

TEST PLATFORM DESIGN AND CONTROL OF A
BICYCLE-TYPE TWO-WHEELED AUTONOMOUS
VEHICLE.

by

XINQI WANG

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF APPLIED SCIENCE

The Faculty of Engineering and Applied Science
Electrical and Computer Engineering
We accept this thesis as conforming
to the required standard.

*Dr. Mikael Eklund, Supervisor, Faculty of Electrical & Computer Engineering,
University of Ontario Institute Of Technology*

*Dr. Lixuan Lu, Faculty of Electrical & Computer Engineering, University of
Ontario Institute Of Technology*

© XINQI WANG, 2011

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Abstract

Bicycle dynamics and behaviors have been vastly studied through modeling and simulation. Due to the complexity, software models are often assumed subjecting to different nonholonomic constraints in order to simplify the models and control algorithms. A real life autonomous bicycle faces perturbances from the road, wind, tire deformation, slipping among other external forces. Limitations of simulations will not always allow these to apply. All these issues make the autonomous bicycle research very challenging.

To study the bicycle control problems a few research results from the literature are reviewed. A nonlinear bicycle model was used to conduct control simulations. Model based nonlinear controllers were applied to simulate the balance and path tracking control. A PID controller is more practical to replace the non-linear controller for the balance control. Simulation results of the different controllers are compared in order to decide the proper control strategies on the hardware platform. The controller design of the platform complies with practicality based on the hardware configuration. Two control schemes are implemented on the test platform; both are developed with PID algorithms. The first scheme is a single PID control loop in which the controller takes the roll angle feedback and balances the running platform by means of steering. If the desired roll angle is zero the controller will try to hold the platform at the upright position. If the desired roll angle is non-zero the platform will be balanced at an equilibrium roll angle. A fixed roll angle will lead to a fixed steering angle as the result of balance control. The second scheme is directional control with balance consisting of two cascaded PID loops. Steering is the only means to control balance and direction. To do so the desired roll angle must be controlled to achieve the desired steering angle. The platform tilts to

the desired side and steering follows to the same side of the tilt; the platform can then be lifted up by the centrifugal force and eventually balanced at an equilibrium roll angle. The direction can be controlled using a controlled roll angle. Many implementation issues have to be dealt with in order for the control algorithm to be functional. Dynamic roll angle measurement is implemented with complementary internal sensors (accelerometer and gyroscope). Directional information is obtained through a yaw rate gyroscope which operates on the principle of resonance. To monitor the speed of the platform, a rotational sensor was formed by using a hard drive stepper motor attached to the axis of the vehicle's driving motor. The optoelectronic circuit plays the vital role to ensure the system functionality by isolating the electromagnetic noise from the motors. Finally, in order to collect runtime data, the wireless communication is implemented through Bluetooth/RS232 serial interface. The data is then plotted and analyzed with Matlab. Controller gains are tuned through numerous road tests.

Field test results show that the research has successfully achieved the goal of testing the low level control of autonomous bicycle. The developed algorithms are able to balance the platform on semi-smooth surfaces.

Contents

Abstract	ii
Contents	iv
List of Figures	vi
Glossary	viii
Acknowledgements	x
Chapter 1 Introduction	1
1.1 State of art of autonomous two-wheeled vehicle	1
1.2 Problem statement	3
1.3 Thesis objective	5
1.4 Project scope	7
1.5 Thesis outline	8
Chapter 2 Literature Review	10
2.1 Bicycle dynamics and stability	10
2.2 Control of bicycle	14
2.3 Sensor signal optimization	23
Chapter 3 Methodology	29
3.1 Bicycle control simulation	29
3.2 Control schemes discussion of the platform	38
3.3 Controller design of the platform	43
Chapter 4 Implementation	48
4.1 Platform description	48
4.2 System architecture	49

4.3	Software structure	52
4.4	RC controller and pulse width measurement	54
4.5	Actuator control	58
4.6	Sensor system	63
4.7	Photoelectric isolation	70
4.8	Controller tuning	72
4.9	Real-time data acquisition	74
Chapter 5 Results & Discussion		79
5.1	Sensor bias calibration	80
5.2	Balance control	82
5.3	Semi-autonomous directional control with balance	84
5.4	Autonomous directional control with balance	89
Chapter 6 Conclusions		92
6.1	Conclusion	92
6.2	Recommendation of future work	94
Bibliography		96
Appendix A Electrical system schematic		102

List of Figures

Figure 1.1	Ghostrider during endurance test in Nevada [27]	2
Figure 1.2	Illustration of counter steering.	5
Figure 2.1	R. Sharp motor cycle	11
Figure 2.2	R.S. Hand bicycle	15
Figure 2.3	Getz's simplified bicycle model [17]	16
Figure 2.4	Balance control with fuzzy controller [12]	19
Figure 2.5	Path tracking strategy [12]	21
Figure 2.6	Path-following control [12]	22
Figure 3.1	3D bicycle model.	30
Figure 3.2	PtolemyII model of bicycle balance control	31
Figure 3.3	Balance control of upright position plot.	32
Figure 3.4	Balance control of non-zero roll angle plot.	33
Figure 3.5	Bicycle status at different time instant	34
Figure 3.6	Path tracking while maintaining balance control	36
Figure 3.7	Tracking result of a sinusoidal ground path	37
Figure 3.8	Tracking result of a circular ground path	38
Figure 3.9	PD controller PtolemyII model	40
Figure 3.10	Balance control of upright position with PD controller	40
Figure 3.11	Balance control of a fix reference roll anlge	41
Figure 3.12	Balance control of sinusoidal reference roll angle	41
Figure 3.13	Step response results comparison	42
Figure 3.14	Balance control scheme of straight running motion	44
Figure 3.15	Equilibrium roll angle control scheme	45
Figure 3.16	Kinematic bicycle directional control	46

Figure 3.17	Directional control with balance scheme	47
Figure 4.1	The current version of the platform	50
Figure 4.2	Deployment diagram of the system architecture	51
Figure 4.3	System software flow chart	53
Figure 4.4	SPEKTRUM DX5e transmitter and AR500 receiver	56
Figure 4.5	PWM signal	57
Figure 4.6	Steering actuator [44]	59
Figure 4.7	Brake actuator [44]	60
Figure 4.8	DimensionalEngineering Syren25 motor controller	61
Figure 4.9	Tilt sensor configuration	64
Figure 4.10	Accelerometer measures change of gravity [13]	64
Figure 4.11	Schematic of pizelectric accelerometer [4]	66
Figure 4.12	Gyroscope measures speed of rotation [13]	66
Figure 4.13	Output rate signal with clockwise rotation [3].	68
Figure 4.14	Rotary encoder	69
Figure 4.15	Motor encoder amplifier	70
Figure 4.16	Accelerometer signal plot at “0g” position	71
Figure 4.17	Photoelectrical isolation	72
Figure 4.18	Minimum connection between two DCE devices	75
Figure 4.19	Parani-SD/100 & the Bluetooth serial port serivce	75
Figure 4.20	Real-time data displayed in Hex format	78
Figure 5.1	Platform during road test on U.O.I.T. south parking lot ..	81
Figure 5.2	Yaw rate gyroscope bias plot	82
Figure 5.3	Balance control plot	85
Figure 5.4	Semi-autonomous directional control	87
Figure 5.5	Counter steering motion when turning	88
Figure 5.6	Autonomous directional control	90

Glossary

Simulation

$\alpha, \dot{\alpha}$	actual roll angle and roll rate
$\alpha_d, \dot{\alpha}_d$	desired roll angle and desired roll rate
α_e	calculated equilibrium roll angle in ground trajectory tracking simulation
v_r, \dot{v}_r	rear wheel speed and acceleration
v_{rd}	desired rear wheel speed
v_{\perp}	rear wheel side slipping speed, $v_{\perp} \equiv 0$
θ	yaw angle
ϕ	steering angle
σ	parameterized steering variable $\sigma := \tan \phi/b$, b is the dimension of the wheelbase of the simulation model
ω_{σ}	steering control output, $\dot{\sigma} = \omega_{\sigma}$
u_r	rear wheel traction control output
β_*^1	balance controller gains
γ_*	path tracking controller gains
$\lambda_d, \dot{\lambda}_d, \ddot{\lambda}_d$	desired roll angle, rate and acceleration calculated from the internal equilibrium manifold

¹‘*’ is a generalized notation; for instances, the actual notation of β_* can be $\beta_{\alpha 0}$, $\beta_{\alpha 1}$ and so on.

Hardware platform

$\alpha(t)$	actual roll angle
$\alpha_{ref}(t)$	reference roll angle
α_0	initial roll angle
$\Delta\alpha$	relative roll angle
$e(t)$	roll angle error
$\tau(t)$	steering output
τ_0	steering offset, i.e., neutral output
K_{p0}	proportional gain of the balance controller
K_{d0}	differential gain of the balance controller
$\theta(t)$	actual yaw angle
$\theta_{ref}(t)$	reference yaw angle
θ_0	initial yaw angle, $\theta_0 \equiv 0$
$\Delta\theta$	relative yaw angle
$e_\theta(t)$	yaw angle error
K_{p1}	proportional gain of the directional controller
K_{d1}	differential gain of the directional controller
β	complementary filter coefficient
dt	sampling period
T	RC output signal PWM duty cycle
T_{offset}	RC neutral output signal PWM duty cycle

Acknowledgements

First of all I would like to thank my supervisor, Dr. Mikael Eklund. Not only his guidance prevented my research work from roundabout, but also his advices and suggestions to my graduate study's career clarified the confusions, gain me confidence, and finally lead to this thesis work.

I am also very thankful to my fellow research partner, Vijyat Bhalla. I truly appreciate his knowledge, inspiration and enthusiasm. Without his great contribution the research could not have progressed so rapid. I will always remember the days when we were doing field test and working in the lab.

Next, I would like to thank the authors of the free software tools I have used in my research, though I do not know who you are. Thanks for your generosity to post your wonderful works on Internet for free downloading.

Finally, I want to send my gratitude to my family, the invaluable spiritual support from all my family have encouraged me to overcome the stresses from study, work and life.

Chapter 1

Introduction

Bicycle control has long been attracting the attention of researchers from many engineering disciplines. Due to the complexity of its dynamics and limited application, autonomous bicycles or motorcycles are far less developed than their four-wheeled counter part. The desire to build such a vehicle in the past was mainly motivated by research or curiosity. However, commercial autonomous bicycles can also be useful when there is a need to navigate into narrow spaces. In fact, in remote areas the percentage of terrain accessible to four wheeled vehicle is small in comparison to the percentage of the terrain that is accessible to a motorcycle [27]. A successful example is the Blue Team's Ghost rider from 2005 DARPA Grand Challenge, a two-wheeled autonomous vehicle which entered the final round.

1.1 State of art of autonomous two-wheeled vehicle

Currently the most accomplished attempt of autonomous two-wheeled vehicle is the Ghost rider [27], Fig. 1.1, which entered the 2005 DARPA Grand Challenge. The Ghost rider is an autonomous motorcycle built by Anthony Levandowski et al. from University of California, Berkeley. Modified from a commercial off-road motorcycle, the Ghost rider is equipped with powerful computers and sensing system allowing



Figure 1.1: Ghost rider during endurance test in Nevada [27]

it to maintain balance, navigate through open area and safely avoid obstacles.

The environmental sensing system is in the center of the system architecture. Navigation mainly relies on vision processing technology to determine the best possible route to follow. Of the whole vision system, there are two complementary subsystems; the color road detection system is to find road color change, and the 3D reconstruction system is to identify obstacles.

GPS product with correction services turned on is used to provide long range guidance though they claimed that they did not benefit from GPS correction service due to the low precision [27].

At the communication layer there are two networks, one for vision image transmission, another for control and navigation.

What is of most interest here is the balance and direction control of the Ghost rider. After the earlier unsuccessful attempt with Fuzzy controllers they developed nested PID controllers which greatly improved the range of stability. The PID controllers with gain values tuned for concrete and asphalt surfaces performed poorly on coarse surfaces such as grass or sand. Intensively used integration contributed to sluggish direction control and in turn affected navigation. They then modified the controller

to a runtime reinforcement learned controller [27] to account for road surface change and self correction. The controller is based on a stochastic model of the vehicle dynamics. Off-line learning was conducted when constructing the controllers. New learned vehicle parameters were fed into the controller and tested in the vehicle. Then these parameters were used to implement the runtime learning model. During runtime the model was constantly updated at $100Hz$ according to the road response from the vehicle so as to determine the gains for optimized control. Their research showed that the control parameters were changed substantially in relation to the surface properties and changes of direction. The balance control, at $1KHz$, could match the ability of an amateur human rider. But a human rider can maintain pose of a motorcycle by shifting the center of mass, this makes human riders over perform the Ghost rider which uses steering as the only means to control balance.

1.2 Problem statement

A fully autonomous bicycle is required to perform self-localization, obstacle avoidance, balance control, and recovering from accidental fall-over when navigating through remote areas. The vehicle must be equipped with GPS sensors, vision system, pose sensors, and powerful computer hardware and software capable to process vast real time information. The technologies required on an autonomous bicycle are no less than that on much larger 4X4 vehicles. In addition bicycles have to face their unstable nature. Fundamental to any *higher level control* is balance and directional control. The *higher level control* here is referred to as the navigation problems. Balance and directional control problems are the center of my research.

A bicycle in this context is referred to as a *Basic Bicycle*[18], i.e. a bicycle without any additional stabilizing mechanism. When a bicycle stands still its dynamics is analogous to an inverted pendulum; it is a nonlinear, non-minimum phase

system. It is unstable when not controlled. According to Vittore Cossalter [14] there are constantly fifteen forces acting on a bicycle when moving on a semi-smoothed surface. Among the fifteen forces, thirteen forces need to be controlled but only two can be actuated, i.e. steering and speed.

A few terms which will be frequently used later in this thesis are introduced here. *Critical velocity* is the minimum speed limit that the bicycle must run above this speed before it can be balanced. Steering plays the main role for balancing. If for example, one simply turns the steering bar to the desired direction the bicycle will lose balance instantly. Balance of a moving bicycle can be achieved by steering to the same side of tilting such that when the centrifugal force balances the gravitational force the bicycle will be held at an equilibrium roll angle. The magnitude of centrifugal force at a certain roll angle is decided by turning radius and speed. The *critical roll angle* is the tilt angle that beyond this angle no matter how much effort to be put into, the bicycle will still fall over. The angle from the critical roll angle of either side to the upright position forms a *manifold*. Within the manifold a bicycle with sufficient speed can always find its equilibria with a suitable control strategy.

The steering angle also determines the direction of motion. The balance control described above implicitly indicates that the directional control can be achieved through the roll angle control. To change the direction of a *Basic Bicycle* the controller must initiate the steering opposite to the desired direction. So the centrifugal force will push the bicycle out of balance, the bicycle starts falling to the opposite side of steering; in order to prevent the bicycle from falling all the way down, the control algorithm should turn the steering bar to the same side of the tilt. The centrifugal force changes direction and starts to pull the bicycle up. This procedure is called counter steering, illustrated in Fig. 1.2 and must be followed whenever changing direction. Once the bicycle is balanced at the equilibrium roll angle, the steering will point to a certain direction. Thus directional control is achieved.

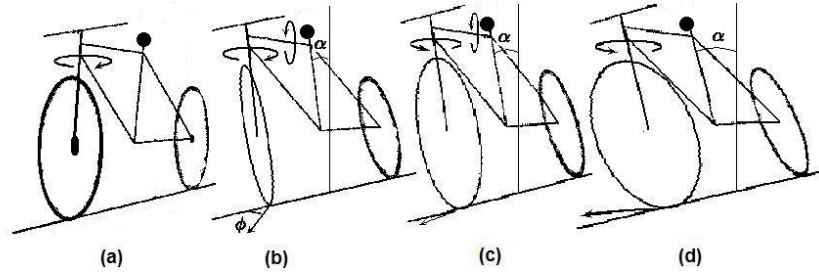


Figure 1.2: Illustration of counter steering. (a) Initiating steering to the left when desired direction is right. (b) Bicycle turns to the left, the centrifugal force rolls the frame to the right. (c) The steering angle decreases. The roll angle continues to increase because the bicycle is still turning left. (d) Steering angle passes through zero and points right. The centrifugal force reverse direction, eventually halt the roll increase and balance the gravity. Steering angle stabilizes, and the bicycle executes the desired right turn.

1.3 Thesis objective

The objective of this thesis is to build a bicycle type two wheeled test platform in order to conduct *lower level control* research and set up the basis for two wheel autonomous vehicle research in the future. The *lower level control* is referred to as balance and directional control.

The mechanical set up continues with an electric scooter which is previously modified by previous undergraduate student research groups [44] [42]. The throttle control circuit is replaced by a motor controller that can be microprocessor controlled. A servo motor is attached to the steering shaft for steering actuation. The original wiring brake handle is also removed and a servo motor is installed as brake actuator.

To perform balance and directional control, roll angle measurement, yaw angle measurement and control problems must be solved. Dynamic roll angle measurement is implemented using accelerometer and gyroscope sensors. An accelerometer measures static tilt angle by transducing the change of gravitational force acting

on its sensing element. When subjected to linear acceleration, it is also affected by external force, such as centrifugal force, as a result the angle reading can be affected. A gyroscope is a roll rate sensor, the signal must be integrated to measure roll angle, thus even a minimum drift error on its calibrated bias will corrupt the result. Thus, the angle estimated by the gyroscope must be periodically corrected. A software filter utilizing features from both sensors to compensate each other needs to be applied to “fuse” the signals [13]. The yaw angle is acquired by integrating the yaw rate signal of a yaw rate gyroscope (also subject to bias drift). My research at this stage is to achieve directional control, not complete navigation; as such, directional deviation during short term is acceptable. Since the yaw rate gyroscope can only tell relative yaw angles the starting direction is always assumed to be the reference direction, i.e. zero yaw angle.

The developed control strategy consists of two parts: at first, simulate balance and path tracking control scenarios with the nonlinear controllers and a nonlinear bicycle model; then use the same model to test the effectiveness of PID controllers which is implemented on the hardware. Under simulation the full dynamics of the bicycle can be tested with the nonlinear controller but the nonlinear controllers are not practical to be used on the HCS12 microprocessor. Model accuracy can be another problem. Thus, I need to find out the suitable conditions to apply PID control. After the PID controller parameters are tuned to be able to control the model, under the same situation the control results of both nonlinear and PID controllers will be compared so as to improve the PID effectiveness. Good control outcomes with PID control can be achieved under certain circumstances. The PID controller is simpler and more suitable to the capability of the microprocessor but the nonlinear controllers can handle wider range of dynamical conditions, for instances, the nonlinear controller is able to lift up the model at a nearly falling roll angle ($\pm 45^\circ$), whereas the maximum roll angle is limited to $\pm 30^\circ$ with PID controller.

Two PID based control scenarios are applied to the platform. The first one maintains the desired roll angle. If the reference roll angle is zero, the control algorithm will attempt to make the bicycle stay upright. To track a non-zero roll angle the bicycle will lean away from upright position meanwhile the direction changes. Under this control strategy there is no direction concept to the vehicle itself; uneven road surface or wind may disturb the balancing and the steering only try to regain balance without any concerns about where to go. The second control algorithm uses a yaw rate sensor to monitor the direction of the bicycle and outputs the desired roll angle as the means to adjust direction. The two control strategies are cascaded together becoming a double close-looped controller, the output of the directional controller is fed into the balance controller as the reference. When the bicycle tracks this reference roll angle by means of steering, the value of the reference roll angle should be in such a way that it causes changes of the steering making the bicycle merging to the desired direction.

The RC control also needs to be implemented through the microprocessor of the system. The RC controller is served as a command center which changes reference roll angles in autonomous mode and controls actuators in manual mode. Control mode switch is carried out using the RC controller as well.

Finally, in order to obtain the real-time states during running, wireless communication using RS232 standard through Bluetooth interface is implemented. Off-line analysis of the dynamical data and simulation of the control results are then conducted with the developed Matlab code.

1.4 Project scope

The tasks involved in this thesis include: constructing a bicycle type prototype test platform and testing different control methods in order to investigate the controlla-

bility of such type vehicle.

Due to the fact that the platform was a commercial scooter without any off-road ability, it is suitable only for operation on hard flat surfaces. The autonomous modification is focused on research and experimental purpose. Most of the actuators are implemented with light duty robot or radio control model servos. Modern control theories have provided sufficient solutions to such kind of non-linear control problems, but sophisticated control algorithms need to be executed on more powerful computer system. Such as the Ghost rider's stochastic model based runtime reinforcement learned controller runs on an embedded PC with an AMD Geode NX1500 1GHz Processor[27]. The physical dimension and power system capability of the platform can not sustain large integrated system. Thus, only non-model based PID control algorithms will be implemented on the platform although more sophisticated model based control have been conducted in a series of simulations[49].

The autonomous bicycle project has a much larger scope. The low level control problem is the foundation of the whole problem. Considering the physical ability many constraints are set. The platform testing is limited on semi-smooth surfaces. The maximum tilt angle (roughly $\pm 20^\circ$ although the actual *critical roll angle* could be much larger) is limited by trainer wheels that supports the platform when stationary.

1.5 Thesis outline

This thesis is organized as follows. Chapter 2 is the literature review of the historical and contemporary bicycle dynamics and control research. Chapter 3 is dedicated to the discussion of the control methods. It compares different control algorithms, analyzes the feasibility of implementing them on the platform, and introduces the final control schemes applied on the vehicle. Chapter 4 presents the implementation

details including the system architecture, software structure, angle measurement, direction measurement, actuation, serial communication and so on. Field test results are discussed in Chapter 5. Finally, in Chapter 6 conclusions and future works are suggested.

Chapter 2

Literature Review

2.1 Bicycle dynamics and stability

One of the earliest bicycle stability studies can be dated back to 1869 by W. J. M. Rankine [36]. The problems were set up as balancing, steering and propelling. The analysis in his work was rather rudimentary and did not give any differential equations. From the modern control point of view it is more of historical interest other than any technical value. The first formal analysis in literature of bicycle self stability is considered by F. J. W. Whipple [50] in 1899. As an undergraduate student in Cambridge University, Whipple developed the famous fourth order model of a straight running bicycle.

Later R. S. Sharp [39] studied the stability in motorcycle design while he was working for B.S.A. motorcycle company, a British based manufacturer. There existed problems regarding tire behaviors in the previous works. In Whipple's work tire sideslip was not allowed. Konda, Nagaoka, and Yoshimura's [25] work allowed tire sideslip but ignored the lag of steering pneumatic tire to build up the side force to the steady states. Sharp [39] developed the linearized equations of motion using Lagrangian method giving an advance in the analysis of the tire behavior and compared the results dependent on three different tire behavior assumptions. The

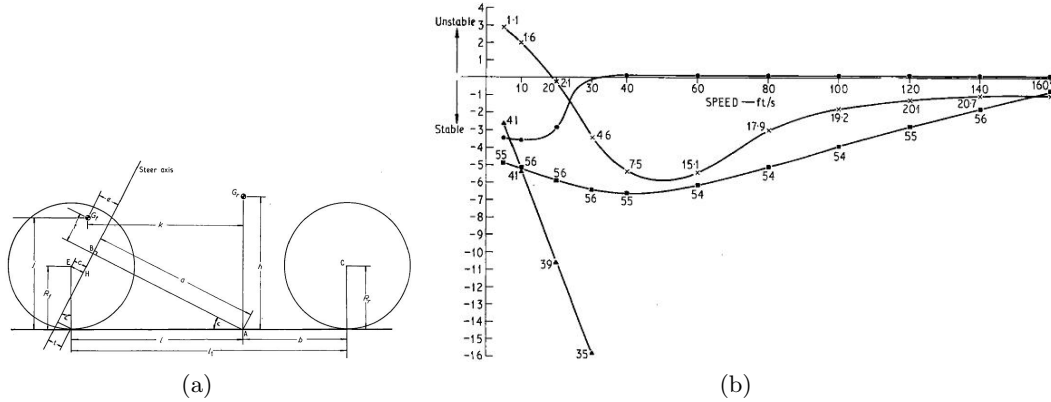


Figure 2.1: R. Sharp motor cycle. (a) Diagrammatic representation of the motor-cycle [39]; (b) Stability and natural frequency of standard machines as a function of forward speed with full tire treatment [39].

diagrammatic representation of a motorcycle and stability as function of forward velocity analysis are shown in Fig. 2.1. Since then Sharp has published a series new researches regarding numerous problems of motorcycle dynamics, stability and control. The paper published in 2001 [40] followed Sharp's prior research in 1978 [38], 1985 [41] comparing the control properties of the motorcycle such as small perturbation from straight line motion and from cornering equilibrium states; steering control by handlebar torque and by rider upper body lean torque; and so on. It indicated that the dominant control of stability is the steering torque and the rider lean is secondary. This paper [40] used speed, acceleration, load, rider stature, accessories as variables to discuss linear stability; and road unevenness to discuss nonlinear stability.

Historical bicycle dynamic studies focus on the stability of the particular type of bicycles popular at its time; did not develop the understanding of relations between design parameters for bicycle stability, according to [18]. The equations of motion did not agree with each other. Hand's master's thesis is part of Cornell university's

bicycle research project. He developed the linearized model using Lagrangian's equation. In order to check the correctness of the model he compared his work with the previous published derivations. A set of twenty historical equations of motion were compared. In the comparison he found out that Whipple [50], Carvallo [11] and Döhrring [15]'s linearized equations, all derived using Newton's Laws, corresponded exactly with his equations; he also found contradictions that Pearsall [33]'s equations differed significantly almost every term from his equation. Although he did not trace the mistakes Hand pointed out there must be major kinematic errors in Pearall's work [18]. Hand discovered inherent connections as well between some historical works such as Sommerfeld and Klein [24]'s equation having all the mass and inertia of the front assembly in the front wheel are somewhat similar to that of Whipple; when compare to Döhrring, are found out to be Whipple's subset. Timoshenko and Young [47] in the *Advanced Dynamics* book derived the nonlinear equation of motion; when linearized, the lean equation agreed with his lean equation. Neïmark and Fufaev [30] referred briefly to Döhrring and derived equation of motion using Lagrange's equation. It's their derivation that Hand mainly followed to develop his equation. After studying Sharp's [39] equation, Hand discovered that if assuming the tires have infinite stiffness (non-holonomic constraint of rolling without sideslip) to reduce the number of equations from four to two an algebraic error occurred. As a result the steering equation was incorrect. He also pointed out two typographical errors in Whipple's linearized equation. Once these errors are corrected, Whipples linearized equations match exactly to the equations derived by Meijaard et al. [29]; these equations are now accepted as benchmark of bicycle dynamics.

2.1.1 Hand's *Basic Bicycle* model and equations of motion

Hand's *Basic Bicycle* model is simplified from the real life bicycle with passive rider (rigid rider, riding no hand, no pedaling) without neglecting major parameters

affecting bicycle self-stability. It consists of four linked rigid bodies, a rigid frame with a fixed rigid rider, front fork/handle bar assembly, and front and rear wheels; a plane of symmetry crosses the center line of the bicycle [18]. Five reference frames are designed in order to derive the equation of motion and measure the inertia of the rigid bodies. The frame system is shown in Fig. 2.2(a).

The *Basic Bicycle Model* then has seven independent generalized coordinates, X_r , Y_r , θ_r , χ_r , ψ , ϕ_r and ϕ_f , which describe its position and rotation. The coordinates X_r and Y_r measures position of the rear wheel contact point P_r in the inertial frame; θ_r is the yaw angle of the rear frame with respect to the OY axis; the angle χ_r measures the roll/lean angle of the rear frame from the vertical; the angle ψ is the steering angle with respect to the rear frame; and the angles ϕ_r and ϕ_f , measure the rotational angle of the rear and front wheel respectively. The position and orientation of the bicycle are described using these generalized coordinates with respect to the inertial frame. The seven independent generalized coordinates are shown in Fig. 2.2(b).

A number of assumptions are used in order that the equation can be linearized:

1. The wheel and tire are considered as a whole knife-edged integrity, no deformation and side-slipping, point contacting with the ground, rolling on a rigid flat horizontal surface with enough friction between the wheels and road to prevent sliding. This is critical in deriving the rolling constraints.
2. The rider does not move relative to the frame.
3. Only small disturbances from the vertical straight-ahead equilibrium position are considered. This assumption allows the equations to be linearized.
4. No internal torque in the system itself or no motion within the bicycle-rider.

5. The bicycle always travels at constant speed, it is as well a consequence of linearization.

Hand's final equations of motion have two coupled second order equations.

$$M_{\chi\chi}\ddot{\chi}_r + C_{\chi\chi}\dot{\chi}_r + K_{\chi\chi}\chi_r + M_{\chi\psi}\ddot{\psi} + C_{\chi\psi}\dot{\psi} + K_{\chi\psi}\psi = M_{\chi_r} \quad (2.1)$$

$$M_{\psi\psi}\ddot{\psi} + C_{\psi\psi}\dot{\psi} + K_{\psi\psi}\psi + M_{\psi\chi}\ddot{\chi}_r + C_{\psi\chi}\dot{\chi}_r + K_{\psi\chi}\chi_r = M_{\psi} \quad (2.2)$$

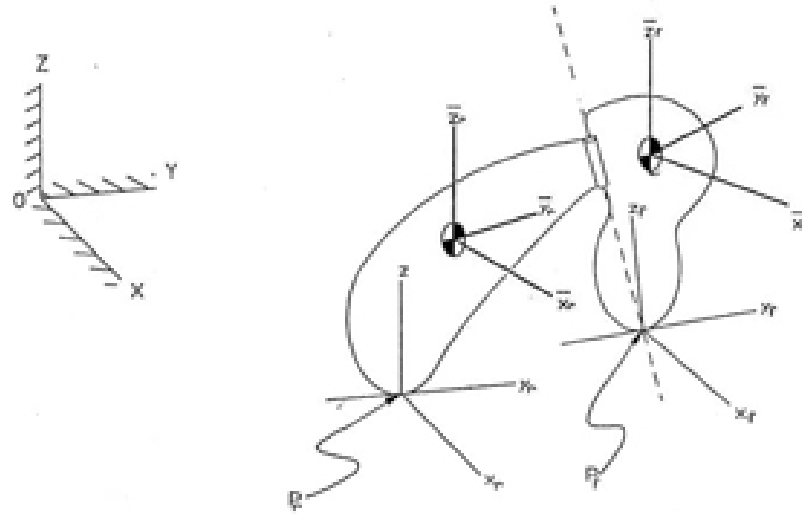
M_{χ_r} represents the torque applied to the rear frame supporting the bicycle from tilting; Eq. (2.1) is named lean equation for this sake. M_{ψ} is steering torque asserted by the rider to the steering bar; thus Eq. (2.2) is called steer equation by Hand. The meaning of the coefficients can be found in [18]. Using these equations Hand studied the self-stability of the bicycle-rider system with different set up of M_{χ_r} and M_{ψ} . Then he figured out several self-stable configurations which can be used to improve bicycle design and to understand the design changes on bicycle stability.

2.2 Control of bicycle

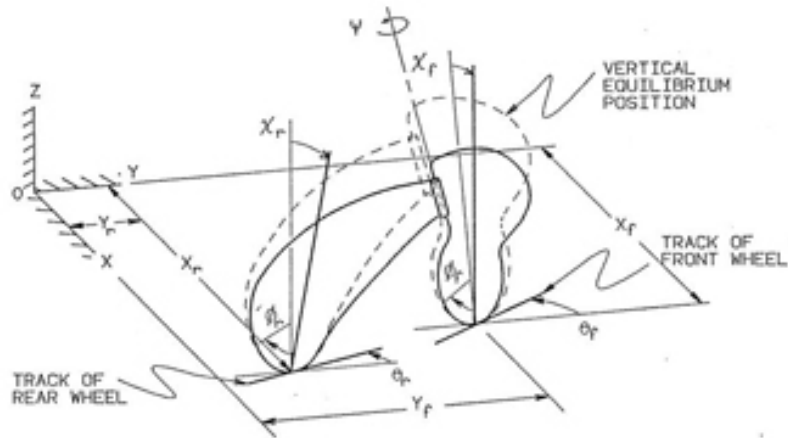
The bicycle control problems offer many challenges in the area of mechanical control systems.

2.2.1 Balance control and internal equilibrium roll angle control

Getz et al. [17] developed a balance controller using steering and rear-wheel torque to control his nonlinear bicycle model recovering from a nearly falling roll angle and later in his another work he applied internal equilibrium control to solve path-tracking while maintaining balance problem. Figure 2.3 shows Getz's simplified bicycle model; (a) is the side view of the bicycle in upright position, the bicycle is considered as a point mass with its value equal to m , steering axis is perpendicular to the ground, the reacting force of ground u_r acting on the rear wheel is parallel to



(a)



(b)

Figure 2.2: R.S. Hand's bicycle. (a) 1) The inertial reference frame XYZ fixed to the ground with origin at point O . 2) A moving reference frame $x_r y_r z_r$ fixed to the rear wheel frame with origin at P_r , the rear contact point. 3) A moving reference frame $x_f y_f z_f$ fixed to the front fork/handle bar assembly with origin at P_f , the front contact point. 4) A moving reference frame $\bar{x}_r \bar{y}_r \bar{z}_r$ fixed to the rear wheel frame with origin at \bar{P}_r , the center of mass of the rear part of the bicycle (rear frame + rider + rear wheel). 5) A moving reference frame $\bar{x}_f \bar{y}_f \bar{z}_f$ fixed to the front fork/handle bar assembly with origin at \bar{P}_f , the center of mass of the front part of the bicycle (front fork/handlebar assembly + front wheel). (b) Generalized coordinates [18].

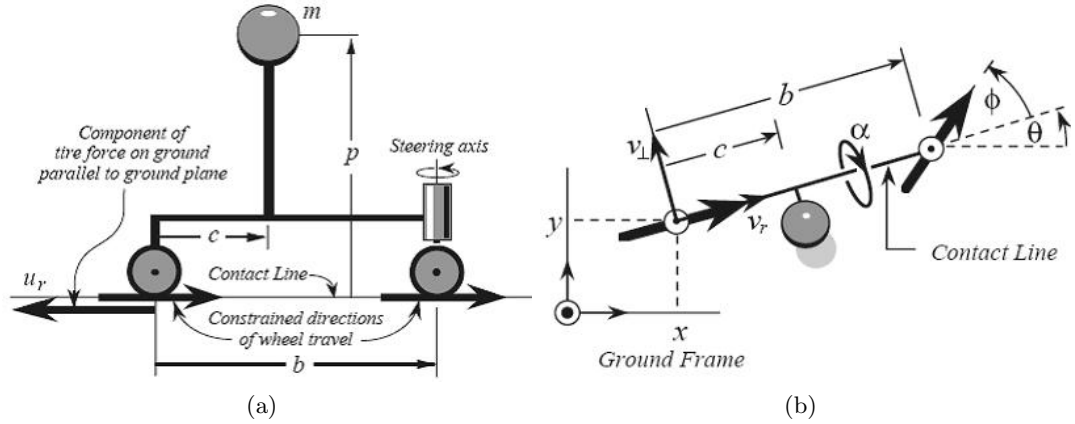


Figure 2.3: Getz's simplified bicycle model. (a) Side view of a bicycle model with roll angle $\alpha = 0^\circ$ [17]; (b) Top view of a bicycle model with a roll angle α , steering angle ϕ , yaw angle θ and coordinates of (x, y) [17].

the ground plane; (b) is the top view of the bicycle; it is located in a ground inertial reference frame with the coordinates of the rear wheel ground contact point of (x, y) and rolls clockwise of a roll angle α , ϕ is the steering angle, v_r is the rear wheel speed, v_\perp is the side-slip speed, θ is the yaw angle of the bicycle about the x axis of the inertial frame, b is the dimension of wheelbase, and c is the projected distance from the center of mass to the rear wheel contact point on the ground. This model comes with one ground inertial frame and five generalized coordinates $(x, y, \alpha, \theta, \phi)$.

The bicycle position in the ground inertial frame can be concluded from Fig. 2.3b as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} v_r \\ v_\perp \end{bmatrix} \quad (2.3)$$

Also define the parameterized steering variable: $\sigma := \tan \phi / b$; the constraints of the wheels roll without side slipping is expressed as:

$$v_\perp = 0, \quad \dot{\theta} = v_r \sigma \quad (2.4)$$

Define steering output $\omega_\sigma := \dot{\sigma}$.

$$\dot{\sigma} = \omega_\sigma \quad (2.5)$$

The equations of motion is derived using Lagrangian equation by Getz [17].

$$M \begin{bmatrix} \ddot{\alpha} \\ \dot{v}_r \end{bmatrix} = F + B \begin{bmatrix} \omega_\sigma \\ u_r \end{bmatrix} \quad (2.6)$$

Where:

$$M = \begin{bmatrix} p^2 & -cp \cos \alpha \sigma \\ -cp \cos \alpha \sigma & 1 + (c^2 + p^2 \sin^2 \alpha) \sigma^2 + 2p\sigma \sin \alpha \end{bmatrix}$$

$$F = \begin{bmatrix} gp \sin \alpha + (1 + p\sigma \sin \alpha) p \sin \alpha \sigma v_r^2 \\ -(1 + p\sigma \sin \alpha) 2p \cos \alpha \sigma v_r \dot{\alpha} - cp \sigma \sin \alpha \dot{\alpha}^2 \end{bmatrix}$$

$$B = \begin{bmatrix} cp \cos \alpha v_r & 0 \\ -(c^2 \sigma + p \sin \alpha (1 + p\sigma \sin \alpha)) v_r & 1/m \end{bmatrix}$$

Equation (2.3), (2.4), (2.5), and (2.6) form the entire state space of the bicycle rolling dynamics and moving kinematics.

The controller is designed using the feedback linearization method.

$$\begin{bmatrix} \omega_\sigma \\ u_r \end{bmatrix} = B^{-1} \left(M \begin{bmatrix} V_\alpha \\ V_r \end{bmatrix} - F \right) \quad (2.7)$$

where:

$$V_\alpha = -\beta_{\alpha 1}(\dot{\alpha} - \dot{\alpha}_d) - \beta_{\alpha 0}(\alpha - \alpha_d)$$

$$V_r = -\beta_{r 0}(v_r - v_{rd})$$

$\beta_{\alpha 1}$, and $\beta_{\alpha 0}$ are the controller gains. The values of these gains should cause the resulting linear system to have negative real part eigenvalues. The balance controller can recover the bicycle from a nearly falling roll angle as well as follow desired roll angle trajectories.

For the path-tracking problems, in order to follow the target path precisely, equation (2.3) is differentiated twice to obtain the kinematic tracking equations. This also requires the target path formula to be C^3 , i.e. third order derivative exists. After variables change, $u^r := \ddot{v}_r$ and $u^\theta := \ddot{\theta}$ the final equation is as follow [17]:

$$\begin{bmatrix} x^{(3)} \\ y^{(3)} \end{bmatrix} = \begin{bmatrix} -2\dot{v}_r \sin \theta - v_r \cos \theta \dot{\theta} \\ 2\dot{v}_r \cos \theta - v_r \sin \theta \dot{\theta} \end{bmatrix} \dot{\theta} + \begin{bmatrix} \cos \theta & -v_r \sin \theta \\ \sin \theta & v_r \cos \theta \end{bmatrix} \begin{bmatrix} u^r \\ u^\theta \end{bmatrix} \quad (2.8)$$

The roll dynamic is the first equation of (2.6):

$$\ddot{\alpha} = \frac{g}{p} \sin \alpha + \frac{1}{p} \left(1 + \frac{p\dot{\theta} \sin \alpha}{v_r} \right) \cos \alpha \dot{\theta} v_r + \frac{c}{p} \cos \alpha u^\theta \quad (2.9)$$

Controllers are also designed with the feedback linearization method. The path following controller is:

$$\begin{bmatrix} u_{ext}^r \\ u_{ext}^\theta \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta / v_r & \cos \theta / v_r \end{bmatrix} \cdot \left(- \begin{bmatrix} -2\dot{v}_r \sin \theta - v_r \cos \theta \dot{\theta} \\ 2\dot{v}_r \cos \theta - v_r \sin \theta \dot{\theta} \end{bmatrix} \dot{\theta} + \begin{bmatrix} V_x \\ V_y \end{bmatrix} \right)$$

Where:

$$V_x := x_d^{(3)} - \sum_{i=0}^2 \gamma_i (x^{(i)} - x_d^{(i)})$$

$$V_y := y_d^{(3)} - \sum_{i=0}^2 \gamma_i (y^{(i)} - y_d^{(i)})$$

The balance controller is named the internal equilibrium controller [17].

$$u_{int}^\theta = \frac{p}{c \cos \alpha} \left(-\frac{g}{p} \sin \alpha - \frac{1}{p} \left(1 + \frac{p\dot{\theta} \sin \alpha}{v_r} \right) \cos \alpha \dot{\theta} v_r + v_{int}^\theta \right) \quad (2.10)$$

$$v_{int}^\theta = \ddot{\lambda}_d - \beta_1 (\dot{\alpha} - \dot{\lambda}_d) - \beta_0 (\alpha - \lambda_d) \quad (2.11)$$

Here v_{int}^θ is equivalent to the steering output. λ_d , $\dot{\lambda}_d$, and $\ddot{\lambda}_d$ are the values calculated from the internal equilibrium manifold.

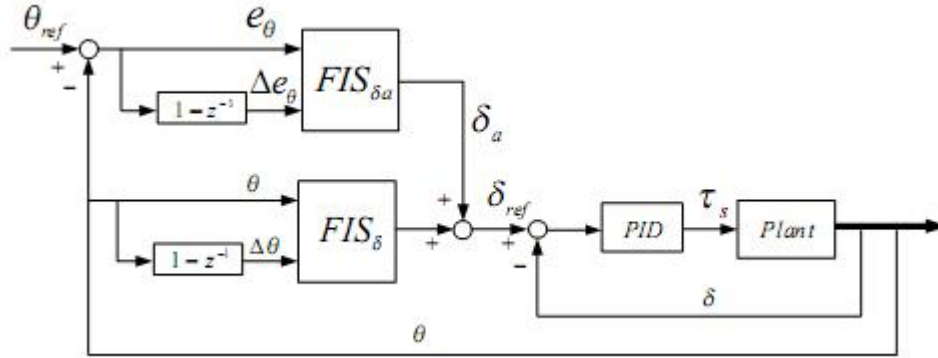


Figure 2.4: Balance control with fuzzy controller [12].

2.2.2 Control a bicycle with Fuzzy controller

Fuzzy controllers [16] formulate human knowledge by rule bases from human experience. The controller design scheme follows the human experience in riding a bicycle. Fuzzy control deals with fuzzified information, in terms of bicycle system, e.g. roll angle is large or small and roll rate is fast or slow, etc. The outputs from fuzzy control are usually decision making. Whereas the interface in between the plant and controller is usually crisp, i.e. sensory signals, actuators, etc., and also exists issues of real-time calculation speed. This makes a fuzzy controller more suitable to be a higher level controller. Chih-Keng Chen and Thanh-Son Dao [12] developed a cascaded structure with a fuzzy controller and low level PID controller for steering control.

Bicycle balance control

Our bicycle riding experience shows that, the bicycle can be balanced by generating an appropriate steering torque to steer the fork into the direction of rolling causing the bicycle make a turn, such that the centrifugal force generated by turning motion balances the roll motion caused by gravity.

The balance control scheme is shown in Fig. 2.4. The steering torque is denoted by τ_s . The PID controller in this diagram is to generate the steering torque according to the difference between the desired steering angle δ_{ref} and the measured steering angle δ . To generate an appropriate reference steering angle δ_{ref} , a fuzzy controller (denoted as FIS_δ) is used. The fuzzy controller FIS_δ uses two inputs: the measured roll-angle θ and its change $\Delta\theta$ to regulate the steering angle until the bicycle turns to the up-right position.

For roll-angle tracking control, suggested in Getz's first scheme, another fuzzy controller (denoted as FIS_{δ_a}) is added. Denote the difference between the desired roll-angle θ_{ref} and the actual roll-angle θ as e_θ . The controller FIS_{δ_a} tries to minimize e_θ and its change, Δe_θ , by creating an additional angle δ_a as shown in Fig. 2.4. The balancing and roll-angle tracking tasks can be accomplished with this scheme. Detailed fuzzy control membership function generation, signal fuzzification and defuzzification can be referred to [16].

Ground path tracking strategy

The goal of bicycle path-following control is to minimize the distance from the reference point c on the bicycle to the target path, shown in 2.5. Let e_L be the shortest distance on the ground plane from c to the target path. From our bicycle riding experience, the bicycle can be pushed towards the target path by leaning our body to the appropriate side. Using this strategy, we can design another controller to regulate the roll angle so that e_L can be minimized. The magnitude of the roll angle is dependent on the distance e_L . In order to retain balance, the value of the roll angle θ should not be too large. The range of θ_{ref} is limited within $[-30^\circ, 30^\circ]$.

For path-tracking control, another control loop is added to the existing one of Fig. 2.4. The control scheme is shown in Fig. 2.6. A fuzzy controller $FIS_{path-trk1}$ generates the corresponding θ_{ref} angle according to e_L and its change Δe_L . θ is

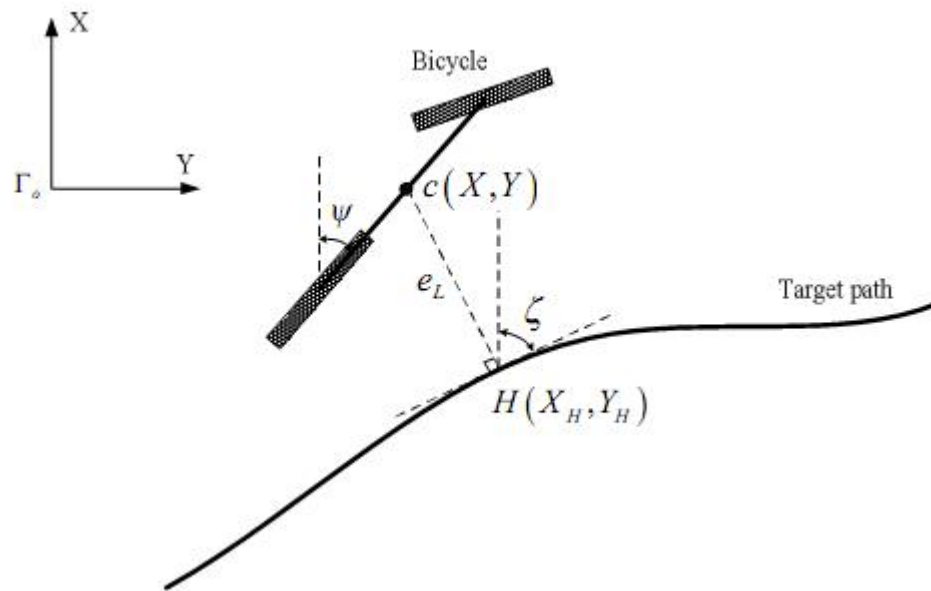


Figure 2.5: Path tracking strategy [12], Γ_0 is the ground inertial reference frame, $c(X, Y)$ is the coordinate of the bicycle reference point in the reference frame, ψ is the yaw angle of the bicycle, $H(X_H, Y_H)$ is the coordinate of the point on the target path where the shortest distance from the bicycle to the target path occurs and e_L is the value of the shortest distance, ζ is the angle of the tangent line at $H(X_H, Y_H)$ of the target path made with the X axis of the inertial frame.

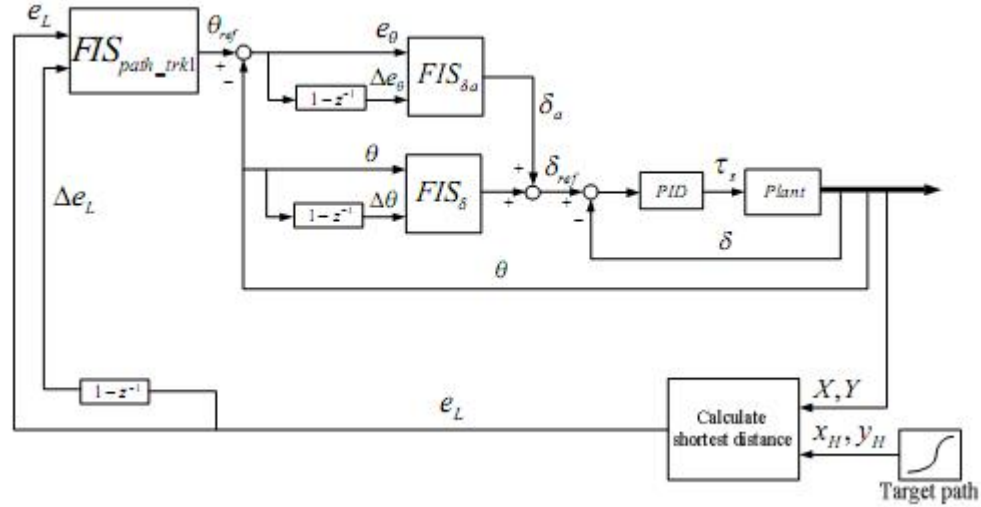


Figure 2.6: Path-following control using e_L and Δe_L for regulating roll-angle [12].

negative when the bicycle leans to the left and positive when it leans to the right. By considering the bicycle in the following three typical situations, the corresponding if-then rules are listed as:

- If the bicycle is far from the target path to the left (e_L is negative large) and moving fast further away from the path (Δe_L is negative large); then one should roll the bicycle to the right at a large angle to make the bicycle go to the right (θ_{ref} positive large).
- If the bicycle is already in its proper position ($e_L=0$; $\Delta e_L=0$); then no control effort is needed. ($\theta_{ref}=0$).
- If the bicycle is far from the target path to the right (e_L is positive large) and moving fast further away from the path (Δe_L is positive large); then one should roll the bicycle to the left at a large angle to make the bicycle go to the left (θ_{ref} negative large).

In a similar fashion, the complete rule-base can be constructed. For this controller with two inputs and seven linguistic values of each case, there are at most $7^2 = 49$ possible rules.

2.3 Sensor signal optimization

Digital filtering is the commonly used signal processing technique to remove noise, fuse information from different sensors in order to predict the optimized states.

2.3.1 Kalman filtering

Kalman filter [23] published in 1960 by R.E. Kalman is one of the most popular digital filters of signal processing. Kalman filtering has been applied in areas of broadcasting, aerospace, marine navigation, nuclear power plant instrumentation, demographic modeling, manufacturing, and many others. It provides a recursive solution to the discrete-data linear filtering problems; it estimates the past, present, and future states even without knowing precise model of a system.

A generalized problem of $x \in \mathfrak{R}^n$ with the input of $u(k) \in \mathfrak{R}^p$, process noise of $w(k) \in \mathfrak{R}^q$, and measurement noise of $v(k) \in \mathfrak{R}^r$ is described as:

$$x(k+1) = A(k)x(k) + B(k)u(k) + w(k) \quad (2.12)$$

with a measurement $z \in \mathfrak{R}^m$:

$$z(k) = H(k)x(k) + v(k) \quad (2.13)$$

$A(k)$ is the $n \times n$ system matrix, $B(k)$ is the $m \times p$ input matrix, and $H(k)$ is the $m \times n$ measurement matrix.

The question addressed by the Kalman filter is: Given the knowledge of the behavior of the system and measurements (with a lot of process and measurement noise) to estimate the best values of the future states; to make the expected value

of the state estimate is equal to the expected value of the true state on average and minimize the expected value of squared deviation or variance of the estimation error.

Assuming that the process noise $w(k)$ is white with a covariance matrix $Q(k)$ and the measurement noise $v(k)$ is white with a covariance matrix $R(k)$, $w(k)$ and $v(k)$ are not correlated, the Kalman filter can be formulated as:

1. State prediction:

$$\hat{x}(k+1|k) = A(k)\hat{x}(k|k) + B(k)u(k) \quad (2.14)$$

2. Measurement Prediction:

$$\hat{z}(k+1|k) = H(k)\hat{x}(k+1|k) \quad (2.15)$$

3. Measurement Residual:

$$v(k+1) = z(k+1) - \hat{z}(k+1|k) \quad (2.16)$$

4. Update state estimate:

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1)v(k+1) \quad (2.17)$$

The “ $\hat{\cdot}$ ” on the top of the symbols represents the estimated states. $W(k+1)$ is the gain matrix or Kalman Gain. The following procedure is the state covariance estimation:

1. State prediction covariance:

$$P(k+1|k) = A(k)P(k|k)A(k)^T + Q(k) \quad (2.18)$$

2. Measurement prediction covariance:

$$S(k+1) = H(k+1)P(k+1|k)H(k+1)^T + R(k+1) \quad (2.19)$$

3. Gain matrix or Kalman Gain:

$$W(k+1) = \frac{P(k+1|k)H(k+1)^T}{S(k+1)} \quad (2.20)$$

4. Update state covariance:

$$P(k+1|k+1) = P(k+1|k) - W(k+1)S(k+1)W(k+1)^T \quad (2.21)$$

The recursive algorithm of Kalman Filter is very time efficient making it suitable to be implemented on digital computer. But for a HSC12 [1] microprocessor it is still a very heavy load.

2.3.2 Complementary filter

To measure roll angle of crafts, accelerometers and gyroscopes need to be used. Accelerometers are affected by horizontal acceleration, gyroscopes subject to bias drift. Shane Colton [13] presents a complementary filter which is simple enough to be easily implemented on microprocessors. It satisfies the following requirements:

- Fusing signals from an accelerometer and a gyroscope together.
- Utilizing features from both sensors to minimize the effect of external forces to the accelerometer and bias drift of the gyroscope in the final angle output.

Fig. 2.7 illustrates the roll angle measurement with the complementary filter. The formula of the complementary filter is:

$$\alpha = \underbrace{(1 - \beta) \times (\alpha_0 + \dot{\alpha}_{gyro} \times dt)}_{\text{highpass filter}} + \underbrace{\beta \times (\alpha_{acc})}_{\text{lowpass filter}} \quad (2.22)$$

Where: α_0 = angle calculated from the last iteration

dt = sampling period

β = filter coefficient

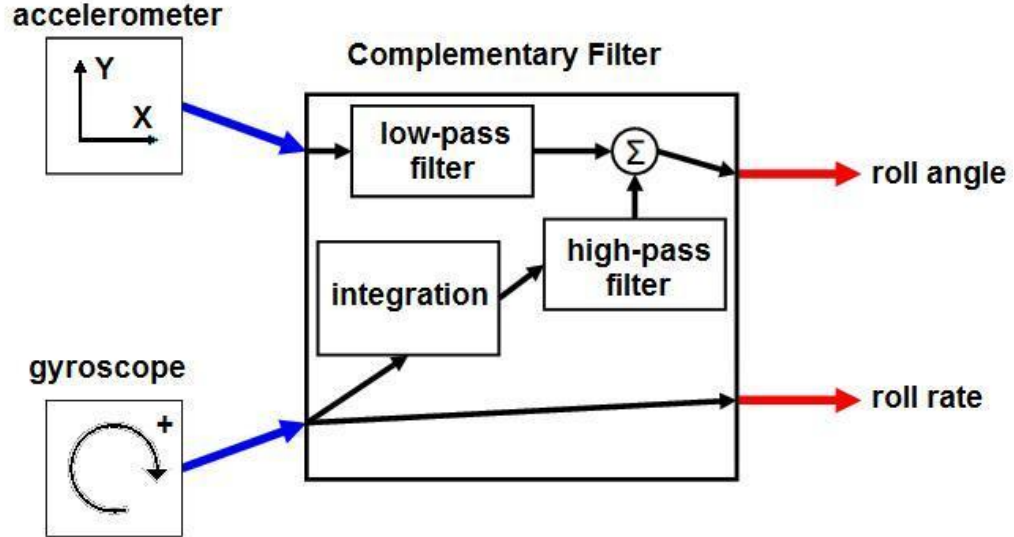


Figure 2.7: Complementary filter to measure roll angle and angular rate.

$\dot{\alpha}_{gyro}$ is the measured roll rate by the gyroscope, it runs on the high-pass filter to filter out the long duration gyro-drift. α_{acc} is the measured roll angle by the accelerometer, it runs on the low-pass filter to filter out the high frequency horizontal acceleration. Increasing the filter coefficient will result in more trust on the accelerometer output, decreasing the coefficient will put more trust to the gyroscope (gyroscope coefficient is increasing). If choose the sampling period $10ms$ and coefficient $\beta = 0.10$, the combined time constant of highpass-lowpass filter can be calculate as [13]:

$$\tau = \frac{(1 - \beta) \times dt}{\beta} = \frac{0.90 \times 10ms}{0.1} = 90ms \quad (2.23)$$

Colton [13] also presents a test result of the filter, shown in Fig. 2.8. He used an ADXRS401, Analog Devices iMEMS $75^\circ/sec$ gyroscope and an ADXL203, Analog Devices iMEMS 2-axis accelerometer on a custom PIC-based wireless controller, 10-bit ADCs, based on the Machine Science XBoard [28]. The sampling rate is 79 Hz,

filter coefficients are 0.98 and 0.02, and time constant is 0.62 sec. The test result shows how the filter handles both horizontal acceleration disturbances while not rotating and gyroscope drift problems.

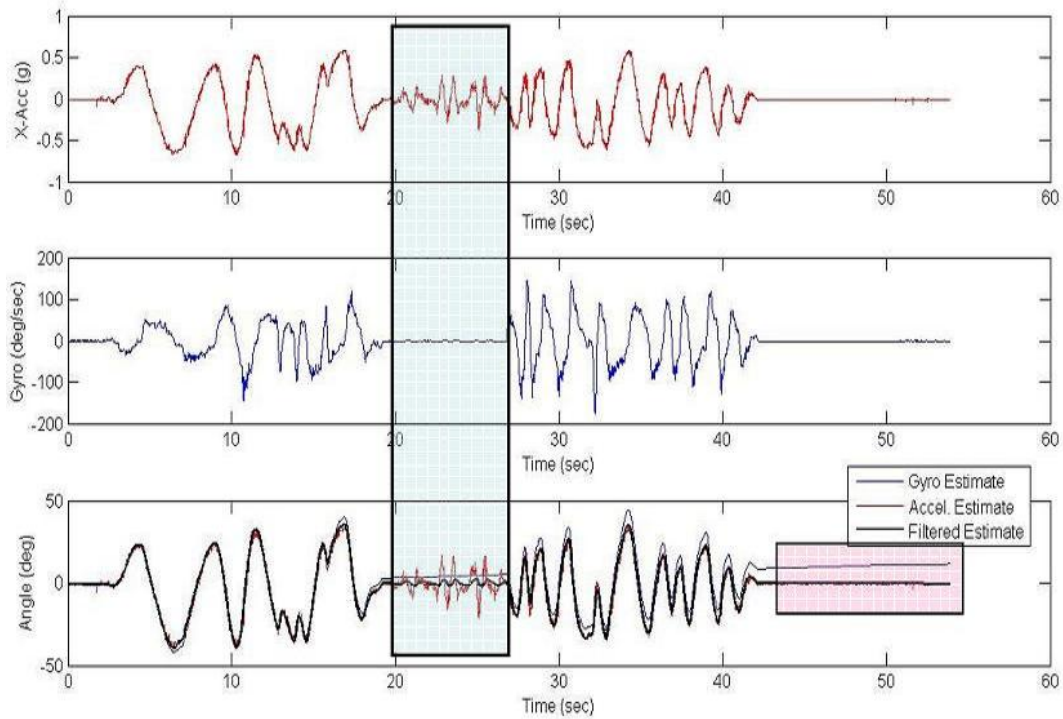


Figure 2.8: Experiment result of the effect of the complementary filter. Top: the accelerometer output plot. Center: the gyroscope output. Bottom: the combined angular outputs plot of the filter, accelerometer, and gyroscope. (a) inside the highlighted blue box, in the top plot, the accelerometer output is affected by the horizontal acceleration while not rotating; at the bottom, the effect of the horizontal acceleration is restrained by the filter. (b) inside the highlighted red box, after a longer time the gyroscope bias drift is more evident (the sensor still not rotating); at the bottom plot, the gyroscope drift is corrected by the complementary filter [13].

Chapter 3

Methodology

Additional stabilization mechanisms may improve stability. For example, a shifting mass to emulate the rider's effort can help balance the bicycle [26]. An internal flywheel can also enhance the gyroscope effect to make a bicycle more stable. Bicycles with non-standard features such as gyroscopes, dampers to enhance or reduce the self-stability can be called *Augmented Bicycle* in contrast to the *Basic Bicycle*. The only way to balance a *Basic Bicycle* is by using proper steering action during running. This chapter compares different control methods and then discusses the final choices of the control algorithms on the hardware platform for this work.

3.1 Model based bicycle control simulation

Based on Getz's model and control methods, two simulation models using Ptolemy II [10] are developed to study the bicycle control problems. The 3D model in Fig. 3.1 visualizes the generalized coordinates of the bicycle.

3.1.1 Balance Control Simulation

Fig. 3.2 is the balance control PtolymeII model, the full scaled model see Appendix B. Let $x_1 = \alpha$, $x_2 = \dot{x}_1$, $x_3 = v_r$. The state space expression of the bicycle dynamics

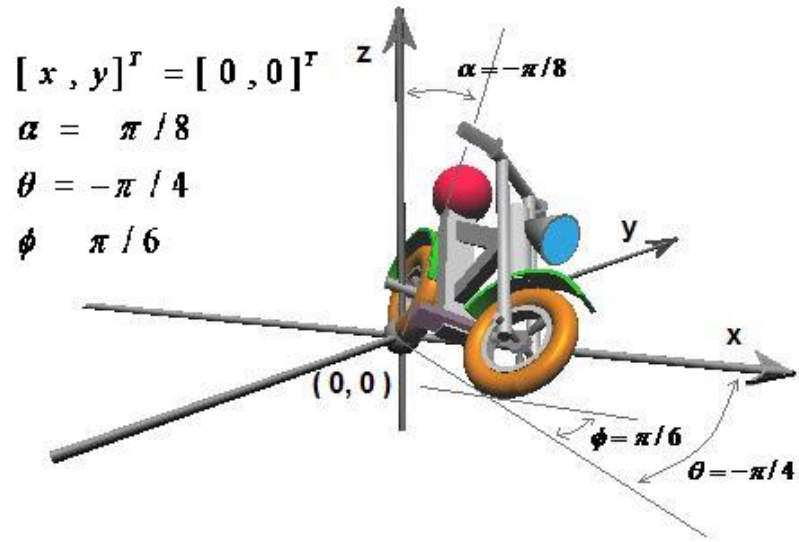


Figure 3.1: 3D bicycle model with positive steering, negative roll angle, negative yaw angle; and located at the origin of the inertial frame.

is derived from Eq. (2.6):

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(x_2, x_1, x_3, \omega_\sigma, u_r) \\ \dot{x}_3 = g(x_2, x_1, x_3, \omega_\sigma, u_r) \end{cases} \quad (3.1)$$

Where the input variables are: ω_σ , u_r , and the output mapping matrix is:

$$\begin{bmatrix} \dot{\alpha} \\ \alpha \\ v \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \\ x_3 \end{bmatrix} \quad (3.2)$$

Given the physics of the bicycle and controller gains:

c	g	p	m	b	$\beta_{\alpha 1}$	$\beta_{\alpha 0}$	$\beta_{r 0}$	h
0.5[m]	9.8[m/s ²]	0.5[m]	20kg	1.0[m]	2.0	6.0	1.0	0.01[s]

Two simulations were conducted with the same model. The first simulation shows the bicycle recovering from a nearly fall roll angle to upright position in

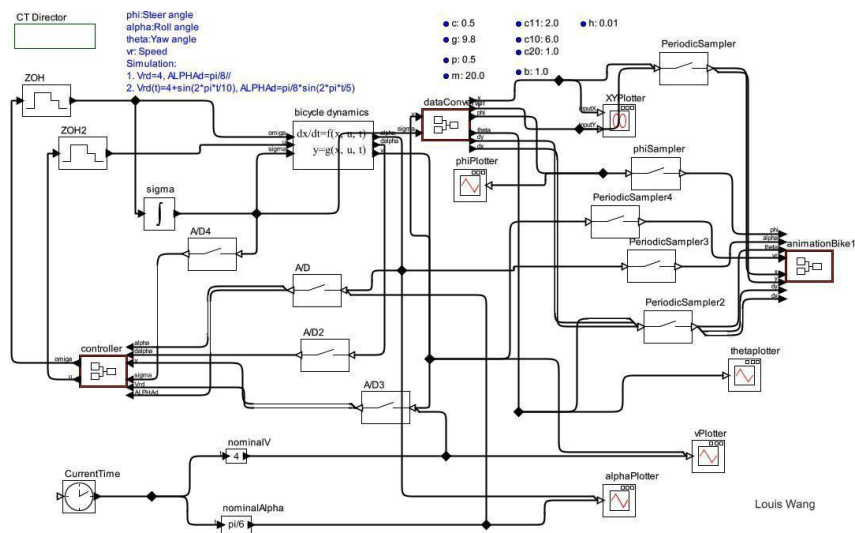


Figure 3.2: PtolemyII model of bicycle balance control [49]. The balance control maintains balance of the bicycle model as well as tracks desired roll angle trajectories. 1) The *CT director* defines the continuous domain of the model. 2) The *Bicycle dynamics* actor contains the bicycle dynamic state space equations. 3) The controller equations are included in the *controller* actor. 4) The *animationBike1* actor is the 3D model of the bicycle. See also Appendix B for the full scaled image.

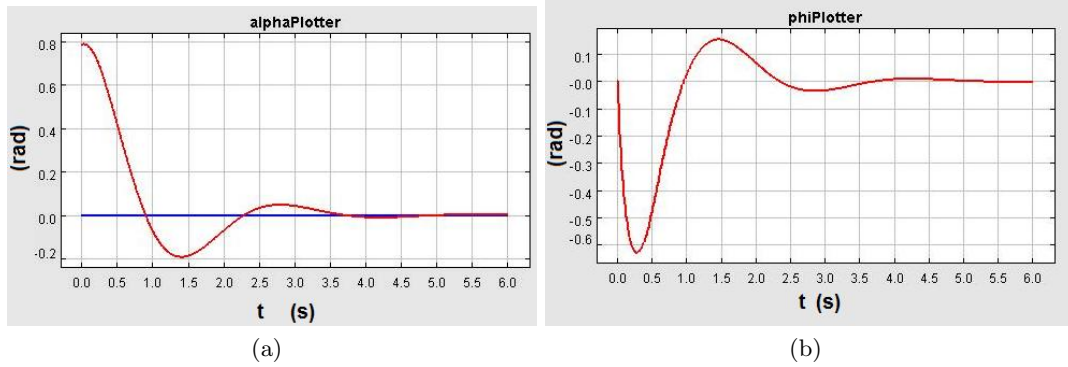


Figure 3.3: Balance control of upright position plot. (a) Roll angle recovers from nearly fall roll angle to upright position; (b) Steering to the same side of rolling pulls the bicycle up.

Fig. 3.3. When the initial roll angle is set at $\pi/4$ and desired roll angle zero, the steering turns to the same side of rolling at a large angle and pulls the bicycle up. The centrifugal force generated by the turning action counteracts the gravity and stops the bicycle from falling.

The second simulation is to control the bicycle tracking a desired, $+\pi/6$, roll angle. The initial condition of the bicycle is: roll angle, $-\pi/6$; and speed, $4m/s$. Fig. 3.4 shows the result of this simulation, where (a) is the trajectory on the ground plane and (b) is the roll angle plot. To demonstrate the balancing behavior a few screen shots at different time instants from this simulation are taken, Fig. 3.5; the first image is taken at $0.3s$ after start, shown in (a) the bicycle roll to the left, correspondingly the steering turns to the left trying to pull it up; (b) at $0.6s$, the bicycle is almost upright, steering is straight as well. (c) at $1.0s$, the roll can not immediately stop, the bicycle starts rolling to the right side, and the steering that follows, (d) at $1.5s$, the centrifugal force is not large enough to balance the gravity, thus, the bicycle falls further and steers further. (e) at $2.0s$, the amount of pulling force exceeds the falling force the bicycle tents to move back to upright position, but

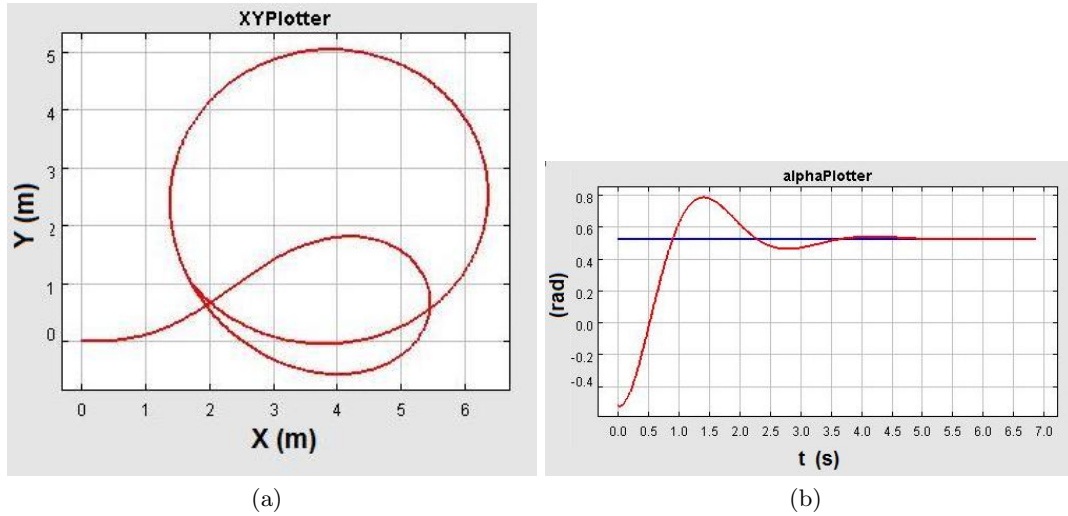


Figure 3.4: Balance control of non-zero roll angle plot. (a) Bicycle trajectory on the ground plane; (b) Roll angle.

the controller makes it follow $+\pi/6$ roll angle. (f) after 4s the bicycle catches up to the equilibrium state and moves at a constant roll angle $+\pi/6$, constant steering angle and constant speed; the bicycle is doing a circular motion.

3.1.2 Ground path tracking with balance Simulation

Fig. 3.6 is the path tracking with balance control PtolemyII model, the full scaled model see Appendix C. The internal equilibrium control [17] is to find out the equilibrium states during path tracking such that the bicycle is balanced while moving along the reference ground path. The equilibrium state is defined as when the sum of the exterior forces acting on the bike equals to zero. The equilibrium state is the solution of α in Eq. (2.9) with rolling velocity and rolling acceleration equal to zero, i.e. $\dot{\alpha} = \ddot{\alpha} = 0$; meanwhile the steering outputs make the bicycle tracking the reference path. By substituting $\ddot{\alpha} = 0$ into Eq. (2.9) and divided by $\cos \alpha$ and times

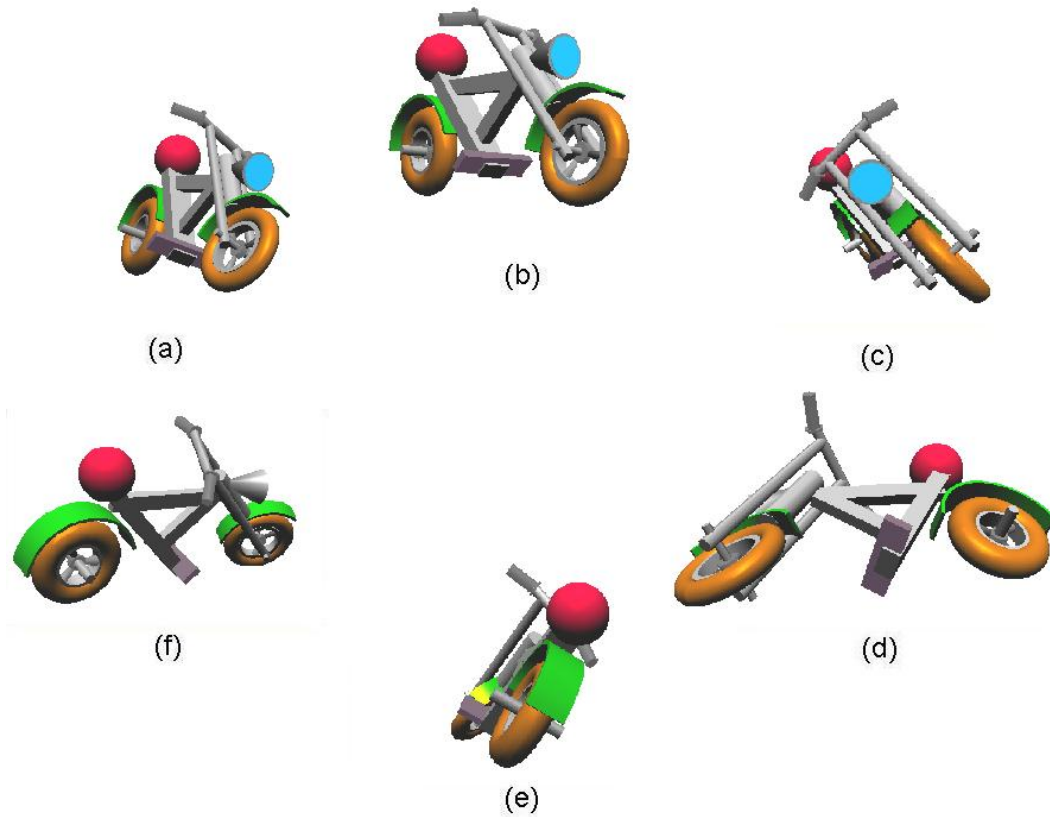


Figure 3.5: Bicycle status of non-zero roll angle control at different time instant. (a) 0.30s, the bicycle leans to the left side, the steering turns to the same side of leaning. (b) 0.60s, the bicycle is almost upright due the centrifugal effect. (c) 1.0s, the lean can not stop immediately; the bicycle leans to the right side, the steering follows the leaning, and the centrifugal force reverses direction. (d) 1.5s, the bicycle leans further and steers further. (e) 2.0s, the bicycle is lift up again by the centrifugal force. (f) After 4.5s, the bicycle catches up the equilibrium state and performs a steady turning motion.

p to both sides of the equation; it is transformed into:

$$0 = g \tan \alpha_e + \left(1 + \frac{p\dot{\theta} \sin \alpha_e}{v_r}\right) \dot{\theta} v_r + c u_{ext}^\theta \quad (3.3)$$

Here, α_e is referred to as desired roll angle and used to replace α in Eq. (3.3).

The state space model of path tracking with balance is expressed as follow:

State variables: $x_1 = x$, $x_2 = \dot{x}_1$, $x_3 = \dot{x}_2$, $x_4 = y$, $x_5 = \dot{x}_4$, $x_6 = \dot{x}_5$, $x_7 = \alpha$,
 $x_8 = \dot{\alpha}$

Input variables: u^r , u^θ .

θ , $\dot{\theta}$, v_r and \dot{v}_r are treated as intermediate variables from definition, $u^r := \ddot{v}_r$
and $u^\theta := \ddot{\theta}$,

State space equation set:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \dot{x}_3 = -2\dot{v}_r \sin \theta \dot{\theta} - v_r \cos \theta \dot{\theta}^2 + \cos \theta u^r - v_r \sin \theta u^\theta \\ \dot{x}_4 = x_5 \\ \dot{x}_5 = x_6 \\ \dot{x}_6 = 2\dot{v}_r \cos \theta \dot{\theta} - v_r \sin \theta \dot{\theta}^2 + \sin \theta u^r + v_r \cos \theta u^\theta \\ \dot{x}_8 = \frac{g}{p} \sin(x_7) + \frac{\cos(x_7)\dot{\theta}v_r}{p} + \sin(x_7) \cos(x_7) \dot{\theta}^2 + \frac{c}{p} \cos(x_7) u^\theta \end{cases} \quad (3.4)$$

Output variables:

$x = x_1$, $y = x_4$, $\dot{x} = x_2$, $\dot{y} = x_5$, $\ddot{x} = x_3$, $\ddot{y} = x_6$, $\alpha = x_7$, $\dot{\alpha} = x_8$

Parameters and controller gains:

c	g	p	m	b	β_1	β_0	γ_2	γ_1	γ_0	h
0.5[m]	9.8[m/s ²]	0.5[m]	20kg	1.0[m]	20.0	100.0	3.0	4.5	3.5	0.01[s]

Initial conditions:

$v(0)$	$x(0)$	$y(0)$	$\dot{x}(0)$	$\dot{y}(0)$	$\ddot{x}(0)$	$\ddot{y}(0)$	$\alpha(0)$	$\dot{\alpha}(0)$
10.0[m/s]	0	5.0[m]	2.5[m/s]	0	0	0	0	0

Fig. 3.7 is the simulation of tracking a sinusoidal ground path. The equation of a sinusoidal wave ground path is given:

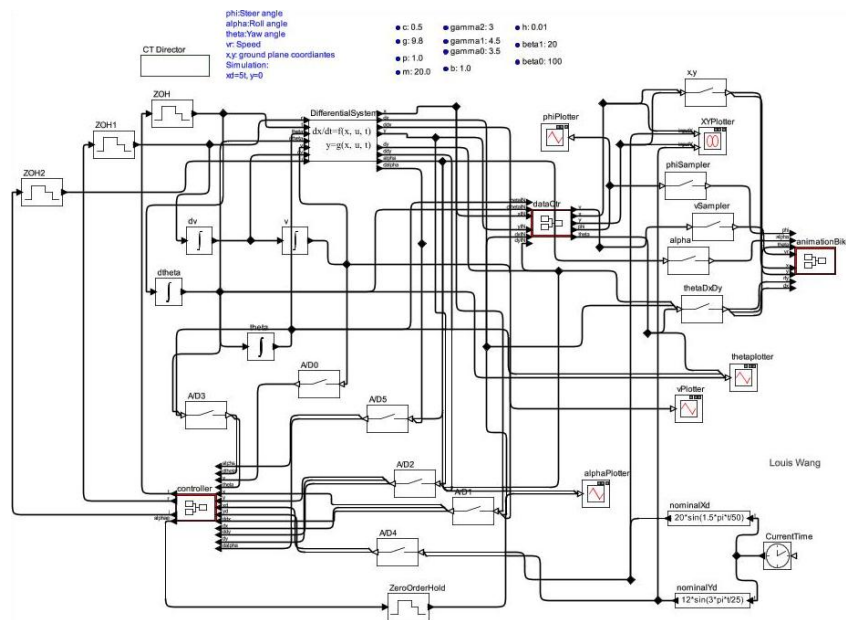


Figure 3.6: PtolemyII model of Path tracking while maintaining balance control [49]. The path tracking control tracks the desired ground path trajectories meanwhile maintains balance of the bicycle model. 1) The *CT director* defines the continuous domain of the model. 2) The *Differential System* actor contains the bicycle dynamic state space equations. 3) The controller equations are included in the *controller* actor. 4) The *animationBike1* actor is the 3D model of the bicycle. See also Appendix C for the full scaled image.

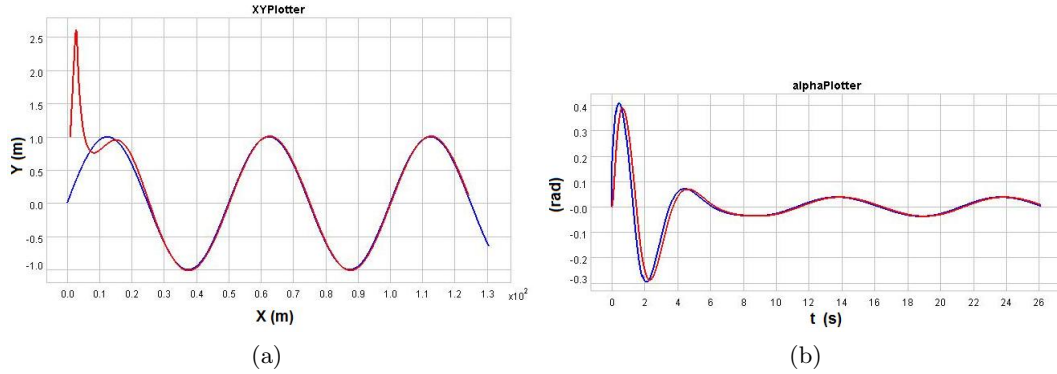


Figure 3.7: Tracking result of a sinusoidal ground path. (a) Result of ground trajectory tracking; (b) α plot of tracking internal equilibrium roll angle.

$$x_d(t) = 5 \cdot t, \quad y_d(t) = \sin\left(\frac{\pi \cdot t}{5}\right)$$

In Fig. 3.7a the blue line is the reference ground path, the red line is the actual trajectory of the bicycle. The blue line in Fig. 3.7b is the calculated reference equilibrium roll angle. The simulation shows that the bicycle immediately tracks the internal equilibria, meanwhile the path tracking controller makes the bicycle converge to the reference path.

The next simulation is the result of tracking a circular ground path, Fig. 3.8. The circular path equation is:

$$x = 8 \cdot \sin(t/5), \quad y = 8 \cdot \cos(t/5)$$

The internal equilibrium roll angle calculation results in tracking of a circular path, shown in Fig. 3.8b.

The wave forms of the internal equilibrium roll angle from both simulations appears cycling around zero roll angle. This can be interpreted as whenever the bicycle crosses the reference path the controller will try to turn it back. Thus, the bicycle tilts back and forth around the upright position.

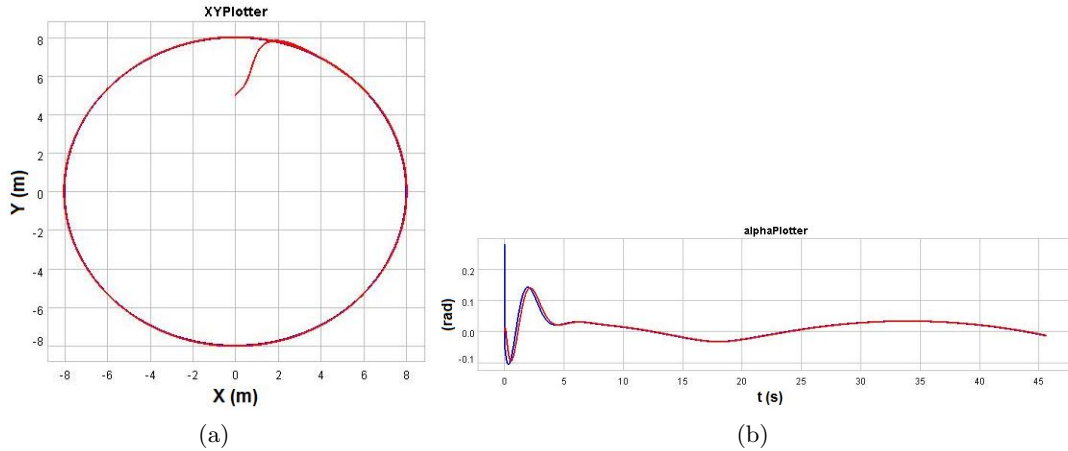


Figure 3.8: Tracking result of a circular ground path. (a) Result of ground trajectory tracking; (b) α plot of tracking internal equilibrium roll angle.

3.2 Control schemes discussion of the platform

In the software bicycle control simulation, the nonlinear control strategies are developed based on the nonlinear models of bicycle dynamics. Modeling accuracy, constraint conditions and assumptions are assumed to be ideal under simulation, but can be problematic in real life control. In Chapter 2, I have discussed that Chih-Keng Chen & Thanh-Son Dao [12] used fuzzy control. Initially, a fuzzy logic controller was designed to control the Ghost rider bicycle, but its complexity made it impossible to tune [27]. The capability of the processor is another consideration to implement very complicated calculation which is unfeasible for the specific one used in the project.

Control algorithm development complies to the principle of simplicity and practicality. PID controller is the most commonly used feedback controller. According to [8] “In the absence of knowledge of the underlying process, a PID controller is the best controller”. Code size of PID controller is also small which is critical for the precious processor time.

3.2.1 Simulation of balance control with PID controller

Before applying PID controller on the platform, balance control was simulated with a PD controller. Using the same model of Fig. 3.2 but replace the model based non-linear controller with a PD controller. Fig. 3.9 is the PtolemyII model of the PD controller. The input signals of the controller are: α the roll angle, $\Delta\alpha$ the roll rate, and α_{ref} the reference roll angle; the output is the steering angle. The proportional gain K_{p0} equals to 6.0, the differential gain K_{d0} equals to 2.0.

Three simulations have been conducted in which the speed is always assumed to be constant. Fig. 3.10 is the simulation of the upright position balance control; the bicycle runs along a straight ground path at zero roll angle and zero steering angle; $\alpha_{ref} = 0$ in this case. In Fig. 3.11, the bicycle is balanced at a fixed equilibrium roll angle, $\alpha_{ref} = \pi/8$; and it draws a circular ground path. Fig. 3.12 is the results that the actual roll angle tracks a sinusoidal reference roll angle, here $\alpha_{ref} = \pi/8 * \sin(2 * \pi * t/5)$; the steering action also appears a sine wave form and lead to a sinusoidal ground path. The simulation results indicates that the PD controller also successfully balanced the bicycle.

3.2.2 Comparison of PD controller and non-linear controller

In order to check the effectiveness of the PD controller, the results of the step response of the bicycle model with PD controller and the non-linear controller are compared, shown in Fig. 3.13. The initial conditions in either case are set exactly the same: $\alpha_0 = -\pi/6$, $v_0 = 4m/s$, given $\alpha_{ref} = 0$, sampling period $10ms$.

Comparing the roll angle transient processes in Fig. 3.13 (a) and (b) it can be seen that the overshoot with PD controller is lower and transient time is shorter than that with the non-linear controller, but the damping ratio is less than that with the non-linear controller. The roll angle oscillation frequency with PD controller is

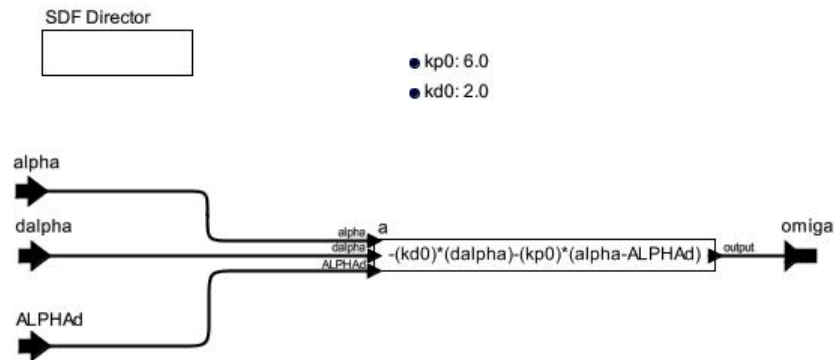


Figure 3.9: PD controller PtolemyII model. Here, α is the actual roll angle, $d\alpha$ is the roll rate, $ALPHAd$ is the desired roll angle, $Kp0$ and $Kd0$ are controller gains, ω is the steering output, the SDF director defines the discrete domain of the controller.

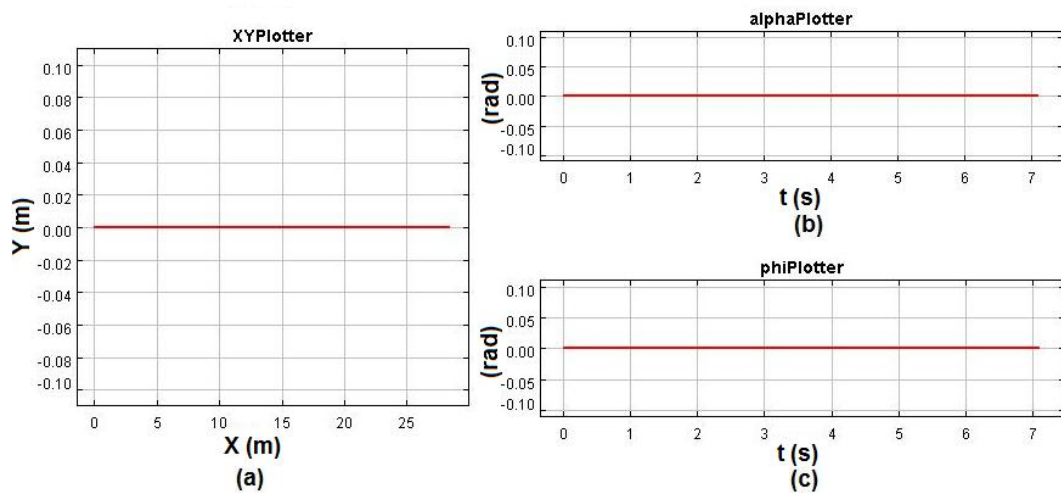


Figure 3.10: Balance control of upright position with PD controller. (a) Ground path. (b) Roll angle plot. (c) Steering angle plot.

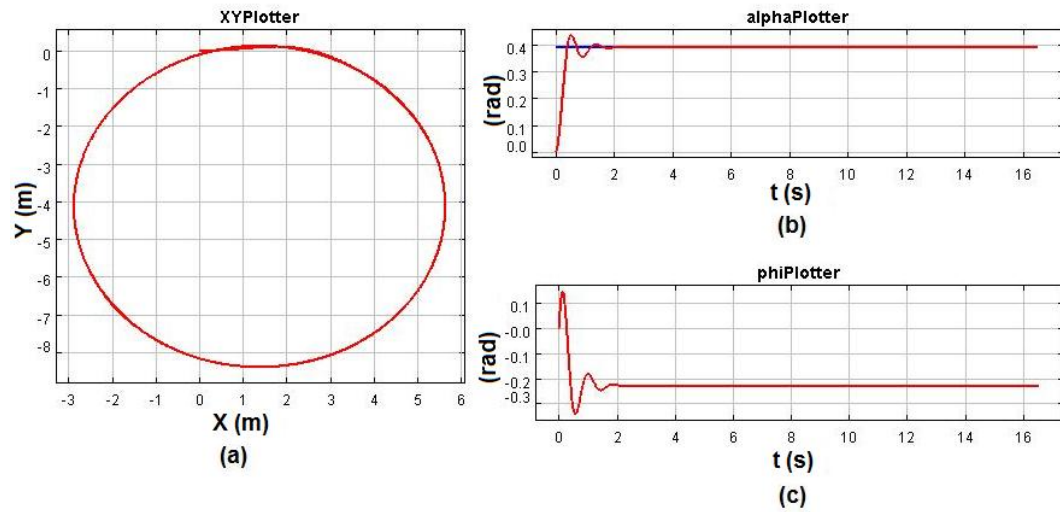


Figure 3.11: Balance control of a fix reference roll angle with PD controller. (a) Ground path. (b) Roll angle plot. (c) Steering angle plot.

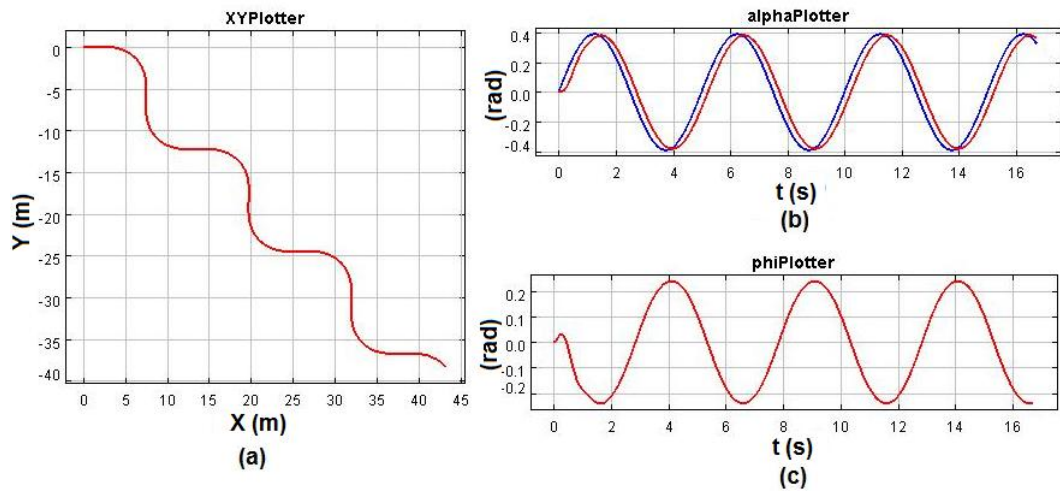


Figure 3.12: Balance control of sinusoidal reference roll angle with PD controller. (a) Ground path. (b) Roll angle plot. (c) Steering angle plot.

higher, this can be explained from the steering output plots, shown in Fig. 3.13 (c) and (d); with PD controller the steering amplitude and frequency are all higher than that with the non-linear controller.

The plot shows that under this certain condition there is no qualitative difference between PD controller and the non-linear controller. The PD controller results in a faster transient process, i.e., the settling time is about 2.5s as shown in the plot, whereas it is about 4s with the non-linear controller; the non-linear controller leads to a smoother motion. I acknowledge that PD controller is not always comparable to non-linear controller under all circumstances. In this approach the PD controller is the better choice according to the feasibility.

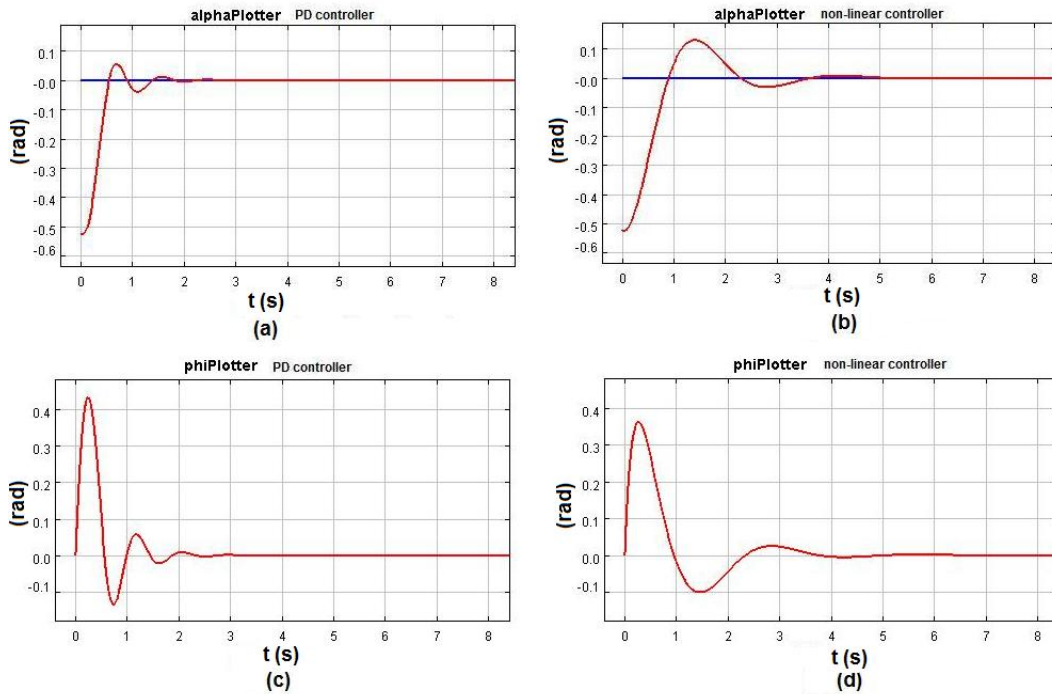


Figure 3.13: Step response results comparison: (a) Roll angle plot with PD controller; (b) Roll angle plot with non-linear controller; (c) Steering angle plot with PD controller; (d) Steering angle plot with non-linear controller.

3.3 Controller design of the platform

A PID algorithm is used to develop all the control schemes in the hardware implementation of the project. The states of a moving bicycle can be described as at upright position for straight-running motion, at an equilibrium roll angle for a cornering motion; or furthermore, to follow desired direction. A number of different control schemes are compared; finally, two of them are appointed to do the road test. In this section, these issues are discussed along with the corresponding control schemes.

3.3.1 Balance control

The balance-only controller is to keep the bicycle at the upright position or hold the bicycle at a non-zero desired roll angle depending on the reference values. At this point, although the bicycle can be controlled to change direction by adjusting the reference roll angle, the vehicle itself does not really know its “actual direction of travel” .

Straight-running motion at upright position

To balance a bicycle with a straight-running motion a simple scheme is developed using PD controller. The block diagram is shown in Fig. 3.14. The consideration of using PD without integration is that the integration term could be subjected to “windup” which could contribute to control sluggishness and instability. Workability was given the highest priority since in this very first approach the question of whether the platform could be physically balanced or not was not known at the time. The final controller equation being tested is:

$$\tau(t) = K_{p0} \cdot \alpha(t) + K_{d0} \cdot \frac{d}{dt}\alpha(t) + \tau_0 \quad (3.5)$$

Here, $\tau(t)$ is the steering output, τ_0 is the offset of steering neutral output, $\alpha(t)$ is the roll angle, $\frac{d}{dt}\alpha(t)$ is roll rate, K_{p0} is the proportional gain, and K_{d0} is the differential gain. This controller will try to maintain the upright position of the platform by generating steering output to the direction of fall. The steering angle is assumed to be exactly tracking the steering actuator output, i.e. the steering angle is proportional to the steering servo PWM signal range; because of the simple set up of the steering mechanism it is difficult to put any feedback. The speed control is not included in this control algorithm. As I have discussed in the previous chapter a stationary bicycle can not be balanced, the speed is controlled manually through the RC throttle channel and always above the *critical velocity*. The scale of centrifugal force to balance a bicycle is also related to velocity (Centrifugal force: $F_c = m \cdot v^2 / r$), so at higher speed less steering angle is required and vice versa. Since there is no measurement of speed yet, the steering output is the natural response of the controller to balance the bicycle according to the current running condition without consideration of the actual speed value.

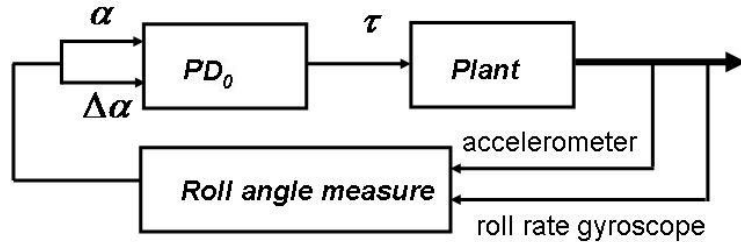


Figure 3.14: Balance control scheme of straight running motion.

Turning motion at non-zero equilibrium roll angle

When an additional non-zero reference angle is added to the input of the upright balance controller, as shown in Fig. 3.15, the bicycle will tilt to one side depending on the sign of the reference. It will finally be held at an equilibrium state by a non-

zero steering angle corresponding to the specific speed, thus leading to a turning motion. Different values and signs of the reference roll angle will cause different steering angles and directions. The equation of the controller are given by:

$$\tau(t) = K_{p0} \cdot e(t) + K_{d0} \cdot \frac{d}{dt}e(t) + \tau_0 \quad (3.6)$$

$$e(t) = \alpha_{ref}(t) - \alpha(t) \quad (3.7)$$

The equilibrium roll angle controller is almost the same as the upright balance controller except the input variables are the roll angle error instead of the roll angle. $e(t)$ is the error of the actual roll angle and the reference roll angle. Changing reference roll angle is achieved through the RC controller roll/aileron channel. Using this approach the direction of the bicycle can be changed manually through the RC controller and the bicycle will maintain balance autonomously.

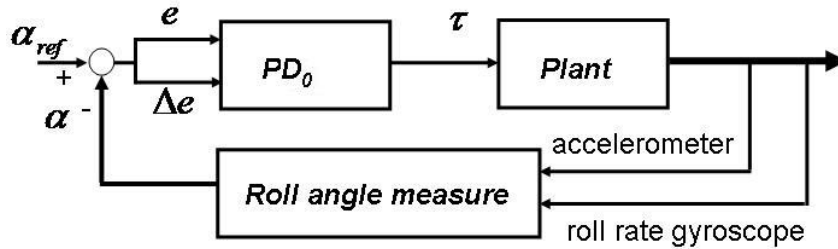


Figure 3.15: Equilibrium roll angle control scheme.

3.3.2 Directional control with balance

In Chapter 2, I have reviewed the directional control and path following; and simulated in the earlier sections of this chapter. On the hardware approach, the path tracking is not practical only directional control is implemented. Provided a bicycle will never fall over, it is called a *kinematic bicycle* [17]. The steering angle is simply

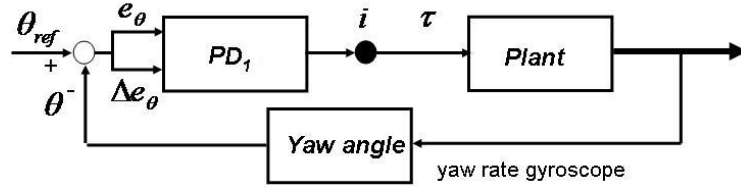


Figure 3.16: Kinematic bicycle directional control.

adjusted according to the changes of the desired direction. The controller of the *kinematic bicycle* can as well be called a *yaw angle controller*, as in Fig. 3.16.

To also maintain balance I can break the link at point ‘ i ’ in Fig. 3.16 and plug in an inner balance control loop, as shown in Fig. 3.17. In other words, it can be said to be adding an outer yaw angle control loop to the balance controller. The resulting controller consists of two cascaded PID controllers. The directional control outer loop equation is:

$$\alpha_{ref}(t) = K_{p1} \cdot e_\theta(t) + K_{d1} \cdot \frac{d}{dt}e_\theta(t) \quad (3.8)$$

$$e_\theta(t) = \theta_{ref}(t) - \theta(t) \quad (3.9)$$

Here, $\theta_{ref}(t)$ is the desired yaw angle, $\theta(t)$ is the actual yaw angle, K_{p1} is the proportional gain of the yaw angle controller and K_{d1} is the differential gain. The output of this controller is the reference roll angle $\alpha_{ref}(t)$. The new roll angle error equation Eq. (3.10) is then obtained by substitute Eq. (3.8) into Eq. (3.7).

$$e(t) = K_{p1} \cdot e_\theta(t) + K_{d1} \cdot \frac{d}{dt}e_\theta(t) - \alpha(t) \quad (3.10)$$

Equation (3.6) together with equation (3.10) form the final controller equation.

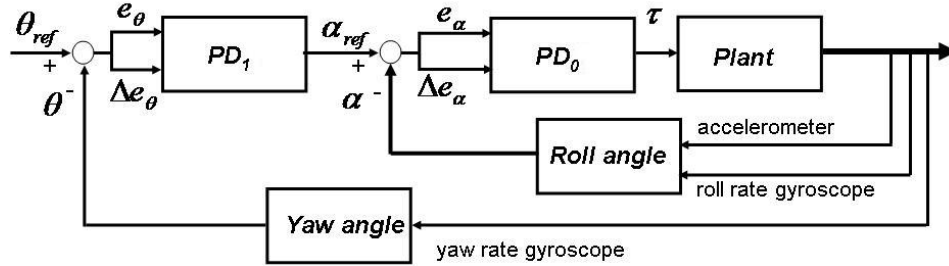


Figure 3.17: Directional control with balance scheme.

3.3.3 Summary of the control schemes

Although the upright balance control and the equilibrium roll angle control are described in two separated diagrams they can be considered as one balance-only controller. The upright balance control is the special case of the equilibrium roll angle control with the reference roll angle always equal to zero, certainly the upright position can also be an equilibrium point. The directional control is based on the balance control. One of the possible problems with the cascaded system is that the two loops may interfere with each other. For quick directional change a larger steering angle is needed, the larger steering angle is generated by the larger roll angle error which requires relative drastic changes of the reference roll angle, thus balance control will be challenged. On the other hand, if the steering action is too slow the direction control will be less effective. A trade-off point between the effectiveness of the directional control and the balance control must be found through tuning the controller which will be discussed in chapter 4 and 5.

Chapter 4

Implementation

To implement the hardware system a number of issues regarding angle measurement, actuation, RC control, data acquisition and so on needed to be solved. Implementation of the control algorithms on the hardware platform is the most important task of the thesis research. The success of this project will lay the foundation of *higher level control* research of a fully autonomous bicycle.

4.1 Platform description

The platform is modified from a commercial electric scooter by a preceding research group of UOIT students. The top structure of the scooter, such as steering bar, seat, cover, etc., was removed to adopt the control devices; only some key parts necessary for autonomous modification are reserved such as wheels, drive motor and so on. Ian Spencer [44] designed the power, actuation and sensory systems. Aravinth Sivarajasingam & Thivakaran Thevarayan [42] implemented partial RC control of steering, motor and braking, etc. through a microprocessor. The previous work laid the first milestone of the project.

The main task of the second step, i.e., the current research, was the control system research. Many upgrades have been done before and during the research

in order to implement the control strategies and solve some of the issues presented by the previous research. Mechanical upgrades include strengthening the structure, new mounting racks to adopt more electronic parts, and so on. The electrical innovations account for the majority of the upgrades including roll angle measuring sensor system, yaw angle measuring sensor, photoelectrical isolation, power supply system, wireless communication, motor encoder, to be specific in here. The circuit schematic diagram is shown in Appendix A.

The center of the system is a Freescale [1] HCS12 microprocessor [5], with 16 bits, 4MHz main frequency, integrated multiple I/O, A/D, timer and PWM [7] channels, making it very suitable for small multitasking project. The main limitation of this microprocessor is its computational ability. The current version of the overall platform is shown in Fig. 4.1. More details will be provided later in this chapter.

4.2 System architecture

A deployment UML diagram of the system architecture with all the components is shown in Fig. 4.2.

The system is powered by three battery groups with three isolated grounds as shown in the circuit schematic diagram.

- DC 24v, for motor power supply; two 5 AH, 12v lead acid batteries are connected in series to provide power to the main motor; one of the two 12v power is also regulated to 5v as the power supply to the collector of the transistor side of the specific optoisolator.
- DC 12v, for computer power supply; 12v 3AH lead acid battery, directly connected to HCS12; also regulated to 5v as the power supply to RC receiver, accelerometer and yaw rate gyroscope; the 5v voltage is then regulated down to 3.3v to supply power to roll/pitch rate gyroscope.

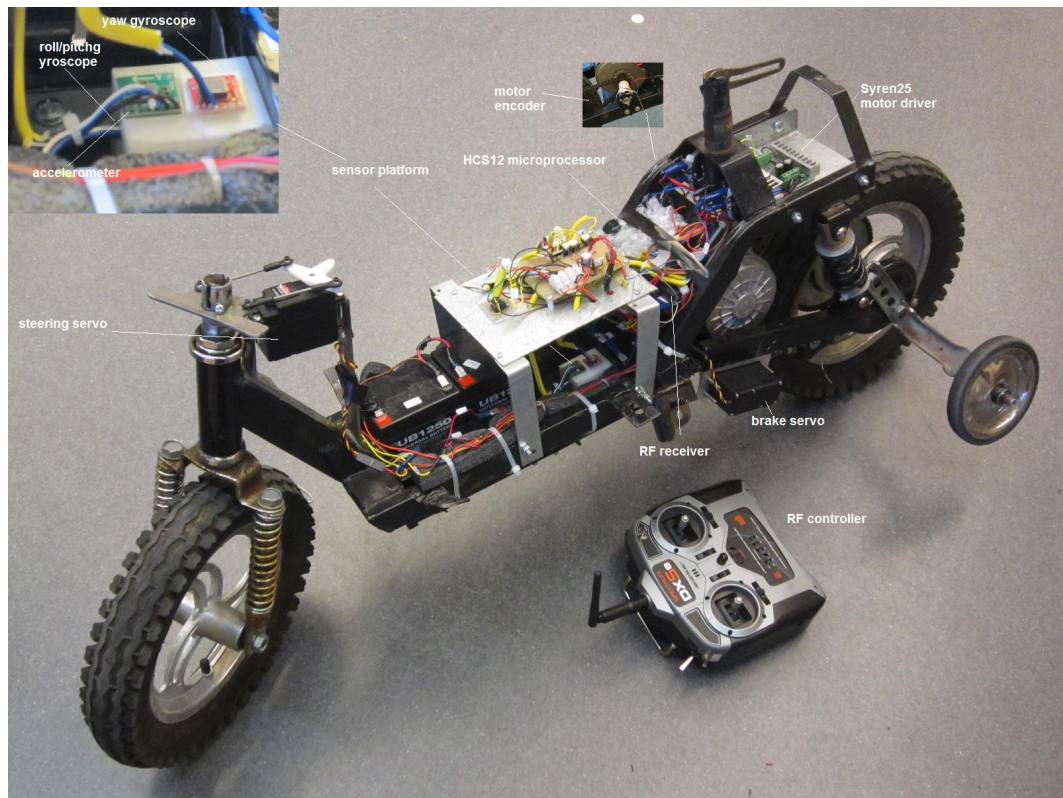


Figure 4.1: The current version of the platform.

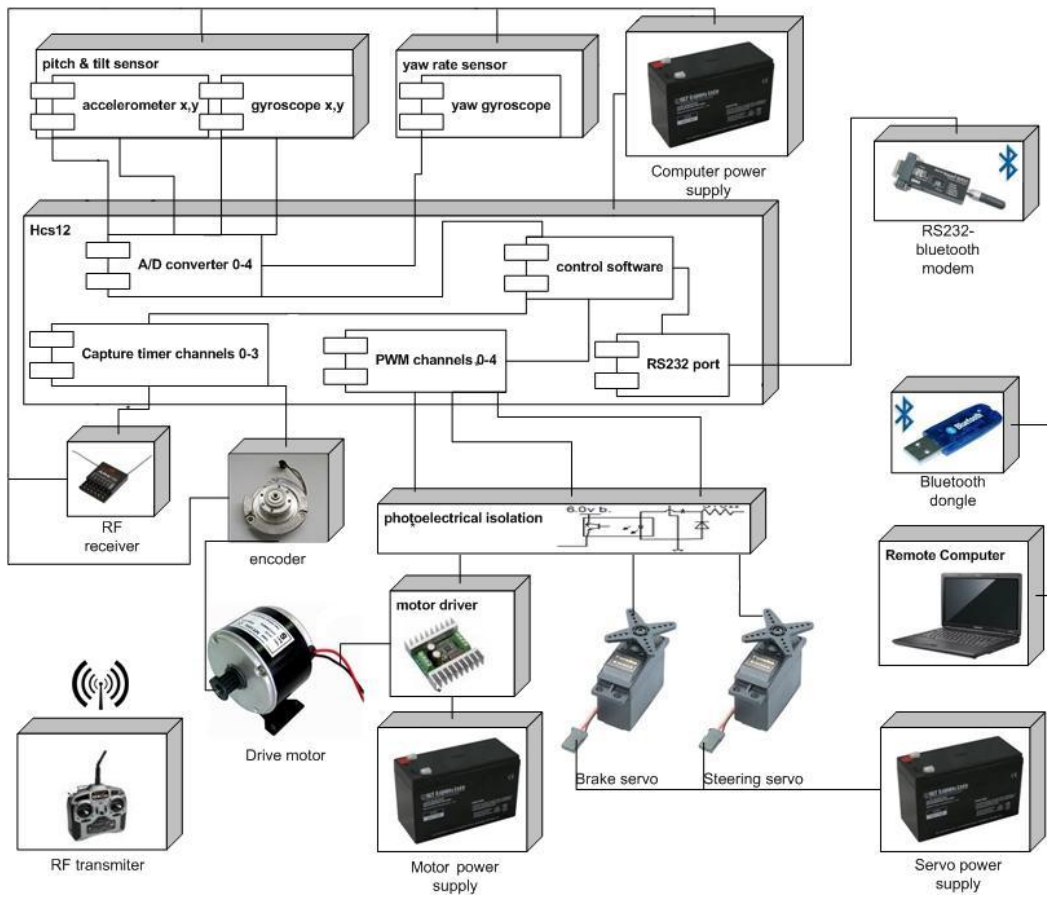


Figure 4.2: Deployment diagram of the system architecture with all the components.

- DC 6v, for servo motor power supply, 6v 3AH lead acid battery, to power steering and brake servo, and serve as the collector voltage to the optoisolators.

The tilt sensors and yaw sensor are connected to the A/D channels of HCS12 providing the roll and directional information to the control software. The balance and directional control software then control the actuators through the PWM output channels. All the actuators are controlled with a $44.5Hz$ frequency, $1100\mu s \sim 1900\mu s$ duty PWM signal. The PWM output channels of the microprocessor are isolated from the actuators through photoelectric isolators to avoid electromagnetic noise to the MPU; the motor driving circuit and steering control circuit also have their own separated ground to eliminate the disturbance between each other (during test it was found when the motor reached high speed it would affect the steering servo).

The RC receiver PWM output signals are connected to the capture timer ports of the microprocessor as the control commands. There are two control modes of the system, RC/manual control mode and autonomous control mode. The modes switch is done through the fifth channel on the DX5e remote controller. The channel outputs two fixed PWM duty value $1100\mu s$ or $1900\mu s$ instead of continuous PWM duty controlled by the lever position on a regular channel. The manual mode is useful for testing the actuators, though it is almost impossible to control balance manually.

The serial communication using RS232-Bluetooth modem broadcasts the vehicle runtime states wirelessly, a computer equipped with a Bluetooth dongle collects the data in order to conduct off-line analysis.

4.3 Software structure

The system software flow chart comes with two main branches, shown in Fig. 4.3. The left branch is the flow of the RC/manual control mode. In this mode the

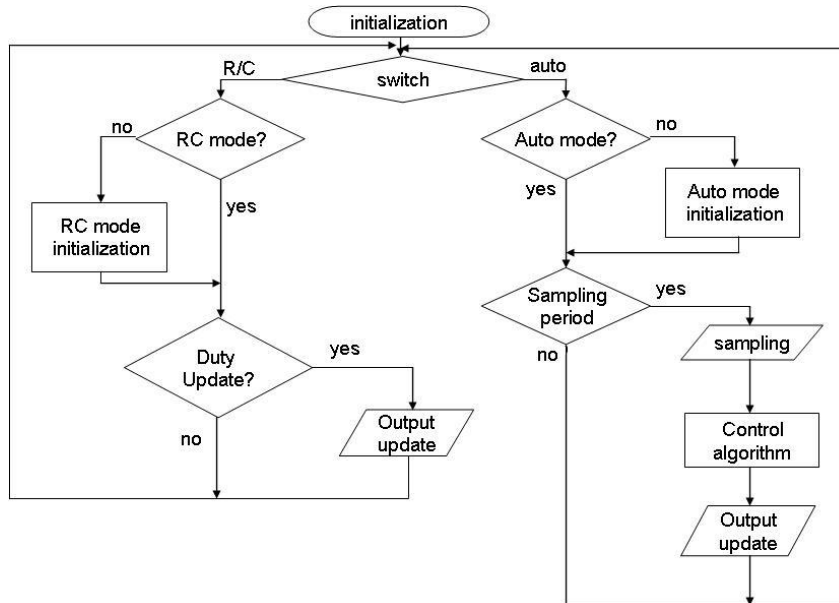


Figure 4.3: System software flow chart.

processor has two major tasks; the first one is capturing the input PWM pulse edges from the RC controller and calculating the duty values; the second one is passing the calculated PWM duty values to the actuators through PWM channel on the processor. There are three actuators controlled by the RC controller: steering actuator, brake actuator and motor speed controller. When the mode switch on the RC controller is moved to the “auto” position the close-looped controller takes over steering control from the RC signal. The control algorithms have been introduced in details in Chapter 3. The motor and brake control are manually controlled and use only the RC controller in either mode.

All the control tasks are precisely timed using timer interrupt. The sampling period of the A/D conversion is 10ms for accelerometer, roll/pitch rate gyroscope and yaw rate gyroscope. The sampled data is then subjected to digital filtering. The sampling period is determined by using the oscilloscope to plot the timing of the

periodical tasks; it is found that 6ms is the best time lower than that the tasks will be delayed. Choosing 10ms is to guarantee enough time for the existing tasks and adapt some additional tasks in case. The control period is not only decided by the system response time but also by the PWM period and the processor clock. PWM signal period of all the actuators is defined by the hardware and is about 22.5ms; this limits the control period can not be faster than this time since the PWM channels must complete the current cycle to ensure the correctness of the output waveform before changing to the new duty value; so the control output period is set to 25ms more typically. The serial communication task is fitted into the same period right after updating the control outputs. One data packet of the real-time states is broadcasted per 25ms. These states will be received on-line and analyzed off-line.

RC signal capturing, A/D converting, PWM signal outputs and serial data broadcasting are performed by hardware functions of the processor. A corresponding hardware interrupt event will be triggered after one task is completed. Comparing to the scanning method which need the processor to check the states of the tasks periodically, the processor load is greatly reduced in this way.

4.4 RC controller and Pulse Width Measurement

The RC controller, such as in a model airplane application, can be used to control the servos directly to fly the plane. But in this system it is used as a commanding device to the microprocessor; the PWM signals must be detected and processed before passing to the corresponding actuators or control tasks. The details of the RC controller and the PWM signal processing method are introduced here.

RC controller The RC controller is the SPEKTRUM DX5e 5-Channel [43] Full Range DSM 2TM 2.4GHz Radio System. The photo of the system is shown in Fig. 4.4. The RC channels are named after the airplane control dialect. The five

channels are throttle channel, aileron channel, elevator channel, rudder channel, gear channel. Gear channel is a special channel, it only outputs two fixed PWM duty values controlled by a double-throw switch. This channel is used as control mode switch. The other channels can output continuous PWM duty values from $1100\mu s \sim 1900\mu s$.

When the gear channel switch is pushed down the channel outputs $1100\mu s$ duty cycle PWM signal and platform is operated at manual mode. Signal from the aileron channel is used to control the steering action. $1100\mu s$ corresponds to -45° maximum left steering angle, $1900\mu s$ corresponds to $+45^\circ$ maximum right steering angle. Signal from the throttle channel is used to control both motor speed and brake servo. The purpose of this configuration is to avoid conflict between motor and brake, i.e., only one can be activated between motor and brake at one time instant. When the lever is held in its center position the throttle channel outputs $1500\mu s$ duty pulse width, and both motor and brake control PWM channels of the microprocessor output $1500\mu s$, and so the actuators stay in neutral; if the lever is pushed forward the microprocessor transfers the duty value to the motor control PWM01 channel driving the motor to rotate and brake control PWM4 output remains $1500\mu s$; if the lever is pulled back the microprocessor transfers duty to the brake control PWM4 to actuate braking and motor control PWM01 output $1500\mu s$ to deactivate the motor.

When the gear channel switch is pulled back the channel outputs $1900\mu s$ duty cycle PWM signal and the platform is operated at autonomous mode. RC control of steering is only effective in RC/manual mode, in autonomous mode the balance controller takes over the steering PWM control. The aileron channel command functions as reference tilt angle input. For now, motor speed and brake are only controlled by the RC controller.



Figure 4.4: SPEKTRUM DX5e transmitter and AR500 receiver.

Pulse Width Measurement The RC controller is designated to produce PWM signals to control servo motors which take PWM input. In this system the PWM control signals are generated by the microprocessor. The RC signals are detected by the microprocessor and then passed to the actuators through the microprocessor PWM channels. This requires that the RC output signals are read by the MCU. A hardware low pass filter is a simple practice by transferring the pulse signal to an analog voltage and the value of the voltage is related to the PWM duty value. The disadvantage of this method is the huge lag caused by the low pass filter. A good real-time solution is Pulse Width Measurement, the maximum lag is only one PWM period (22.5ms). Pulse Width Measurement is implemented by capturing PWM both rising and falling edges, the time interval in between one rising edge and the next falling edge is the duty cycle, the time interval in between two rising or falling edges is the period.

Each PWM output pin of the AR500, the remote receiver, is connected directly

to one capture timer port of the microprocessor. The timer capturing mechanism is described below:

- The HCS12 has eight 16 bits enhanced capture timer channels. One 16 bits free run counter is shared by the 8 channels.
- The counter clock is equal to the bus clock frequency divided by the prescaler, $1\mu s$ in our project.
- The timer is set up to capture both rising and falling edge mode.
- The edge will cause an interrupt of the corresponding channel; thus the edge is captured.
- Use the current counter value to subtract the last counter value to calculate the time interval.

A PWM signal wave form is shown in Fig. 4.5.

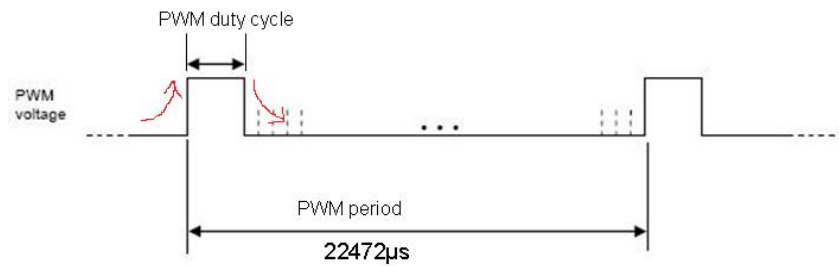


Figure 4.5: PWM signal.

The processor does not distinguish whether the interrupt is caused by the rising or falling edge. When the calculated time interval is longer than the maximum duty cycle used in this system it is rejected, this is the interval of the falling edge to the next rising edge. The PWM period is constant in our system so only duty cycle is calculated using the follow algorithm:

```
old_counter_value=current_counter_value
    // get the last counter value
current_counter_value=timer_channel-register
    // get the current counter value
if current_counter_value > old_counter_value
    duration=current_counter_value - old_counter_value
    // calculate the time interval
if duration > 5000us
    // the time interval too big is not the duty cycle.
    new_duty_cycle=old_duty_cycle
if duration<5000us
    new_duty_cycle=duration
if current_counter_value < old_counter_value
    //counter overflowed ,
    //it starts over in between two capture events
duration=0xFFFF-old_counter_value+current_counter_value+1
    // calculate the time interval
if duration > 5000us
    // the time interval too big is not the duty cycle.
    ignore duration
if duration<5000us
    new_duty_cycle=duration
```

4.5 Actuator Control

To control a *Basic Bicycle* [18], three actuators are required to control steering, speed and brake. All these actuators take PWM signals with frequency $44.5Hz$ and

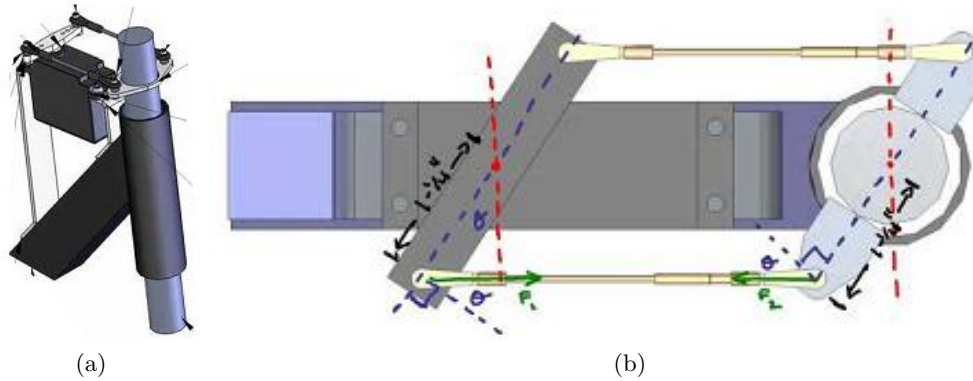


Figure 4.6: (a) Steering actuator design; (b) Steering mechanism top view [44].

duty cycle $1100\mu s \sim 1900\mu s$ as inputs. So they can be controlled with a single RC controller or microprocessor conveniently.

Steering actuator The steering actuator assembly, shown in Fig. 4.6 [44], is consisted of HS-805++BB Servo motor [19], mounting brackets, threaded rods with ball joints, and steering bracket. The maximum stall torque of the servo is 21 lb·in. Fig. 4.6(b) shows the steering is rotated at an angle θ , thus the torque exerted to the steering shaft is:

$$\tau = 21 \times \cos^2 \theta \text{ [44]}$$

At 45° the actual torque to the steering is reduced by 50%. This mechanism provides minimum 10.5 lb·in stall torque and maximum $\pm 45^\circ$ steering angle.

Brake actuator The same HS-805++BB servo motor as in the steering actuator is used as the brake actuator [44]. An one inch long arm is attached to the axis of the servo, and the other end of the arm is jointed to the scooter's rear wheel braking drum with a steel brake cable, Fig. 4.7. A nearly one inch long stroke of the cable which corresponds to a 28.5° rotating angle is required to actuate the brake. Since the servo is a rotary device the braking force will not be a constant at different



Figure 4.7: Brake actuator.

angle. The highest torque happens when the servo arm is normal to the steering cable; the arm and cable assembly is adjusted so that when the servo motor is at neutral position the servo arm is perpendicular to the braking cable. If the servo stalls at angle θ the maximum force that the servo can exert to the brake is:

$$F_{brk} = 21 \times \cos \theta \text{ [44]}$$

At 28.5° , the physical braking stroke, the servo will stall; the braking force is approximately 18.5 lbs which satisfies the minimum 12 lb force requirement [44] to actuate the brake drum. The braking force is adjustable, the less rotating angle of the servo the less braking force.

Motor control actuator The motor controller is a Syren25 [45], regenerative motor driver from Dimensional Engineering. Syren25 is a very flexible and powerful DC motor driver using $6v \sim 24v$ input DC voltage and $25A$ output DC current making it suitable to power a wide range of DC motors. Its regenerative circuit will return power back to source during deceleration, this is very important for battery powers. Syren25 can take analog, PWM or serial input as control signal. PWM is used in our system, $22500\mu s$ period, $1500\mu s$ neutral, maximum reversed rotating

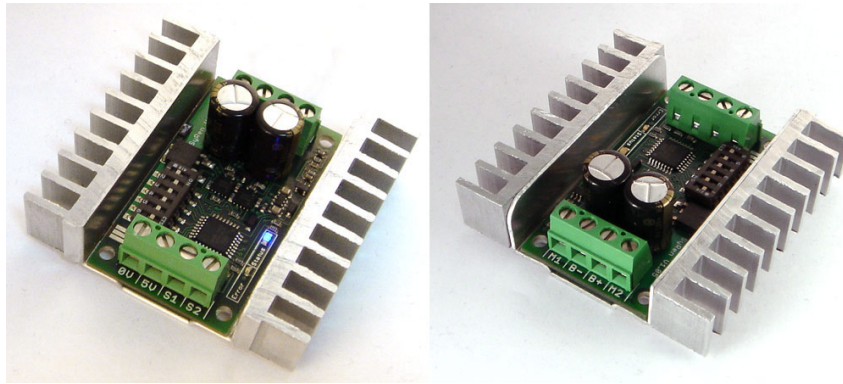


Figure 4.8: DemensionalEngineering Syren25 motor controller.

speed at $1100\mu s$ duty cycle, and maximum forward rotating speed at $1900\mu s$ duty cycle. Since our platform is a bicycle type vehicle only forward rotation is required, i.e. duty cycle range from $1500\mu s$ (neutral) to $1900\mu s$ (fully forward speed).

Control of actuators Each independent PWM channel has as an 8 bit duty register, one 8 bit counter, and one 8 bit period register [5]. The hardware function of PWM output mechanism is as follow:

1. Write the desired period into the period register.
2. Write the desired duty cycle into the duty register.
3. If the duty level is set to high, then the PWM port outputs high level.
4. The counter starts from 0 to 255, increases by one for every PWM clock tick.
5. When the counter number matches the duty cycle in the duty register PWM output flip-flop to low level.
6. When the counter number matches the period register value, the counter reset to zero and output flips to high. The counter will also start over if it is full.

The processor bus frequency is $1MHz$. The PWM clock frequency is equal to the bus frequency divided by the prescaler value. The prescaler value must be the multiple of 2. So the prescaler value can be 1, 1/2, 1/4, ... 1/128 and so on. Considering the PWM period is $22.5ms/44.5Hz$ frequency, to ensure the maximum resolution the prescaler value should be 1/128, such that each bit of the counter represents $128\mu s$; counter overflow time is $128\mu s \times 255 = 32.64ms$. The calculated duty cycle for $1100\mu s$ is 8 or 9, $1900\mu s$ is 15 or 16. The rotating range of the steering actuator is from -45° to $+45^\circ$ corresponding to PWM duty cycle from $1100\mu s$ to $1900\mu s$. If use one independent PWM channel, the steering action from -45° to $+45^\circ$ is divided into only 7 or 8 steps ($1100\mu s \sim 1900\mu s$ is divided into 7 or 8 steps). The servo behaves jerking.

The PWM can be set up to use 16 bits by concatenating two 8 bits PWM into one 16 bits PWM, i.e., PWM0/1, PWM2/3, PWM4/5. The maximum 16 bits counter number is 65535; prescaler value 1, i.e. using bus clock straight as PWM clock; one increment of the counter represents $1\mu s$. The deadband of HS-805BB+ is $8\mu s$, -45° to $+45^\circ$ is divided into 100 steps equal to resolution 0.9° per step.

In this project PWM 2 and PWM 3 channels are concatenated into one PWM 2/3 channel using PWM 3 output pin to control steering servo, PWM 0 and PWM 1 channels are concatenated into one PWM 0/1 channel using PWM 1 output pin to control the Syren25, the motor controller. Brake servo is connected to an independent PWM channel, PWM 4, since there is no resolution requirement for brake control. Both motor and brake servo rotate only one direction so only use $1500\mu s \sim 1900\mu s$ duty cycle range for motor and $1100\mu s \sim 1500\mu s$ for brake.

4.6 Sensor system

The main focus of the research is testing the basic balancing and directional control of the platform. To serve this purpose three sensory units are integrated into the system. The tilt sensor measures the dynamic roll angle for balancing control; the yaw gyroscope sensor provides directional information; motor encoder measures motor rotating speed related to the linear velocity of the platform.

4.6.1 Roll angle measurement

The tilt sensor is implemented with an accelerometer and a gyroscope. Either the accelerometer or gyroscope has its own limit to measure a dynamic roll angle. IMU (Inertial Measurement Unit) [20] is an application of using combination of gyroscopes and accelerometers to measure the orientation, gravitational force, etc. of the crafts. The roll angle sensor consists of an accelerometer ADX203 [34], and a gyroscope IDG300 [21]. ADX203 is a dual axes, X and Y, accelerometer with full range $\pm 1.7g$; IDG 300 is a dual axis (roll, pitch) gyroscope, full range $\pm 500^\circ/s$. The two sensors are aligned along the central line of symmetry on the platform frame in such a way that tilting to right side from the upright position will produce a positive roll angle, positive roll rate; and a negative roll angle, negative roll rate if tilt to the left, Fig. 4.9. This configuration can also be used as pitch angle sensor. Informations from both accelerometer and gyroscope are fused together to calculate the roll angle.

The accelerometer measures acceleration. The theory of accelerometer is Newton's second law of motion, i.e. $F = m \cdot a$; it actually measures the change of force per unit mass. When measuring the change of gravity the accelerometer can be used to measure the tilt angle. Fig. 4.10 illustrates how the dual-axis accelerometer works. At the left the accelerometer is positioned upright, i.e. Y axis points upward

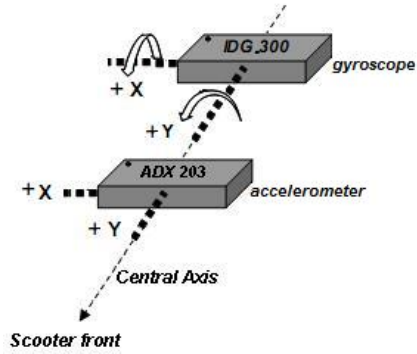


Figure 4.9: Tilt sensor configuration.

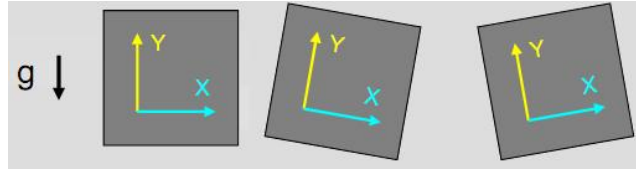


Figure 4.10: Accelerometer measures change of gravity [13].

opposite to gravity direction, X axis points horizontally. Now Y reads $-1g$, X reads $0g$. At the center and right the accelerometer is tilted the X axis reads slightly positive and negative gravity respectively. At such configuration “ $-1g$ ” on X axis represents bicycle laying down on the floor to the left, “ $+1g$ ” represents laying all the way down to the right, “ $0g$ ” represents upright. If the sensor is tilted forward by an angle α , but stationary (not accelerating horizontally).

$$X_{\text{reading}} = 1g \times \sin \alpha.$$

In terms of measuring tilt angle the information of a single axis is sufficient. Only when measuring 360° rotation information from both axes are required to decide the current quadrant. The formula of measuring tilt angle with accelerometer is:

$$V_{OUT} = V_{OFFSET} + \left(\frac{\Delta V}{\Delta g} \times 1.0g \times \sin \alpha \right) \quad [34] \quad (4.1)$$

Where: V_{OUT} = Accelerometer Output in Volts

V_{OFF} = Accelerometer 0g Offset

$\frac{\Delta V}{\Delta g}$ = Sensitivity

1.0g = Earth's Gravity

α = Angle of Tilt

Solving for the angle:

$$\alpha = \arcsin\left(\frac{V_{OUT} - V_{OFFSET}}{\Delta V/\Delta g}\right) \quad (4.2)$$

For tilting, the most important angles to measure are close to vertical. If the bike tilts more than 30° in either direction, there is probably not much the controller can do other than steering full angle to try to correct it. Within $\pm 30^\circ$ using small angle approximation can save the processor time. This is critical for a microprocessor like HCS12, 16bit 4MHz.

$$\sin \alpha \approx \alpha; \quad -30^\circ \leq \alpha \leq +30^\circ. \quad (4.3)$$

The sensitivity of ADX203 accelerometer is $\Delta V/\Delta g = 1000mv/g$ [8], output range 0 – 5v, A/D full range 1023. The tilt angle:

$$\alpha_{acc} \approx (V_{OUT} - V_{OFFSET})/(\Delta V/\Delta g) = (X_{acc} - X_{offset}) \times 5/1023 \quad (4.4)$$

Here X_{acc} is A/D value of accelerometer, X_{offset} is the offset A/D value, i.e the A/D reading with the sensor X axis pointing horizontally, in this case X_{offset} is about 0x200.

ADX203 operational principle is pizeoelectric effect [4]. A pizeoelectric accelerometer, Fig. 4.11, utilizes pizeoelectric effect to measure the changes of the gravitational force on the sensor. If a bicycle is balanced at an equilibrium roll angle the centrifugal force will balance the gravitational force exerted on the accelerometer, the roll

angle reading will be zero. An accelerometer can not be the only means to measure the dynamical roll angle.

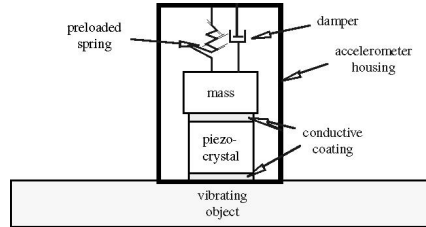


Figure 4.11: Schematic of piezoelectric accelerometer [4].

Gyroscope measures speed of rotation. Fig. 4.12 shows three different states of gyroscope sensor.

The gyroscope is stationary as shown in the left side of Fig. 4.12; the sensor should read zero speed; the A/D reading is the zero angular velocity offset value. In the center and right side the sensor reads positive speed and negative speed respectively. The angular velocity can be directly obtained from the gyroscope reading. The relative tilt angle can be calculated through integrating the angular rate.

$$v_{\text{angularVelocity}} = (V_{\text{gyroOUTPUT}} - V_{\text{gyroOFFSET}}) / \text{gyroSensitivity} \quad (4.5)$$

$$\Delta\alpha = v_{\text{angularVelocity}} \times dt \quad [21] \quad (4.6)$$

Where: dt = sampling period



Figure 4.12: Gyroscope measures speed of rotation [13].

The sensitivity of IDG300 gyroscope sensor is 2 mV/(degree/second) [21], A/D full range 1023, the offset is about 0x127. The actual roll rate is calculated as:

$$\text{rollRate} = (\text{gyro}_{\text{AD}} - \text{gyro}_{\text{OFFSET}}) \times 5 / (1023 \times 0.002) \quad (4.7)$$

As in the previous discussion accelerometer is affected by the centrifugal force and horizontal acceleration. Gyroscope is not affected by horizontal accelerations but can only measure angular rate and relative angle. Accelerometer is required to tell the initial absolute roll angle. The gyro-drift, i.e. when gyroscope is stationary it does not always read perfect constant offset value, the small rate will be added up to the angle through integration until it is far away from the actual angle. The complementary filter [13] is used to fuse these two sensor signals together to calculate the tilt angle. The function of the filter is already described in detail in Chapter 2. The formula of the complementary filter is written in here again:

$$\alpha = (1 - \beta) \times (\alpha_0 + \text{rollRate} \times dt) + \beta \times (\alpha_{\text{acc}}) \quad (4.8)$$

The iteration starts from the stop status of the bicycle. The first absolute tilt angle is read from the accelerometer only, and then it runs both high-pass filter to filter out the long duration gyro-drift and low-pass filter to filter out accelerometer horizontal acceleration.

4.6.2 Yaw angle measurement

The yaw sensor is a Z-axis rate sensing gyroscope ADXRS614 [3], measurement range $\pm 50^\circ/s$. This device operates on the principle of resonance of polysilicon sensing structures. “Two polysilicon sensing structures each contains a dither frame that is electrostatically driven to resonance, producing the necessary velocity element to produce a Coriolis force during angular rate” [3], and then a capacitive sensing structure on each frame monitors the Coriolis force [9]. The sensor outputs positive

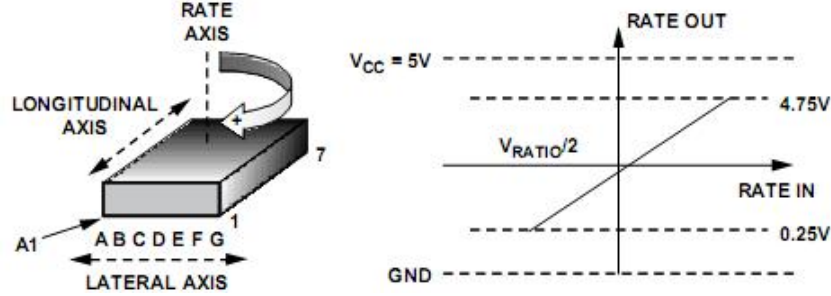


Figure 4.13: Output rate signal with clockwise rotation [3].

rate when rotating clockwise and negative when counterclockwise. Fig. 4.13 shows the relation of output rate and clockwise rotation. The yaw angle is obtained by integrating the yaw rate.

$$v_{\text{yawRate}} = (V_{\text{gyroOUTPUT}} - V_{\text{gyroOFFSET}}) / \text{gyroSensitivity} \quad (4.9)$$

$$\Delta\theta = v_{\text{yawRate}} \times dt[3] \quad (4.10)$$

Where : dt = sampling period

$$\theta = \theta_0 + \Delta\theta$$

The start direction is always assumed to be zero yaw angle despite the actual orientation. θ_0 is the yaw angle of previous iteration and is always set to zero during system initialization.

The parameters of ADXRS614 gyroscope sensor: sensitivity 22.5 mv/(degree/second) [3], A/D full range 0x3ff, the offset is about 0x1ff. The actual yaw rate can be calculated as:

$$\text{YawRate} = (\text{gyroAD} - \text{gyroOFFSET}) \times 5 / (1023 \times 0.0225) \quad (4.11)$$

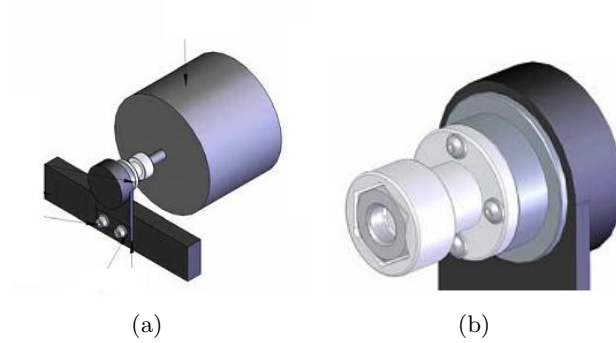


Figure 4.14: Rotary encoder. (a) Motor encoder assembly; (b) Encoder plastic coupling. [44].

4.6.3 Rotary encoder

The rotary encoder is made from a computer hard drive stepper motor [44] Fig. 4.14. A custom made coupling Fig. 4.14b , designed by Ian Spencer is used to connect the encoder and the main motor axis. The coupling was initially made with ABS using fast prototype plastic processing technology and later is machined with nylon6/6 after the first one broke. The stepper motor comes with two signal channels (serve as input signals in motor mode) when driven by mechanical force will output electrical pulses with shifted phases. The frequency of the signal represents the rotating speed and the phase difference can be used to identify the rotating direction. Since only the forward movement of the platform is required only one channel is used. The encoder amplifier circuit is an operational amplifier LM353 configured as a comparator, shown in Fig. 4.15. The amplifier outputs TTL level square wave signal of the same frequency of the stepper motor output.

The gear ratio from the motor driving shaft to the scooter rear wheel driving gear is 11:65, and the rear wheel diameter measures 317.5 mm. The linear velocity of the vehicle can be calculated as:

$$V_{scooter} = (1397/52000)\omega_{motor} \quad [44]$$

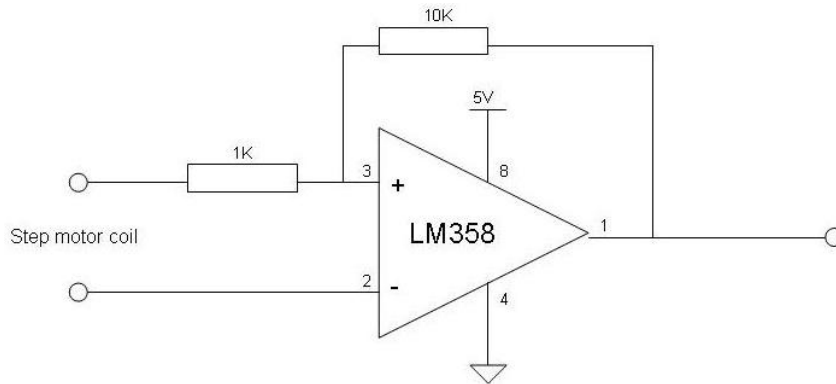


Figure 4.15: Motor encoder amplifier.

The motor angular velocity ω_{motor} is obtained through plotting the output frequency of the encoder and the motor running at several different known speed. The plotting result [44] indicates the linear relationship of the frequency and the angular velocity.

$$\omega_{motor} = 2.3058f - 0.3026$$

4.7 Photoelectric isolation

The magnetic load in the system, such as the motor, servos etc, are the sources of electromagnetic noise. During the system debugging significant noise were observed from oscilloscope. When the whole system is wired into one common ground the noise will be introduced to computer system through the ground line. Fig. 4.16 is the image from the oscilloscope when the accelerometer was at upright position. The window at the left side is the signal without any noise comparing the right side plot with the motor or the servo turned on. The noise level and frequency are also affected by the motor running at different speed. Software filters were implemented, however none of them can effectively restrain the effect of the noise. The high noise level directly caused the sensors reading failure. The roll angle reading fluctuated

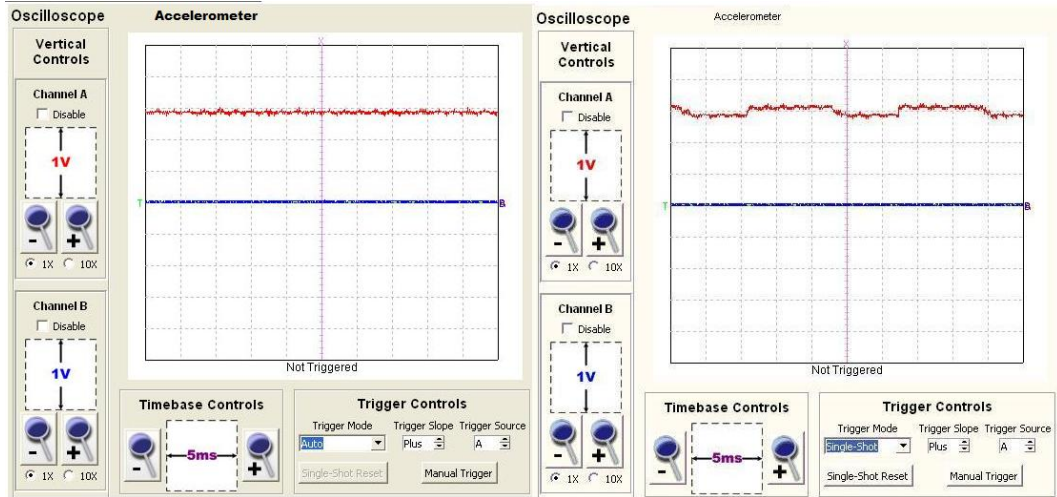


Figure 4.16: Accelerometer signal plot at “0g” position. Left: accelerometer output signal without noise. Right: accelerometer output signal with magnetic noise introduced by the driving motor.

$\pm 15^\circ$ under the noise.

The hardware photoelectric isolation circuit is added to the system. It separates the noise sources electrically and completely removes the noise from the system, the circuit diagram is shown in Fig. 4.17. The element used in the circuit is Fairchild 4N25 [2] optoisolator with $3\mu s$ switching time which ensure the PWM signal (22.5ms period) can be transferred without distortion. The circuit parameters guarantee 4N25 switching correctly and working at very low level of load to increase the reliability. The maximum rating of the input LED current is $80mA$, in our circuit it is less than $10mA$; the maximum current of the output transistor is $100mA$, the working load in our circuit is also less than $10mA$. Thus, the whole system totally comes with three independent ground systems, i.e. GND0 is the ground on the computer side, GND1 is the ground of the servo motors, and GND2 is the ground of the main motor, see Appendix A.

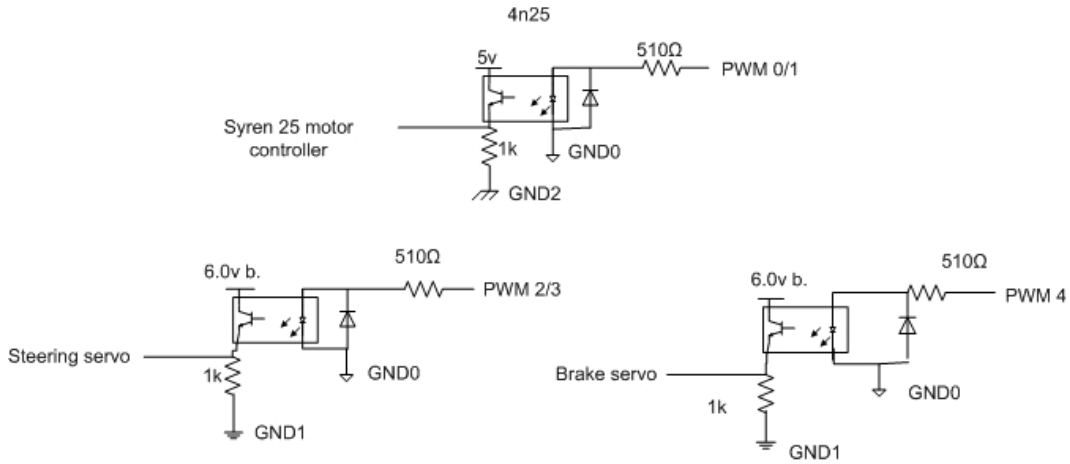


Figure 4.17: Photoelectrical isolation.

4.8 Controller tuning

The controller tuning can only be done through observing the runtime performance. It was not possible to attach any experimental equipment on the vehicle when running. The controller gains were tuned manually through road tests. Generally, PID controller tuning requires to start from K_p at first; and K_i and K_d are set to zero [6]. Increase K_p to the point that the system starts oscillating; and then reduce K_p to the value before oscillating and increase K_d gradually (there is no integration in our case) until the control loop quickly finishes the transient process with acceptable overshoot. Here the balance control loop is considered as the inner loop of the directional controller. The problem becomes tuning up two cascaded control loops. The principle of implementing two cascaded PID controllers is that the inner loop should be the fast loop, and the outer loop should give the inner loop enough time to respond to its output change. Typically adjusting outer loop to ensure that the response time of outer loop is 3-5 times slower than that of the inner loop [46].

4.8.1 Inner loop tuning

The cascaded system tuning should start from inner loop, that is to open the outer loop and manually change the reference of the inner loop [48]. The inner loop is the balance controller taking the roll angle as input, ranging from -20° to $+20^\circ$; and the steering output PWM signal range is $1100\mu s \sim 1900\mu s$, $1500\mu s$ as neutral duty. PWM signal period of all the actuators is $22.5ms$ so the minimum control period should not be less than that. Assuming the fastest rolling rate is $10^\circ/s$, the steering servo can turn about 7° in $22.5ms$ [19]. If using the maximum steering to correct maximum roll angle, the initial K_{p0} was estimated around 30.00. To be conservative K_{p0} was initially set up as 10.00. The critical K_{p0} value of system stability is found to be 27.00. Then K_{d0} was added. The final gain values of the inner loop are $K_{p0} = 25$, $K_{d0} = 1.5$.

Due to the error of the mechanical setup the actual steering neutral is not at the PWM neutral output. This became evident every time after the front end of the vehicle was crashed to the curb and it would no longer be the same as before. After the gains were fixed the steering neutral were adjusted accordingly in order to achieve the same behavior.

4.8.2 Outer loop tuning

When tuning the outer loop the inner loop was put into cascade. The control input of the inner loop, i.e. the reference roll angle, is the control output of the outer loop; the road disturbance is the noise input. The initial gain values of directional control loop are also based on the ratio of input and output parameters. Initially, if using the range of the yaw gyroscope, $\pm 50^\circ/s$ as the input range, maximum allowable roll angle as the output range $\pm 20^\circ$; the grade of magnitude of K_{p1} should be of the unit of one. The initial K_{p1} value was chosen as 2. The final gain values of the outer

loop are $K_{p1} = 2.5$, $K_{d1} = 0.5$.

There are two considerations of control period selection, the first one is to follow the rule that the outer loop should be slower, the second one is the effectiveness of the directional control. If the directional control too fast it will affect balance, too slow the direction will deviate too much. The *trade off* of the sampling period ratio of the two control loops is found about 1:3; the inner loop $25ms$, the outer loop $75ms$. When the outer-loop control period was $50ms$, it appeared saturating the balance control loop too quick, the vehicle leaned to the maximum allowable roll angle to correct the direction. When the outer-loop control period was $100ms$, the vehicle deviated too much from the reference.

4.9 Real-time data acquisition

4.9.1 Bluetooth-RS232 wireless serial communication interface

In order to obtain the dynamic data during system running Bluetooth devices are used to bridge the serial ports connection in between the HCS12 and PC. Parani-SD/100 [32] is a RS232/Bluetooth Class 1 modem with wireless range of 100m. HCS12 is built as a DCE (data communication equipment) [37] device so is Parani-SD/100. The connection must be made in the way called “crossover”. Fig 4.18 shows the minimum connection of two DCE devices in our application.

Since the standard serial port has been removed from the most latest laptop computers a Bluetooth device is used to emulate the serial port on the computer side. The BlueSoleilTMsoftware [22] is used to configure the Bluetooth dongle as a serial-port Bluetooth service[22] provider and a serial port address is assigned to the Bluetooth dongle as well. Parani-SD/100 is configured as slave mode; once it is found inside the BlueSoleilTMit can be manually connected to the master device (Bluetooth dongle), shown in Fig. 4.19. After establishing the connection the Bluetooth interface

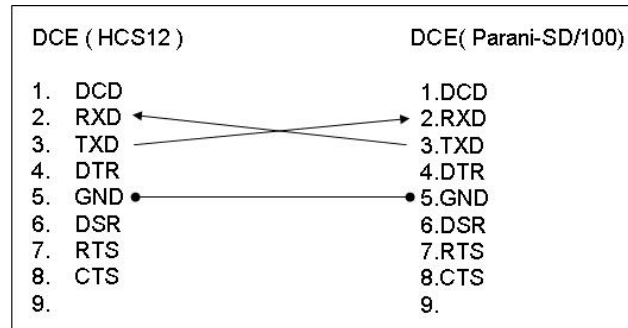


Figure 4.18: Minimum connection between two DCE devices.

can be abstracted. HCS12 is connected with the PC through a virtual serial port. The serial port can then be controlled by the softwares on the PC such as Matlab serial object, HyperTerminal®, etc.

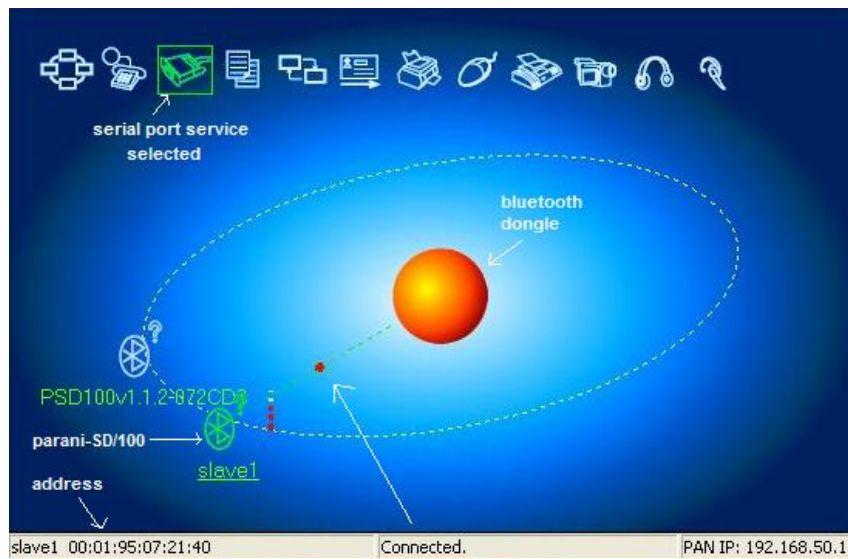


Figure 4.19: Parani-SD/100 is connected to the Bluetooth serial port service.

4.9.2 Data transfer

To insure the real-time data acquisition the data transferred should be as simple as possible considering the serial communication speed. The actual roll angle, for instance, is treated as float point data which takes more bytes than a short integer; it is not suitable for transfer in this case. Alternatively the raw data from A/D convertors of the sensors are sent directly; and the actual angles and controller outputs are calculated in Matlab using the same formula as in the platform. The control period of platform is $25ms$, that says the states are updated every $25ms$. The processor can not stop control to hand-shaking with the serial communication, so I did not design any communication protocol. The data are simply broadcasted at 9600 baud rate.

In the balance-only control scheme one data packet is consisted of total ten bytes including two bytes of starting mark (0xeeff) followed by roll rate gyroscope A/D data (2 bytes), accelerometer A/D data (2 bytes), yaw rate gyroscope A/D data (2 bytes) and two bytes of PWM duty from the RC controller aileron channel(as reference roll angle). Whereas in the autonomous directional control scheme one packet is eight bytes same as the first eight bytes in the balance-only control case. An example data frame from the autonomous directional control scheme can be:

```
“ ee ff 01 31 02 42 02 0d”
```

On the computer side HyperTerminal® is used to capture the broadcasting from HCS12 and the captured data are saved in files. The data is analyzed off-line using the Matlab code.

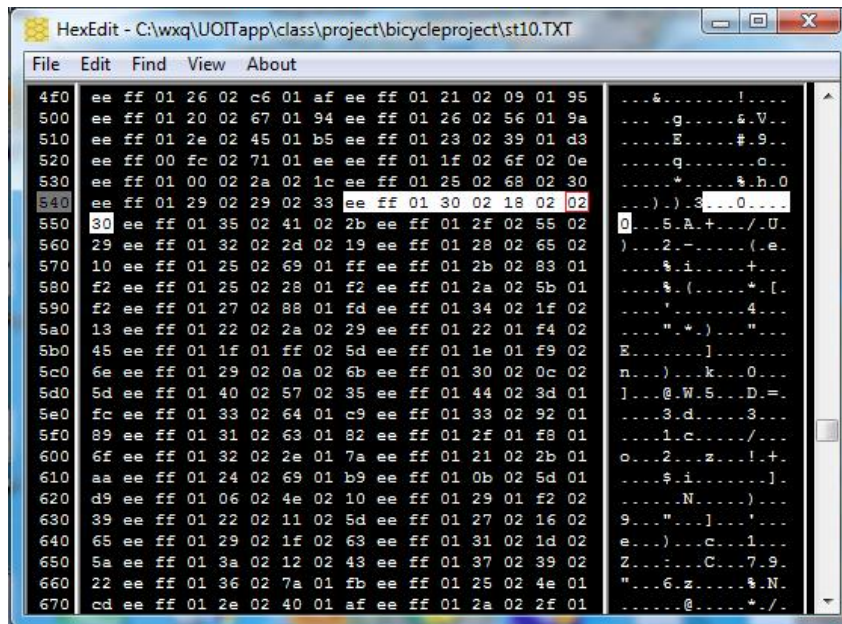
4.9.3 Data analysis and simulation

Although the data captured by HyperTerminal® is save as “.txt” file I am aware that the actual date is binary. Without a sophisticated communication protocol

data error could happen due to baud rate error or some other reasons. Fig. 4.20 is the screen-shot from the binary file editing software HexEdit v1.03[35]. The data was captured during one of the road tests. The highlighted data frame starting from address 0x548 has received an extra byte, a duplicated byte shown in the red box. The mistake could cause dislocation of all the remaining data reading. To filter these mistakes out I designed a filter in the Matlab code to diagnose the correctness of the frame. Always check if the two leading bytes are 0xeeff; if not , the just received frame will be discarded and go on finding the next 0xeeff. The following pseudo code describes the data reading and filtration.

```
fid=fopen('...\bicycleproject\stlab0.TXT','r');
leading_bytes=fread(fid,1,'uint16');
while (leading_bytes <> 0xeeff)
    leading_bytes=fread(fid,1,'uint16');
for i=1:data_size {
    roll_rate_AD=fread(fid,2,'uint8');
    accelerometer_AD=fread(fid,2,'uint8');
    yaw_rate_AD=fread(fid,1,'uint8');
    leading_bytes=fread(fid,1,'uint16');
    if (leading_bytes == 0xeeff)
        real_time_data(i)=[roll_rate_AD, accelerometer_AD, yaw_rate_AD];
    else{
        while (leading_bytes \neg 0xeeff)
            leading_bytes=fread(fid,1,'uint16'); } }
```

The roll angle, roll rate, yaw angle, yaw rate, control input and output are calculated from the real-time data using the same algorithm implemented in the platform. A ground path is also simulated. The test results will be discussed in the



The screenshot shows a HexEdit window titled "HexEdit - C:\wxq\UOITapp\class\project\bicycleproject\st10.TXT". The window displays a list of hexadecimal addresses and their corresponding data. The data is organized into columns: the first column shows addresses from 4f0 to 670 in increments of 10; the second column shows hex data in pairs of two bytes; the third column shows the ASCII equivalent of the hex data. The data appears to be a stream of characters, possibly representing a message or a set of coordinates. The window has a menu bar with "File", "Edit", "Find", "View", and "About".

```
HexEdit - C:\wxq\UOITapp\class\project\bicycleproject\st10.TXT
File Edit Find View About
4f0 ee ff 01 26 02 c6 01 af ee ff 01 21 02 09 01 95 ...&.....!...
500 ee ff 01 20 02 67 01 94 ee ff 01 26 02 56 01 9a ...g.....&.V..
510 ee ff 01 2e 02 45 01 b5 ee ff 01 23 02 39 01 d3 ...E.....#.9..
520 ee ff 00 fc 02 71 01 ee ee ff 01 1f 02 6f 02 0e ...q.....o...
530 ee ff 01 00 02 2a 02 1c ee ff 01 25 02 68 02 30 ...*.....%h.0
540 ee ff 01 29 02 29 02 33 ee ff 01 30 02 18 02 02 ...).).3...0...
550 30 ee ff 01 35 02 41 02 2b ee ff 01 2f 02 55 02 0...5.A.+.../.U.
560 29 ee ff 01 32 02 2d 02 19 ee ff 01 28 02 65 02 )...2.-...(.e.
570 10 ee ff 01 25 02 69 01 ff ee ff 01 2b 02 83 01 ...%.i.....+...
580 f2 ee ff 01 25 02 28 01 f2 ee ff 01 2a 02 5b 01 ...%.(......*.[.
590 f2 ee ff 01 27 02 88 01 fd ee ff 01 34 02 1f 02 ...'......4...
5a0 13 ee ff 01 22 02 2a 02 29 ee ff 01 22 01 f4 02 ...".*.)...".
5b0 45 ee ff 01 1f 01 ff 02 5d ee ff 01 1e 01 f9 02 E.....].....
5c0 6e ee ff 01 29 02 0a 02 6b ee ff 01 30 02 0c 02 n...).k...0...
5d0 5d ee ff 01 40 02 57 02 35 ee ff 01 44 02 3d 01 ]...@.W.5...D.=.
5e0 fc ee ff 01 33 02 64 01 c9 ee ff 01 33 02 92 01 ...3.d....3...
5f0 89 ee ff 01 31 02 63 01 82 ee ff 01 2f 01 f8 01 ...1.c...../.
600 6f ee ff 01 32 02 2e 01 7a ee ff 01 21 02 2b 01 o...2...z...!+.
610 aa ee ff 01 24 02 69 01 b9 ee ff 01 0b 02 5d 01 ...$.i.....].
620 d9 ee ff 01 06 02 4e 02 10 ee ff 01 29 01 f2 02 ...N.....).
630 39 ee ff 01 22 02 11 02 5d ee ff 01 27 02 16 02 9..."....]...'.
640 65 ee ff 01 29 02 1f 02 63 ee ff 01 31 02 1d 02 e...).c...1...
650 5a ee ff 01 3a 02 12 02 43 ee ff 01 37 02 39 02 Z...C...7.9.
660 22 ee ff 01 36 02 7a 01 fb ee ff 01 25 02 4e 01 "...6.z.....%N.
670 cd ee ff 01 2e 02 40 01 af ee ff 01 2a 02 2f 01 .....@.....*/.
```

Figure 4.20: Real-time data displayed in Hex format.

next chapter.

Chapter 5

Results & Discussion

Numerous number of road tests have been conducted on UOIT south parking lot by the research group since late spring of 2010 and lasted till the end of September 2010. Fig. 5.1 is the picture captured from the road test video. The tests happened most intensively in the month of July 2010; the tasks include testing the stability of the platform, comparing different control schemes, tuning controller gains, and so on. On average we spent three days a week, two to three hours a day on the parking lot during the time period. In November 2010, we resumed a few more tests to collect the runtime data of the platform after implementing the wireless communication.

The successful test results validate the control algorithms as well as the implementations of measurement, actuation, and communication. Three different types of test have been conducted. Balance control checks if the free-run platform with sufficient speed can be balanced regardless the direction. Semi-autonomous directional control is to manually change the direction of the balanced platform. The autonomous control tests if the platform can move along a straight line. The difference of the directional concepts between semi- and fully-autonomous mode is that in semi-autonomous mode the platform itself does not know the direction it only follows the command from the RC controller; whereas in fully-autonomous mode

the platform measures its own position and direction by means of the sensors and software. The following list is the road test procedure:

- At the beginning the platform sits still on one of the trainer wheels. The roll angle at the trainer wheel touching the ground is set around $\pm 20^\circ$.
- Establish the Bluetooth connection of the platform and a site laptop; start up HyperTerminal® on the laptop and make it ready to capture data.
- Use RC throttle channel to start up the main motor in manual mode.
- Once the platform starts running, push the control mode lever to autonomous, meanwhile push the throttle lever all the way out to reach full speed.
- In autonomous mode the close-looped control algorithm takes over the steering control and autonomously maintains balance.
- The data broadcasting starts at the moment when the balance controller kicks in and stops after the control mode switches back to manual.
- To finish the test, pull the throttle channel lever down to brake the vehicle, and switch to manual mode.
- Stop data capture of the HyperTerminal® in order to save the current data to file.

5.1 Sensor bias calibration

Sensors bias calibration is part of the test. The method used is the same as the road test except inside the lab. The biases of the two gyroscopes are critical since the angle calculation is done through integrating the sensor output, the error is accumulative. The calibration procedure is as follow:



Figure 5.1: Platform during road test on UOIT south parking lot.

- The platform is positioned upright and checked with the level.
- The platform is also held steady free of any vibration.
- Running balance control algorithm but disconnected the power of all the actuators.
- Collecting the run-time states.
- Plotting and calculating the mean or median of the data from each sensor individually.
- Using the new calculated mean or median as the new bias.

For instance the yaw rate gyroscope bias calibration, total 800 data collected at 25ms sampling rate, the result shows that the data error appears random distribution Fig. 5.2. The mean is 525.25 and median is 525. Using this method to obtain the bias of the rest sensors. The bias of roll rate gyroscope, accelerometer and yaw rate gyroscope are $0x132$, $0x204$, and $0x20D$ respectively.

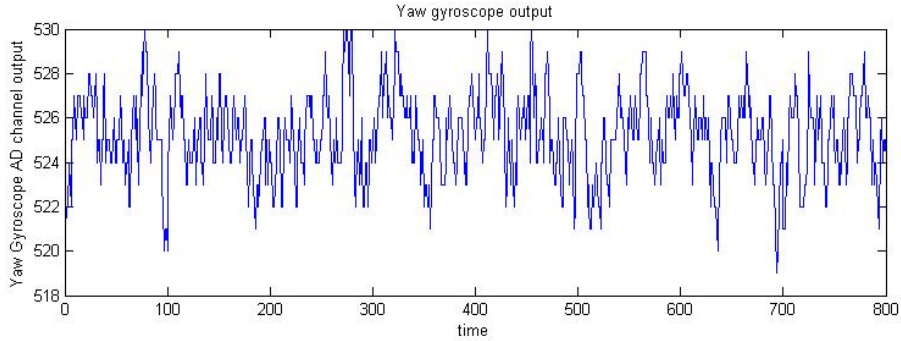


Figure 5.2: Yaw rate gyroscope bias plot.

5.2 Balance control

Before applying the close-looped control the RC controller was implemented to test the platform in manual mode. During the test it has been found that it was almost impossible to balance the platform manually. After the platform reached higher speed the trainer wheels could not hold it from falling. The natural reaction of the passive steering can help balance an unmanned bicycle with enough speed, but in this case the steering shaft is fixed on the servo. The steering must be controlled properly according to the change of roll angle and direction. If we explicitly turned the steering to the same side of tilting trying to pull the platform up; the platform was then lifted up by the centrifugal force but would pass the upright point and roll to the opposite side; before we could change the direction of steering the platform already fell over. Manually, we were never able to keep up the steering with the tilting.

The balance is achieved with the close-looped PD controller based on the correct implementation of roll angle measurement. The balance-only control is to maintain the upright position, i.e. zero roll angle; ideally the steering angle should be zero and the platform should run along a straight line. In reality this never happened due to two reasons one is the error of the steering neutral parameter, another is

the road noise such as the unevenness, wind, etc. The vehicle appears running on arbitrary directions in reaction to the disturbance. One of these test results is collected and plotted, shown in Fig. 5.3., Fig. 5.3 (a) shows the real-time roll angle plot; the blue line is the zero reference roll angle, the reference is the neutral output value from RC controller, due to the output error there is a small offset from zero; the controller successfully controlled the platform to follow the reference. Fig. 5.3 (b) shows the steering output plot; this plot is the simulation of the actual steering value, it is calculated with the same PD control algorithm on the platform. Fig. 5.3 (c) shows the real-time yaw angle. Using the yaw angle information the ground path is simulated in Fig. 5.3 (d). The equation Eq. (5.1) represents the kinematics of the ground path.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix} \cdot v_r \quad (5.1)$$

θ is the real-time yaw angle of the platform, v_r is the constant speed, 5m/s. This equation is solved using Matlab ODE45 solver to calculate the ground path. The following Matlab code describes the ODE and ground path calculation.

ODE fuction of ground path:

```
1: function dxdt= simupath(t,x)
2: global yaw vr;
3: dxdt=zeros(2,1);
4: dxdt(1)=cos(yaw)*vr;
5: dxdt(2)=sin(yaw)*vr;
```

Script of calculating the ground path :

The time span of ODE45 is equal to the sampling rate on the platform, i.e. 25ms.

```
1: global vr yaw;
2: vr=5;
3: Ts=0.025;
4: x0=zeros(2,1);
5: for i=1:m
6:   yaw=theta(i)/180*pi;
7:   tspan = [0,Ts];
8:   ode = @(t,x) simupath(t,x);
9:   [t,x] = ode45(ode, tspan, x0);
10: n=size(x);
11: x_groundpath(i)=x(n(1),1);
12: y_groundpath(i)=x(n(1),2);
13: x0=x(n(1),:);
14: end
```

The result indicates that the platform has been running with balance during the test, but deviates from the start direction and never tends to follow any certain direction.

5.3 Semi-autonomous directional control with balance

This test is to check if the directional control of a bicycle can be achieved through changing the roll angle. In addition to the zero roll angle balance control, the aileron channel from the RC controller is used to change the reference roll angle. The RC channel PWM output signal range $1100\mu s \sim 1900\mu s$ corresponds to roll angle $-20^\circ \sim 20^\circ$. Fig. 5.4 is the result of the test. The relationship of the reference roll angle and the PWM duty cycle is:

$$\alpha_{ref} = (T - T_{offset})/400 * 20^\circ \quad (5.2)$$

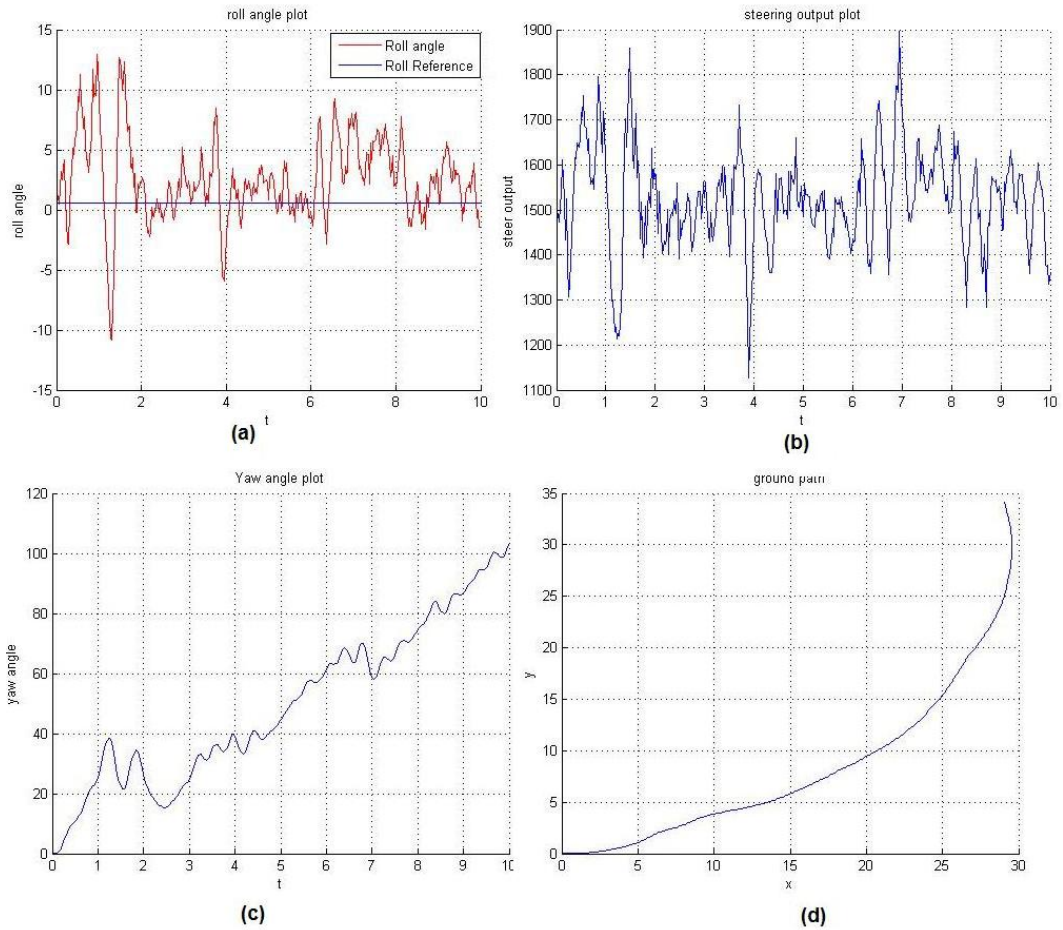


Figure 5.3: Balance control plot. (a) Roll angle plot. (b) Steering output plot simulation. (c) Yaw angle plot. (d) Simulated ground path.

Here, T is the actual duty output from the RC controller channel, T_{offset} is the neutral PWM duty value. Fig. 5.4(a) shows the roll angle changes according to the change of the reference roll angle; during the test the reference roll angle has been held to -15° for few seconds, the bicycle is then stabilized by the controller at the equilibrium roll angle. Fig. 5.4(b) is the steering output simulation. The yaw angle plot in Fig. 5.4(c) shows the platform has been doing left turning motion until the last two second when the RC controller output returns to neutral. The simulation result of the ground path, plotted in Fig. 5.4(d), also agrees with the discussion in case (c) that the bicycle runs at a steady circular motion on the ground during the period within which the non-zero roll angle control is applied.

During test it has been found the platform can be controlled to any direction by changing the reference roll angle. This proves the discussion in the previous chapters that the directional control of a bicycle can be done through roll angle control.

Counter steering motion has been described previously in Chapter one , i.e. for a *Basic Bicycle* [18] in order to make a turn it has to initiate the steering to the opposite direction. The phenomena has also been observed during the test. Fig. 5.5 is the same plot of Fig. 5.4(a) and (b) in different scale to facilitate the demonstration. In this plot the counter steering motion has shown up twice. First at around 2.5 second, the platform received the reference roll angle change command, -15° , that is to let it tilt to the left; the steering immediately turns to the right, the platform tilts to the left and then the steering angle follows the tilt. The second one is more evident at 10 second when the reference roll angle change back to neutral again, at that moment the platform is tilting to the left, the steering turns a even larger angle to the left, the centrifugal force overcomes the gravitational force and the platform tilts back to upright, steering angle then follows. Since there is no steering feedback applied the steering is not really accurate. The approximate

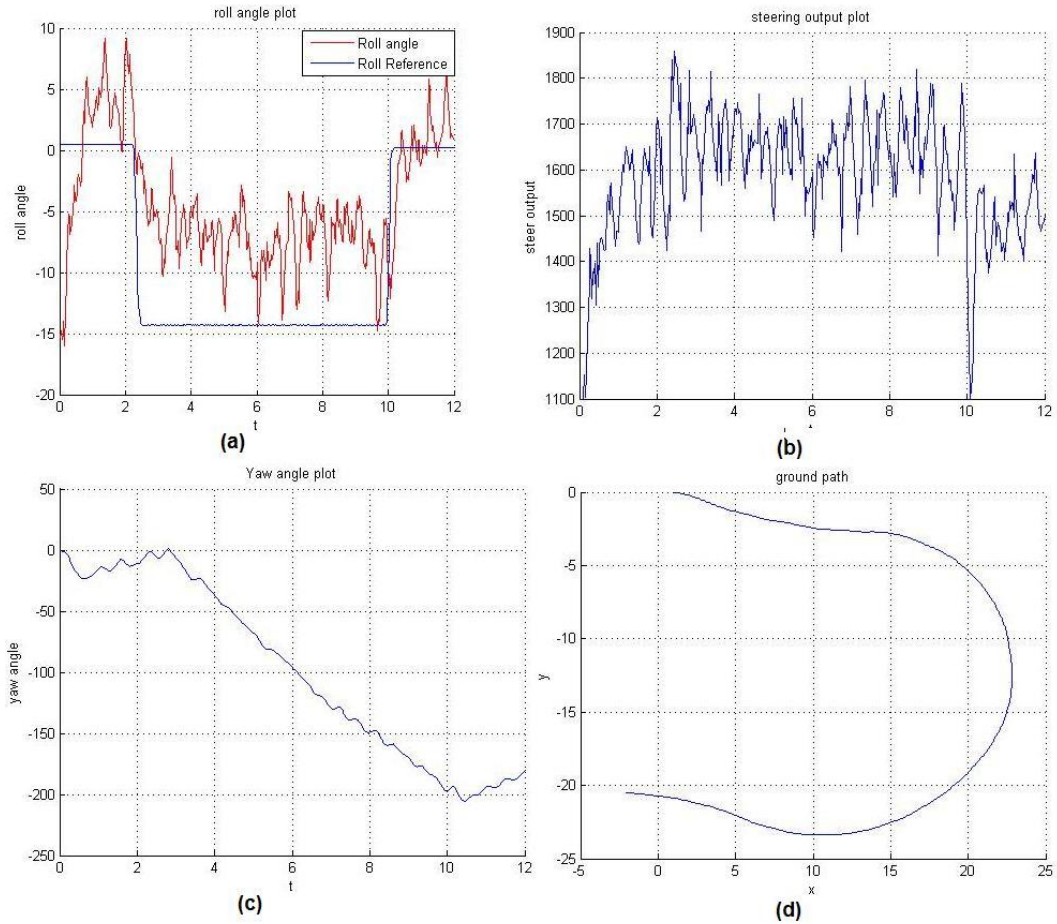


Figure 5.4: Semi-autonomous directional control. (a) Roll angle plot. (b) Steering output plot simulation. (c) Yaw angle plot. (d) Simulated ground path.

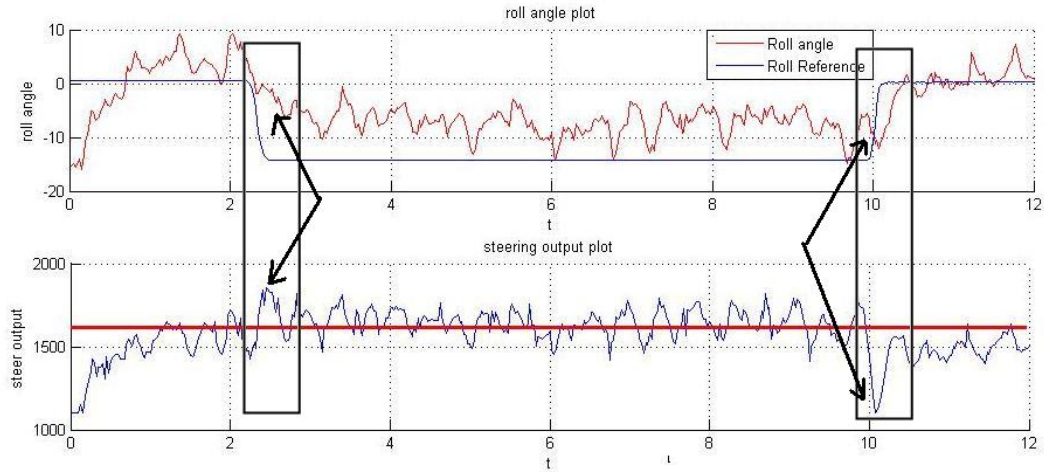


Figure 5.5: Counter steering motion when turning.

neutral steering output is marked with a red line in the diagram. There is a dead band of the steering at neutral position, it is not a constant and different from left to right due to wearing, crashing, and adjusting. The plot can only qualitatively show the *counter steering* action, not precise quantity. But it is clear enough that *counter steering* must happen every time to change the direction of a *Basic Bicycle* [18].

From the results of both balance control tests roll angle static errors are observed, i.e. when the bicycle is balanced, the equilibrium roll angle is not equal to the reference roll angle due to the absence of the integration term of the controller. This can be realized from the PD controller equation, whenever the error remains constant the output stays on the same value. Although integration term can be used to reduce the static error, the prolonged “windup” state could be very harmful to the balance. In the second test the platform is controlled to change direction, but the vehicle does not measure the direction itself. To autonomously control both direction and balance, the direction control need to be put into the control loop.

That is the test in the next section.

5.4 Autonomous directional control with balance

This test is to verify the autonomous directional control scheme. There are a few limitations of this test. The computational capability of the processor making it difficult to track a complex reference ground path. We decided to control it to follow a straight line using yaw angle information. The yaw rate gyroscope has the problem of zero drift during time. If using yaw gyroscope to control direction it must be corrected periodically due to the drift. Although the bias has been carefully plotted the integration algorithm could accumulate any small error to a very large grade. The directional control in this way can only be tested during short periods.

The test results are shown in Fig. 5.6. Fig. 5.6 (a) is also the roll angle trajectory plot as in the previous plots; the actual roll angle is able to track the changes of the reference roll angle, and the reference roll angle is the run-time output from the directional PD controller instead of a fix value or a clean RC command. The steering output is plotted in Fig. 5.6 (b). The yaw angle plot, as in Fig. 5.6 (c), shows that with the directional control the platform tries to return back to zero yaw angle. In the ground path simulation plot, Fig. 5.6 (d), the trend that the platform tries to merge to the reference line can be clearly seen; here the reference ground path is a straight line, $y = 0$.

The tests also encountered some objective limits: the length of the parking lot only allow the platform to run a few seconds before hitting the curb; Bluetooth effective range is another problem; though ParaniSD100 has claimed 100m range with the standard antenna, in fact it hardly reached about 70m to 80m before the Bluetooth connection being cut off. We were aware that we might not be able to collect as many data as in other tests. However, the data were still enough to tell

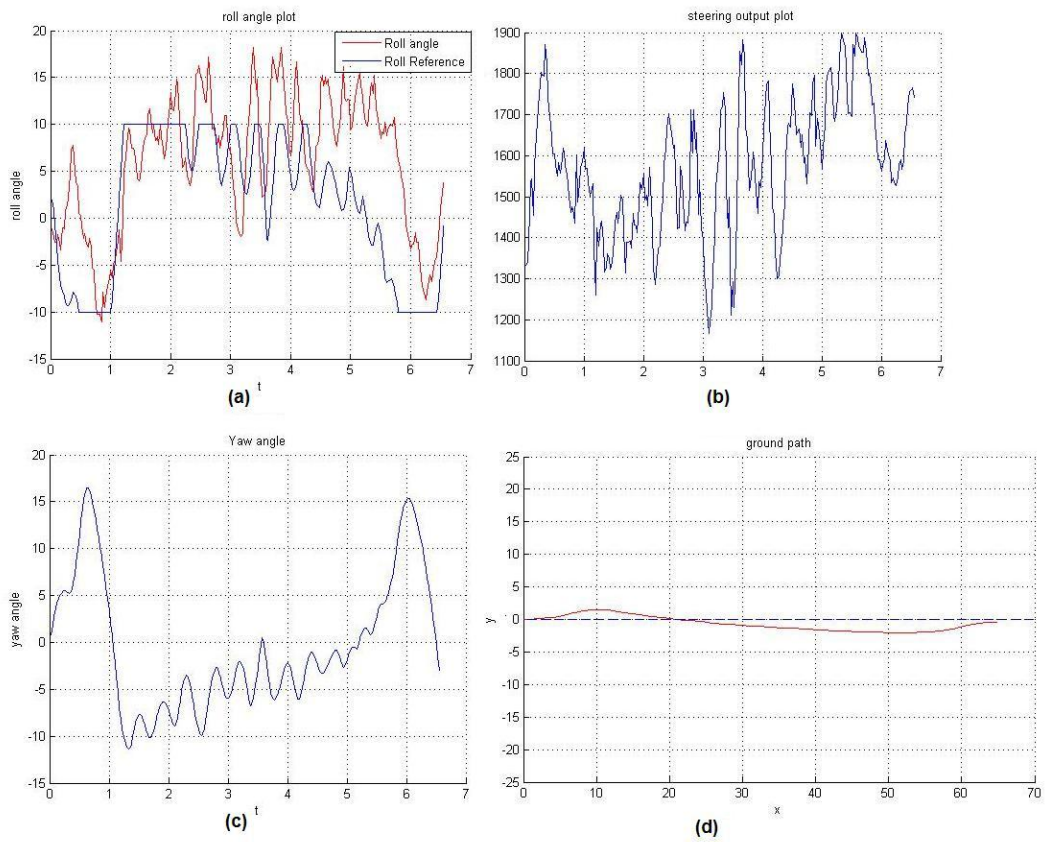


Figure 5.6: Autonomous directional control. (a) Roll angle plot. (b) Steering output plot simulation. (c) Yaw angle plot. (d) Simulated ground path.

that the platform did follow the straight line. The performance of the platform on the road is better than the simulation results since the actual running time was much longer than the time period within which the platform can be reached by the Bluetooth. The test has proved that the proposed control scheme is able to control the direction of the platform autonomously.

Chapter 6

Conclusions and recommendation of future work

6.1 Conclusions

Prior to this thesis work there were few projects related to studying bicycle dynamics, control algorithm investigation, and mobile robotic control, which constituted the theoretical preparation for this hardware approach.

The developed hardware platform is similar to a *Basic Bicycle*, so a software *Basic Bicycle* model was chosen to study the bicycle control problem. I implemented the non-linear controllers to perform balance and ground path tracking control. The simulation results validated that the balance and direction of a *Basic Bicycle* could be controlled with steering and velocity. On the hardware platform, although steering and speed can be used to balance the vehicle, a simpler controller need to be considered due to the capability of the microprocessor. I simulated the balance control with a PID controller. To further simplify the control algorithm, the speed is assumed to be constant and is above the *critical velocity*. The control results of both the nonlinear controller and the PID controller were compared and the results indicated that under some specific circumstances, the PID controller can also generate good results. The settling time and maximum overshoot with PID controller are

all less than that with the nonlinear controller. But the maximum controllable roll angle with the nonlinear controller could be $+45^\circ/-45^\circ$; the maximum roll angle with PID controller was limited to $\pm 20^\circ$ which is good enough for this project.

On the hardware platform, I have faced different challenges from simulations. I implemented RC remote control with Pulse Width Measurement so that the platform has instant response from the RC signals. Traditional low pass filters could cause more than 500 ms lag. The next, I solved the tilt angle measurement problem. The absolute static roll angle is measured with an accelerometer. The roll rate is measured with a roll/pitch rate gyroscope. The dynamic roll angle is then obtained through fusing the signals from the two sensors together by means of a complementary filter which can restrain the effects of the external force to the accelerometer and drift of the gyroscope. Directional information is obtained using a yaw rate gyroscope. At this point it is assumed that any starting direction is the reference direction. The system test indicated that the magnetic loads including the drive motor, servos and so on had caused severe electromagnetic noise and interfered to each other, to the sensors signals, and to the microprocessor. The useful signals were totally swamped. To make correct and accurate feedback is the assurance of implementing the control algorithm. The system is then separated into three power groups each with its own independent power supply and ground. Between the actuators and microprocessor optoelectronic elements are used to isolate the noise. This is the key to ensure the success of the whole research.

Several control scenarios have been compared including balance control, directional control, ground path tracking, etc. Finally two control schemes, i.e. balance-only and directional control with balance, are applied and tested on the platform. The balance control scheme has maintained the upright position or equilibrium roll angles so that the platform can go straight or circle. The directional control scheme has also moved the bicycle to the reference direction meanwhile maintaining bal-

ance. The controller gains were estimated at the beginning and then tuned during several road tests. The two control algorithms have been proved to be suitable to the processor's capability and fulfilled the control tasks.

At last wireless serial communication between the platform and computer is implemented. A Bluetooth device is installed on a PC to emulate a serial port. At the platform side a RS232/Bluetooth modem is used to convert the RS232 signal to Bluetooth wireless. After establishing connection the Bluetooth interface can be abstracted, the platform virtually communicates with a PC using RS232 standard.

6.2 Recommendation of future work

This research is the first bicycle type two-wheeled autonomous vehicle control approach. The results have indicated the feasibility to control such a vehicle. The controllers are based on PID algorithms; PID with fixed gain values is only suitable for conditions of a narrow scope. The bicycle controllability has not been fully investigated. The limits of the hardware make it hard to implement faster control nor complicated calculations. For comparison, the Ghost rider bike balance can be controlled at 1KHz; the control frequency of our actuators is limited by the PWM signal with a 44.5Hz frequency; at the microprocessor side, the control output task will be delayed if the control frequency higher than 100Hz. From the hardware point of view, the digital RC servos can be used to increase the control frequency. Real time states acquisition is very important for analyzing the control effect. Using Bluetooth as the means of wireless communication is another drawback due to the limitation of the Bluetooth range. Serial ports using RF module[51] which covers up to six miles can be a very promising choice. Currently, two more implementations could be conducted to improve the directional control; the first one is using the IMU unit, a rate gyroscope can not carry the task alone; the second one is the feedback of

steering angle, this will correct the parameter errors of the steering mechanism but a new control strategy will be involved. In the future more advanced computer system or distributed systems are required to perform the complex computing tasks. To improve the stable range more sophisticated modeling according to the responses to different road conditions can be considered. Stochastic model predictive control for instance as in the Ghost rider bike could be a good way to deal with the problems. Online controller gain adaptation can ensure the best performance in different environment. All these need more efficient team work and dedication. The mechanical potential also limits our tester to be good only for prototyping. A more capable platform will be required to build the next autonomous bicycle.

Autonomous bicycle control is a much broader concept than balance control. Navigation problem can be the next big topic to explore into, it is comparable to the four wheeler navigation problem but there are something more of the bicycle due to its unstable nature. The current research has just laid the second milestone of the autonomous bicycle research of UOIT.

Bibliography

- [1] “16-bit Microcontrollers.” *Freescale*. ©2006 Freescale Semiconductor, Inc. Web. 5 May. 2009.
<http://www.freescale.com>.
- [2] “4N25, 4N26, 4N27, 4N28 General Purpose 6-Pin Phototransistor Optocouplers.” *4N25 Datasheet*. ©2000 Fairchild Semiconductor Corporation. Web. 5 May. 2010.
<http://www.fairchildsemi.com/ds/4N/4N25-M.pdf>.
- [3] “ $\pm 50^\circ/s$ Yaw Rate Gyro ADXRS614.” Rev. A, *Analog Devices*. ©2007 - 2010 Analog Devices, Inc. Web. 1 Apr. 2010.
http://www.analog.com/static/imported-files/data_sheets/ADXRS614.pdf.
- [4] Alciatore, G. David and Hestand, B. Michael. *Introduction to Mechatronics and Measurement Systems*. Third Edition, McGraw-Hill Higher Education 2007. 384.
<http://mechatronics.colostate.edu/>.
- [5] “Application Module Student Learning Kit Featuring Freescale MC9S12DT256.” *Freescale*. ©Freescale Semiconductor, Inc. 2006. Web. 5 May. 2009.
<http://www.freescale.com>.
- [6] Araki, M. “Control Systems, Robotics, and Automation Vol. II - PID Control.” *Encyclopedia of Life Support Systems (EOLSS)*: all. EOLSS Web. 1 Jun. 2010.
<http://www.eolss.net/ebooks/Sample%20Chapters/C18/E6-43-03-03.pdf>.
- [7] Barr, Michael and Massa, Anthony. “Programming Embedded Systems: With C and GNU Development Tools.” 2nd ed. Sebastopol, CA: O’Reilly Media Inc. 2006. 103 - 4. Print.

-
- [8] Bennett, Stuart. *A history of control engineering, 1930-1955*. London, United Kingdom: Peter Peregrinus Ltd., 1993. 48. Print. ISBN 9-780863412998.
- [9] Bhatia, V.B. *Classical Mechanics: With introduction to Nonlinear Oscillations and Chaos*. New Delhi, India: Narosa Publishing House, 1997. 268-357. Print. ISBN 81-7319-105-0.
- [10] Brooks, Christopher. et al. "Heterogeneous Concurrent Modeling and Design in Java (Volume 3: Ptolemy II Domains)." April 1, 2008. Technical Report No. UCB/EECS-2008-30. Electrical Engineering and Computer Sciences University of California at Berkeley.
<http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-30.html>.
- [11] Carvallo. "Théorie Du Mouvement Du Monocycle. Part 2: Theorie de la Bicyclette." *Journal de L'Ecole Polytechnique, Series 2, Volumn 6 (1901)*: all. Print.
- [12] Chen, Chih-Keng and Dao, Thanh-Son. "Fuzzy control for equilibrium and roll-angle tracking of an unmanned bicycle." *Multibody Syst Dyn* 15 (2006): 325-350. © Springer Science+Business Media B.V. 2006. DOI 10.1007/s11044-006-9013-7.
- [13] Colton, Shane. "Complementary filter." web.mit.edu/Colton MIT., n.d. Web. 24 Feb. 2009.
Email. scolton@mit.edu.
- [14] Cossalter, Vittore. *Motorcycle Dynamics* 2nd ed. (English Edition), LULU publisher. 2006. Print. ISBN-9781-14303-0861-4.
- [15] Döhning, E. "Stability of Single-Track Vehicles." *Forschung Ing.-Wes.*, Vol.21, No.2, (1955): 50-62. (English translation by J. Lotsof, March 1957; Copy received from Calspan Corp. library, Buffalo, N.Y., through Cornell Engineering library. Many authors incorrectly reference as Döhning and Brunswick.)
- [16] Fakhreddine, O. Karray and Clarence, de Silva. *Soft Computing and Intelligent Systems Design, Chapter 3: Fuzzy logic control*. Harlow, England: Pearson/Addison Welsley, 2004. 137-200. Print.
- [17] Getz, H. Neil. "Dynamic Inversion of Nonlinear Maps with Application to Nonlinear Control and Robotics." (1995). Ph.D. Dissertation, Dept. of Mechanical Engineering, University of California at Berkeley.

- [18] Hand, R. S. "Comparisons and Stability Analysis of Linearized Equations of Motion for a Basic Bicycle Model." (1988). Master thesis, Cornell University.
- [19] "HS-805BB+ MEGA 1/4 SCALE SERVO SPECIFICATION." *Robotshop*. n.p., n.d. 1 Jun. 2010.
<http://www.robotshop.ca/PDF/hs805.pdf>.
- [20] *Inertial Measurement Units (IMUS)*. ©1995 - 2011 Analog Devices, Inc. Web. 10 Oct. 2010.
<http://www.analog.com/en/mems/imu/products/index.html?ref=ASC-PR-486>.
- [21] "Integrated Dual-Axis Gyro IDG-300." *InvenSense*. 2007 InvenSense, Inc. Web. 9 Sep. 2010.
<http://www.invensense.com>
- [22] "IVT BlueSoleil™ User Manual." Version: 1.4. *IVT Corporation*. CD.
<http://www.ivtcorporation.com>.
- [23] Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems." *Transaction of the ASME Journal of Basic Engineering* (1960): 35-45. Print.
- [24] Klein, F. and Sommerfeld, A. "Theorie des Kreisels (Theory of Gyroscopes)." Vol. III. *Technical Application, Section IX, Number 8*. (1903): 863-64. Berlin and Leipzig. In German; translation available from Cornell Bicycle Research Project/Prof. Ruina.
- [25] KONDO, M. et la. "Theoretical study on the running stability of the two-wheelers." *Trans. SAE Japan*, Vol. 17, No. 1, (1963): 8. Print.
- [26] Lee, Sangduck and Ham, Woonchul. "Self Stabilizing Strategy in Tracking Control of Unmanned Electric Bicycle with Mass Balance." *Proceedings of the 2002 IEEE/RSI Intl. Conference on Intelligent Robots and Systems EPFL*. Lausanne, Switzerland (October 2002).
- [27] Levandowsk, Anthony. et al. "Autonomous Motorcycle Platform and Navigation Blue Team DARPA Grand Challenge 2005." *Blue Team Autonomous Motorcycle Platform - DARPA*. DARPA. Web. 1 Jun. 2010.
<http://www.darpa.mil/default.aspx>.
- [28] *Machine Science Web*. © 2005-2011 Machine Science Inc.
<http://www.machinescience.org>.

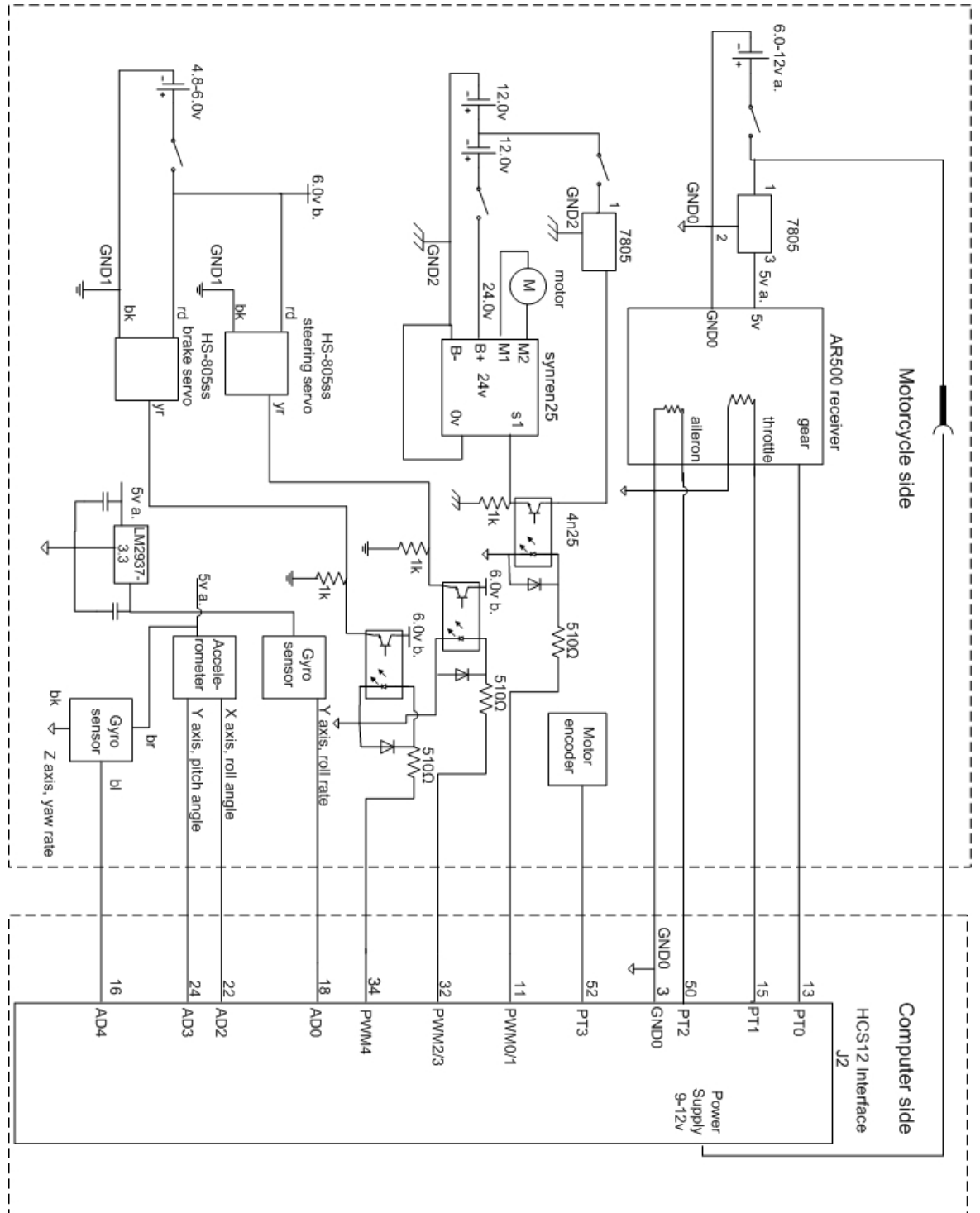
- [29] Meijaard, J. P. et al. "Linearized dynamics equations for the balance and steer of a bicycle: a benchmark and review." *Royal Society Series A*. Jun. 2007: 463. <http://audiophile.tam.cornell.edu/~als93/Publications/06PA0459BicyclePaperv45.pdf>
- [30] Neĭmark, J. I. and Fufaev, N. A. "Dynamics of Nonholonomic Systems" *American Mathematical Society Translations of Mathematical Monograph*. Vol. 33, (1967): 326. Print.
- [31] *Ordinary Differential Equations*. Feb. 2008. The MathWorks Inc. 1 Jan. 2010.
- [32] "Parani-SD100/200 User Guide." Ver. 1.1.6, Firmware version 1.1.X., Seoul, Korea: Sena Technologies, Inc. 2009. Print.
- [33] Pearsall, R. H. "The Stability of a Bicycle." *The Institute of Automobile Engineerings, Proceeding Session 1922-23 Part I*. Vol. XVII, (1922): 395-404. Print.
- [34] "Precision $\pm 1.7g$ iMEMS Accelerometer ADXL103/ADXL203". Rev. C. *Analog Devices*. © 2004 - 2010 Analog Devices, Inc. Web. 1 Apr. 2010. http://www.analog.com/static/imported-files/data_sheets/ADXL103_203.pdf
- [35] Prewett. "HexEdit v1.03." Hex Edit Software. n.p., n.d. 10 Sep. 2010. Email. prewett@pacific.mps.ohio-state.edu. <http://www.physics.ohio-state.edu/~prewett/hexedit>.
- [36] Rankine, W. J. M. "On the dynamical principles of the motion of velocipedes." *The Engineer* 28. 1869. all. Print.
- [37] *RS-232 Standard*. n.d. Omega Engineering Inc. 1 Dec. 2010. <http://www.omega.com/techref/pdf/rs-232.pdf>.
- [38] Sharp, R. S. "A Review of Motorcycle Steering Behavior and Straight Line Stability Characteristics." © SAE Internal. 1978: Technical Paper No. 780303. DOI: 10.4271/780303.
- [39] Sharp, R. S. "Stability and control of motorcycles." *JOURNAL MECHANICAL ENGINEERING SCIENCE* 2 (1971): 316 -329. Print.
- [40] Sharp, R.S. "Stability, Control and Steering Responses of Motorcycles." *Vehicle System Dynamics*. Vol. 35, No. 4-5, © Swets & Zeitlinger Publisher 2001: 291 - 318.

- [41] Sharp, R.S. "The lateral dynamics of motorcycles and bicycles." *Vehicle System Dynamics*. Vol. 14, Issue. 4 - 6 © Swets & Zeitlinger Publisher 1985: 265 - 83. DOI: 10.1080/00423118508968834.
- [42] Sivarajasingam, Aravinth and Thevarayan, Thivakaran. "Stabilization And Design And Implementation Of RC Transmitter For An Autonomous Two Wheeler Vehicle." 2009. Undergraduate thesis, University of Ontario Institute of Technology, 2000 Simcoe Street North, Oshawa, Ontario, Canada L1H 7K4.
- [43] "SPEKTRUM DX5e 5-Channel Full Range DSM2TM 2.4GHz Radio System." Horizon Hobby Inc. Apr. 2009. Print.
- [44] Spencer, Ian. "Design And Development Of An Actuation And Sensory System For A Two Wheeled Electric Vehicle." 2008. Undergraduate thesis, University of Ontario Institute of Technology, 2000 Simcoe Street North, Oshawa, Ontario, Canada L1H 7K4.
- [45] "SyRen 10/SyRen 25 motor driver users guide." *SyRen 25*. Dimensional Engineering 2007. Web. 1 Jun. 2010.
<http://www.dimensionengineering.com/SyRen25.htm>.
- [46] Thomas, Peter. "How to Tune Cascade Loops." *Control Specialists Ltd. ExperTune User Conference*. Austin, Texas. Apr. 2007.
<http://www.controlspecialists.co.uk>
- [47] Timoshenko, S. and Young, D. H. *Advanced Dynamics*. New York: McGraw-Hill Book Company, Inc., 1948. 239-243. Print.
- [48] Vilanova, Ramon and Arrieta, Orlando. *ExperTune User Conference. Austin, Texas. Apr. 2007*. Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. III (2008), Suppl. issue: Proceedings of ICCCC 2008: 521-25.
- [49] Wang, X. "Embedded real-time control systems course project report: Simulation of a two wheel autonomous vehicle with ptolemyII." 2009, Department of Electrical and Computer Engineering, University of Ontario Institute of Technology, 2000 Simcoe Street North, Oshawa, Ontario, Canada L1H 7K4.
Email. xinqi.wang@uoit.ca
- [50] Whipple, F. J. W. "The Stability of the Motion of a Bicycle." *Quarterly Journal of Pure and Applied Mathematics*, vol. 30, (1899): 312 - 348. Print.

-
- [51] “XBee-PRO®900/DigiMesh™900 OEM RF Modules.” *XBeeo/XBee-PROo ZB RF Modules*. Digi International Inc. 3 Mar. 2009. Web. 10 Jan. 2011.
<http://www.digi.com/>

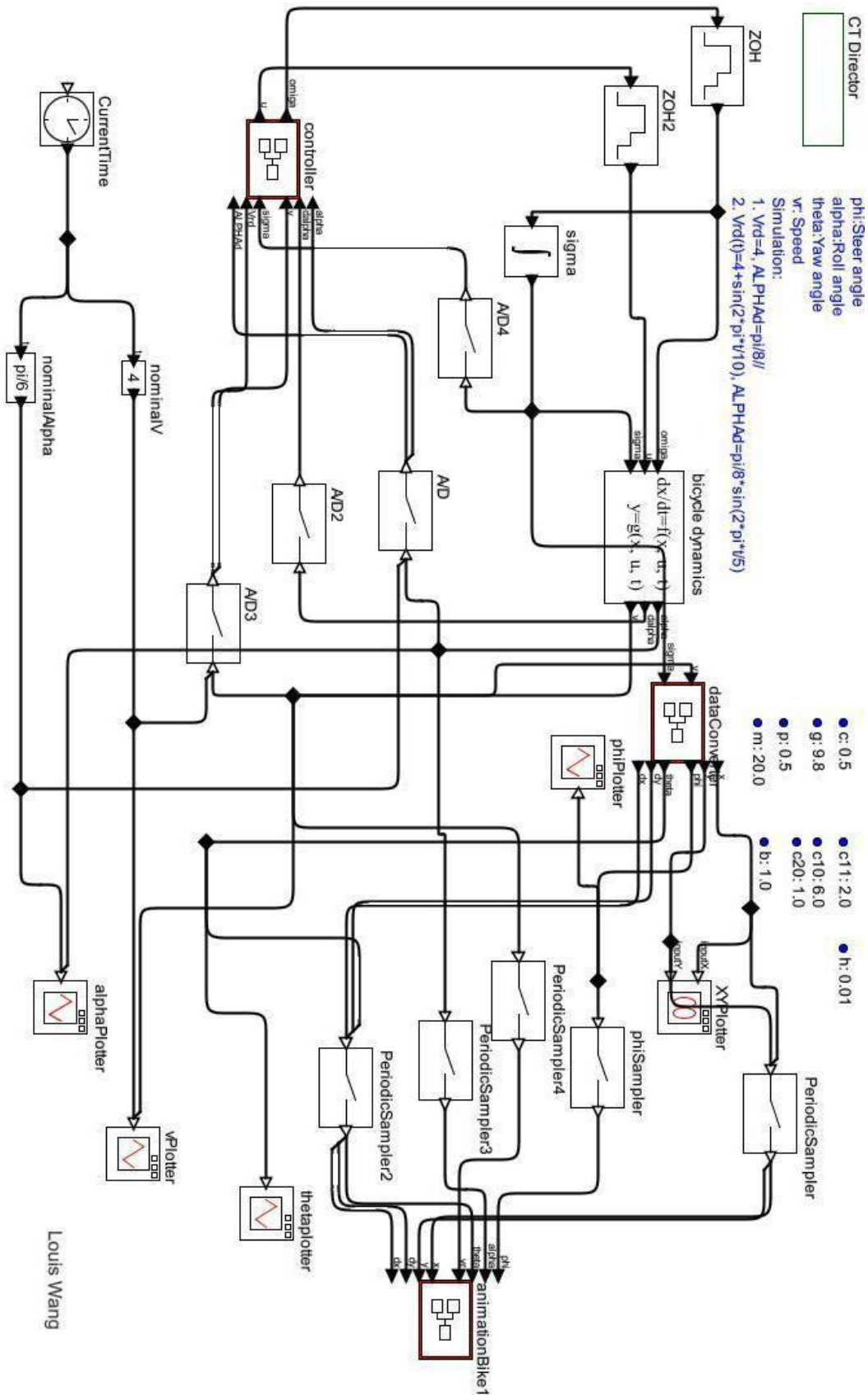
Appendix A

Electrical system schematic



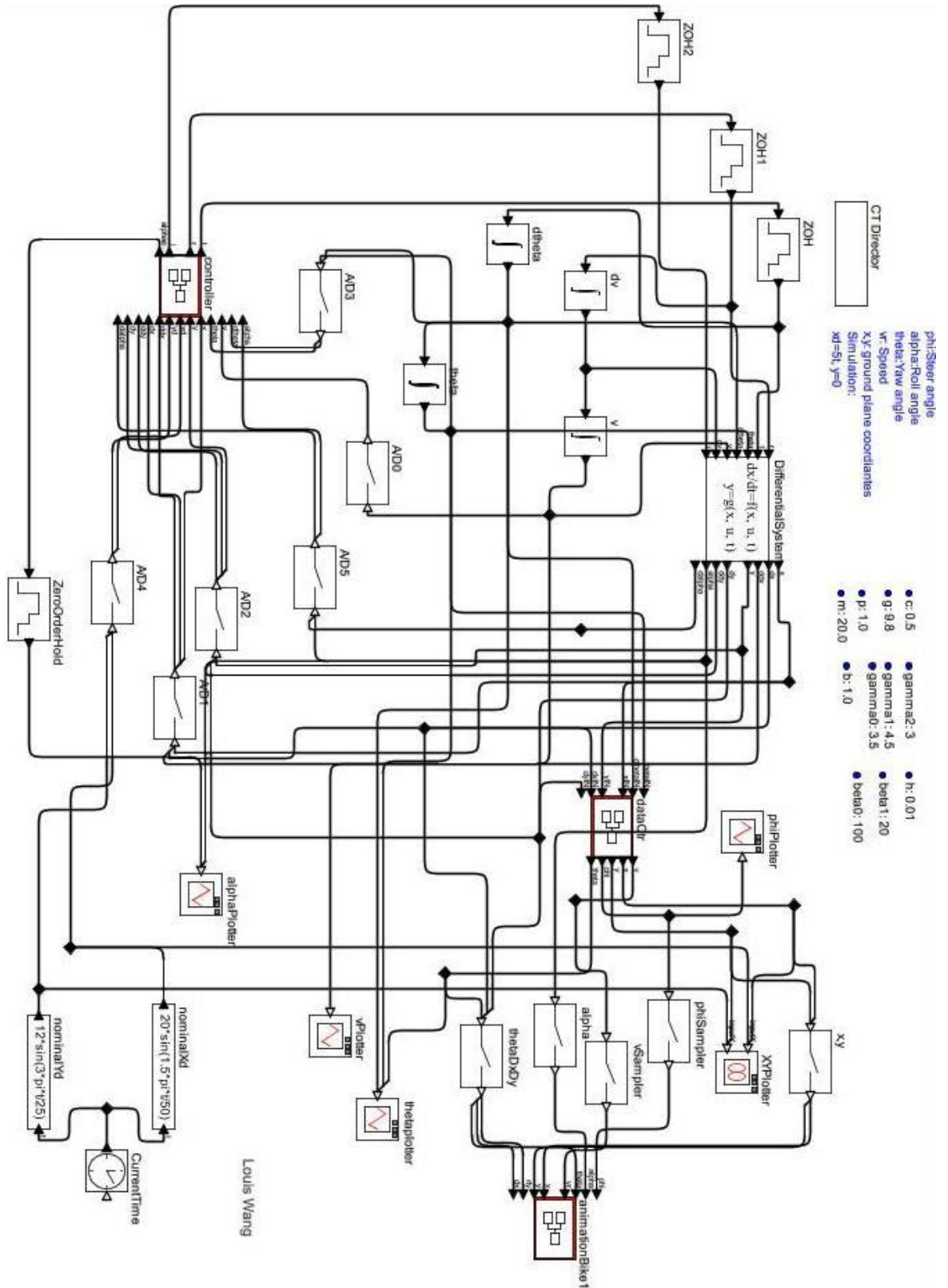
Appendix B

Balance control PtolemyII model



Appendix C

Ground path tracking with balance PtolemyII model



Partial Copyright License

I hereby grant the right to lend my thesis to users of the University of Ontario Institute of Technology Library, or in response to a request from the Library of any other university or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the university designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

TITLE OF THESIS:

TEST PLATFORM DESIGN AND CONTROL OF A BICYCLE-TYPE TWO-WHEELED
AUTONOMOUS VEHICLE.

Author: _____

LOUIS XINQI WANG

(Date)