The Beagle Project

Marcus Greiff - marcusgreiff.93@hotmail.com

This document details the current status of the beagle project with some helpful references. It is a work in progress, so read it with a pinch of salt!

April 24, 2017

Contents

Project Objectives

The objective of the project is to implement a set of miniature lab processes primarily for use at LTH. If successful, these will also be made open source to make simple experiments involving control theory more accessible. The considered processes are (i) a simple DC servo, (ii) the ball and beam in (ii) 1D and (iii) 2D, as well as (iv) the Furuta pendulum. In order to reach the end-goal, the project will naturally require some theory in terms of modelling (see **Section 1**) and development of controllers (see **Section 2**). This will primarily be done in Simulink, but could eventually be migrated to Python/Julia. As for the real-time implementation, the Beagle Bone Black (BBB) [1] will be used as middleware, for which the communication protocols and software must be written (see **Section 4**). In addition, the electronics (see **Section 5**) and hardware (see **Section 6**) for the processes will be designed from scratch. Initially, will be running Ubuntu 14.04 with a ROS Indigo installation [2] [3], but at later stages in the project, the interface on the host side will be re-written in Julia [4] and provide plugins to Simulink as well using the BBB/Simulink interface developed by Mathworks [5].

1 Modelling

As the processes themselves are relatively simple, much of the modelling has already been done. See for instance [6] for DC motor models, [7] for the ball and beam type models and [8] for furuta pendulum models. The purpose of this section is to present the models coherently and define the parameters and states used in later sections and the on controllers and software implementations.

1.1 DC Motor

In this section we consider the modelling of a brushed DC motor using Newton's second law and Kirchhoff's laws. Consider a motor where J_m $[kg \cdot m/s^2]$ denotes the rotor shaft moment of inertia and $\ddot{\theta}_m(t) = \dot{\omega}_m(t) = \alpha_m(t)$ $[rad/s^2]$ denotes the angular acceleration of the rotor shaft. If the motor generates a torque $\tau_m(t)$ along the rotational axis while subjected to a load torque $\tau_l(t)$, Newton's second law yields

$$J_m \alpha_m(t) = \tau_m(t) - \tau_l(t) - b\omega_m(t) \qquad [N \cdot m]. \tag{1}$$

where the viscous friction inherent to the motor is considered proportional to the rotational speed with some constant b $[N \cdot m \cdot s]$. Now, if we let the generated torque $\tau_m(t)$ be proportional to the armature current $I(t)$ by some constant K_t $[Nm/A]$,

$$\ddot{\theta}_m(t) = \alpha_m(t) = \frac{1}{J_m}\Big(K_t I(t) - \tau_l(t) - b\omega_m(t)\Big) \qquad [rad/s^2]. \tag{2}$$

Similarly to the mechanical equation, we denote the motor inductance L $[H]$ and the coil resistance of R $[Ohm]$. Applying a voltage $U(t)$ $[V]$ to the motor and denoting the resulting current $I(t)$, Kirchhoff's voltage law results in the equation

$$L\dot{I}(t) = -RI(t) + U(t) - e(t) \tag{3}$$

Now, assuming that the electro-motive force (back-EMF) $e(t)$ $[V]$ is proportional to the rotor speed $\omega_m(t)$ by a constant K_e $[V/rad/s]$, this equation be written

$$\dot{I}(t) = \frac{1}{L}\Big(- RI(t) + U(t) - K_e\omega_m(t)\Big). \tag{4}$$

Combining (2) and (4), we may define a state vector and control signal vector

$$\mathbf{x}_m(t) = \begin{bmatrix}\theta_m(t) & \dot{\theta}_m(t) & I(t)\end{bmatrix}^T, \qquad \mathbf{u}_m(t) = \begin{bmatrix}\tau_l(t) & U(t)\end{bmatrix}^T, \tag{5}$$

resulting in an LTI model

$$\mathbf{x}_m(t) = \mathbf{A}\mathbf{x}_m(t) + \mathbf{B}\mathbf{u}_m(t) \tag{6}$$
$$\mathbf{y}_m(t) = \mathbf{C}\mathbf{x}_m(t) \tag{7}$$

where

$$\mathbf{A}_m = \begin{bmatrix}0 & 1 & 0\\ 0 & -b/J_m & K_t/J_m\\ 0 & -K_e/L & -R/L\end{bmatrix}, \quad \mathbf{B}_m = \begin{bmatrix}0 & 0\\ 0 & -1\\ 1/L & 0\end{bmatrix}, \tag{8}$$

and $\mathbf{C}_m$ is chosen to reflect the available sensory information. This model may be used for state estimation in the real time implementation and model based rotor speed control, both in continuous and discrete time. Comparisons between the discrete and continuous time models can be done by running the `run_project.m` script with the `model="DC_motor"` and `controller="response"`. The model is implemented in both continuous and discrete time, both in Simulink and in Python.

1.2 Stepper motor

TODO: Not yet written, but the idea is to simulate the system with a second order process and force the real-time system to comply with it's speed profile to enable analysis of combined models using steppers.

1.3 Ball and Beam process in one dimension

The ball and beam process may, similarly to the Furuta pendulum, be modelled using the Lagrangian method. Consider a ball of radius R [m] and mass m $[kg]$ with a moment of inertia J_b $[kg/m^2]$ about the rotational axis, moving with a rotational speed $\omega(t)$ on the beam or arm. The arm itself rotates about the origin in the $\hat{\mathbf{y}}\hat{\mathbf{z}}$-plane, has a moment of inertia J_a $[kg/m^2]$ and a length L [m]. In this setup, the position of the ball on the arm is denoted $r(t)$ $[m]$ from the origin. First, we form the substate vector $\mathbf{q}(t) = [\alpha(t), r(t)]$ and find the system kinetic and potential energies $E_{kin}(\mathbf{q}, \dot{\mathbf{q}})$ and $E_{pot}(\mathbf{q}, \dot{\mathbf{q}})$, such that the total energy of the system is expressed

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = E_{kin}(\mathbf{q}, \dot{\mathbf{q}}) - E_{pot}(\mathbf{q}). \tag{9}$$

The potential energy of the system may be written

$$E_{pot}(\mathbf{q}) = MgL\sin(\alpha(t)) + mgr(t)\sin(\alpha(t)) = (MgL + mgr(t))\sin(\alpha(t). \tag{10}$$

Similarly, the kinetic energy of is found as

$$E_{kin}(\mathbf{q},\dot{\mathbf{q}}) = \underbrace{\frac{1}{2}J_a\dot{\alpha}(t)^2}_{arm} + \underbrace{\frac{1}{2}mr(t)^2\dot{\alpha}(t)^2 + \frac{1}{2}m\dot{r}(t)^2 + \frac{1}{2}J_b\omega(t)^2}_{ball} = \frac{1}{2}\Big[(J_a+mr(t)^2)\dot{\alpha}(t)^2+\frac{7}{5}m\dot{r}(t)\Big] \tag{11}$$

where, for the last equality, we have made use of the relation $\dot{r}(t) = R\omega(r)$ and fact that the moment of inertia of a sphere, may be expressed

$$J_b = \frac{2}{5}mR^2. \tag{12}$$

The Netwon-Euler equation takes the form

$$\frac{d}{dt}\Big(\frac{\partial\mathcal{L}(\mathbf{q},\dot{\mathbf{q}})}{\partial\dot{\mathbf{q}}}\Big) - \frac{\partial\mathcal{L}(\mathbf{q},\dot{\mathbf{q}})}{\partial\mathbf{q}} = \begin{bmatrix}\tau_\alpha \\ f_r\end{bmatrix}. \tag{13}$$

which with the definition of the Lagrangian (9) and expressions (11) (10) may be simplified as

$$= (J_a + mr(t)^2)\ddot{\alpha} + 2mr(t)\dot{r}(t)\dot{\alpha}(t) + \Big[mgr(t) + LMg\Big]\cos(\alpha(t)) = \tau_\alpha(t) \tag{14a}$$

$$= \frac{7}{5}\ddot{r}(t) - r(t)\dot{\alpha}(t)^2 + g\sin(\alpha(t)) = f_r(t) \tag{14b}$$

As no external forces act on the ball itself, we assume $f_r(t) = 0 \;\; \forall t$, allowing the non-linear system may be written on the more compact form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \begin{bmatrix}\tau_\alpha \\ 0\end{bmatrix}, \tag{15}$$

with

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix}J_a + mr^2 & 0 \\ 0 & 7/5\end{bmatrix}, \qquad \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \begin{bmatrix}mr\dot{r} & mr\dot{\alpha} \\ -r\dot{\alpha} & 0\end{bmatrix}, \qquad \mathbf{g}(\mathbf{q}) = \begin{bmatrix}(mgr + LMg)\cos(\alpha) \\ g\sin(\alpha)\end{bmatrix} \tag{16}$$

Remarks

This process is highly unstable. Setting $\dot{r} = \ddot{r} = 0$ and $\dot{\alpha} = \ddot{\alpha} = 0$ in (14a) (14b) and assuming $f_r(t) = 0 \;\; \forall t$, we find equilibrium points

$$\alpha^* = k\pi, k \in \mathbb{Z}, \qquad \tau_\alpha^* = (mgr^* + LMg)\cos(\alpha^*). \tag{17}$$

Now, consider a subset of the state-space within the bounds of the saturations in the physical process, where

$$-\pi < \alpha_- < \alpha(t) < \alpha_+ < \pi, \qquad 0 \leq r(t) < 2L. \tag{18}$$

for some $\alpha_-, \alpha_+ \in \mathbb{R}$. The equilibrium points of the system are found along $[\alpha, r, \dot{\alpha}, \dot{r}]^T = [0, r^*, 0, 0]^T$, occurring when and only when $\tau_\alpha^* = mgr^* + LMg$. This provides shows just how the ball and beam process may be connected to one of the motors (DC or stepper), and also shows the approximate torque specifications required of the motor, which should be *at least* $\tau_{max} \gtrsim (2m + M)Lg$. Interestingly, (ii) the expression $\dot{\mathbf{M}}(\mathbf{q}) - 2\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ is not skew symmetric, making controller synthesis by the Lyapunov method difficult. Finally, (iii) the matrix $\mathbf{M}(\mathbf{q})$ is positive definite as $J_a + mr^2 > 0 \;\; \forall(\mathbf{q},\dot{\mathbf{q}})$ by (18), making the dynamics well conditioned at all times.

Implementation

The model may be run using the `run_project.m` script with the `model="ball_and_beam"` and `controller="response"`. In addition, a script has been written to generate function handles for generating the continuous time system dynamics on matrix form, as a vector valued function and finally in the linearised error dynamics. All of which are generated by the `linearize_ball_and_beam.m` script, have been checked against hand computed linearisations and numerically checked against the

1.4 Furuta Pendulum

This model is largely based on the work of [8], where similarly to the Ball and Beam process, the Lagrangian method is used to find the governing equations. The considered system consist of a motor axis with a moment of inertia J $[kgm^2]$ to which an arm of length l_a $[m]$ with mass m_a $[kg]$ is attached. To this arm, a second arm of length l_p $[m]$ and mass m_p $[kg]$ is attached with a weight M $[kg]$ at its end. Just as in, we define

$$\begin{aligned} \alpha &\triangleq J + \Big(M + \frac{1}{3}m_a + m_p\Big)l_a^2 \qquad & \beta &\triangleq \Big(M + \tfrac{1}{3}m_p\Big)l_p^2 \\ \gamma &\triangleq \Big(M + \frac{1}{2}m_p\Big)l_a l_p & \delta &\triangleq \Big(M + \frac{1}{2}m_p\Big)g l_p \end{aligned} \tag{19}$$

and denote the torques about $\psi(t)$ and $\theta(t)$ as $\tau_\psi(t)$ and $\tau_\theta(t)$ respectively. Letting $\mathbf{q}(t) = [\phi(t), \theta(t)]$ and finding the kinetic and potential energies of the system, the Lagrangian $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = E_{kin}(\mathbf{q}, \dot{\mathbf{q}}) - E_{pot}(\mathbf{q})$ are given in [8]. The Euler-Lagrange equation

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \end{bmatrix} = \frac{d}{dt}\Big(\frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}}\Big) - \frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})}{\partial \mathbf{q}}, \tag{20}$$

may then be simplified as

$$\begin{aligned} \tau_\phi(t) &= [\alpha + \beta \sin^2(\theta(t))]\ddot{\phi}(t) + \gamma \cos(\theta(t))\ddot{\theta}(t) + 2\beta \cos(\theta(t)) \sin(\theta(t))\dot{\phi}(t)\dot{\theta}(t) - \gamma \sin(\theta(t))\dot{\theta}(t)^2 \\ \tau_\theta(t) &= \gamma \cos(\theta(t))\ddot{\phi}(t) + \beta\ddot{\theta}(t) - \beta \cos(\theta(t)) \sin(\theta(t))\dot{\phi}(t)^2 - \delta \sin(\theta(t)) \end{aligned} \tag{21}$$

The governing equations may be written in the equivalent matrix form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \begin{bmatrix} \tau_\phi \\ \tau_\theta \end{bmatrix}, \tag{22}$$

with

$$\mathbf{M}(\mathbf{q}) \triangleq \begin{bmatrix} \alpha + \beta \sin^2(\theta) & \gamma \cos(\theta) \\ \gamma \cos(\theta) & \beta \end{bmatrix}, \qquad \mathbf{g}(\mathbf{q}) \triangleq \begin{bmatrix} 0 \\ -\beta \sin(\theta) \end{bmatrix}, \tag{23}$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \triangleq \begin{bmatrix} \beta \cos(\theta) \sin(\theta)\dot{\theta} & \beta \cos(\theta) \sin(\theta)\dot{\phi} - \gamma \sin(\theta)\dot{\theta} \\ -\beta \cos(\theta) \sin(\theta)\dot{\phi} & 0 \end{bmatrix}. \tag{24}$$

Remarks

Note that (i) as $J, M, m_a, m_p, l_a, l_p > 0$ implies $\alpha\beta > \gamma^2$, hence, the matrix $\mathbf{M}(\mathbf{q})$ is positive definite $\forall \mathbf{q}, \dot{\mathbf{q}}$. Furthermore, (ii) the expression $\dot{\mathbf{M}}(\mathbf{q}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is skew symmetric $\forall \mathbf{q}, \dot{\mathbf{q}}$. With this knowledge, the structure of the governing equations may be used in proofs of GAS by non-linear controllers.

Implementation

The model may be run using the `run_project.m` script with the `model="furuta_pendulum"` and `controller="response"`, and is currently only implemented in continuous time. Similarly to the ball and beam, a script was written to generate function handles for finding continuous time system dynamics on matrix form, as a vector valued function and finally in the linearised error dynamics asa function of the system parameters and states. All of which are generated by the `linearize_ball_and_beam.m` script. These have been checked against hand computed linearisations, and also by comparison to the linearization about $[0, 0, 0, 0]^T$ and $[0, \pi, 0, 0]^T$ in [8].

1.5 Discretisation of the non-linear models

As the real-time implementation will be running in discrete time, we consider a simple approach of in-the-loop ZOH sampling of the non-linear system

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{u}, \tag{25}$$

with $\mathbf{q} \in \mathbb{R}^{n\times 1}$, $\mathbf{u} \in \mathbb{R}^{m\times 1}$. This clearly encompasses all considered non-linear models, and due to the positive definiteness of $\mathbf{M}(\mathbf{q})$, its inverse always exists allowing the formation of the system

$$\frac{d}{dt}\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{0}_n & \mathbf{I}_n\\ \mathbf{0}_n & -\mathbf{M}^{-1}(\mathbf{q})\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\end{bmatrix}}_{\mathbf{A}_c(\mathbf{q}, \dot{\mathbf{q}})}\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix} + \underbrace{\begin{bmatrix}\mathbf{0}_n\\ \mathbf{M}^{-1}(\mathbf{q})\end{bmatrix}}_{\mathbf{B}_c(\mathbf{q})}\mathbf{u} + \underbrace{\begin{bmatrix}\mathbf{0}_{n\times 1}\\ \mathbf{g}(\mathbf{q})\end{bmatrix}}_{\mathbf{G}_c(\mathbf{q})} \tag{26}$$

where the subindex n refers to a square matrix in $\mathbb{R}^{n\times n}$, and the subindex $n \times 1$ refers to a vector of length n. Equivalently, letting $\mathbf{x}(t) = [\mathbf{q}(t)^T, \dot{\mathbf{q}}(t)^T]^T \in \mathbb{R}^{2n\times 1}$, this system may be written

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{A}_c(\mathbf{x}(t))\mathbf{x}(t) + \mathbf{B}_c(\mathbf{x}(t))\mathbf{u}(t) + \mathbf{G}_c(\mathbf{x}(t)) \tag{27}$$

The objective is then to find a discrete time system, whose state vector $x(h(k+1))$ may be computed from a state and control signal at the time $t = hk$ for some time step h $[s]$, such that

$$\mathbf{x}(h(k+1)) = \mathbf{A}_d\mathbf{x}(hk) + \mathbf{B}_d\mathbf{u}(hk) + \mathbf{G}_d. \tag{28}$$

In simpler dynamics, this may be done by the regular methods of discretisation in closed form expressions. However, due to the complexity but for the considered dynamics, we instead take a more general approach and use a zero order hold approximation, such that the control signals are assumed constant over h. By an extension to the exponential matrix theorem, the discrete time system may be propagated in time from $t = hk$ by defining

$$\mathbf{E} = \begin{bmatrix}\mathbf{A}_c(\mathbf{x}(hk)) & \mathbf{B}_c(\mathbf{x}(hk)) & \mathbf{G}_c(\mathbf{x}(hk))\\ \mathbf{0}_{m+1,2n} & \mathbf{0}_{m+1,m} & \mathbf{0}_{m+1,1}\end{bmatrix} \in \mathbb{R}^{2n+m+1} \tag{29}$$

where then

$$e^{\mathbf{E}h} = \sum_{k=0}^{\infty} \frac{1}{k!}(\mathbf{E}h)^k = \begin{bmatrix} \mathbf{A}_d & \mathbf{B}_d & \mathbf{C}_d \\ \mathbf{0}_{m\times 2n} & \mathbf{I}_m & \mathbf{0}_{m\times 1} \\ \mathbf{0}_{1\times 2n} & \mathbf{0}_{1\times m} & 1 \end{bmatrix}. \tag{30}$$

To illustrate this method and validate the model implementation, consider the open loop simulation of the furuta pendulum with a constant torque $\tau_\phi > 0$ and $\tau_\theta = 0$. As there is no energy dissipation in the system, the angular rate $\dot{\phi}(t)$ increases close to linearly with the corresponding angle $\phi(t)$ increasing quadratically. The small oscillation is due to the initial condition of the pendulum $\mathbf{x}(0) = [\phi(0), \theta(0), \dot{\phi}(0), \dot{\theta}(0)]^T = [0, \pi, 0, 0]^T$, and remains constant throughout the simulation in the continuous time. When using the above method of discretisation, the continuous open loop response is close to identical with the discrete model when $h^{-1} < 250\ [Hz]$, but the numerical errors grow rapidly in time for slower loop rates due to approximations made in the discretisation. For instance, the ZOH assumption becomes less and less valid as h increases, accumulating an error which leads to an non-physical injection of energy and instability (see Figure (1)).

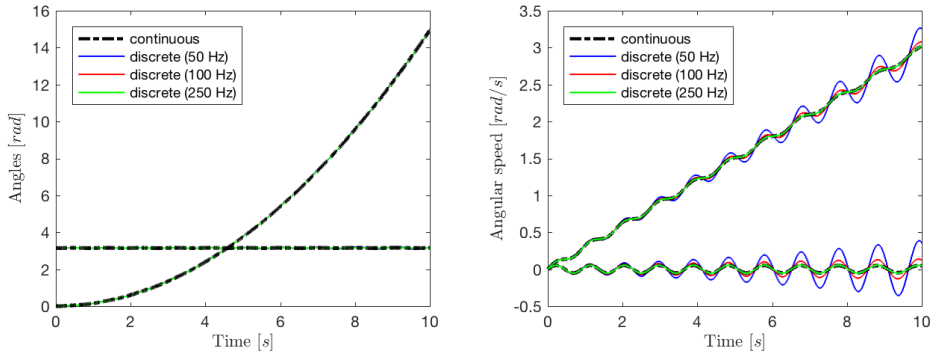

Figure 1: Continuous and discrete time responses of the furuta pendulum with angular rates (left) and angular rates (right) at various loop rates.

2 Control

In this section, various method of control are considered for the processes. Their details, while presently omitted, will be included in this section.

3 DC motor control

Various methods of control have been considered and implemented for the LTI DC motor model (6), including LQR, SMC, MRAC, and many others. All of which may be accessed through the `run_project.m` script.

4 Communication and software

The design of communication protocols and software is arguably the most challenging aspect of the project, and can be approached in many ways. As the BBB supports ethernet and USB, one or the other may be used for commutation, the main difference being the bandwidth increasing when using ethernet. One could imagine communication using the (i) Ethernet/IP, (ii) CIP or TCP/IP protocols. As long as the interfaces on both sides of the communication is designed properly, the two applications may be developed independent of each other, using a wide range of languages and operating systems. The implemented communication protocol, presented in 4.1, sends packages of compressed data in a hexadecimal format, tailored to the considered processes and intended for use with double floating point precision.

On the BBB side, the simplest approach is to use the java script library "Bonescritpt" developed specifically for the BBB by BeagleBoard.org [9]. Alternatively, applications can be developed using more common languages, where many wrappers exist for the Bonscript API. One such example is Adafruit's "BBIO" python module [10], and an interesting prospect is to develop such a module for Julia. The modules in their corresponding languages are described in 4.2. The final part of the software concerns the host side, where various approaches may be considered. Initially, an application will be developed in the robot operating system ROS, but there exist methods for accessing the GPIO pins of the BBB from Simulink in-the-loop which should also be considered, and one could also envision powerful implementations in Julia.

4.1 Communication

In order to make communication versatile and compatible with as many applications as possible, we will only consider the TCP/IP protocols, as they are supported in Matlab/Simulink stack. In addition, there exists good modules in Python, C++ and Julia for socket based communication. While Ethernet/IP communication is supported in Python using the "cpppo" module [11], it adds unnecessary complexity when interfacing with Simulink and Julia. We are then left with a choice of using either UDP or TCP in the TCP/IP suite. For the considered processes, the high reliability of the TCP is not needed. In addition, sue to the Nagle algorithm [12], performance will likely be improved by using UDP if sending many short packets in rapid succession.

TODO: write out the currently implemented protocol.

4.2 The Beagle modules

4.2.1 Python

The Python module implements the Adafruit's "BBIO" python module, and allows the declaration of sensors and actuator objects. The code is written to connect the hardware on user specified pins on the BBB, and may be called in the loop to communicate received data to the controller or control objects directly. This module also implements the sender and receivers which code and decode data sent to and from the host computer.

TODO: High level description of the currently implemented module.

4.3 The Host modules

4.3.1 ROS

The robot operating system ROS has many strengths when it comes to controlling many modular systems, a property which will not be necessary in the considered project. However, it is widely used in the robotics community and is therefore an interesting platform to consider for educational purposes. In addition it provides great methods of visualisation using RVIZ, and could easily be used to interface motion capture with the ball and beam process when considering a two-dimensional plane.

Caution must be taken when implementing control systems in ROS, as inter-node communication yields non-deterministic time delays on the millisecond scale. This is a string argument for implementing controllers on the BBB directly,

TODO: High level description of the ROS stack.

4.3.2 Julia

5 Electronics

6 Hardware

References

[1] "BeagleBone Black," 2017, (Accessed: 04/23-2017). [Online]. Available: https://beagleboard.org/black

[2] "Ubuntu ARM install of ROS Indigo," 2017, (Accessed: 04/23-2017). [Online]. Available: http://wiki.ros.org/indigo/Installation/UbuntuARM

[3] "ROS Indigo Igloo," 2017, (Accessed: 04/23-2017). [Online]. Available: http://wiki.ros.org/indigo

[4] J. Bezanzon, S. Karpinski, V. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," in *Lang.NEXT*, Apr. 2012. [Online]. Available: http://julialang.org/images/lang.next.pdf

[5] "Add Support for BeagleBone Black Hardware," 2017, (Accessed: 04/23-2017). [Online]. Available: https://www.mathworks.com/help/supportpkg/beaglebone/

[6] L. Zaccarian, "Dc motors: dynamic model and control techniques," *Lecture given–2012*, 2012.

[7] W. Yu, "Nonlinear pd regulation for ball and beam system," *International Journal of Electrical Engineering Education*, vol. 46, no. 1, pp. 59–73, 2009.

[8] M. Gäfvert, "Modelling the furuta pendulum," *Department of Automatic Control, Lund Institute of Technology*, 1998.

[9] "BoneScript Library," 2017, (Accessed: 04/23-2017). [Online]. Available: http://beagleboard.org/Support/BoneScript

[10] "Adafruit BBIO, A module to control BeagleBone IO channels," 2017, (Accessed: 04/23-2017). [Online]. Available: https://github.com/adafruit/Adafruit_Python_GPIO

[11] "The CPPPO module," 2017, (Accessed: 04/23-2017). [Online]. Available: https://pypi.python.org/pypi/cpppo

[12] J. C. Mogul and G. Minshall, "Rethinking the tcp nagle algorithm," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 1, pp. 6–20, 2001.