Self Balancing Bike Control [Mechatronics] Report - Ibnul Firdous

Introduction:

This project involved the testing and optimising a controller for a self-balancing motorcycle by the help of using MATLAB and the Arduino Engineering kit that came along with the bike. This report will highlight my approach to optimising and testing the control system to allow the motorcycle to balance and control and show how it had its effect on the system's performance. Performance of the motorcycle was judged by how it could balance

Design/Science:

To achieve balance of the motorcycle, a full state feedback controller was implemented into the design which relied on three vital variables which were "theta", "thetadot" and "omega" (IWSpeed). Theta denotes the tilt angle of the motorcycle, whereas thetadot is the rate of change of the tilt angle of the motorcycle. As for omega, it represents the angular velocity of the flywheel on the motorcycle. A lot of optimising and calibrating between these values were done to ensure there were all tuned to achieve stability and optimal performance. I was able to obtain values that showed good performance in balancing whilst being mindful of keeping the motorcycles surroundings relatively consistent to reduce the slightest impediment in the motorcycle's performance and variability. For locomotion, I used an open-loop controller, which meant that the control actions for the bike did not depend on any feedback. This open-loop controller was used to drive the back motor, which was controlling speed independently. This was simply done by adding a constant block to the signal path to the M1M2 block. In MATLAB, the discrete transfer functions with filtered derivates were added to the sensor preprocessing subsystem to supress the higher frequency oscillations. To test the purpose and optimal value, a range of time constants from 0.1 to 0.6 was tested. I discovered that the lower time constant increased noise sensitivity but also increased the reduced response time and therefore an improved system responsiveness. Conversely, higher values reduced noise but showed latency. Through iterative testing, the optimal filter constant to be determined was 0.2. This value allowed an optimal balance between responsiveness and stability, as shown in Figure 3, the improved and latest feedback response. A gain block was added to the signal inertiaWmotor shown in Figure 1, which controlled how much by how forcefully the motor worked in response to the changes in the system without changing any of the values in the balancing controller. This was very helpful since I could solely control values for the motor if it seemed too weak whilst the other values were optimal. By adjusting the gain value, the response intensity could easily be tweaked without compromising the over system's values and stability. This was helpful since I was able to test different values for the gain under various conditions. For

further Stability, digital signal filtering was introduced to the system to reduce noise from the "theta" and "thetadot" signals to reduce interference.

Analysis:

To examine the effectiveness of the motorcycle's controller, performance benchmarks were instilled to match a good target to ensure good performance. The benchmarks focused on how long the bike could maintain being self-balanced whilst in locomotion and stationery.

Benchmarks:

Time balancing on a Flat Plane Surface (s)	30
Time in locomotion on a Flat Surface (s)	10

The benchmarks provided a clear target to use as metrics to match when testing system performance and used to guide me through future improvements. The Engineering Kit Manual had set values on the control parameter, however the bike wasn't able to balance effectively and for long at all with the initial values of 80 (theta), 10 (thetadot) and 0.01 (omega). With these parameters the bike would not balance at all and so I had to optimise the values one gain at a time whilst keeping others fixed. When one of the values was set too low, the bike seemed to tip even being upright, and this highlighted that I needed to change the bias value or the control gains. Once the bike was able to balance for more than a few seconds I kept varying parameters and observing how they impact the bikes performance. Increasing the bias to the tilt angle and retesting the motor response through a gain block allowed improved performance of the bike balancing, which allowed more control over the torque of the flywheel. To ensure that the flywheel was operating correctly, the help of the data inspector and scope in MATLAB allowed me to optimise the bias value some value close to 0 for the balance point on the scope to read at 0 for theta. This made sure that the bike was more stable in the middle and not tilting on the same side after every 20-40 seconds. I would then repeat this process of optimisation with the other two parameters, until the optimised values I obtained was **64 (theta), 5 (thetadot), 0.004 (omega)**. With these optimal values, I obtained a balancing time of more than 6 minutes and a time of 11 seconds when in locomotion in a straight line. Challenges and uncertainties such as excessive heating of the motor causes higher resistance of the motor and would result in less torque as an output. This also indicates that less current was delivered due to increased resistance.

Engineering Design:

As part of the bike, the flywheel used in the motorcycle consisted of three disks. By the equation, $\boldsymbol{I = \frac{1}{2}MR^2}$, since mass is proportional to inertia, if the number of disks were to increase, the increased mass would result in a higher moment of inertia which improves the system's ability to sudden changes in environment and abrupt interferences. However, that

would mean that a heavier flywheel would require much more torque from the motor and drain the battery more quickly. On the other hand, reducing the number of disks would result to a longer battery life and less power needed proportional to the torque being generated, however, the inertia may not be enough to stabilise the bike. Fine-tuning the full-state feedback consisted of changing around different values of proportional gain and derivative gaining. Proportional gain referred as stiffness of the system and how much the flywheel needed to spin whenever the bikes tilting, whereas derivative gain referred to as damping. When testing the values, a high proportional gain caused the bike to tilt excessively and as an observation the theta value would deviate too much from the balance point as seen on Data Inspector on MATLAB. Reducing the proportional gain value improved the stability of the bike. Adjusting the derivative hain helped regulate the response time for the flywheel to changes in angle/tilt. It also damps out any oscillations, which further improves stability and performance. These values were tuned to optimal values for a great performing flywheel for the bike.

To measure the motorcycles tilt/lean angle (theta) and its rate of change of tilt angle (thetadot), an IMU was used. For this project, the IMU used was a BNO055 9-axis Inertial Measurement Unit (IMU). This sensor consists of a gyroscope, an accelerometer and a magnetometer. The BNO055's sensor senses the gravitational vector and the magnetic field with the magnetometer, therefore, a way to calibrate the IMU was to flip the bike upside down and flipping it back upright ensuring it was rotated in all 3 axes. This also ensured that the data that was sent to the controller was accurate and reliable. Decision blocks were implemented to verify that the IMU was calibrated, and ensured the conditions were at acceptable ranges such as, the lean angle (<12°) and the battery voltage being above a threshold (>3V). The bike would only start balancing once these conditions were met.

Practice + Results:

Going back to initial testing and response tests for the bike balancing, these values were tested for different parameters and seeing how they effected the performance of the bike: Keeping Proportional Gain and Integral gain the same whilst changing derivative.

Bias	Proportional Gain	Derivative Gain	Integral Gain	Time Balanced (s)
-0.005	5	53	0.004	19
-0.005	5	60	0.004	25
-0.005	5	64	0.004	374 (6 mins +)
-0.005	5	74	0.004	43

K_P	64
K_D	5
K_{PW}	0.004
Bias	-0.005

Flywheel Gain	255

Appendix:

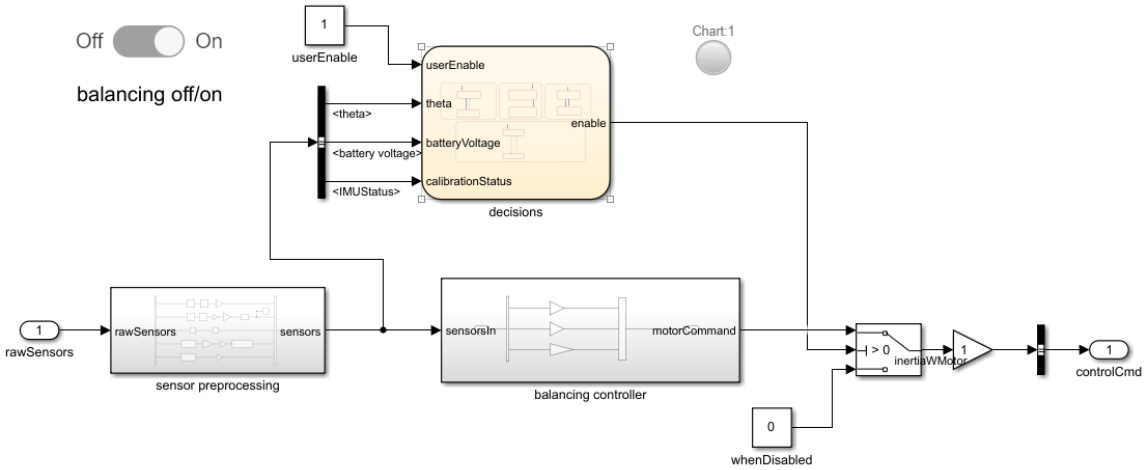

Figure 1: Simulink Model of Controller

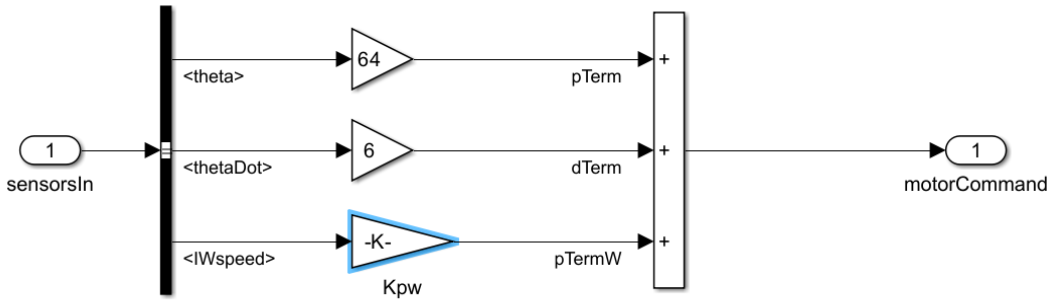

Figure 2a: Simulink Model of the Balancing Controller

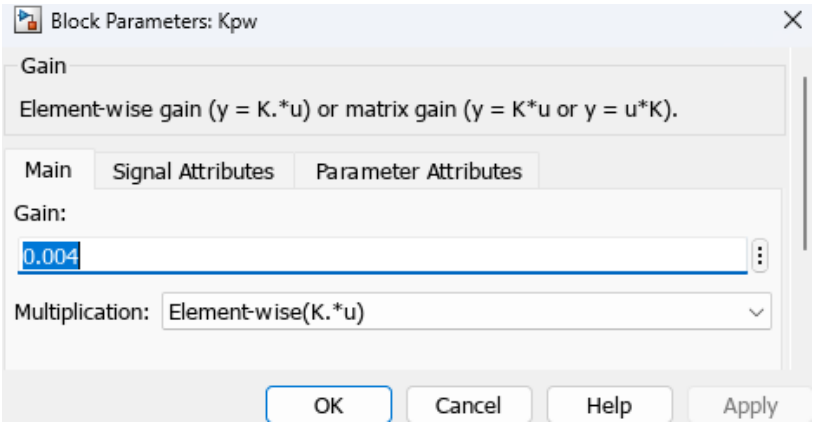

Figure 2b: Value of K_{PW}

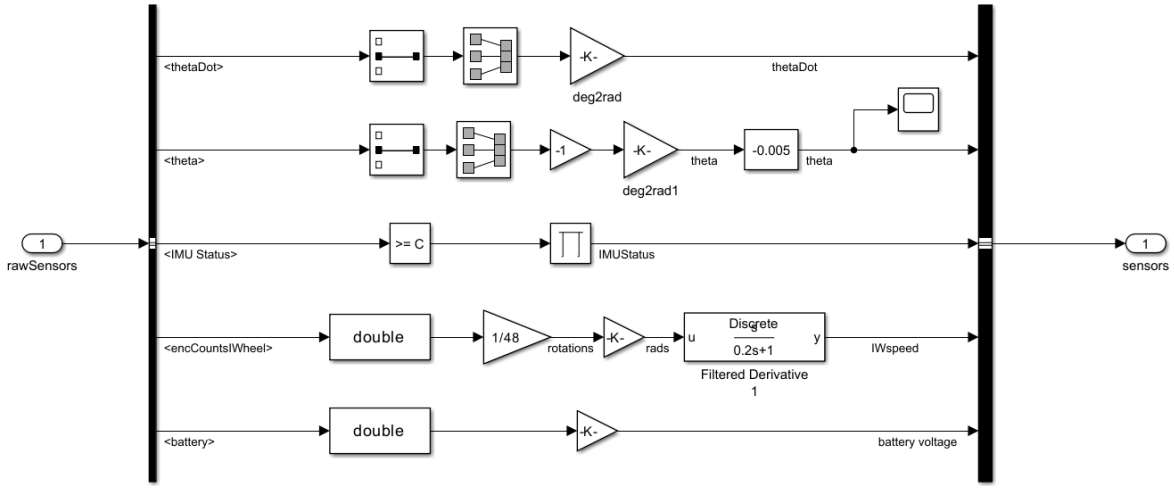

Figure 3: Low pass filter + Bias block – Sensor Preprocessing.