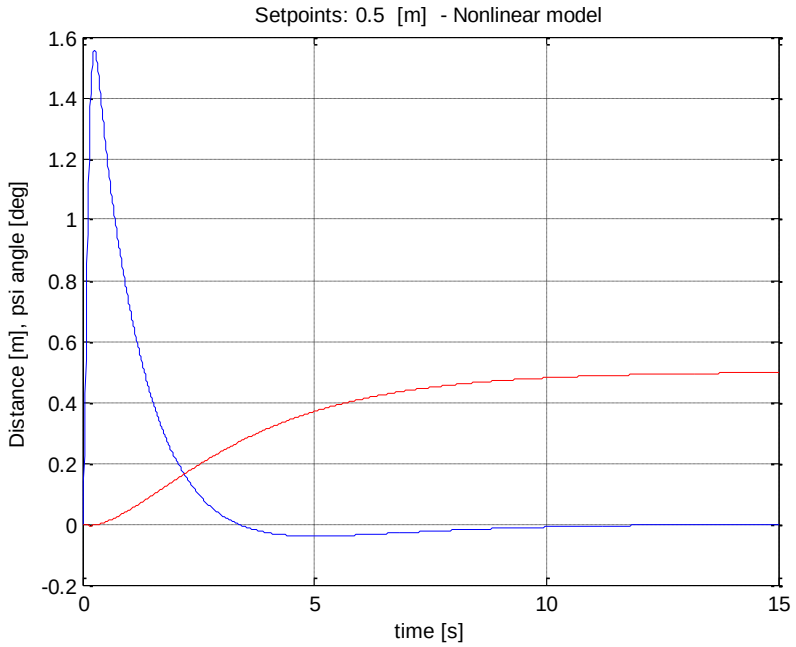


Fig. 5.3 LQ control: the nonlinear simulation model

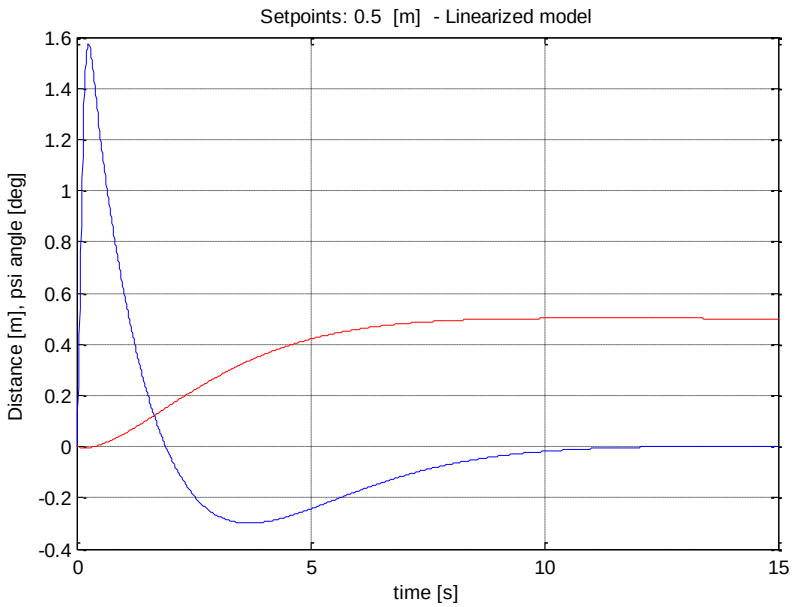


Fig. 5.4 LQ control: the linearized simulation model

5.3 REAL-TIME EXPERIMENTS- IMPLEMENTATION

In this section the preprogrammed examples of the UnTrans control system are illustrated. This demo is used to familiarize a user with the UnTrans system operation and help to create user-defined control algorithms.

5.3.1. EXAMPLE 1 – STABILISATION AND FORWARD DISPLACEMENT

After clicking on the *LQ Control* button the model of the real-time controlled UnTrans system appears (Fig. 5.5).

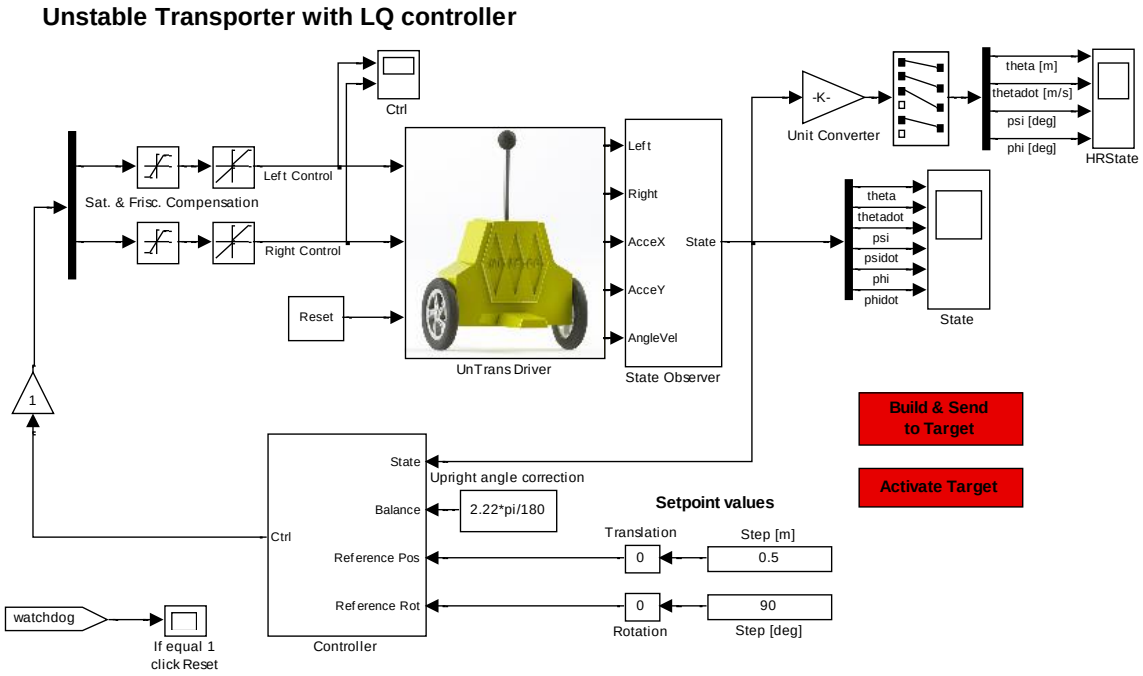


Fig. 5.5 Control system with the LQ controller

Note that this is a typical Simulink model. The device driver shown in Fig. 3.2 is applied in the same way as other blocks from the Simulink library. The only difference consists in applying the dedicated *Simulation/Configuration Parameters* tab configuration of the *External Mode* (see details at section 6). Also there are two special buttons: *Build & Send to Target* and *Activate Target*.

The State Observer block follows the *Unstable Transporter driver* block. The State Observer calculates the transporter state variables $\theta, \dot{\theta}, \psi, \dot{\psi}, \varphi, \dot{\varphi}$ on the basis of measurements $Enc_l, Enc_r, Acc_x, Acc_y, Gyro$.

The *State Observer* block contains the following blocks:

- the Kalman filter which calculates the vertical position of the transporter using the Acc_x, Acc_y and $Gyro$ signals,

- digital filters for calculating velocities.

In the model (and in the other demos) there are three scopes:

- Ctrl – where controls are displayed and stored,
- State – where the system states are shown. In this scope the states are displaced in rad and rad/s.
- HRState (human readable state) – where the selected states are shown. In this case the applied units are: m, m/s, deg and deg/s.
- If the *Watchdog* block breaks experiment (see page 14) the reset must be done. To do it click *Reset* and in the opened window switch back and forth the *Reset Encoders* switch.

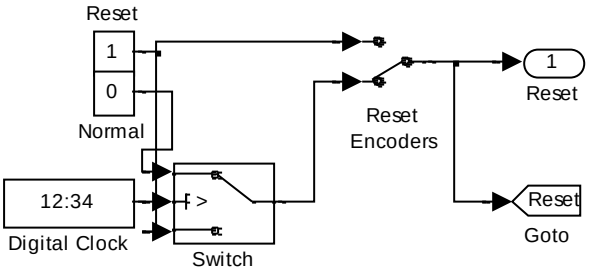


Fig. 5.6 Interior of the *Reset* block

To cause basic movements i.e. to define the setpoints for the translation and rotation two sliders are used: *Translation* and *Rotation* (see Fig. 5.7). After click on the right side of the slider the translation setpoint is modified i.e. increased + 0.5 [m]. In the case of the rotation slider the setpoint increases + π/2 [rad]. After click on the left side of the slider the setpoints are changed – 0.5 [m] or – π /2 [rad].

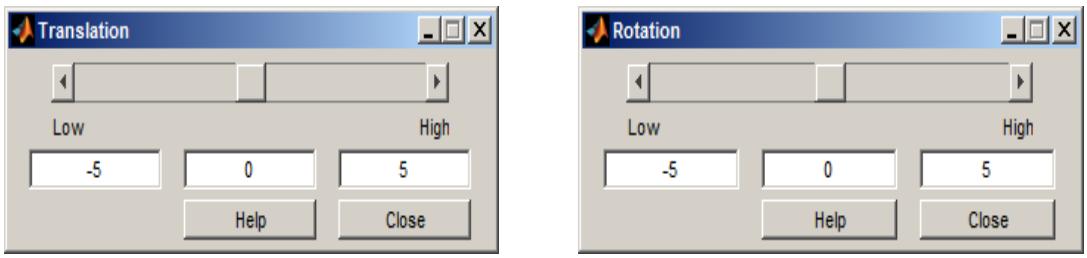


Fig. 5.7 *Translation* and *Rotation* sliders

To start experiment click the *Build & Send to Target* button to create and send the executable file to the PandaBoard . Next click the *Activate Target* button to activate real-time application in the transporter. Then press *Connect to target* icon on the top bar of the Simulink model. Now you can start real-time application by pressing the *Start Simulation* icon .

Results of the experiment are shown in Fig. 5.8 and Fig. 5.9. In the experiment the transporter is stabilized (red plot) and in 35 second the setpoint + 2 radian is set.

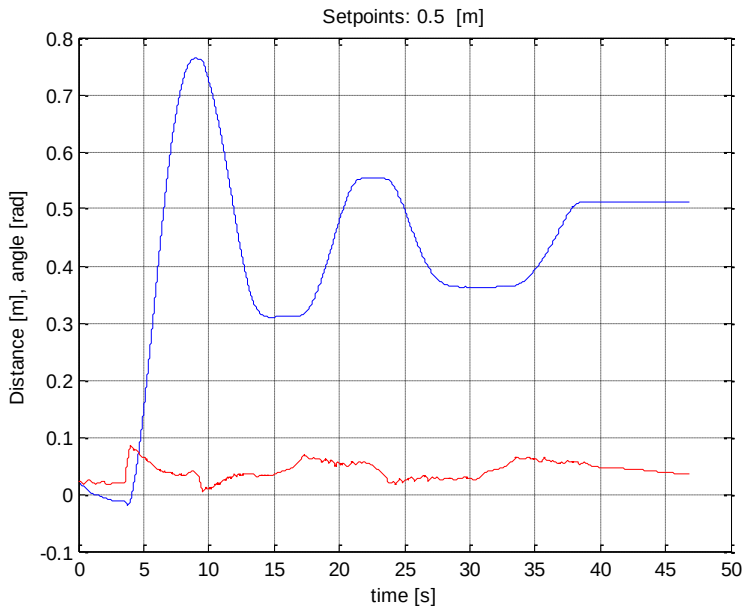


Fig. 5.8 State variables: θ-blue and ψ - red. The forward displacement experiment

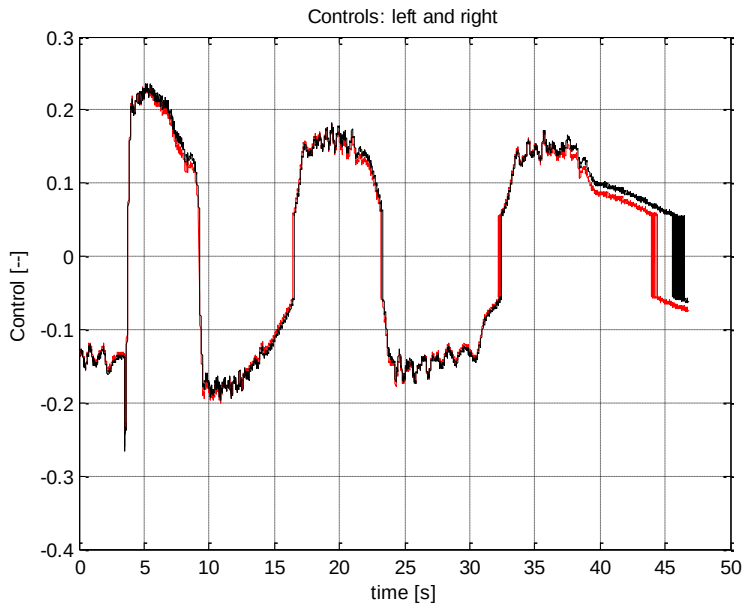


Fig. 5.9 Control signals for the left and right motors

5.3.2. EXAMPLE 2 – STABILISATION AND ROTATIONS

In this case the LQ with integral controller is used.

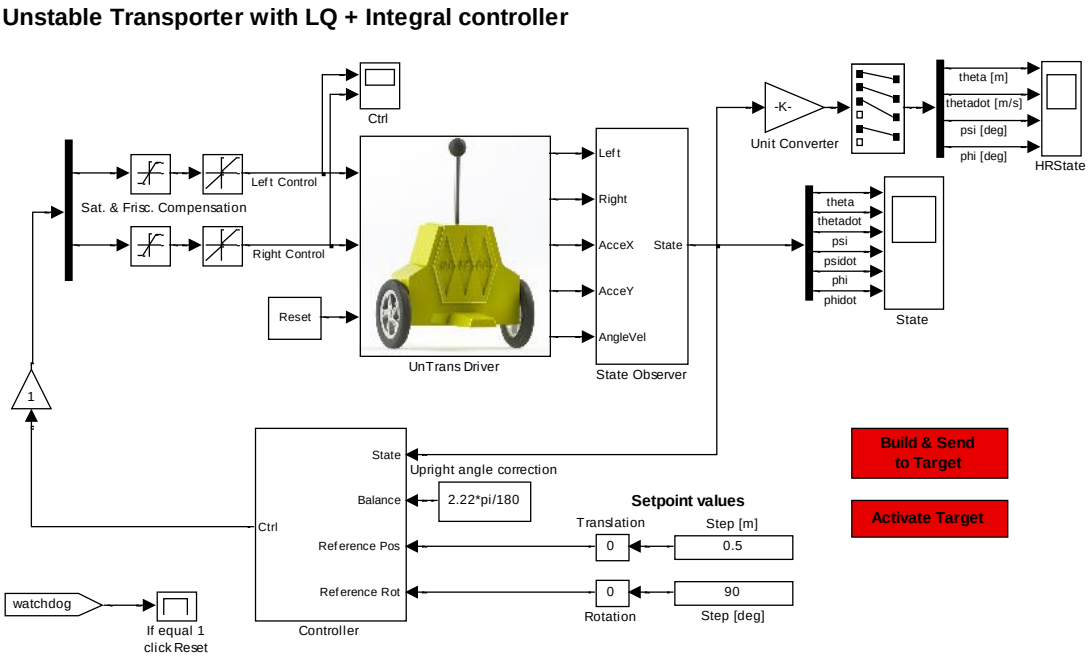


Fig. 5.10 Control system with the LQ+Integral controller

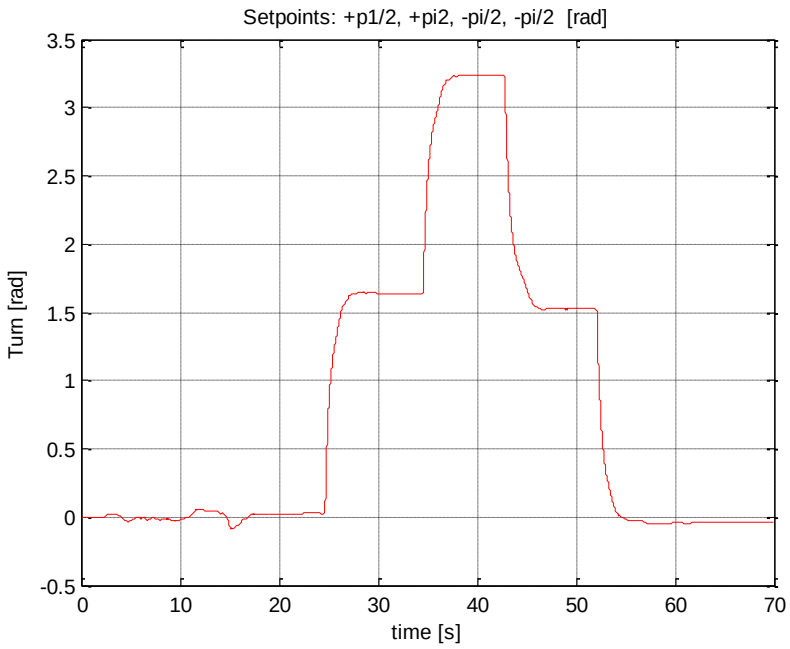


Fig. 5.11 State variable φ. The rotation experiment

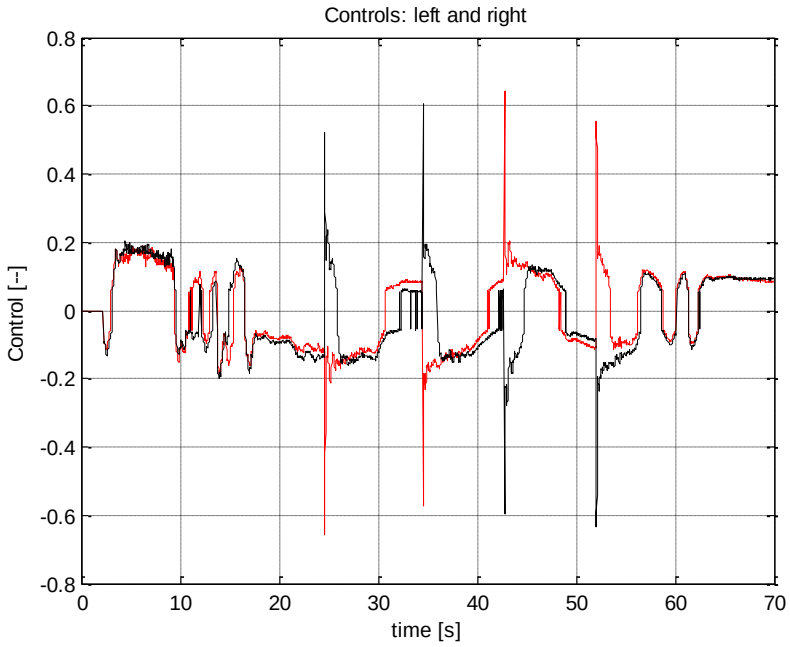


Fig. 5.12 Control signals for the left and right motors

5.3.3. EXAMPLE 3 – STABILIZATION AND ROTATION

In this case the LQ with two integrals controller is used.

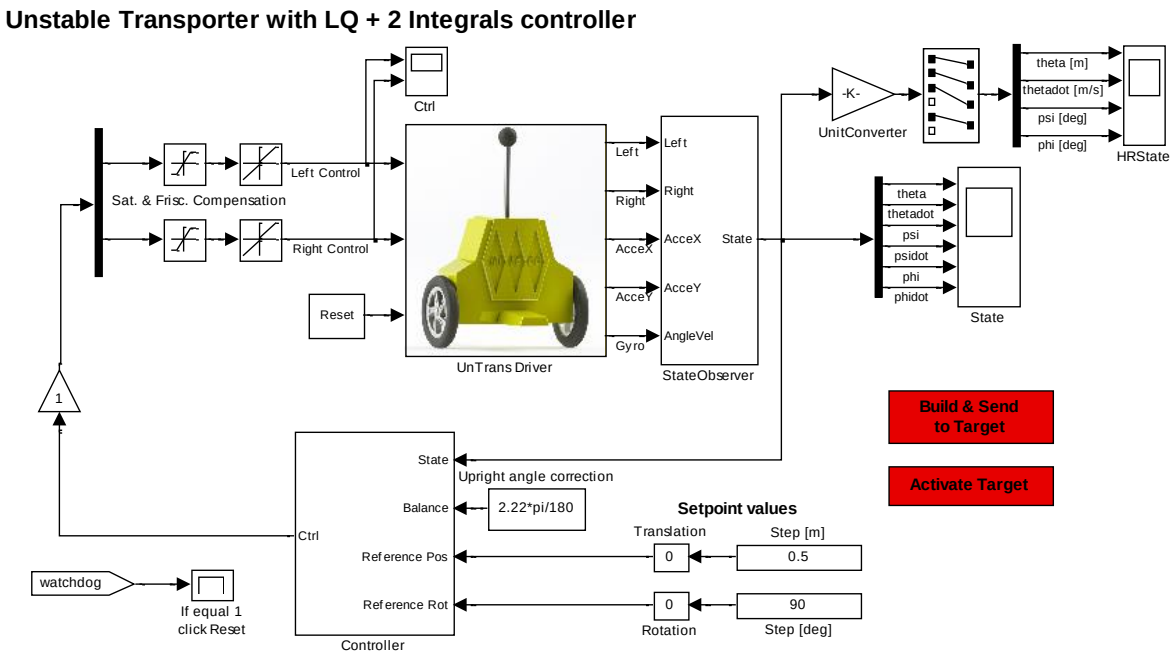


Fig. 5.13 Control system with the LQ+2 Integrals controller

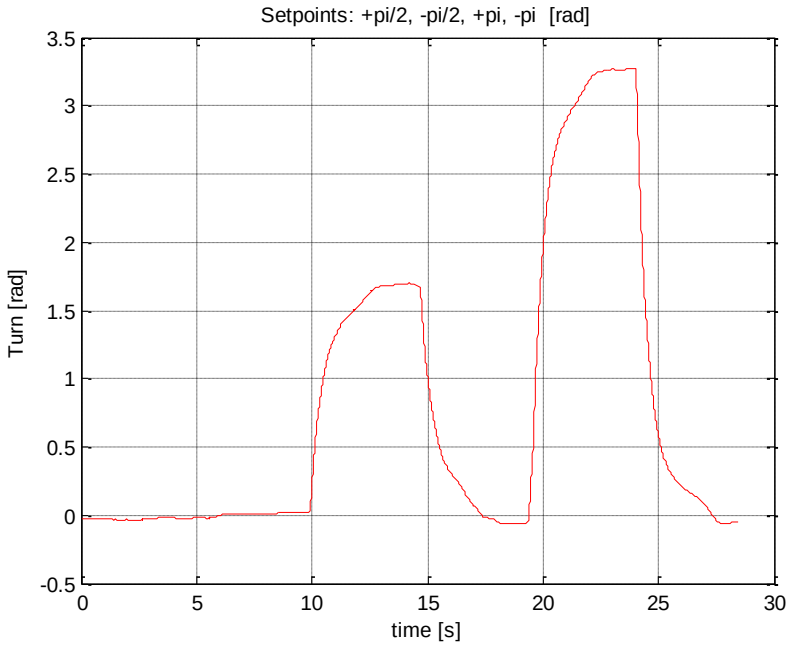


Fig. 5.14 State variable φ. The rotation experiment

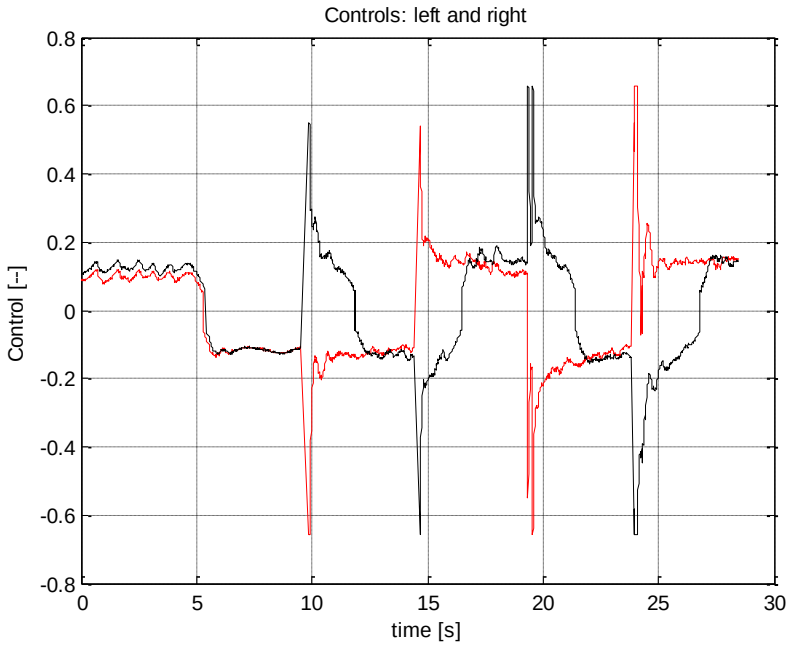


Fig. 5.15 Control signals for the left and right motors

5.3.4. EXAMPLE 4 – STABILIZATION AND DISPLACEMENT

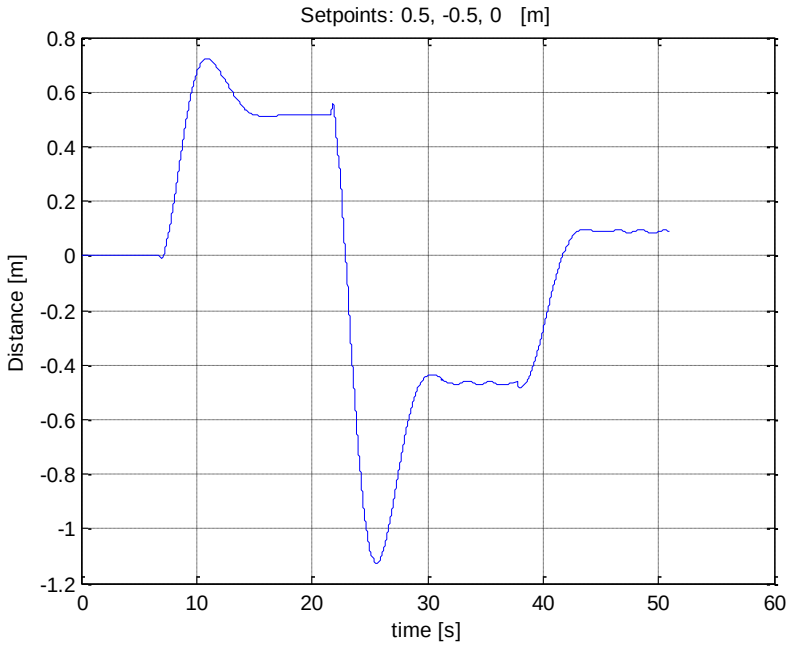


Fig. 5.16 State variable θ. The displacement experiment

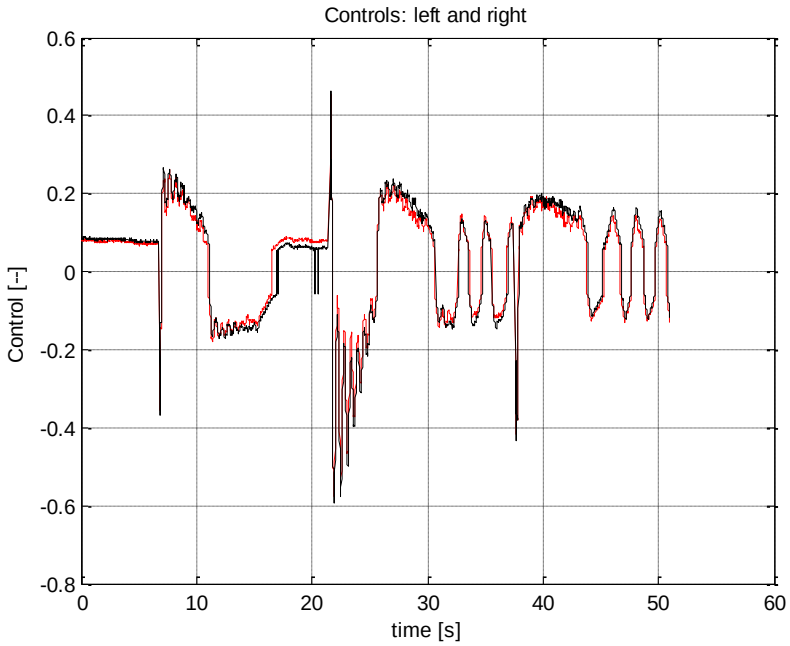


Fig. 5.17 Control signals for the left and right motors

5.3.5. EXAMPLE 5 – STABILIZATION. DISPLACEMENT AND ROTATION

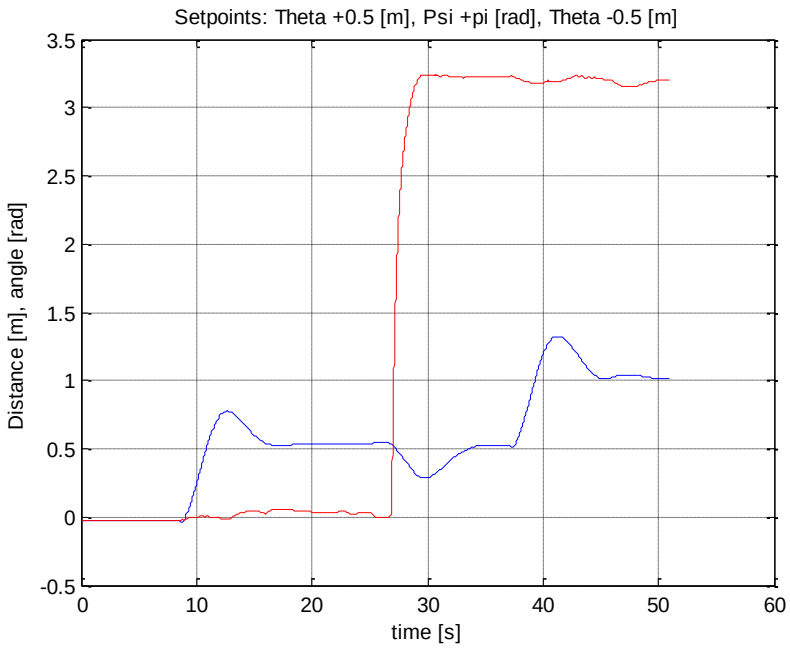


Fig. 5.18 States θ-blue and φ- red, displacements and rotation experiment

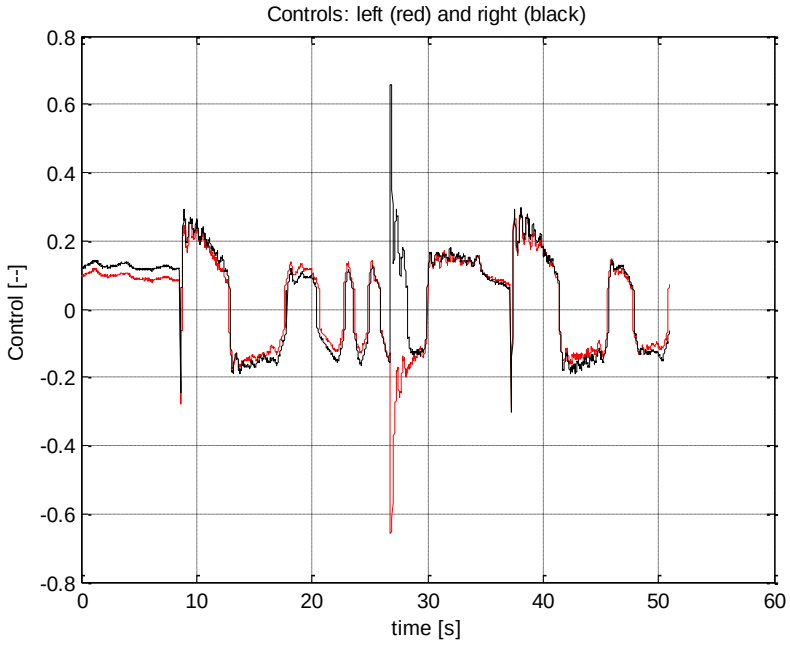


Fig. 5.19 Control signals for the left and right motor

6. PROTOTYPING AN OWN CONTROLLER IN MATLB/SIMULINK ENVIRONMENT

In this section the method of building your own controller is described. The Simulink Coder (formerly Real-Time Workshop) toolbox is used. Section 9 shows how to use the *UnTrans* software. Here we introduce the reader how to proceed in the Simulink Coder environment.

6.1 CREATING A MODEL

The simplest way to create a Simulink model for the UnTrans system is to use the *LQ controller* as a template. Save this model as the *MySystem.mdl* Simulink diagram. The *MySystem* Simulink model is shown in Fig. 6.1.

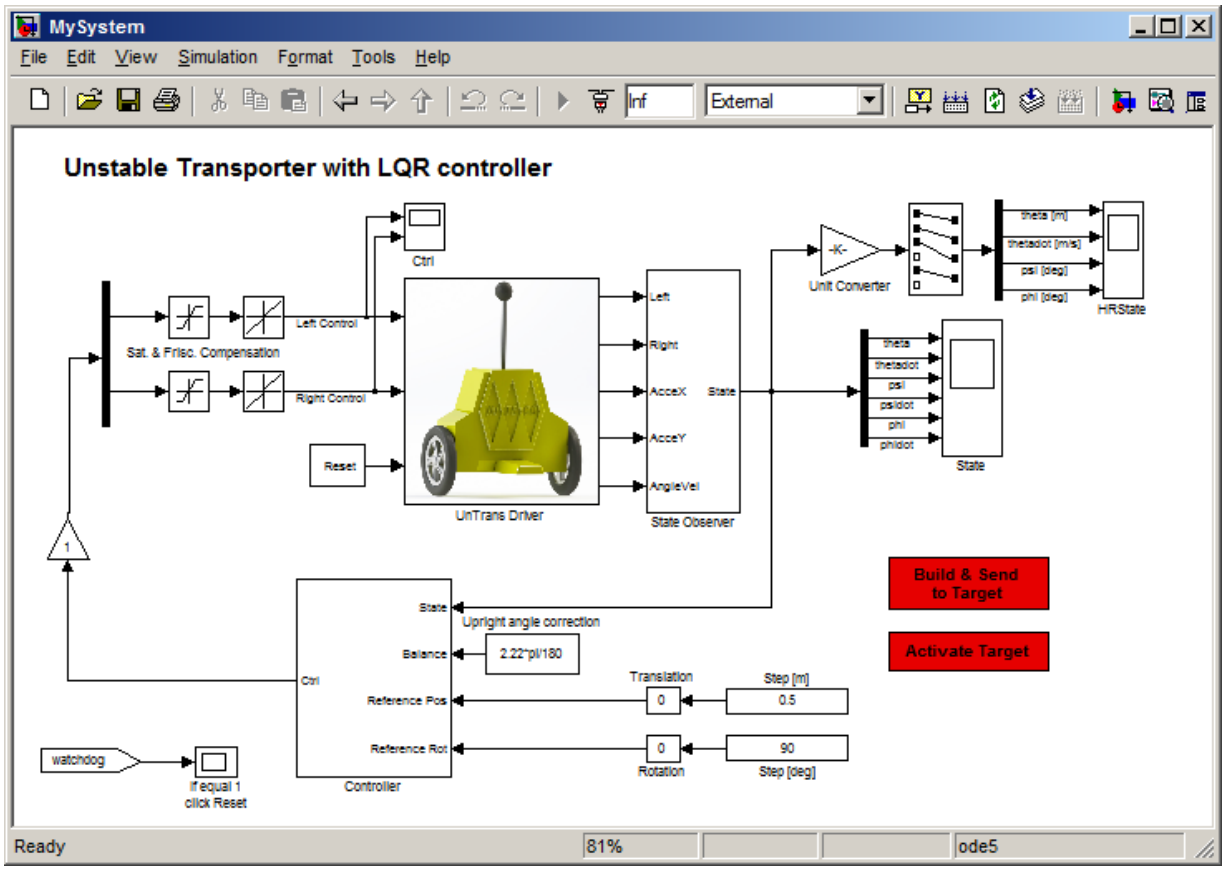


Fig. 6.1 The *MySystem* Simulink model

Now, you can modify the model. You have absolute freedom to develop your own controller. Remember, do not delete the *Unstable Transporter* (driver) and *State*

Observer blocks. Also do not delete the *Build & Send to Target and Activate* buttons. These special buttons activates the real-time control procedure. These items are necessary to support the data communication with the PandaBoard in the transporter. Though it is not obligatory, we recommend you to leave the scope (*State* block in Fig. 6.1). You need a scope to watch how the system runs. Other blocks in the window are not necessary for a new project.

Creating your own model on the basis of the old example ensures that all internal options of the model are set properly. These options are required to proceed with compiling and linking in a proper way. To put the *Unstable Transporter Driver* into the real-time code a target language compiler and a template makefile are required. Those files are included to the system software.

You can use the most of the blocks from the Simulink library. However, some of them cannot be used (see *MathWorks* references manual for details).

The Fig. 6.2, Fig. 6.3 and Fig. 6.4 show the proper configuration of all parameters of the model to create a real-time working executable.

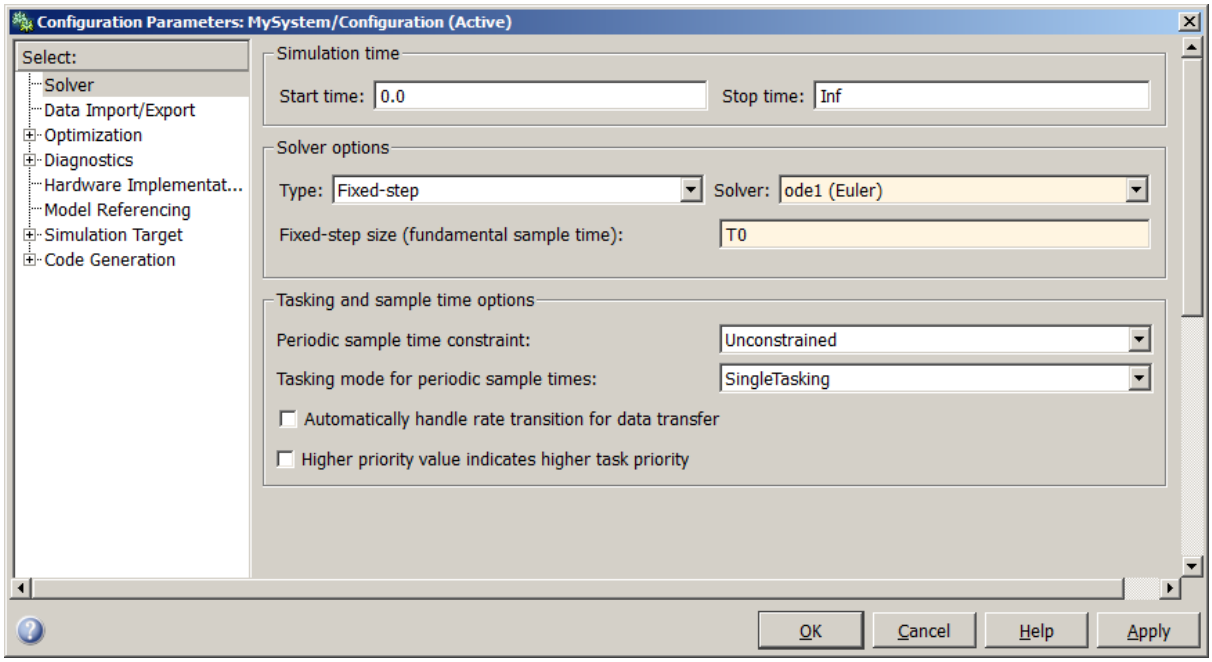


Fig. 6.2 Solver tab

The *Fixed-step* solver is obligatory for real-time applications. If you use an arbitrary block from the discrete Simulink library or a block from the driver's library remember, that different sampling periods must have a common divider.

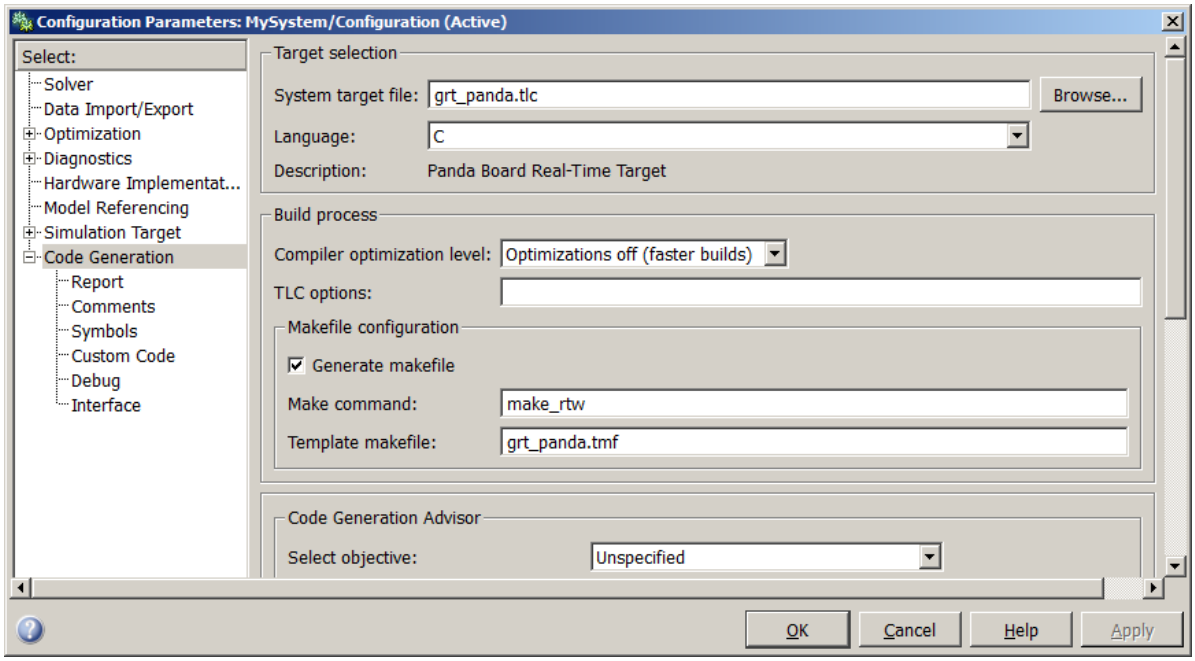


Fig. 6.3 Configuration Parameters – *Code Generation*

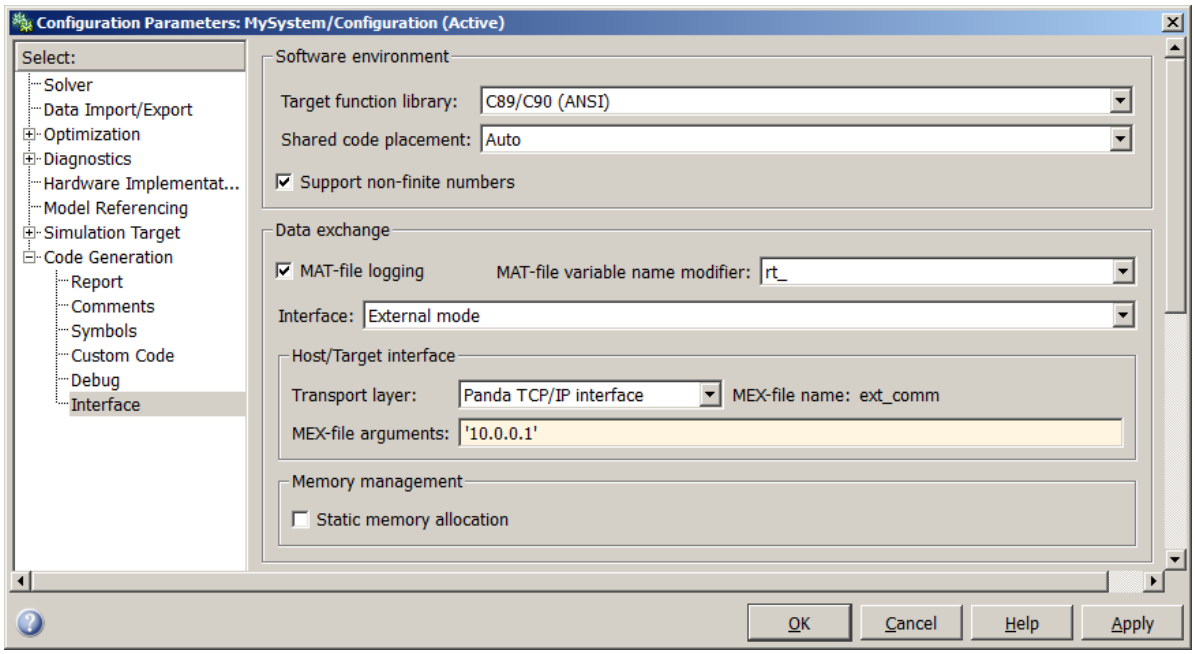


Fig. 6.4 Configuration Parameters – *Interface* tab

6.2 TROUBLESHOOTING

Problem	*Error message* / Cause	Solution
Failure to connect to target.	*Checksum mismatch. Target code needs to be rebuilt.* Model structure has changed since the last time code was generated.	**Rebuild, send and activate model.**
	No established connection between host computer and target device (UnTrans).	**Connect to device.**
	Error occurred while executing External mode Mex-file 'ext_comm' Real Time model not activated.	**Activate model using ACTIVATE TARGET button on Simulink diagram.**
Unable to connect or no PaAP WiFi Access Point.	Wrong password.	**Correct WiFi password (default *panda123*).**
	Battery is discharged.	**Charge battery.**
Simulink model fails to build.	Model structure error.	**Examine Matlab console to determine error.**
	Model has continuous states.	**Determine if model has states other than discrete. Discretize continuous states.**
	No particular reason.	**Delete MODEL_NAME_grt_rtw folder and try to rebuild.**
Demo controller has constant 0 control signal.	UnTrans is tilted more than +/- 12 degree.	**Tilt the UnTrans system to the upright position.**
	Watchdog flag has value = 1.	**Use Reset Switch**

		inside Reset Subsystem.
Demo controller does not stabilize system.	Upright position measured incorrectly.	**Adjust Upright Position Correction Coefficient.**
	System is too far from equilibrium.	**Reset encoders.**
	Watchdog flag has value = 1.	**Use Reset Switch inside Reset Subsystem.**
Signals are not present on the scopes.	Model is not running.	**Run the model.**
	Signals are NaN.	**Examine Scopes in Device Driver with raw data. If measurements signals are present, restart model.**
	No measurements in device driver.	**Try to rebuild driver. If problem persist try one of the Demo Controller.**
After start of Demo Controller UnTrans moves from initial position.	Reference Position has value different from 0.	**Change Reference position to 0.**
	Upright position is observed incorrectly.	**After start of the model wait a couple of second to allow state observer converge.**

7. ALGORITHM FOR TRACKING THE TRAJECTORY

7.1 FORMULATION OF THE PROBLEM

The section describes the task of tracking the trajectory of an unstable two-wheeled mobile transporter. The purpose of the algorithm is to maintain the vertical, unstable position during any motion at the transporter plane. The constructed control system contains two regulators.

The first is the overriding controller to maintain upright position of the vehicle.

The second controller is responsible for keeping a desired transporter trajectory. Kinematics of the transporter is described by three coordinates – two relate to the position of the transporter and one relates to the orientation. At the same time the transporter has two controls responsible for the movement of the right and left wheels. In carrying out the trajectory of the wheels one obtains specific routes of the system. However, due to the description of the model and convenience in controlling the orientation of the transporter it is better to describe the transporter by translations and rotations. The translation describes a displacement of the transporter base. The rotation is a rotary motion around the vertical axis passing through the center of the Transporter base. The rotation is proportional to the difference in motion of the wheels.

8. APPENDIX - RTOS CONFIG

8.1 COMMUNICATION

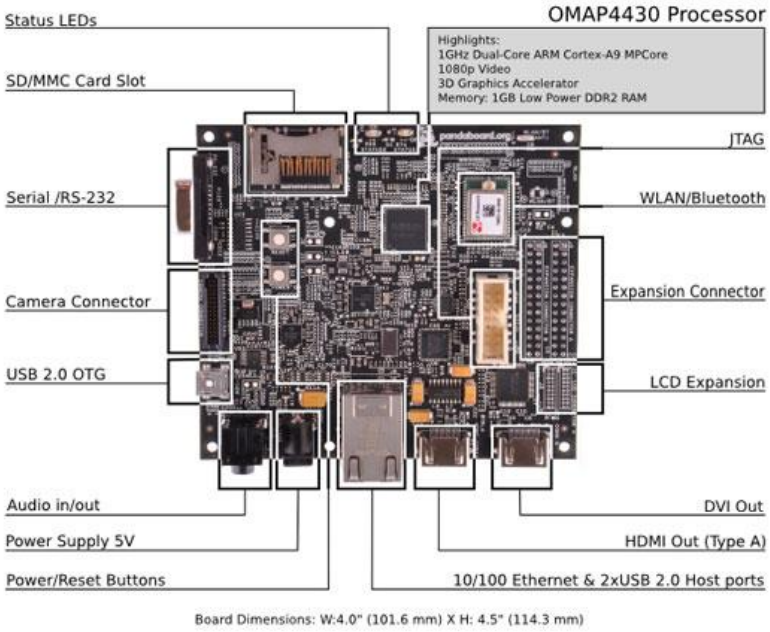


Fig. 8.1 PandaBoard diagram

To connect to remote system for the first time you can choose one of the interfaces listed below.

1. WiFi (Default communication)
    a. PandaBoard sets up WiFi Acces Point named *PaAP* with password *panda123.*
2. LAN
    a. Disassemble UnTrans housing (unscrew 6 plastic bolts)
    b. Connect PandaBoard to your router or a computer with DHCP server installed. Look for PandaBoard IP address on configuration page of your network.
    c. Use ssh terminal to connect to Pandaboard. User: *panda*, password: *panda*
3. Local (keyboard & display)
    a. Disassemble UnTrans housing (unscrew 6 plastic bolts)
    b. Connect USB keyboard and HDMI cable to HDMI output of the board (P2).
    c. Power up board and login in. User: *panda*, password: *panda*
4. Serial Terminal
    a. Disassemble UnTrans housing (unscrew 6 plastic bolts) and PandaBoard (unscrew 2 plastic nuts)

b. Connect RS232 cable to access console terminal.

PandaBord default configuration allows you to connect to the via Access Point provided by board. The SSID of an Access Point is PaAP and the password is panda123.

To change default config it is necessary to perform changes in networking configuration files.

For example to change WiFi config from default Acces Point to connection to your existing WiFi network perform those steps:

1. Turn off hostapd service:
   a. edit **/etc/default/hostapd** file by commenting line:
      DAEMON_CONF=/etc/hostapd/hostapd.conf
   b. turn off service:
      service stop hostapd
2. Configure network interfaces file **/etc/network/interfaces**:
   a. comment lines:
      address 10.0.0.1
      netmask 255.255.255.0
   b. change line:
      iface wlan0 inet static
      to
      iface wlan0 inet dhcp
   c. add your WiFi configuration:
      wpa-ssid "YourSSID"
      wpa-psk "YourPassword"

If any issues appear see at section 6.2.

Build & Send

UnTransTCPStop('10.0.0.1',20004); MODEL_NAME = gcs; slbuild(MODEL_NAME) cd([MODEL_NAME '_grt_rtw']) PSCP = ['!' matlabroot '\toolbox\UnTrans\tools\pscp']; eval([PSCP ' -pw panda ' MODEL_NAME ' panda@10.0.0.1:/home/panda/inteco/rt_model']) cd('..') fprintf('\nReady to activate\n')	Deactivate currently running model. Get model name. Build model. Change directory to build directory. Set path to pscp (SCP client). Send builded executable to UnTrans and rename to rt_model Back to working directory. Display message.

Activate Target

UnTransTCPStart('10.0.0.1',20004) fprintf('\nActivated\n')	Activate real time model on UnTrans. Display message.

9. REFERENCES

[1] Jun Ye. Tracking control for nonholonomic mobile robots: Integrating the analog neural network into the backstepping technique. *Neurocomputing*, 71(16):3373 - 3378, 2008. URL http://www.sciencedirect.com/science/article/pii/S0925231207003712.

[2] Omid Mohareri, Rached Dhaouadi, Ahmad B. Rad. Indirect adaptive tracking control of a nonholonomic mobile robot via neural networks. *Neurocomputing*, 88(0):54 - 66, 2012. URL http://www.sciencedirect.com/science/article/pii/S092523121200015X. Intelligent and Autonomous Systems.

[3] Yorihisa Yamamoto, NXTway-GS Model-Based Design - Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT, 2009

[4] Grasser, F. D'Arrigo, A. ; Colombi, S. ; Rufer, A.C., JOE: A Mobile, Inverted Pendulum, Industrial Electronics, IEEE Transactions on , vol.49, no.1, pp.107,114, Feb 2002