University of Oxford

Trinity Term

CourseWork Module (CWM)

LEGO Football

Lab organizers: Jack Umenberger (jack.umenberger@eng.ox.ac.uk)
Junaid Memon (junaid.memon@eng.ox.ac.uk)

Location: Control Laboratory, 5th Floor, Thom Building

May 20, 2024

Learning outcomes

The "LEGO Football" coursework module has four main objectives:

1. Apply in a more practical example tools from control theory and become familiar with the underlying concepts.
2. Provide an introduction to the entire control design process, that involves: physical modelling; design and analysis of a feedback control mechanism (stability and performance); model-based performance evaluation of the developed controller (in simulation); actual implementation using the systems' sensor-actuators (in hardware).
3. Become familiar with the use of MATLAB and Simulink for control design of systems with different characteristics: unstable, critically stable, stable.
4. Interact with robots and make them play football!

In these notes we consider a self-balancing, two-wheeled LEGO Mindstorms EV3 robot, and provide a companion for the laboratory tasks, providing details on the modelling and control design aspects. The robot is represented by a two-wheeled inverted pendulum, with the wheels being driven by DC motors. A nonlinear dynamical system governing its behaviour is derived based on the equations of motion for the wheels and the inverted pendulum, and on a simplified representation of the DC motor dynamics. The constructed model is then transformed to a linear dynamical system by means of linearization. Treating the DC motors' input voltage as control inputs, a Proportional-Integral (PI) controller is developed to ensure that the robot (two-wheeled inverted pendulum) can self-balance, a Proportional-Integral-Derivative (PID) controller is developed to allow forward/backward motion, and a Proportional (P) controller to allow turning.

Acknowledgements

These notes follow the notation and problem description of [1], [2], where some of the figures are also taken from (modified as appropriate), but the system dynamics are derived following an analysis similar to [3, 4]. We refer also to [5] for an alternative description and physical interpretation.

Robot assembling, hardware communication and the gamepad functionality is due to Dr. Izzi Mear (Teaching and Design Engineer, University of Oxford); the contribution of Dr. Filiberto Fele (University of Oxford) is also gratefully acknowledged. We are grateful to Dr. Coorous Mohtadi (MathWorks) for insightful discussions and input on the hardware implementation. The contribution of Dr. Luca Deori (Politecnico di Milano) in the first lab design involving LEGO Mindstorms NXT robots in Trinity Term 2015-16 is also gratefully acknowledged.

Contents

1 Physical description

Consider a two-wheeled LEGO Mindstorms EV3 robot, as shown in Figure 1, where both wheels are identical, and are driven by identical motors. The main brick is responsible for any communication and processing when interacting with MATLAB/Simulink. Our configuration is equipped with two sensors: a gyro sensor that as will be discussed in the sequel contains information on the robot's angular displacement from the vertical axis, and a touch sensor that will be used to reset the motors. Actuation inputs to the robot constitute the voltages applied to the left and right motor, respectively.

From a mathematical point of view we can represent the robot by a two-wheeled inverted pendulum (see Figure 2), whose side and top view are shown in Figure 3. It can be thought of as a combination of a unicycle-like model and an inverted pendulum.

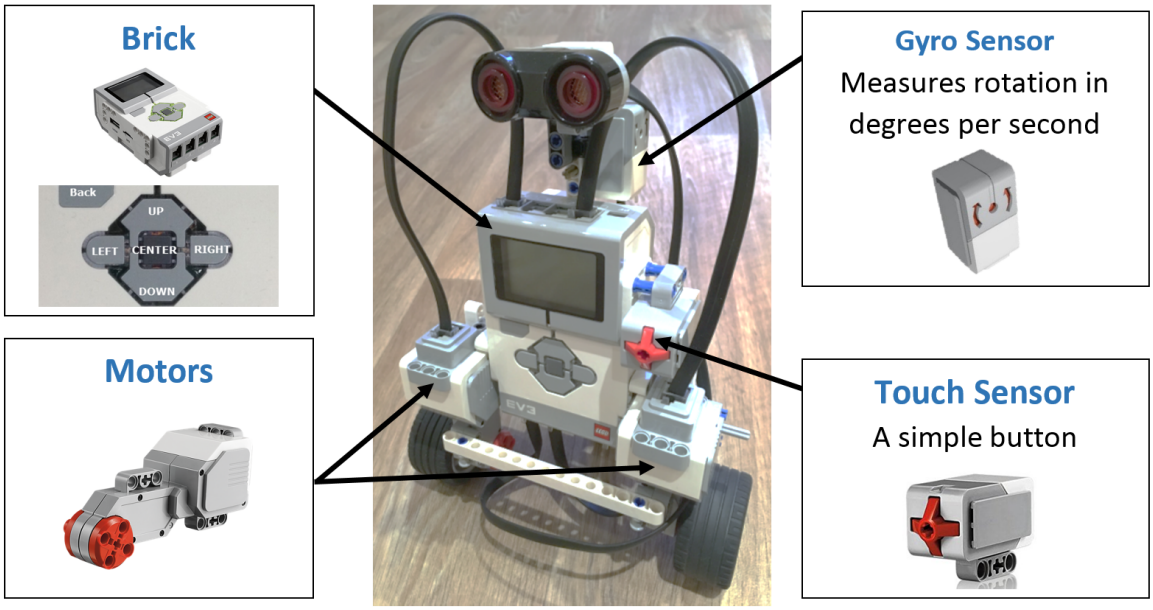

Figure 1: LEGO Mindstorms EV3 robot.

Variable θ denotes the wheels' angle, ψ denotes the pitch angle of the pendulum, and ϕ the yaw angle of the robot. Before providing the equations of motion of the two-wheeled inverted pendulum, it would be convenient to establish certain relations between the cartesian coordinates (x_w, y_w, z_w) of the center of each wheel $w \in \{l, r\}$, where l, r, indicate the left and right wheel, respectively, the coordinates of the middle point (x_m, y_m, z_m) along the wheel axis, and the ones of the middle point (x_b, y_b, z_b) of the upper body, as a function of the angles θ, ψ and ϕ.

To this end, consider the reference coordinate frame of Figure 3, and denote by

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

the average angle of the right and left wheel, respectively. The rate of change of the yaw angle $\dot{\phi}$ can be written as a function of $\dot{\theta}_r$ and $\dot{\theta}_l$ given by

$$\dot{\phi} = \frac{R}{W}(\dot{\theta}_r - \dot{\theta}_l). \tag{2}$$

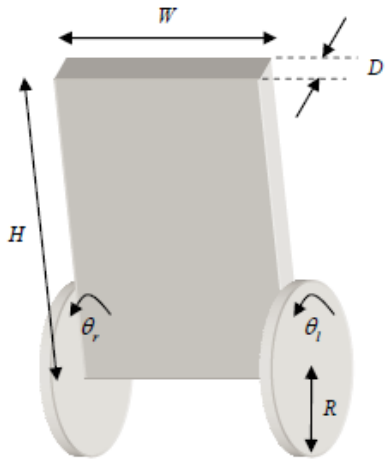

Figure 2: Inverted pendulum representation of the LEGO Mindstorms EV3 robot.

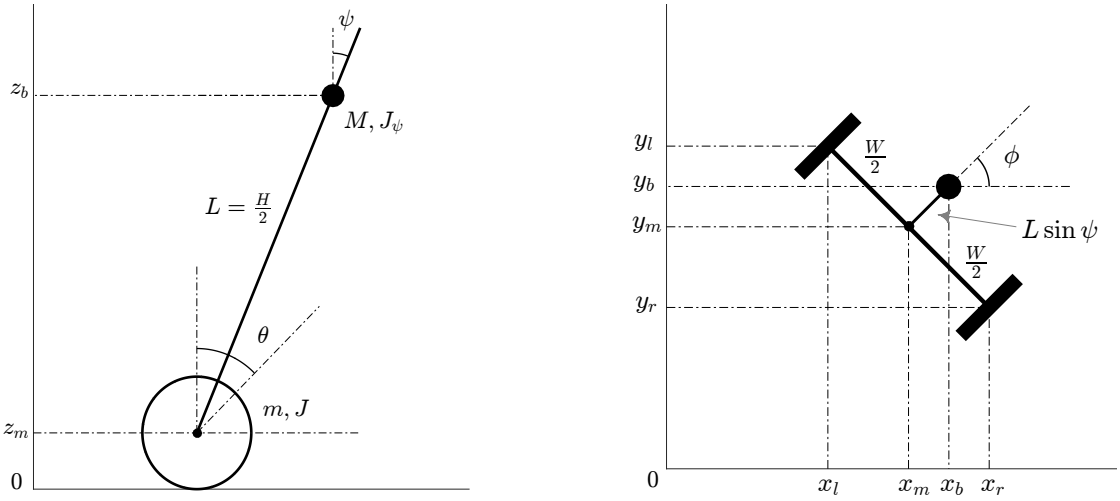

Figure 3: Side and top view of the two-wheeled inverted pendulum.

To see this notice that by the left panel of Figure 4 we have that the arc length travelled by wheel w for an angular displacement $d\theta_w$, incurred within some time interval dt, is equal to $Rd\theta_w$, $w \in \{l, r\}$. Consider now the right panel of Figure 4, where $x - y$ denotes the reference coordinate frame of Figure 3. We then have that for dt (and hence also for $d\phi$) small enough, $d\phi \approx \tan d\phi = \frac{Rd\theta_l - Rd\theta_r}{W}$. However, since an increase by $d\phi$ tends to decrease ϕ according to the convention of Figure 4, we have that

$$\dot{\phi} = -\lim_{dt \to 0} \frac{d\phi}{dt} = -\frac{R}{W}(\dot{\theta}_l - \dot{\theta}_r), \tag{3}$$

which leads to (2). The notation $\frac{d\phi}{dt}$ should not be confused with the time-derivative of ϕ, which is in turn denoted by $\dot{\phi}$, but is the ratio between $d\phi$ and dt. By (2) it follows that the yaw motion requires a differential voltage input.

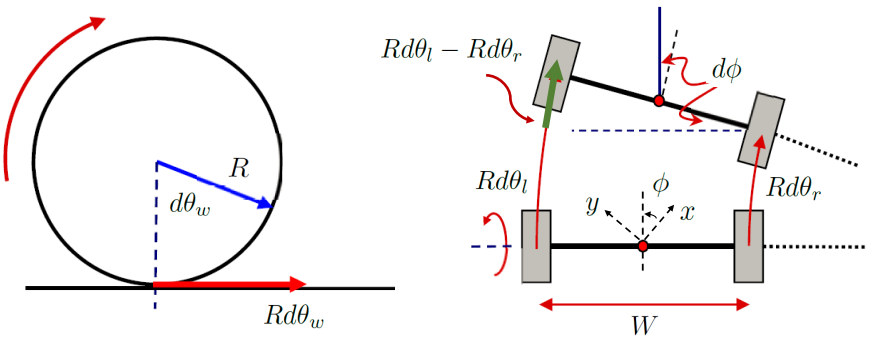

Figure 4: Left panel: Wheel's side view for an angular displacement $d\theta_w$, $w \in \{l, r\}$. Right panel: Top view of the robot's displacement during a right turn.

We have that

$$(x_m, y_m, z_m) = \Big(\int \dot{x}_m dt, \int \dot{y}_m dt, R \Big), \tag{4}$$

where

$$(\dot{x}_m, \dot{y}_m) = (R\dot{\theta}\cos\phi, R\dot{\theta}\sin\phi). \tag{5}$$

Note that $\dot{x}_m, \dot{y}_m$, correspond to the projection of the velocity vector of magnitude $R\dot{\theta}$ on the x and y axis, respectively.

Q 1. Calculate the coordinates (x_l, y_l, z_l) and (x_r, y_r, z_r) of the left and right wheel, respectively, and the coordinates of the middle point of the upper body, as a function of x_m, y_m, z_m, ϕ, and ψ.

For each $w \in \{l, r\}$, let $F_{s,w}$ denote the static friction force due to the contact between wheel w and the ground, whereas let $F_{h,w}$ denote the horizontal reaction force due to the contact between the wheel and the main body. Denote also by $F_{p,w}$ the vertical force due to the contact between each wheel and the main body. Let T_w denote the torque causing rotation of wheel w, produced by the associated DC motor, and by $T_{f,w}$ the component of the friction torque due to the contact between the DC motor and the main body with direction opposite to that of T_w.

Q 2. Provide free body diagrams for each of the wheels and the pendulum.

Each wheel $w \in \{l, r\}$ is driven by a DC motor. The behaviour of the DC motor is captured by the electric circuit of Figure 5, where R_{dc} and L_{dc} denote the resistance and inductance, respectively, of the armature circuit, which is assumed to be the same for both motors. When the armature circuit is placed within a fixed, separately excited field, torque causing rotation is generated. Variable u_w denotes the voltage input, which serves as a control variable for our purposes, I_w denotes the flowing current, and $u_{e,w}$ denotes the induced voltage, referred to as back electromotive force (back EMF), which is proportional to the angular velocity of the motor at wheel w, with proportionality constant K_e.

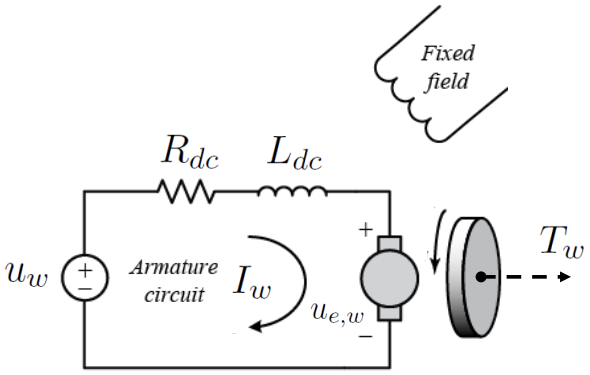

Figure 5: Electric circuit describing the behavior of the DC motor at wheel w, $w \in \{l, r\}$. Torque is generated when the armature circuit is placed within a fixed, separately excited field.

Q 3. Assume that there are no current transients, and let

$$u_1 = \frac{1}{2}(u_l + u_r), \;\; u_2 = \frac{1}{2}(u_l - u_r). \tag{6}$$

Show that

$$(T_l + T_r) - (T_{f,l} + T_{f,r}) = 2\frac{K_T}{R_{dc}}u_1 - 2\Big(\frac{K_T K_e}{R_{dc}} + b\Big)(\dot{\theta} - \dot{\psi}), \tag{7}$$

$$(T_r - T_l) - (T_{f,r} - T_{f,l}) = -2\frac{K_T}{R_{dc}}u_2 - \frac{W}{R}\Big(\frac{K_T K_e}{R_{dc}} + b\Big)\dot{\phi}. \tag{8}$$

Hint: For each $w \in \{l, r\}$, work according to the following steps:

1. **Provide an expression for the back EMF $u_{e,w}$.**
2. **Write I_w as a function of u_w, $\dot{\theta}_w$ and $\dot{\psi}$; you may assume $\dot{I}_w \approx 0$.**
3. **Use the fact that T_w is proportional to the armature current I_w, i.e., $T_w = K_T I_w$.**
4. **Model $T_{f,w}$ as a damper, using the friction coefficient b in Table 1, Appendix A.**

2 Mathematical modelling

We will first design a controller that allows the robot to self-balance and move forward/backward, and, at the next stage, we will design a controller for the yaw motion. To achieve this and decouple the yaw motion we assume that $\phi = 0$, $\dot{\phi} = 0$, and $\ddot{\phi} = 0$, i.e., the robot's heading is aligned with the x axis, when deriving the equations of motion for the wheels and the pendulum. This analysis is approximate, but accurate enough from a control point of view, leading to the same linearized dynamical system. Definitions for all other parameters, treated as constants in our analysis, and their numerical values, are taken from [1], and can be found in Table 1, in Appendix A.

2.1 Pitch and forward/backward motion

Q 4. For each $w = \{r, l\}$, derive the equations of motion for the wheel w due to its horizontal and its rotational motion. Show that they can be combined to form

$$(2mR^2 + 2J)\ddot{\theta} = (T_l + T_r) - (T_{f,l} + T_{f,r}) - (F_{h,l} + F_{h,r})R - 2\bar{b}\dot{\theta}. \tag{9}$$

Hint: To obtain (9), work according to the following steps:

1. **Consider the cartesian motion. Under the assumption $\phi = 0$, obtain an expression for $\ddot{x}_w$, which denotes the wheel's linear acceleration along the x axis.**
2. **Consider the rotational motion. Obtain an expression for $\ddot{\theta}_w$, which denotes the wheel's angular acceleration. In addition to $T_{f,w}$, the friction torque due to rotation along the wheel axis needs to be taken into account as well. To model this, use $\bar{b}$ from Table 1, in Appendix A.**
3. **Express $\ddot{x}_w$ as a function of $\ddot{\theta}_w$ using (5), and use (1) to obtain (9).**

Q 5. Derive the equations of motion for the pendulum, considering both its horizontal motion and its rotation along the pitch angle direction. Show that they are given by

$$M(R\ddot{\theta} + L\cos\psi\ddot{\psi} - L\sin\psi\dot{\psi}^2) = (F_{h,l} + F_{h,r}), \tag{10}$$

$$M(-L\sin\psi\ddot{\psi} - L\cos\psi\dot{\psi}^2) = (F_{p,l} + F_{p,r}) - Mg, \tag{11}$$

$$J_\psi\ddot{\psi} = (T_{f,l} + T_{f,r}) - (T_l + T_r) + (F_{p,l} + F_{p,r})L\sin\psi - (F_{h,l} + F_{h,r})L\cos\psi. \tag{12}$$

Hint: To obtain (10) − (12), work according to the following steps:

1. **Consider the cartesian motion. The pendulum is moving both in the horizontal direction (only along the x axis since $\phi = 0$) and along the vertical direction. Hence, obtain an expression for $\ddot{x}_b$ and $\ddot{z}_b$, which denote the linear acceleration of the centre of mass of the main body along the x axis and z axis, respectively.**
2. **Express $\ddot{x}_b$ and $\ddot{z}_b$ as a function of θ, ψ and their derivatives using (5) and the equations you derived in Q1. This leads to (10) and (11).**
3. **Consider the rotational motion. Obtain an expression for $\ddot{\psi}$, which denotes the angular acceleration along the pitch direction, thus leading to (12).**

Q 6. Combine (9), (10) , (11), **and** (12), **by eliminating unknown force vectors, and show that the nonlinear equations of motion that govern the robot's self-balance and forward/backward motion take the form**

$$MLR\cos\psi\ddot{\theta} + (ML^2 + J_\psi)\ddot{\psi} - MgL\sin\psi = -(T_l + T_r) + (T_{f,l} + T_{f,r}), \tag{13}$$

$$(2mR^2 + 2J + MR^2)\ddot{\theta} + MLR\cos\psi\ddot{\psi} - MLR\sin\psi\dot{\psi}^2 = (T_l + T_r) - (T_{f,l} + T_{f,r}) - 2\bar{b}\dot{\theta}. \tag{14}$$

Q 7. Let $x_1 = [\theta\ \psi\ \dot{\theta}\ \dot{\psi}]^\top \in \mathbb{R}^4$**. Show that the linearization of** (13) **and** (14) **around** $x_1^* = [0\ 0\ 0\ 0]$ **and** $u_1^* = 0$ **(recall the definition of** u_1 **in** (6)**), can be expressed as**

$$\begin{bmatrix} MLR & ML^2 + J_\psi \\ 2mR^2 + 2J + MR^2 & MLR \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = 2 \begin{bmatrix} K_T K_e/R_{dc} + b & -K_T K_e/R_{dc} - b \\ -K_T K_e/R_{dc} - b - \bar{b} & K_T K_e/R_{dc} + b \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & MgL \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \psi \end{bmatrix} + 2 \begin{bmatrix} -K_T/R_{dc} \\ K_T/R_{dc} \end{bmatrix} u_1.$$

Moreover, show that the full linearized equations of motion can be written in state space form as

$$\dot{x}_1 = A_1 x_1 + B_1 u_1, \tag{15}$$

$$y = C_1 x_1 + D_1 u_1, \tag{16}$$

where

$$A_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -gRM^2L^2\frac{1}{q} & -2(MRL + ML^2 + J_\psi)(\frac{K_T K_e}{R_{dc}} + b)\frac{1}{q} - 2(ML^2 + J_\psi)\bar{b}\frac{1}{q} & 2(MRL + ML^2 + J_\psi)(\frac{K_T K_e}{R_{dc}} + b)\frac{1}{q} \\ 0 & gML((2m+M)R^2 + 2J)\frac{1}{q} & 2((2m+M)R^2 + 2J + MRL)(\frac{K_T K_e}{R_{dc}} + b)\frac{1}{q} + 2MRL\bar{b}\frac{1}{q} & -2((2m+M)R^2 + 2J + MRL)(\frac{K_T K_e}{R_{dc}} + b)\frac{1}{q} \end{bmatrix},$$

$$B_1 = \begin{bmatrix} 0 \\ 0 \\ 2(MRL + ML^2 + J_\psi)\frac{K_T}{R_{dc}}\frac{1}{q} \\ -2((2m+M)R^2 + 2J + MRL)\frac{K_T}{R_{dc}}\frac{1}{q} \end{bmatrix},\ C_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ D_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

and $q = ((2m+M)R^2 + 2J)(ML^2 + J_\psi) - M^2R^2L^2$**.**

Note that the output corresponds to θ and $\dot{\psi}$. In practice, $\dot{\psi}$ is directly available via the gyro sensor, whereas θ can be calculated by means of the DC motor angle measurement $\theta - \psi$, where ψ is calculated from $\dot{\psi}$ by means of numerical integration.

2.2 Yaw motion

Notice that the sum of the motors' voltages $u_l + u_r$ is responsible for the forward/backward motion of the robot. As suggested by **Q 3**, the yaw motion emanates from a differential voltage, i.e., $u_l - u_r$. Setting $x_2 = [\phi\ \dot{\phi}]^\top \in \mathbb{R}^2$, the yaw dynamics linearized around $x_2^* = [0\ 0]^\top$, $u_2^* = 0$ are given by the state space form

$$\dot{x}_2 = A_2 x_2 + B_2 u_2, \tag{17}$$

$$y = C_2 x_2 + D_2 u_2, \tag{18}$$

where

$$A_2 = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{W^2}{2R^2}(\frac{K_T K_e}{R_{dc}} + b + \bar{b})\frac{1}{\frac{1}{2}mW^2 + J_\phi + \frac{JW^2}{2R^2}} \end{bmatrix},\ B_2 = \begin{bmatrix} 0 \\ -\frac{W}{R}\frac{K_T}{R_{dc}}\frac{1}{\frac{1}{2}mW^2 + J_\phi + \frac{JW^2}{2R^2}} \end{bmatrix},\ C_2 = \begin{bmatrix} 1 & 0 \end{bmatrix},\ D_2 = 0.$$

Note that we have assumed that the yaw angle ϕ can be measured directly.

2.3 Modeling in MATLAB Simulink

Go to the `LEGO_SW` folder in your directory. Two files are needed for this task: the `run_LEGO_SW.m` MATLAB file, where all parameters shall be defined, and the Simulink file `LEGO_SW.slx`. The overall Simulink architecture exhibits the form of Figure 6, where the `Signal Builder` blocks provide the means to design reference trajectories for $\dot{\theta}^{\text{ref}}$ and $\dot{\phi}^{\text{ref}}$.

Q 8. Define values for all necessary parameters according to Table 1, Appendix A, in `run_LEGO_SW.m`. Use the `Simulink Library Browser` to fill in the subsystem `robot` of `LEGO_SW.slx` shown in Figure 6.
Hint: Check the `State-Space` block as a modelling option from the `Simulink Library Browser`.

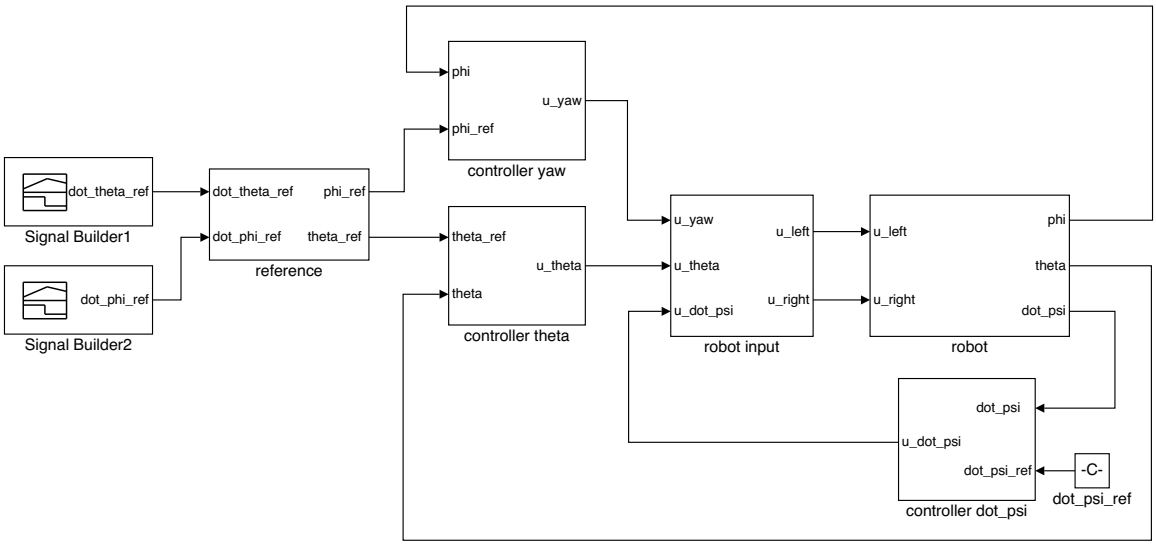

Figure 6: Simulink architecture of `LEGO_SW.slx`.

3 Control design – Self-balancing and forward/backward motion

We aim at designing u_1 for the two-wheeled inverted pendulum to self-balance, and track some reference signal $x_1^{\text{ref}} = [\theta^{\text{ref}} \ \psi^{\text{ref}} \ \dot{\theta}^{\text{ref}} \ \dot{\psi}^{\text{ref}}]^\top$, such that $\psi^{\text{ref}} = \dot{\psi}^{\text{ref}} = 0$, i.e., the robot should move forward/backward with the pendulum balancing along the vertical position. Notice that, since it can be shown that x_1^* is an unstable equilibrium of (15), even when $x_1^{\text{ref}} = x_1^*$ and the robot is not moving, a controller is needed to stabilize the system at x_1^*, and the pendulum to self-balance. To achieve this task we consider the feedback control structure of Figure 7.

Recall that the output y includes the states θ and $\dot{\psi}$. We employ a Proportional-Integral (PI) controller and a Proportional-Integral-Derivative (PID) controller to regulate $\dot{\psi}$ and θ, to their reference values, respectively. Notice that the PI controller for $\dot{\psi}^{\text{ref}} - \dot{\psi}$ is effectively a proportional-derivative controller assuming feedback of the states ψ, $\dot{\psi}$, with gains $K_{\dot{\psi}}^I$, $K_{\dot{\psi}}^P$, respectively, and the state ψ being reconstructed from $\dot{\psi}$ via numerical integration. Similarly, the PID controller could be thought of as the composition of a PI controller concerning $\theta^{\text{ref}} - \theta$, with a proportional controller with gain K_θ^D for $\dot{\theta}^{\text{ref}} - \dot{\theta}$, with the latter being constructed via numerical differentiation from $\theta^{\text{ref}} - \theta$.

3.1 Self-balancing pitch PI controller

Consider the system of Figure 7. We start by designing the inner, PI controller so that the robot can self-balance. To this end, we aim at choosing the feedback gains $K_{\dot{\psi}}^P$, $K_{\dot{\psi}}^I$ that are related to the pendulum's pitch angle so that the unstable system becomes stable.

Q 9. Using MATLAB, determine the system's transfer function (matrix) $G(s)$ from $u_1 \to y$. Denote by G_θ and $G_{\dot{\psi}}$ the transfer functions from $u_1 \to \theta$ and from $u_1 \to \dot{\psi}$, respectively. Is the system stable?
Hint: Check MATLAB commands `ss` and `zpk`.

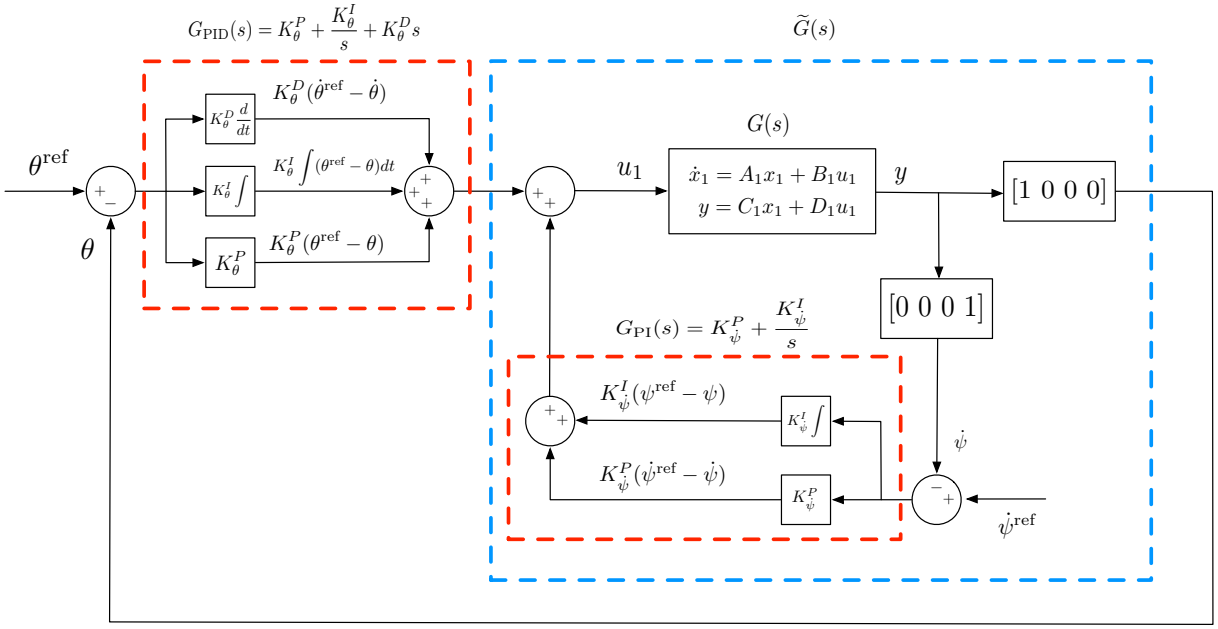

Figure 7: Feedback control structure.

The transfer function of the PI controller is given by

$$G_{\mathrm{PI}}(s) = K_{\dot{\psi}}^P + \frac{K_{\dot{\psi}}^I}{s} = K_{\dot{\psi}}^P \left(\frac{s + K_{\dot{\psi}}^I / K_{\dot{\psi}}^P}{s} \right).$$

Note from Figure 7 that $G_{\mathrm{PI}}(s)$ is connected with negative feedback. However, roughly speaking, $G_{\dot{\psi}}$ maps "positive input" u_1 to "negative output" $\dot{\psi}$. Consequently, we should expect that positive feedback will be more appropriate to stabilize $G_{\dot{\psi}}$, with $\dot{\psi}^{\mathrm{ref}} = 0$. This can be achieved by using a negative $K_{\dot{\psi}}^P$ gain. MATLAB's `rlocus` command allows only positive gains, so to consider negative gains, simply negate the argument to `rlocus`.

Q10. Using MATLAB, choose the gains $K_{\dot{\psi}}^P$, $K_{\dot{\psi}}^I$ of the PI pitch controller.

1. **Place first the zero $z_{\mathbf{PI}} = -K_{\dot{\psi}}^I / K_{\dot{\psi}}^P$ of the controller appropriately.**
2. **Determine the gain $K_{\dot{\psi}}^P$ by checking the root locus of the open-loop transfer function $G_{\dot{\psi}} \frac{s + K_{\dot{\psi}}^I / K_{\dot{\psi}}^P}{s}$. Plot the negation of this function since MATLAB allows only for positive gains in root locus.**

Q11. Determine the transfer function $\widetilde{G}(s)$ of the inner loop system of Figure 7. Is the corresponding system stable?
Hint: Check MATLAB command `minreal`, and use the equivalence between the system's state space representation and the associated transfer functions computed in Q9 as shown in Figure 8.

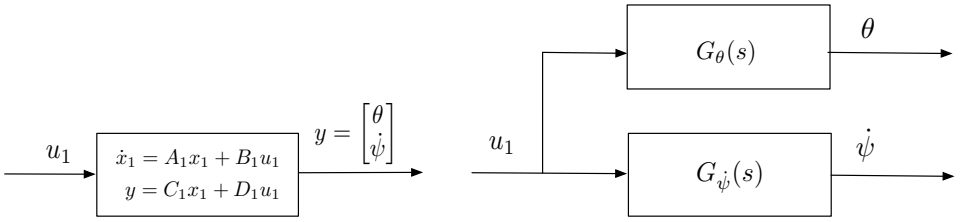

Figure 8: State space representation of the plant appearing in Figure 7 and the associated transfer functions.

3.2 Forward/backward motion PID controller

By inspection of $\widetilde{G}(s)$ it should be observed that there is one pole at the origin. As a result, the system will be unstable after a step input. We will thus design a PID controller to ensure a stable closed-loop performance. Moreover, $\widetilde{G}(s)$ exhibits one (odd number) of right-half plane zeros and is strictly proper; we thus anticipate an initial undershoot in the system's response after a step reference increase. Note that the presence of a right-half plane zero imposes limitations on the cross-over frequency, and hence the bandwidth. In particular, the "slower" the right-half plane zero, the lower the cross-over frequency is expected to be (see Chapter 11.7 in [6]).

Control design problem: Consider the following stability and performance specifications:
1. Phase margin PM greater than $50°$.
2. Step response settling time T_s less than $10s$.

The transfer function of the PID controller is given by

$$G_{\text{PID}}(s) = K_\theta^P + \frac{K_\theta^I}{s} + K_\theta^D s = K_\theta^D \frac{s^2 + (K_\theta^P/K_\theta^D)s + (K_\theta^I/K_\theta^D)}{s}. \tag{19}$$

Q12. Using MATLAB, choose the gains K_θ^P, K_θ^I and K_θ^D of the PID controller.

1. **Turn the control design specifications into requirements for the damping ratio ζ and the natural frequency ω_n.**
2. **Place the zeros of the PID controller in the desired $\zeta - \omega_n$ region, by determining $r_{\mathbf{PD}} = K_\theta^P/K_\theta^D$ and $r_{\mathbf{ID}} = K_\theta^I/K_\theta^D$.**
3. **Determine the range of admissible K_θ^D values by the root locus of the open loop transfer function $\frac{s^2+(K_\theta^P/K_\theta^D)s+(K_\theta^I/K_\theta^D)}{s}\widetilde{G}(s)$ (so that the closed loop system is stable). Plot the negation of this function since MATLAB allows only for positive gains in root locus.**

4. **Choose the gain K_θ^D in this range, ensuring (from the root locus) that the closed loop system will operate in the desired $\zeta - \omega_n$ region. Verify by the Bode plot that the desired phase margin specification is satisfied.**
   Hint: Check MATLAB command `margin`.

Ensure to negate the selected gains since the root locus of the negation of the transfer function was considered.

Q13.
1. **Calculate the transfer function of the closed loop system, i.e., from $\theta^{\text{ref}} \to \theta$ with reference to Figure 7.**
2. **Verify the settling time specification by simulating a unit step response of the closed loop system.**
   Hint: Check MATLAB command `step`.

Q14. Comment on the stability of the closed loop system, by a method of your choice, e.g. computing the closed-loop pole locations, the Nyquist diagram, or Bode plot.

For a similar PID design procedure we also refer to [7] (Example 9.10, Chapter 9).

3.3 Modelling in MATLAB Simulink

Go to the `run_LEGO_SW.m` MATLAB file, define values for the control gains as calculated in the aforementioned tasks, and recall the general architecture as shown in Figure 6.

Q15. Use the `Simulink Library Browser` to fill in subsystems `reference`, `controller dot_psi`, `controller theta` and `robot input` of `LEGO_SW.slx` shown in Figure 6.

4 Control design – Yaw motion

4.1 Yaw motion

We consider now the task of designing u_2, which governs the yaw motion of the two-wheeled inverted pendulum. By inspection of A_2, B_2, it can be observed that (17) corresponds to a critically stable dynamical system. In particular, it is in controllable canonical form (see Chapter 9 in [8]), hence the choice of gains for a state feedback controller via pole placement is anticipated to be straightforward. For this particular structure, however, applying a constant input u_2 is sufficient for stabilizing the second component $\dot{\phi}$ of x_2 to some desired reference value $\dot{\phi}^{\text{ref}}$ (we assume that $\ddot{\phi}^{\text{ref}} = 0$); using a state feedback we could also enforce the desired stabilization rate.

To see this, denote by $A_2^{(2,2)}$, $B_2^{(2)}$ the corresponding elements of A_2 and B_2, respectively. Let

$$u_2 = -\frac{A_2^{(2,2)}}{B_2^{(2)}}\dot{\phi}^{\text{ref}}. \tag{20}$$

By (20), (17), we then have that

$$(\ddot{\phi} - \ddot{\phi}^{\text{ref}}) = A_2^{(2,2)}(\dot{\phi} - \dot{\phi}^{\text{ref}}). \tag{21}$$

Notice that $A_2^{(2,2)} < 0$, hence we can stabilize $\dot{\phi}$ to $\dot{\phi}^{\text{ref}}$, and as a result ϕ to ϕ^{ref}, where the latter is the integral of $\dot{\phi}^{\text{ref}}$.

Q 16. Is (20) **an open or a closed loop control design? What would you expect if controller** (20) **is applied to the LEGO Mindstorms EV3 robot?**

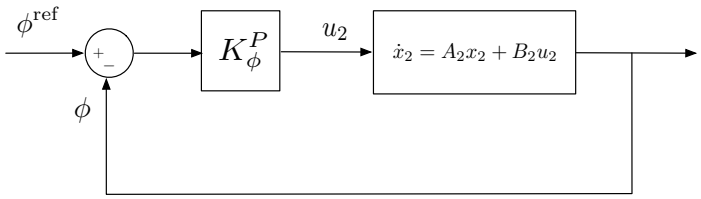

Figure 9: Feedback control for yaw motion.

Consider the block diagram structure of Figure 9. A Proportional (P) controller with gain K_ϕ^P is employed now to regulate ϕ to ϕ^{ref}.

Q17. Consider the matrices C_2 and D_2 defined in Section 2.2, and choose the gain K_ϕ^P of the proportional yaw controller. This can be done via simulation to ensure that the closed loop system is stable and sufficiently fast.

4.2 Modelling in MATLAB Simulink

Go to the `run_LEGO_SW.m` MATLAB file, define values for the gains associated with the yaw controller as calculated in the aforementioned tasks, and recall the general architecture as shown in Figure 6.

Q 18. Use the `Simulink Library Browser` to fill in subsystem `controller yaw` of `LEGO_SW.slx` shown in Figure 6. Consider two separate cases according to whether (20) **or the proportional controller of Q17 is employed.**

5 Control deployment

5.1 Simulation in MATLAB Simulink

Go to the `LEGO_SW` folder in your directory, and consider the `run_LEGO_SW.m` MATLAB file, and the Simulink file `LEGO_SW.slx`. With reference to Figure 6, all subsystems should be now filled in and connected.
Ensure that all parameters and control gains needed in the Simulink file are defined in the file `run_LEGO_SW.m`, and run both files.

Q 19. Plot the time evolution of the states x_1 and x_2 of the robot to investigate its performance in terms of tracking reference signals. Use the `Signal Builder` block of the `Simulink Library Browser` to construct reference signals that involve both positive and negative step changes.
Hint: Simulink scopes can be used to log signals to the workspace; see `view / configuration properties.../logging` of the scope of interest.

Q 20. Plot in separate graphs the trajectory traversed by the wheel's middle point x_m and y_m, respectively, as a function of time.

5.2 Deployment on LEGO Mindstorms EV3

Go to the `LEGO_HW` folder in your directory. Two files are needed for this task: the `run_LEGO_HW.m` MATLAB file, where all parameters shall be defined, and the Simulink file `LEGO_HW.slx`. The following tasks are to be completed:

Software. The Simulink architecture in `LEGO_HW.slx` is similar to that of Figure 6, with the difference that the controllers are now implemented in discrete time, and the state space representation of the robot should now be replaced by LEGO Mindstorms EV3 blocks from the relevant Simulink library. There are also some additional blocks to allow for control via the gamepad, but we can ignore these for now.
To prepare for implementation on the robot, follow the subsequent steps:

1. Go to the `robot` subsystem, and then enter in the `EV3 Robot` subsystem that can be found therein.
2. Open the `Simulink Support Package for LEGO MINDSTORMS EVE3 Hardware` that can be found in the `Simulink Library Browser`. You should find a list of blocks as shown in Figure 10.
3. Drag and drop from this list to the `EV3 Robot`, two `Motor`, two `Encoder` and one `Gyro Sensor` blocks.
4. Connect to the `Motor` blocks the outputs of the `robot input` subsystem, and use the `Encoder` and the `Gyro Sensor` to obtain sensor measurements for the angle of the left and right motors, and the gyro output, respectively.
5. Connect all sensor signals to a `Bus Creator` block selected from the `Simulink Library Browser`. The output of this bus constitutes the `Raw Sensors` output of the `EV3 Robot` subsystem.

Ensure that the EV3 brick input port appearing in the `Gyro Sensor` block corresponds to the port where the gyro sensor is connected in the robot. Moreover, since we have a discrete time implementation, ensure that in all blocks the sample time is set to T_s (its numerical value is provided in the `run_LEGO_HW.m`). As an example see Figure 11, while the same comments apply for the other blocks. Moreover, the name of the signals appearing in the `Bus Creator` block should be in correspondence with the ones appearing in the `Bus Selector` block in the `Sensor Processing` subsystem.
Once the `EV3 Robot` subsystem is filled in, **proceed to fill in the remaining blocks**, namely `Sensor Processing`, `controller yaw`, `controller theta`, and `controller dot_psi`. Ensure that all parameters and control gains needed in the Simulink file are defined in `run_LEGO_HW.m` (and don't forget to run the file).

Hardware. The LEGO Mindstorms EV3 robot requires configuring so as to communicate with MATLAB/Simulink via Wi-Fi. To connect, follow the procedure outlined in `LEGO_Football_guidelines_to_connect.pdf`. Again, the Gamepad will not be required at this stage, as you will re-use the reference signals you designed in Q19. To deploy the controller on the robot, run `LEGO_HW.slx`.

Reset. The `LEGO_HW.slx` diagram also includes *reset* functionality to facilitate testing. Pressing the `Touch Sensor` (red button) on the EV3 will cut the power to the robot's motors. Pressing the `Touch Sensor` a second time will reset the controller's integrators (and any reference trajectories) and restore power to the motors. For example: if the robot falls, press the `Touch Sensor` once to stop the motors. Manually place the robot upright again, and press the `Touch Sensor` again to continue.

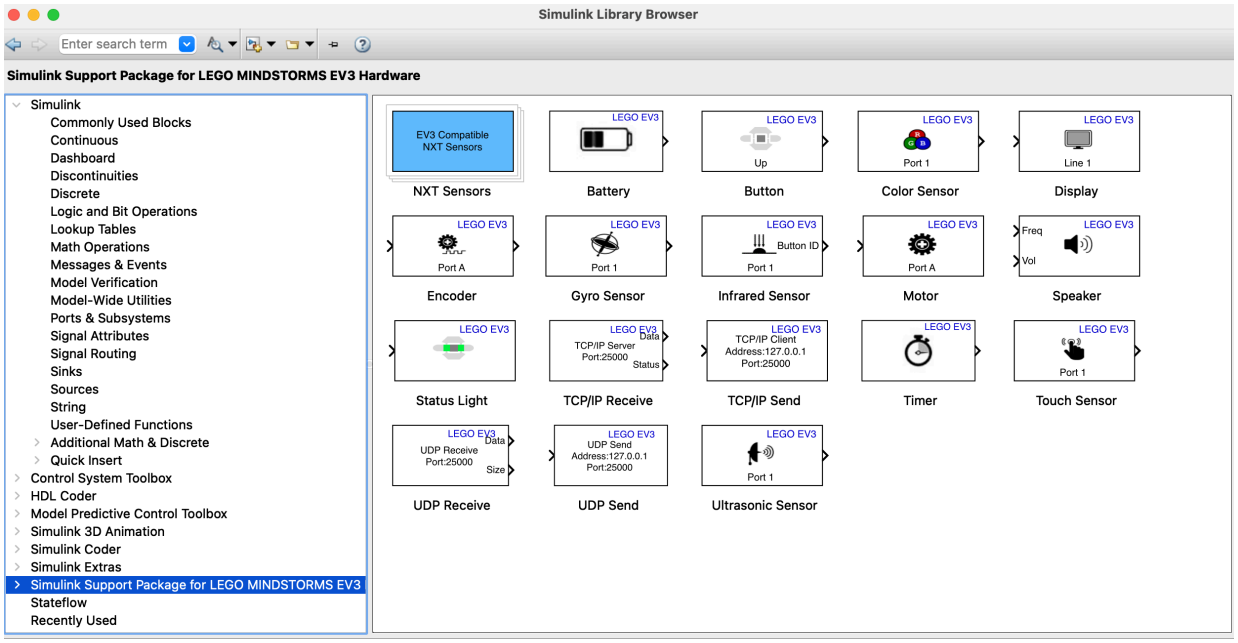

Figure 10: Simulink library related to the support package for LEGO Mindstorms EV3 hardware.

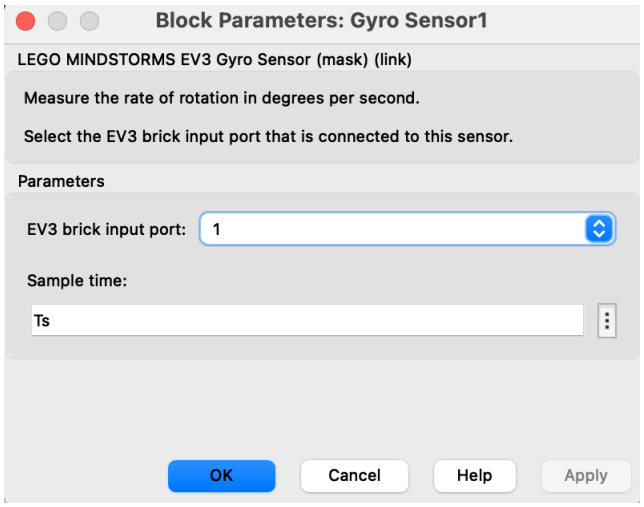

Figure 11: `Gyro Sensor` block.

Q 21. Plot the time evolution of the states x_1 and x_2 of the robot to investigate its performance in terms of tracking the reference signals.

Q 22. Compare reference tracking when using the yaw controller of (20) versus the proportional controller of Q17. What do you observe compared to your answer in Q18?

Q 23. Determine appropriate reference signals such that the robot travels from the goal box, to the halfway line, where it performs a 180 degree turn, before returning to the goal box.
Hint: Remember that the reset button also resets the reference trajectories. There is no need to rebuild the Simulink model.

5.3 Driving the robot via a gamepad

The `LEGO_HW.slx` diagram already contains an interface to the gamepad controller; cf. the `Gamepad` block in the top left of the `LEGO_HW.slx` diagram. The `Gamepad` block provides four outputs:

1. `button_L1`, which stops the motors of the EV3 robot and resets the integrators in the Simulink controllers.
2. `dot_theta_ref`, which is controlled by the up/down motion of the left stick.
3. `dot_phi_ref`, which is controlled by the left/right motion of the left stick.
4. `kicker`, which is controlled by the left/right motion of the left stick.

All output from the gamepad sticks varies between -1 and 1.
Reconfigure `LEGO_HW.slx` so that `Gamepad` block provides reference signals to your controller. Note that the `Reference Selector` block already contains a switch to toggle between the gamepad and the signal generators. Ensure that you scale the output of the `Gamepad` block

appropriately. Note also that the same reset functionality can be achieved with the gamepad: the L1 button on the gamepad has the same function as the red reset button on the EV3.

Q 24. Drive your robot around using the gamepad. Be sure to tune the gains that amplify the signals from the gamepad to ensure that you can track reference signals suitable for playing football (e.g. abrupt stopping and turning). Use an additional motor so that your robot can kick/pass the ball.
Hint: when building the Simulink model, use `Deploy` instead of `Monitor and Tune`. This reduces the communication load, and helps the controller to run with a more consistent sample time.

Your robot should be ready to play football! The overall set-up should look like the one of Figures 12 and 13.

Figure 12: Place your robot on the football rug and start playing football!

Figure 13: Laboratory set-up with 6 Mindstorms EV3 robots.

6 Control design via pole placement

We consider an alternative control design technique. To this end, consider the architecture of Figure 7. Set $z = \int(\theta - \theta^{\text{ref}})dt$, thus leading to $\dot{z} = \theta - \theta^{\text{ref}}$. Let $K = \begin{bmatrix} K_\theta^P & K_{\dot{\psi}}^I & K_\theta^D & K_{\dot{\psi}}^P & K_\theta^I \end{bmatrix}^\top \in \mathbb{R}^5$, and denote by $\bar{C}_1$ the first row of C_1. By inspection of Figure 7, we then have that

$$u_1 = -K \begin{bmatrix} x_1 \\ z \end{bmatrix} + K \begin{bmatrix} x_1^{\text{ref}} \\ 0 \end{bmatrix}. \tag{22}$$

By (15) and (22), we have for the closed-loop system that

$$\begin{bmatrix} \dot{x}_1 \\ \dot{z} \end{bmatrix} = \left(\begin{bmatrix} A_1 & 0_{4\times 1} \\ \bar{C}_1 & 0 \end{bmatrix} - \begin{bmatrix} B_1 \\ 0 \end{bmatrix} K \right) \begin{bmatrix} x_1 \\ z \end{bmatrix} + \begin{bmatrix} B_1 K \\ [-\bar{C}_1 \; 0] \end{bmatrix} \begin{bmatrix} x_1^{\text{ref}} \\ 0 \end{bmatrix}. \tag{23}$$

Note that he open-loop system is unstable, since A_1 has one eigenvalue with positive real part. We can determine K by placing the poles (eigenvalues) of $\left(\begin{bmatrix} A_1 & 0_{4\times 1} \\ \bar{C}_1 & 0 \end{bmatrix} - \begin{bmatrix} B_1 \\ 0 \end{bmatrix} K \right)$ at some targetlocation $p = [p_1 \; p_2 \; p_3 \; p_4 \; p_5]$, so that the closed-loop system is stable (all elements of p should have negative real parts). This can be achieved by means of the so called *pole placement* procedure.

Pole placement procedure:

1. **Compute the target charectiristic polynomial (whose roots are $p_1, \ldots, p_5$). This is given by $\Pi_{i=1}^5 (s - p_i)$.**
2. **Compute the characteristic polynomial associated with the closed loop system whose state space matrix is given by $\left(\begin{bmatrix} A_1 & 0_{4\times 1} \\ \bar{C}_1 & 0 \end{bmatrix} - \begin{bmatrix} B_1 \\ 0 \end{bmatrix} K \right)$. This is given by $\det \left(sI - \left(\begin{bmatrix} A_1 & 0_{4\times 1} \\ \bar{C}_1 & 0 \end{bmatrix} - \begin{bmatrix} B_1 \\ 0 \end{bmatrix} K \right) \right)$.**
3. **Equate the coefficients of the two characteristic polynomials. Notice that the coefficients of the one of step 2 depend on the gains in K that need to be determined.**

Note that the characteristic polynomial of a given matrix A is given by the determinant $\det(sI - A)$, which is in turn the denominator of the system's transfer function, and when equating it with zero, its roots correspond to the poles of the system. I denotes an identity matrix of appropriate dimension. The aforementioned procedure results in solving a system of 5 equations (since the characteristic polynomials are of 5-th order), linear in the gains in K. For more details the reader is referred to [8] (Chapter 9). However, this can be achieved using MATLAB.

Q 25. Choose p so that a complex pole pair is dominant, meeting the control design specifications of Section 3.2. Calculate the control gain vector K that ensures that the poles of the closed-loop system are in the locations contained in p.
Hint: Check the MATLAB command `place`.

Q 26. Compare the performance of the set of gains calculated via pole placement with those calculated via root locus and bode plots. Perform any fine tuning if appropriate.

Q 27. Consider the Simulink architecture of `LEGO_SW.slx`, as illustrated in Figure 6. Simplify the model to include only two state space Simulink blocks and two state feedback control loops, with the one feedback gain being the vector K.

The overall laboratory set-up is in the form of Figure 14. Each robot is still automatically stabilized via the developed control scheme, but is driven manually by setting the reference values $\dot{\theta}^{\text{ref}}$ and $\dot{\phi}^{\text{ref}}$ in real-time via the gamepad according to Figure 14.

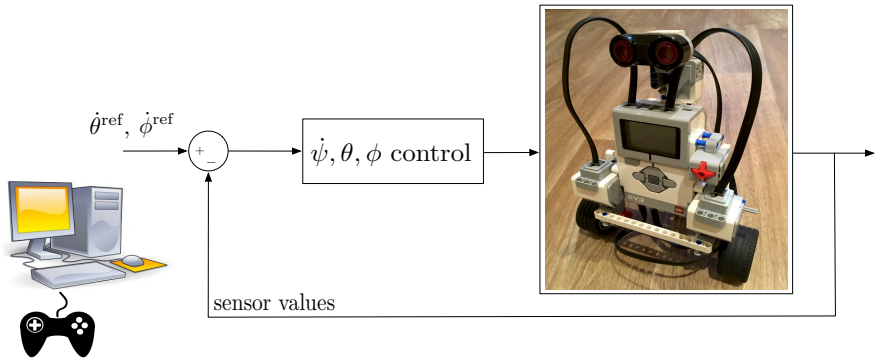

Figure 14: Real-time steering principle, where $\dot{\psi}$ control refers the the PI controller regulating the pitch angle to zero, θ control refers to the PID controller regulating the forward/backward motion of the EV3, and ϕ control refers to the P controller regulating the yaw motion.

Q 28. How many feedback control loops do you recognize in Figure 14?

Appendix A

Table 1: Parameters and numerical values.

Symbol	Value	Measurement units	Physical meaning
g	9.81	m/s^2	gravity acceleration
m	0.024	kg	wheel's mass
M	0.6	kg	main body's mass
R	0.027	m	wheel's radius
D	0.04	m	main body's depth
W	0.14	m	main body's width
H	0.22	m	main body's length
L	$\frac{H}{2}$	m	main body's half-length
J	$\frac{mR^2}{2}$	kgm^2	wheel's moment of inertia
J_ψ	$\frac{ML^2}{3}$	kgm^2	pitch moment of inertia
J_ϕ	$\frac{M(W^2+D^2)}{12}$	kgm^2	yaw moment of inertia
J_{dc}	$1e^{-5}$	kgm^2	DC motor's moment of inertia
R_{dc}	6.69	Ω	DC motor's resistance
K_e	0.468	Vs/rad	back EMF constant
K_T	0.317	Nm/A	DC motor's torque constant
b	0.0022	Nms/rad	friction coefficient between main body and DC motor
$\bar{b}$	0.0022	Nms/rad	friction coefficient due to rotation along the wheel axis

References

[1] Y. Yamamoto, "NXTway-GS Model-Based Design - Control of Self-balancing two-wheeled robot build with LEGO Mindstorms NXT," *Technical Report*, pp. 1–73, 2009. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/19147-nxtway-gs–self-balancing-two-wheeled-robot–controller-design

[2] "Simulink and LEGO MINDSTORMS NXT," *Technical Report, MathWorks, Inc.*, pp. 1–23, 2013.

[3] Y. Kim, S. Kim, and Y. Kwack, "Dynamic analysis of a non-holonomic two-wheeled inverted pendulum robot," *Journal of Intelligent and Robotic Systems*, vol. 44, no. 1, pp. 25–46, 2005.

[4] B. Bonafilia, N. Gustafsson, P. Nyman, and S. Nilsson, "Self-balancing two-wheeled robot," *Technical Report, Chalmers University of Technology*, pp. 1–11, 2013. [Online]. Available: http://sebastiannilsson.com/wp-content/uploads/2013/05/Self-balancing-two-wheeled-robot-report.pdf

[5] "NXT SegWay Robot, Lesson 1," *Technical Report, SolidWorks*, pp. 1–45, 2013.

[6] K. Astrom and R. Murray, "Feedback Systems – An Introduction for Scientists and Engineers," *Princeton University Press, Twelfth Edition*, 2008.

[7] R. Dorf and R. Bishop, "Modern Control Systems," *Prentice Hall, Twelfth Edition*, 2011.

[8] J. Lygeros and F. Ramponi, "Lecture Notes on Linear System Theory," *ETH Zurich*, pp. 1–158, 2013.