Team 8 ROB 550 Balancebot Report

Deepika Baskar (deepikab@umich.edu), Nitish Sanghi (uniq@umich.edu), Tor Shepherd (tors@umich.edu)

***Abstract*—The concept of mobile robots and the implementation of modern control theory in controlling these bots has been the most sorted out in topic, as it furthers our understanding in both, an inverted pendulum and robots with coaxial wheels. Hence they act as learning tools in understanding and testing the controllers that go into making a real-time robot with an industrial application. In this project, began by extracting the motor parameters followed by calculating the moment of inertia about the three principal axes. We implemented a PID controller with cascading loops to balance the robot and then implemented fusion techniques to account for the drift in the gyro data. The values estimated by the odometry was verified and the error between the ground truth and the estimated value was found to be around 2.7%.**

I. Introduction

MOBILE Inverted Pendulum (MIP) robots are commonly used by roboticists for benchmarking control algorithms in research. These robots are also an excellent tool for learning basics of robotics by applying control theory to a relatively simple system. In the field of robotics, theses concepts have a widespread application in the development of Vertical Take-Off and Landing aircraft (VTOL) [1], ball and beam systems and bicycles. With Mobile Robots being omnipresent these days, the wheeled inverted self balancing type MIPs find application in a variety of industries ranging from entertainment to transport. These two wheeled robots, although unstable are easy to control as than a legged robot if care is taken to prevent the over balancing of these robots as they have relatively more number of degrees of freedom and impact force with the ground.

MIPs typically have a base with two motors attached to the wheels and a vertical body, characterized by the center of gravity lying above the pivot point giving rise to the control problem that has to balance this unstable base.

In this project, we worked with a toolkit for building a Mobile Robotics Cape for the Balancebot with the main goal being to integrate the Mobile Robotics Cape with the Beaglebone Green, to implement a gyro-odometric-based dead-reckoning navigation system, and to control the Balancebot using sensor feedback so it can balance upright even when perturbed (i.e. someone pushes it) or when traversing a commanded path.

Section II describes the physical modeling of the system. Section III describes how that physical model can be controlled by PID controllers in successive loop closure. Section IV expands the controller to include heading with respect to the robot's starting frame, and section V details how the robot is able to navigate the environment and keep track of its location and orientation using odometry from the motor encoders and gyroscopic data from the inertial measurement unit.

II. System Modeling

A. Moment of Inertia Calculation

The first step is to characterize the MIP robot's moment of inertia about its principal axis. This is done by conducting an experiment to determine oscillation time periods of the robot. The robot is setup as a bifilar pendulum and manually rotated to initiate back-and-forth periodic motion. The IMU on the robot captures (using **measure_moments.c**) gyroscopic data which is used to determine the period of oscillation. This procedure is repeated for all 3 principal axes, Table I. The time period of oscillation can be determined by,

$$T_0 = 4\pi\sqrt{\frac{J_{ZZ}L}{m_0 g d^2}} \tag{1}$$

Using the oscillation time period the moment of inertia are calculated from the equation relating time period and moment of inertia as follows,

$$J = \frac{m_0 g d^2}{16\pi^2 L} T_0^2 \tag{2}$$

Principal Axis	Moment of Inertia
X	0.00417
Y	0.00884
Z	0.00680

TABLE I: Moment of Inertia about the three principle axes

B. Estimation of Motor Parameters

The motor parameters were determined using the **test_motors_params** executable. The executable takes in resistance values for the left and right motors as input arguments. The resistance values, determined using a multi-meter, for the left and right motor are 5.5Ω and 6.9Ω. Motor parameters found are summarized in Table II.

Motor	ω_{NL}	τ_{NL}	I_{NL}	K	μ	J
Left	36.4	1.96	94.9	0.0153	0.0008	4.23
Right	34.9	1.71	52.7	0.0163	0.0005	2.07

TABLE II: Left and Right motor parameters

where, ω_{NL} is the no load angular velocity(rad/s^2), τ_{NL} is the no load torque ($N - m$), I_{NL} is No stall current(mA), K is coefficient of motor torque, μ is coefficient armature friction and J is the moment of inertia of the shaft with no load($kg - m^2$).

III. Feedback Control

A. Balance controller (inner loop)

The equations of motion governing the underactuated system are derived from the force and moment balances for the wheels and rod.

$$(m_b R_w L \cos\theta)\ddot{\phi} + (I_b + m_b L^2)\ddot{\theta} = m_b g L \sin\theta - \tau \quad (3)$$

$$(I_w + (m_b + m_w)R_w^2)\ddot{\phi} + (m_b R_w L \cos\theta)\ddot{\theta} = m_b R_w L \dot{\theta}^2 \sin\theta + \tau \quad (4)$$

$$\tau = \tau_s u - \frac{\tau_s}{\omega_{NL}}\omega \quad (5)$$

By incorporating the relation between motor output torque and duty cycle (equation 5), the transfer function from motor duty cycle to body angle can be described in the Laplace domain by equation 6.

$$\frac{\Theta(s)}{U(s)} = \frac{-b_2(a_1+a_2)s^2}{(-a_2^2+a_1a_3)s^4+b_2(a_1+a_3+2a_2)s^3-a_1a_4s^2-a_4b_2s} \quad (6)$$

$$a_1 = I_w + (m_w + m_b R_w^2) \quad (7)$$

$$a_2 = m_b R_w L \quad (8)$$

$$a_3 = I_b + m_b L^2 \quad (9)$$

$$a_4 = m_b g L \quad (10)$$

$$b_1 = 2\tau_s \quad (11)$$

$$b_2 = \frac{-2\tau_s}{\omega_{NL}} \quad (12)$$

With the dynamical system modeled, a controller is proposed to stabilize the robot around a setpoint body angle θ. A PID controller can approximate the behavior a second-order spring-mass-damper system by

- applying gain proportional to the difference between the setpoint body angle and measured body angle from the gyro (error) similarly to the spring in a second-order system
- applying gain proportional to the change in error over time similarly to the damper in a second-order system
- applying gain proportional to the accumulation of error over time

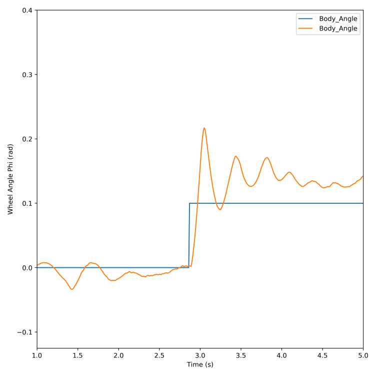
$$u_{motor} = K_p e_{\theta,k} + K_i \sum_{n=1}^{k} e_{\theta,n} + K_d \frac{e_{\theta,k} - e_{\theta,k-1}}{dt} \quad (13)$$

After taking the Laplace transform, the transfer function $D1(s)$ is given in equation 14.

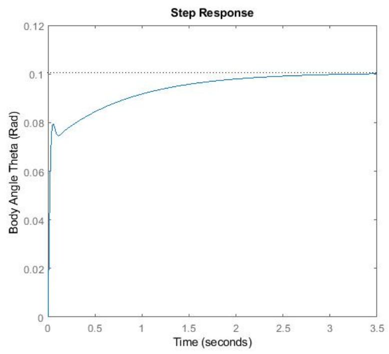
$$D1(s) = K_p + \frac{K_i}{s} + K_d s \quad (14)$$

TABLE III: Body angle PID Controller

Parameter	Value
K_P	8.0
K_I	14.0
K_D	0.1
$t_{rolloff}$	0.0286



(a) Step input (blue) and robot response (orange)



(b) Step input (blue) and simulated response (orange)

Fig. 1: Balancebot step response to reference body angle $\theta = 0.1$, hardware and simulated

One major issue with the PID controller alone is that derivative gain tends to amplify high-frequency noise from the sensor readings and large derivatives associated with step inputs. A low-pass filter is implemented as well, with transfer function shown in equation 15.

$$L1 = \frac{a}{s+a} \tag{15}$$

The dynamic model and PID controller for balancing are modeled in MATLAB and tested on the robot to converge on the gain values shown in table III, with $t_{rolloff}$ representing the time constant of the LPF.

To validate the control scheme, the balancebot is provided a step input of 0.1 radians body angle. The measured response is figure 1a, while the simulated MATLAB model is figure 1b.

At the greater offset from the vertical linear approximation, the robot accelerates continuously and has trouble tracking the body angle. When it overshoots, it falls quickly.

B. Linear displacement controller (outer loop)

While the ”inner loop” balancing PID controller does a good job of keeping the robot upright, the wheels are unconstrained and can drift with no repercussion. The transfer function from body angle to wheel position (equation 16 determines how the system will move.

$$\frac{\Phi(s)}{\Theta(s)} = \frac{\left(-\left(I_b + m_b R_w L\right) s^2 + m_b g L\right)}{\left(I_w + \left(m_w + m_b\right) R^2 + m_b R_w L\right) s^2} \tag{16}$$

To ensure that the robot tracks a desired linear position (in one dimension), a PID controller (equation 17) with low-pass filter is designed around the error between the reference wheel angle ϕ_{ref} and the measured average of the encoder readings, which is the output of the system described by the transfer function above. The output is the reference body angle to the inner loop controller.

$$\theta_{ref} = K_p e_{\phi,k} + K_i \sum_{n=1}^{k} e_{\phi,n} + K_d \frac{e_{\phi,k} - e_{\phi,k-1}}{dt} \tag{17}$$

The Laplace form of the controller is given by equation 18.

$$D2(s) = K_p + \frac{K_i}{s} + K_d s \tag{18}$$

Tuning involves increasing proportional gain until the robot overshoots the setpoint wheel angle with a desirable rise time, then increasing derivative gain to damp oscillations. The values implemented in the robot are described by table IV.

The overall stabilized system can be modeled with the block diagram shown in figure 2 with the inner loop controller D_1, duty cycle to body angle transfer function G_1, outer loop controller D_2, and body angle to wheel position transfer function G_2.

TABLE IV: Wheel Position PID controller Parameters

Parameter	Value
K_P	-0.008
K_I	0.0
K_D	-0.008
$t_{rolloff}$	0.3

The system is tested on a step response from linear displacement of 3cm, which corresponds to $\phi_{ref} = 0.714$

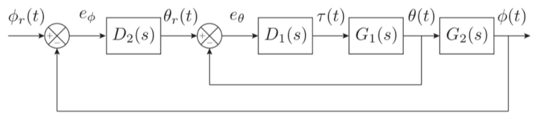
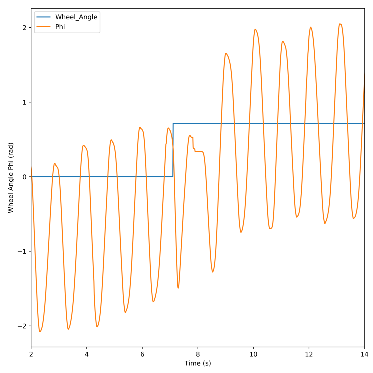
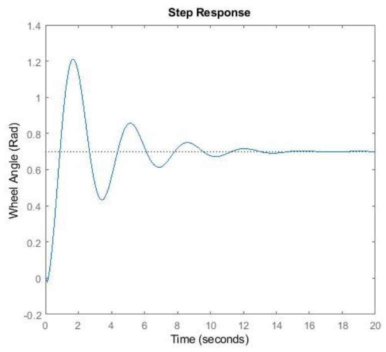


Fig. 2: Body Angle and Wheel position Controller Block Diagram

radians. The measured response is figure 3a, while the simulated MATLAB model is figure 3b.

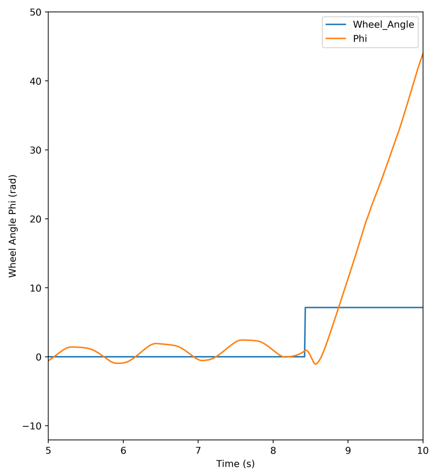


(a) Step input (blue) and robot response (orange)

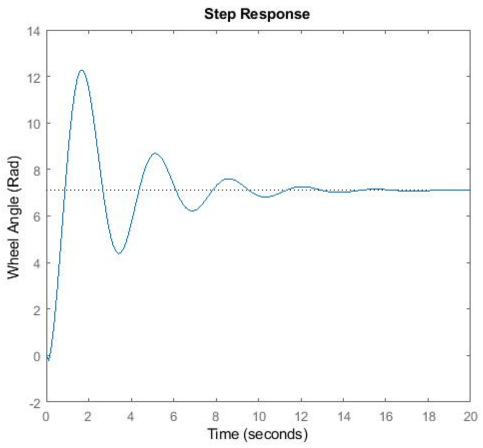


(b) Step input (blue) and simulated response (orange)

Fig. 3: Balancebot step response to reference wheel angle $\phi = 0.714$, hardware and simulated



(a) Step input (blue) and robot response (orange)

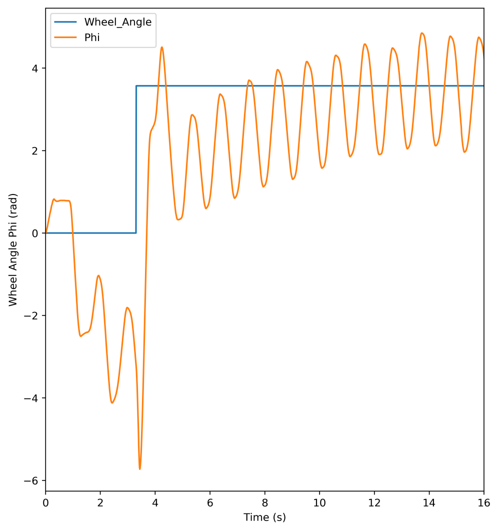


(b) Step input (blue) and simulated response (orange)

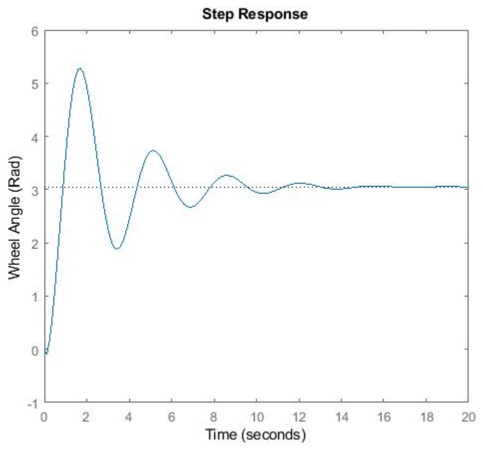
Fig. 4: Balancebot step response to reference wheel angle $\phi = 7.14$, hardware and simulated

Similarly, the system is tested with a step response of 30cm, or $\phi_{ref} = 7.14$ radians. The results are shown in figures 4a and 4b.

Clearly, the step input is of too great magnitude for the PID controller and it commands a body angle outside of the linear region and falls over. The solution in practice is to smooth the reference wheel angle signals to the outer loop controller to limit the derivative portion and restrict the input signal to never command a reference greater than 15cm away from the robot at any point. The step response associated with a 15cm input is shown in figures 4a and 4b.



(a) Step input (blue) and robot response (orange)



(b) Step input (blue) and simulated response (orange)

Fig. 5: Balancebot step response to reference wheel angle $\phi = 3.57$, hardware and simulated

IV. Manual Control

By adjusting the setpoint wheel angle, the robot can be controlled to move back and forth in one dimension. However, the measurement of wheel angle is an average of both wheel encoders, so the robot is free to pivot about the vertical axis. To constrain this, the two-wheeled car model of motion is introduced, with $\dot{x}$ representing the linear velocity and $\dot{x}$ representing the angular velocity.

$$\dot{x} = \frac{\omega_{left} + \omega_{right}}{2} \tag{19}$$

$$\dot{\gamma} = (\omega_{right} - \omega_{left})\frac{r_{wheel}}{w_{base}} \tag{20}$$

Discretizing these equations and multiplying by Δt yields

$$\Delta x = \frac{\Delta\phi_{left} + \Delta\phi_{right}}{2} \tag{21}$$

$$\Delta\gamma = (\Delta\phi_{right} - \Delta\phi_{left})\frac{r_{wheel}}{w_{base}} \tag{22}$$

Equation 22 can be accumulated over time and used to control the rotation by the PID loop given in equation 23

$$\delta = K_p e_{\gamma,k} + K_i \sum_{n=1}^{k} e_{\gamma,n} + K_d \frac{e_{\gamma,k} - e_{\gamma,k-1}}{dt} \tag{23}$$

where the error term e_γ is calculated as the difference between the integral of the turn rate (reference heading) and the integral of all small changes in γ (equation 24).

$$e_\gamma = \sum \omega_{robot,ref} - \sum \Delta\gamma \tag{24}$$

The control signal sent to the motors is revised to be

$$u_{left} = u_{motor} - \delta \tag{25}$$

$$u_{right} = u_{motor} + \delta \tag{26}$$

with u_{motor} representing the output of the inner loop. This is commonly known as differential (or tank) steering. In practice, only proportional gain is necessary to achieve good tracking of a setpoint gamma, as listed in table V.

TABLE V: Heading controller parameter

Parameter	Value
K_P	0.5

The step response of the heading controller to an input of $\gamma_{ref} = 90° = 1.57 rad$ is shown in figure 6. The oscillation could potentially be damped by increasing derivative gain, but the performance meets the requirements of the task.

The Spektrum DSM controller is incorporated by multiplying the normalized values from the velocity and turn rate joysticks by the max speed and turn rates, respectively. The resultant setpoint $\dot{x}$ and $gam\dot{m}a$ are integrated at each time step to get the overall reference values for the outer loop and heading controllers.

V. Odometry

Encoder and IMU data can be used to estimate the change in position of the robot over time. The 2D planar translation of the MIP is determined using the estimated change in wheel angle $\Delta\phi$ and robot heading $\Delta\gamma$. $\Delta\phi$ is determined as the average of the change in left and right wheel angle over time Δt. Initially, $\Delta\gamma$ is determined using equation 22. Discretized equations 27, 28, and 29

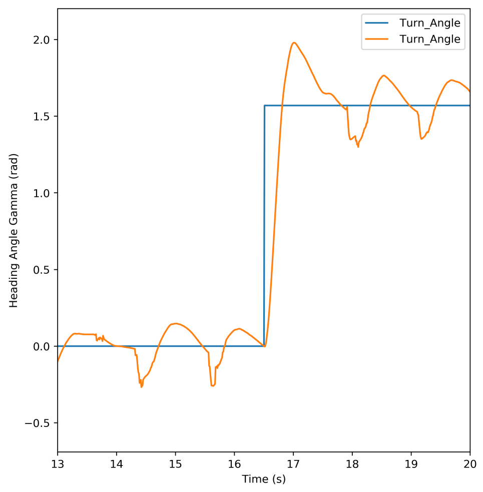


Fig. 6: Step heading input (blue) and balancebot measured response (orange)

compute the coordinate location of the robot with respect to its initial start position.

$$\gamma_{k+1} = \gamma_k + \Delta\gamma_{k+1} \tag{27}$$

$$x_{k+1} = x_k + \Delta\phi_{k+1} r_{wheel} \cos(\gamma_{k+1}) \tag{28}$$

$$y_{k+1} = y_k + \Delta\phi_{k+1} r_{wheel} \sin(\gamma_{k+1}) \tag{29}$$

$\Delta\gamma$ previously used is calculated using the encoder counts. Another method to determine the $\Delta\gamma$ is to use the change in the **yaw** angle of the robot. The change in **yaw** angle is determined by reading the IMU gyro channel for yaw angle, which is measuring the change in **yaw** angle. $\Delta\gamma$ calculated using the encoders and measured by the IMU can be fused together to take advantage of data from both sensors and determine a robust orientation value.

Odometry assumes wheel revolutions can be translated into linear translation. But in cases of non-systemic error like wheel slippage or minor bumps the odometry based orientation change will be inaccurate. In this case the gyroscopic orientation change can be used to determine heading. On the other hand, only using gyroscopic values is not feasible as gyroscope drift over time and are affected by temperature changes. Thus fusing data from encoder and gyroscope provides a robust oreintation change value.

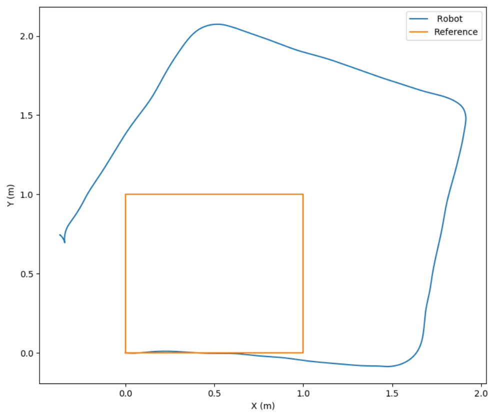
$$\Delta\gamma_{k+1} = \begin{cases} \Delta\gamma_{gy\_k+1}, & (\Delta\gamma_{gy\_k+1} - \Delta\gamma_{od\_k+1}) > \Delta\gamma_{th} \\ \Delta\gamma_{od\_k+1}, & otherwise \end{cases} \tag{30}$$

Equation 30 uses an experimentally determine threshold change value to determine which sensor data to use. If the difference between the two sensors is larger than the threshold, $\Delta\gamma$ determined from the IMU is used otherwise the encoder based $\Delta\gamma$ is used.

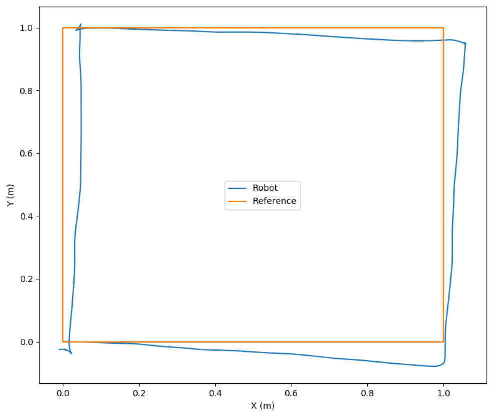
Parameter	Value
Left_Wheel_Scaler	0.62
Right_Wheel_Scaler	0.6138
Wheel_Base_Scaler	1.0204
$\Delta\gamma_{th}$	0.0037

TABLE VI: Odometry Calibration Parameters

In order to get accurate odometry of the robot, the systemic errors arising due to misalignment of wheels, difference in wheel diameters, uncertainity in the effective wheelbase, and limited encoder resolution need to be tuned out. These are errors which are inherent to the system due to manufacturing and assembly variances and can be accounted for. UMBark procedure is used to calibrate the odometry calculation. Table VI lists the parameters which are tuned to calibrate the robot.



(a) UMBark input track (orange) and robot response (blue)



(b) UMBark input track (orange) and robot response (blue)

Fig. 7: Balancebot trajectory following. 7a shows the response prior to calibration, while 7b is after tuning.

The encoder values are scaled down to 62% of the measured reading to calibrate 1 meter travel input to output actual 1 meter travel. The right wheel diameter is also scaled to 0.99 of measured wheel diameter. Furthermore, effective wheel base is found to have an effective value to 2.04% larger than the measured wheel base. These scaling parameters can be due to wheel point of contact and wheel alignment between the left and right wheel. Figures 7a and 7b graph the before and after calibration trajectories of the robot moving along the UMBark track. Before calibration the final location of the robot with respect to its initial position is ($x = -0.373$ m, $y = 0.745$ m) and after calibration ($x = 0.027m$, $y = -0.029m$). The normed error of position of robot from the initial position reduces by an order of magnitude after calibration from 83.3% to 0.039%. With the odometry calibrated, the robot moves in straight lines and performs accurate 90 degree turns. The angular offset which can be observed in **??** is due to the initial orientation condition which was noticed to be slight off when the data was collected.

Parameter	Value
ρ	0.1
α	0.2
β	0.1

TABLE VII

Using lecture notes for feedback control to a point tuned ρ, α, and β to values in Table VII.

VI. Results

A. PID Controllers

The PID controllers designed, were capable of stabilizing around their nominal points for setpoints that lies within the linear region of stability and becomes unstable and tips over for values larger than that as seen in Fig.4a.

B. Odometry

The accuracy of the states estimated by the odometry was estimated using the UMBark track as discussed in section V. The states estimated by the bot was indeed off it's reference as seen in Fig.7a and by appropriately tuning the values the error between the reference and the estimated state was a nominal 3%. Error in the initial alignment with the reference track start point are the potential contributors to the drift in the generated trajectory.

VII. Discussion

This project helped us understand how Mobile Robots can be used as a tool in understanding the effect of controllers, (a simple PID in our case) over the real time system. Furthermore, it enabled us to intuitively understand the effect of tuning the gains (K_P, K_d and K_I) over the system characteristics like, steady state error, settling time and overshoot. The tuned values of the PID controllers indeed achieved its steady state in less than 10 seconds.

A. Competition Performance

The balance bot was able to successfully stabilize around the given mark (setpoints) for around 10 seconds. Moreover, the controllers were robust enough to return to it's nominal point for any small disturbances imparted on it.
For the drag race, although our bot was able to complete the course collision-free, the time required to complete it was not satisfactory and it can be accounted for the gaps in the tuning of the manual control.

B. Potential Improvements

During the step response data collection we observed that a steady state error was built around estimated states and later was rectified by off setting the body angle a constant value. This can be resolved by either introducing K_I in the outer loop or adding a pre-compensator to the open loop.

References

[1] P. B. Martin P., Devasia S., "Robot modeling and control," *Automatica*, vol. 32, 1996.
[2] M. Spong, S. Hutchinson, and M. Vidyasagar. Wiley, 2005. [Online]. Available: https://books.google.com/books?id=wGapQAAACAAJ