**MODEL-BASED DESIGN FOR A SELF-BALANCING ROBOT**

**USING THE ARDUINO MICRO-CONTROLLER BOARD**

**ENPM667 – PROJECT 1**

**A TECHNICAL REPORT**

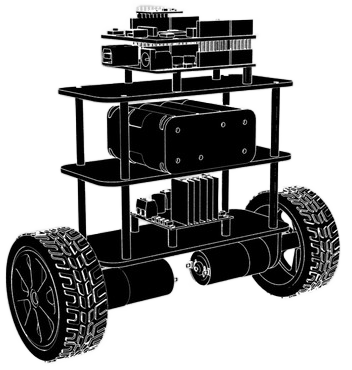
*Submitted by*

**ARSHAD SHAIK (118438832)**

**DHINESH RAJASEKARAN (119400241)**

*Under the guidance of*

**Dr. Waseem A. Malik**

****

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| S.No | Title | Page No. |
| 1. | Abstract | 1 |
| 2. | Introduction | 1 |
| 3. | Block Diagram | **2** |
| 4. | Methodology | **3** |
| 5. | Dynamic Model & State Space | **5** |
| 6. | Complementary Filter | **9** |
| 7. | Motor Synchronization & Simulink Model | **10** |
| 8. | PID Controller & Simulink Model | **14** |
| 9. | LQR Controller & Simulink Model | **17** |
| 10. | Hardware Design/Representation | **21** |
| 11. | Discussion and Conclusion | **22** |
| 12. | References | **23** |

**ABSTRACT**

Self – Balancing Robot is one of the basic hardware models where various traditional and modern-day controllers can be experimented and tested directly. It is an unstable non-linear dynamic system. In the research paper, they have designed and implemented PID and LQR controller on a self-balancing robot. The dynamic model was simulated in MATLAB Simulink and tested on real hardware using a set of DC motors, accelerometer, and Arduino. By utilizing cost effective simple hardware and model-based design, they implemented both PID and LQR controller as well as a simple PD controller for motor synchronization.

**INTRODUCTION**

The research paper aims at developing a model-based design of a self-balancing robot in MATLAB Simulink and experimenting the results on real cost-efficient hardware.

The paper involves 2 aspects:

1. Synchronization of the 2 Motors on the robot
2. Balancing of the Robot

3 different controllers are used for this purpose. A PD controller is used for motor synchronization which takes as input and regulates the output torque accordingly so that both the motors rotate in synchronous manner without causing drift because of unequal speeds or torque.

Balancing of the robot is implemented in both PID and LQR controller. The gain values are tuned accordingly, and output is achieved. The input for these controllers is the robot body angle of the robot and output is the linear and angular motion. The controller tries to bring the angular motion which is robot body angle to zero, so the robot is balanced.

Self-balancing robots caught attention and gathered widespread research interest as one of the future forms of short-range indoor mobility when Segway proved the first successful human-friendly two wheeled balancing robot as a personal transport device. As the auxiliary wheels are removed in a self-balancing robot, it becomes an inverted pendulum which must be balanced in the event of external disturbances.

Another goal of this paper is to implement model-based design using Simulink which is highly useful for mathematically expressing complex cyber-physical systems and create a virtual model of them. Simulink PID tuner app and LQR controller in MATLAB will be used to realize this research paper.

**BLOCK DIAGRAM**

Graphical user interface

Description automatically generated with low confidence

**Figure 1: Overall Block Diagram**

**METHODOLOGY**

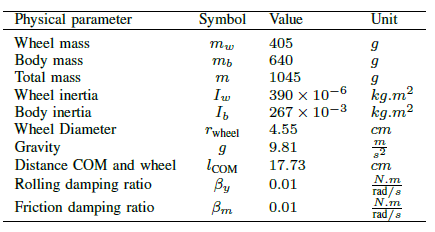
* First, we start with creating the mathematical model of the robot which can be used in Simulink and at the end deriving the state-space representation.
* The dynamic model of the robot is derived using Lagrange’s method which gives a nonlinear model. It is linearised and final equations are derived which is represented in the form of state-space.
* Accelerometer is used to estimate the state of the system which gives acceleration along the 3 cartesian axes. The resulting acceleration is calculated as per the formula provided in the accelerometer datasheet [22]. The resulting gyroscope angle is also computed in the same way.
* The & calculated is fed to a complimentary filter which is basically a combination of low pass and high filter which is perfect for a complementary device like accelerometer and gyroscope.
* The is obtained as an output of the complimentary filter which is used as the estimated state of the system.
* Now the PD controller comes into picture for motor synchronization. The derived motor transfer function will be used for the motor output torque. Its input is motor armature voltage and output is Torque. Since an actual hardware was used in this paper, the wheel angle is computed from the data provided by the motor encoders using following relation.

A picture containing diagram

Description automatically generated

* By appropriately adjusting the proportional gain kp and derivative gain kd , the torque provided to each motor is balanced based on the value.
* This motor synchronization block will be used in the overall system where the focus is to balance the robot.
* The which was computed by the complimentary filter will be fed to the PID controller and gains were adjusted accordingly.
* The PID controller will adjust the gains such that, the is zero which is the balanced condition of the robot.
* The same setup is repeated is with a LQR controller and both the outputs are compared at the end.
* The entire setup was deployed on a hardware based on Arduino Due which is the microcontroller controlling the motors and data acquisition.
* RN-42 Bluetooth module is interfaced with Arduino which sends and receives data with the computer which is running the Simulink models. Once the computation is done, it sends back the data to control the motors to Arduino using the Bluetooth module.
* The Arduino uses a L298N dual channel 2A motor driver to control the motors speed and direction. It’s an H-bridge IC which can operate without consuming lot of power and dissipating heat.
* DC motors of 150 RPM each is used as the main actuator.
* MPU9250 9-axis IMU is interfaced with Arduino which sends accelerometer, gyroscope, and magnetometer data to Arduino. The Arduino sends this data to the Simulink model via Bluetooth module.
* The motors are also equipped with encoders, which is magnetic quadrature 2 channel encoder which is interfaced with Arduino using its built-in interrupt pins. All the 4 channels of the 2 encoders are monitored by the Arduino, which gathers the signal, processes it and computes the wheel angle and sends it to Simulink model. The direction of rotation of the motor can also be figured out using the encoder output.
* The entire setup is powered by 3 Li-ion batteries of 3.7V 2000mAh each which makes up a total of 11.1V 2000mAh sufficient to meet the requirements of the DC motors.
* The batteries must be charged using external chargers. The PC which is running the Simulink model will be powered separately and is an isolated system from the robot.
* A voltage divider configuration is used in the RX (receiver) line to interface the RN-42 Bluetooth module with Arduino as RN-42 is 3.3V operating device. Refer Fig. 3.

Below are the various parameters which will be used in the upcoming derivations (attached from the paper for your easy ref during the derivation part):



Table

Description automatically generated

**DYNAMIC MODEL & STATE SPACE**

Diagram, engineering drawing

Description automatically generated

**Figure 2: Self-Balancing Robot Dynamic Model**

From the above dynamic model, using Lagrange’s method, we derive the dynamic equations for this system.

We know,

Potential energy of the system is given by, (potential energy is zero at the upright position)

Kinetic energy of the system is given by,

Lagrangian is the difference between the kinetic energy and potential energy so it’s given by,

There are only 2 variables of interest in the system which will be included in the state-space representation which are & . Thus, the dynamic equation of the system is computed using & .

1. coordinate:

From the above equations, we can compute the partial derivates and final dynamic equation,

(1)

1. coordinate:

From the above equations, we can compute the partial derivates and final dynamic equation,

(2)

and are used for simple representation purposes, so that the long values can be easily written inside matrices. They are generalised torques for each coordinate.

We can write the dynamic equations (1) & (2) in a matrix form as follows:

(3)

Next, we calculate the generalised torque using our known parameters. It is defined as the difference between actual torque applied to the system and torque dissipated or wasted due to friction. In our system, we have 2 frictional surfaces:

1. Friction between wheel and axle
2. Friction between wheel and the floor

Let be the torque dissipated due to friction in the axle and be the torque dissipated due to friction between the ground and the wheel. The applied torque is taken as .

We know, rolling damping ratio and friction damping ratio , using this, we can write and as follows:

Now, we have a non-linear dynamic equation (3), which must be linearised. The system can be linearised for a small range of motion from the stable upright position which is . Within this range it is safe to assume that .

Thus, the equation (3) can be written as,

Equation (4) is a simple representation of the above equation, so that it’s easier to write the state-space representation.

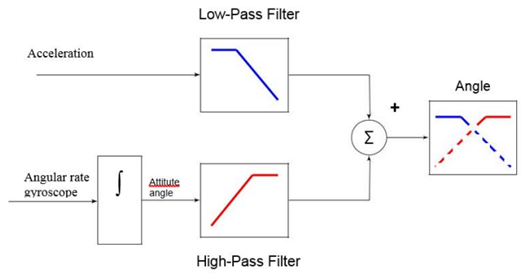
(4)

Thus, the state-space representation of the system with as the state variables is as follows:

The input to the system is applied torque and output is linear distance moved by the robot ( and the rotational angle of the robot body . We assume there is no slipping between the ground and wheel.

Simpler representation of the state-space is as follows (for convenience in referring and representing):

**COMPLIMENTARY FILTER**



**Figure 3: Complimentary Filter Representation**

* is obtained from the MPU9250 module. Since the module is oriented perpendicular to surface, the formula for calculating the as per the module’s datasheet [22] is given by,
* The complimentary filter Fig. 3 is a combination of high pass and low pass filter for processing the MPU9250 data. The from the above result is passed through the low pass filter and the high pass filter respectively and the resulting angle is obtained.
* The mathematical equation of a complimentary filter is given by, where is filter factor whose value is obtained from [19] paper’s practical experiments.
* In the current Simulink model, the feedback is directly fed to the input, without considering the noise from the actual sensor, as the gyroscope and accelerometer sensor outputs are within the scopes of the hardware. For simulation purposes, the sensor is assumed to be ideal sensor, without the noise aspect.

Table

Description automatically generated**MOTOR SYNCHRONIZATION & SIMULINK MODEL**

Diagram

Description automatically generated

Text, letter

Description automatically generated

**The Block diagram of Motor synchronization from the Paper:**

**Diagram

Description automatically generated**

**Actual implementation of the Motor Synchronization in the Simulink:**

Diagram

Description automatically generated

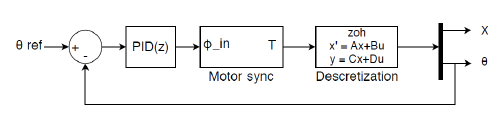
**Conclusion from the simulation:**

The two motors (left and right) are modelled independently and the respective torques are summed together. The PD control is used to regulate the angular position to the desired value. The control loop is tested individually. Though the outputs are regulated perfectly, the cascaded PID control shows the erroneous output. Hence, this individual control loop of angular position of the shaft is not included in the robot angle control. For the purposes, of adding to the completeness in alignment with the paper and the illustration, the above diagrams are included here.

**PID CONTROLLER & SIMULINK MODEL**

* The PID controller is implemented using Simulink PID tuner app with of 0.01s and initial condition of . Phase margin is 60 and bandwidth is 32 rad/s.
* The block diagram of the model and its Simulink implementation are shown below in the following figures.

**From Paper:**

****

Diagram

Description automatically generated**Actual Implementation in Simulink:**

**Graphs:**

* The simulation graphs, as shown in the paper and the equivalent graphs from running the Simulink models are shown below.

**From Paper:**

Chart, line chart

Description automatically generated

**Simulation Graphs:**

Graphical user interface

Description automatically generated

Line chart

Description automatically generated with low confidence

**Conclusion from the above graphs:**

* It can be seen that the same trend is replicated through the simulation, as shown in the technical paper.
* The following assumptions are made:
* The angular position is regulated in the given paper whereas in the current simulation mode, the angular position is assumed to be in synchronization between the two motors. and hence there is no individual PD control implemented.

**LQR CONTROLLER & SIMULINK MODEL**

We are interested in finding when the robot becomes uncontrollable, so that the hardware model’s mass distribution can be altered accordingly. So, to design a full state feedback controller is required which gives us more information about the controllability of the system than the PID. We can compare the gains of the controller and free response of the closed-loop system to determine when it goes uncontrollable.

Thus, we choose a LQR controller which is one of the best full state feedback controllers where we will use all the state variables unlike PID.

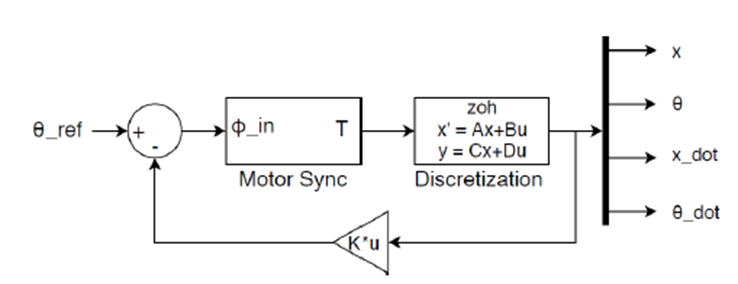
are the state variables and input being . We consider all the state variables and input equally, thus weighting matrices are as follows:

The K matrix is computed from the MATLAB as below:

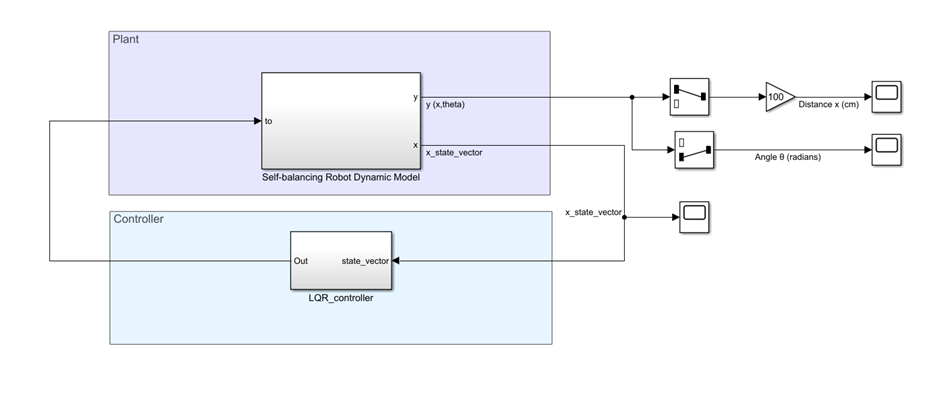
K\_lqr = [1.0000 135.0544 0.8343 69.1891]

The above is generated from the ‘lqr(A,B,C,D,Q,R)’ function from MATLAB. It is found that the motor synchronization function is not considered in the computation of K matrix and hence, it may be attributed to the difference in the K values from the paper and the actual implementation in the Simulink. In any case, it can be noted that the robot angle is regulated through the above implementation.

**From the Paper:**



**Actual Implementation in the Simulink model:**



**From Paper:**

Chart

Description automatically generated with medium confidence**Chart, line chart

Description automatically generatedGraphs from simulation:**

Graphical user interface

Description automatically generated

**Conclusion from the above graphs:**

* It can be seen that the similar trend is replicated through the simulation, as shown in the technical paper.
* The following assumptions are made: The angular position is regulated in the given paper whereas in the current simulation mode, the angular position is assumed to be in synchronization between the two motors. and hence there is no individual PD control implemented.
* Motor transfer function is not considered in the simulation to simplify the simulation process. Only the state space equation as derived in the paper is being used as the plant to regulate the robot angle.

**HARDWARE DESIGN/REPRESENTATION**

Diagram, schematic

Description automatically generated

**DISCUSSION AND CONCLUSION**

Using a self-balance robot, as a system or plant, its mathematical equations of a dynamic model are derived and modelled using Simulink model-based design.

Two methods of controlling algorithms (PID, LQR) are used to demonstrate the controllability of the system.

Simulations show the efficacy of the two control algorithms. It can be noted that the hardware implementation is not pursued as it is not in scope of the current project. Further, the complimentary design filter as it described in the paper, is not included in the simulated study, as it is associated with the hardware sensor output.

Testing a control system on a real-world application is often complicated because of the additional hardware and programming knowledge required. By using an existing platform and Simulink to program the hardware, this process is far more simplified and shortens the time from idea to a working prototype. It is concluded that using the model-based design approach for model creation, simulation and implementation on cheap hardware such as the Arduino allows for a more educational experience in a control-oriented curriculum.

**REFERENCES**

[1] J. Morantes, D. Espitia, O. Morales, R. Jim´enez, and O. Aviles, “Control system for a segway,” International Journal of Applied Engineering Research, vol. 13, no. 18, pp. 13 767–13 771, 2018.

[2] E. Najafi, G. A. Lopes, and R. Babuska, “Balancing a legged robot using state-dependent Riccati equation control,” in Proceedings of the 19th IFAC World Congress, 2014, pp. 2177–2182.

[3] H. Yu, Y. Liu, and T. Yang, “Closed-loop tracking control of a pendulum-driven cart-pole underactuated system,” Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, vol. 222, no. 2, pp. 109–125, 2008.

[4] E. Najafi, R. Babuska, and G. A. Lopes, “An application of sequential composition control to cooperative systems,” in Proceedings of the 10th International Workshop on Robot Motion and Control, 2015, pp. 15–20.

[5] G. A. Lopes, E. Najafi, S. P. Nageshrao, and R. Babuska, “Learning complex behaviors via sequential composition and passivity-based control,” in Handling Uncertainty and Networked Structure in Robot Control, L. Busoniu and L. Tamas Eds. Springer, 2015, pp. 53–74.

[6] F. Dai, X. Gao, S. Jiang, W. Guo, and Y. Liu, “A two-wheeled inverted pendulum robot with friction compensation,” Mechatronics, vol. 30, pp.116–125, 2015.

[7] J. Krizan, L. Ertl, M. Bradac, M. Jasansky, and A. Andreev, “Automatic code generation from Matlab/Simulink for critical applications,” in Proceedings of the Canadian Conference on Electrical and Computer Engineering, 2014, pp. 1–6.

[8] R. Grepl, “Real-time control prototyping in Matlab/Simulink: Review of tools for research and education in mechatronics,” in Proceedings of the IEEE International Conference on Mechatronics, 2011, pp. 881–886.

[9] E. Najafi, R. Babuska, and G. A. Lopes, “Learning sequential composition control,” IEEE transactions on cybernetics, vol. 46, no. 11, pp.2559–2569, 2016.

[10] P. Reguera, D. Garc´ıa, M. Dom´ınguez, M. Prada, and S. Alonso, “A low-cost open source hardware in control education. case study: Arduino-Feedback MS-150,” IFAC-PapersOnLine, vol. 48, no. 29, pp. 117–122, 2015.

[11] E. V. Kumar and J. Jerome, “Robust LQR controller design for stabilizing and trajectory tracking of inverted pendulum,” Procedia Engineering, vol. 64, pp. 169–178, 2013.

[12] J. C. Jensen, D. H. Chang, and E. A. Lee, “A model-based design methodology for cyber-physical systems,” in Proceedings of the International Wireless Communications and Mobile Computing Conference, 2011, pp. 1666–1671.

[13] E. Najafi, A. Shah, and G. A. Lopes, “Robot contact language for manipulation planning,” IEEE/ASME Transactions on Mechatronics, vol. 23, no. 3, pp. 1171–1181, 2018.

[14] E. Najafi, T. Laugs, P. N. Rubio, and S. Alers, “An implementation of AutomationML for an Industry 4.0 case study: RoboCup@Work,” in Proceedings of the International Conference on Research and Education in Mechatronics, 2019, pp. 1–6.

[15] J. L. Guzman, R. Costa-Castello, S. Dormido, and M. Berenguel, “An interactivity-based methodology to support control education: How to teach and learn using simple interactive tools,” IEEE Control Systems, vol. 36, no. 1, pp. 63–76, 2016.

[16] A. Martyanov, E. Solomin, and D. Korobatov, “Development of control algorithms in matlab/simulink,” Procedia engineering, vol. 129, pp. 922–926, 2015.

[17] Y. Ding, J. Gafford, and M. Kunio, “Modeling, simulation and fabrication of a balancing robot,” Harvard University, Massachusettes Institute of Technology, vol. 151, 2012.

[18] E. Najafi and G. A. Lopes, “Towards cooperative sequential composition control,” in Proceedings of the 55th IEEE International Conference on Decision and Control, 2016.

[19] N. H. Ariffin, N. Arsad, and B. Bais, “Low cost mems gyroscope and accelerometer implementation without Kalman filter for angle estimation,” in Proceedings of the International Conference on Advances in Electrical, Electronic and Systems Engineering, 2016, pp. 77–82.

[20] E. Najafi, G. A. Lopes, S. P. Nageshrao, and R. Babuska, “Rapid learning in sequential composition control,” in Proceedings of the 53rd IEEE International Conference on Decision and Control, 2014, pp. 5171–5176.

[21] H. G. Min and E. T. Jeung, “Complementary filter design for angle estimation using mems accelerometer and gyroscope,” Department of Control and Instrumentation, Changwon National University, Changwon, Korea, pp. 641–773, 2015.

[22] <https://www.nxp.com/docs/en/application-note/AN3461.pdf> (Accelerometer Angle Derivation)