Platform for ergonomic steering methods investigation of "Segway-style" balancing scooters

A thesis submitted in partial fulfilment
of the requirements for the degree of

Master of Engineering

in physics and Electronic Engineering
at the
University of Waikato

By

Weiqian (Viking) Zhou

2008

ACKNOWLEDGEMENTS

I would like to have this opportunity to acknowledge all people who had help me through my project and thesis.

Specially thanks for my academic supervisor, Jonathan Scott and his wife Kay, they help me a lot much more than academic problems. Though many of my questions were silly, Jonathan always explained in details with great patience.

Thanks for Nihal Kularatna, he gave me much assistance for the power electronic part, especially the advice of snubber circuits for protecting MOSFET. Also appreciate Dr. Howell Round for his helping me to build up the mathematical model of the whole system and complete the control system simulation.

Last but not least, my great gratitude is to my family, my kindly parents, my beautiful wife Leman and my lovely child Levi.

ABSTRACT

Segway has been a popular production as an alternative transporter since its invention at the end of 20th century. Millions of people like for its ergonomic design and high-tech elements. It is predicted to be an innovational product to change a person's life style.

This thesis focuses on building a simple low cost, home-made Segway style scooter. This project uses two electric scooter motors, two 12V car batteries, one accelerometer and several microprocessors to build up the whole system.

Significantly, this project also explains how to build a Brushed Direct Current (BDC) motor driver with a rated output power of more than 350W and the capability of coping with up to 120A transient peak current and up to 40A continuous current. Four-quadrant operation and eight modes of DC motor operation are discussed.

A mathematical model of the Segway style scooter is also introduced in details. This including the modelling of a BDC motor, a two-wheeled inverted pendulum and their combination. The linearization of these models is used. At the end the linearized model is simulated in computer software.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 The project background

Segway™ is a self-balancing, non-tandem, electric powered transporter invented by Dean Kamen in 1999. (See *Fig 1.1*) The name “Segway” came from the word “segue”. It transforms a person into an “empowered pedestrian”, allowing him/her to go farther, move more quickly and carry more. [1] Segway is an innovated product to improve the communication manner of people in workshops, indoor offices, campus, Golf courses, etc. It has been used by police for patrol and to get to the spot in a short time in a busy street; it is also popular as a tourist-transport in cities.

Fig 1.1 View of Segway

1.2 Working theory

Segway's working is based on a new technology which is termed "dynamic stabilization". It enables the Segway to work seamlessly with the body's movement. Since the wheels of the Segway are parallel, it does not keep itself upright mechanically. When the rider stands still, it resembles an inverted pendulum. This is an inherently unstable system. To make the system stable and keep it in equilibrium, the wheels should always be right under the centre of gravity.

Human beings and other two-foot walking mammals have their own sensing organs so they can find out where the centre of gravity is and move their bodies just on their feet so that they can keep balance. When they walk, they move the centre of gravity forward and step up their feet at the same time, the faster they move the centre of gravity and their feet, the faster they go. Similarly, Segway keeps balance when the rider stands still and responds to tilting forward and leaning back to accelerate and stop. Its speed, direction and stopping (braking) are controlled by the rider's shifting weight. However it implements steering using a handlebar grip turning mechanism which is similar to that used for acceleration on conventional motorbikes and some electric scooters such as the eGo. [2] This is considerably less elegant a method compared with what is used for thrust.

1.3 Aim of the project

This paper describes making a low cost simple Segway-style scooter from “OFF-THE-SHELF” materials. The purpose of this is to produce a machine to allow the testing of an alternative steering method for Segway-style bikes. To achieve this purpose, two scooter driving motors are used to drive the wheels. These motors are powered by car batteries. If the scooter is running under heavy load, the motor will produce a huge current. This situation will also happen when the scooter is switching between running directions. Therefore, the motor driver must have a relatively high specification.

The mathematical model of this system is based on a two-wheeled inverted pendulum model. This model is a multiple-input, unstable, nonlinear system. Linearization is applied to the mathematical model to meet the requirement of the PID controller. The simulation carried out in software shows that this scooter system can be controlled and the expected result can be achieved, although time did not allow the full control to be implemented in this work.

Overall, the Segway is a very high-tech product but actually we can build a simple one with standard materials if the system backup and redundancy are neglected. Many people have built their own Segway-style scooters successfully as reported on web. [3][4]

1.4 Thesis structure

This thesis focuses on how to build a simple Segway-style, self-balancing, two-wheeled scooter. Chapter 2 represents the hardware used in this project and gives specifications to explain how it meets the requirements of the project. Chapter 3 covers the circuit design and software design for the electrical part of this project. Subsection 3.1 describes what the difficulties of the motor driver circuit and what has been done to achieve the goal; subsection 3.2 concerns the theory of making the software; and subsection 3.3 is about the key part of the algorithm. Chapter 4 illustrates how model the system in detail, including the inverted pendulum model and the model of a BDC motor. Chapter 5 gives the simulation results which show the whole system can be controlled in theory. Chapter 6 gives the author's result and future recommendations.

2 EXPERIMENTAL PLATFORM

Compared to the real Segway product, all materials used to build this Segway-style scooter can be easily found. Even the electrical components are low cost and can be bought in store easily.

2.1 Mechanical devices

The scooter is built up with a “T” form bar and a steel chassis with room for batteries. Two wheels are arrayed in parallel by the chassis’s side, and they have a 0.5° toe in to ensure the scooter would be more stable when steering. Motors are arrayed off the back end under the chassis; they drive the wheels by pulley belts. This driving method can lower the noise compared with gears. An overview of the scooter is shown in *Fig 2.1*.

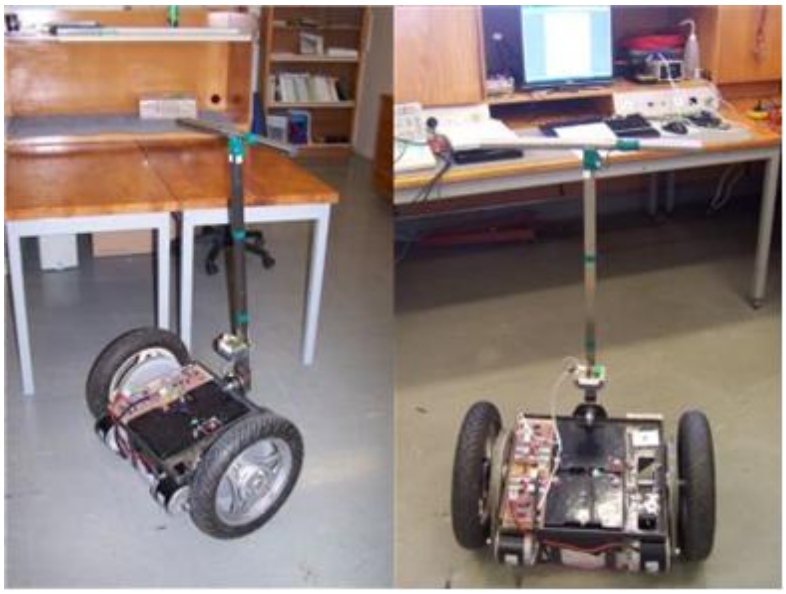
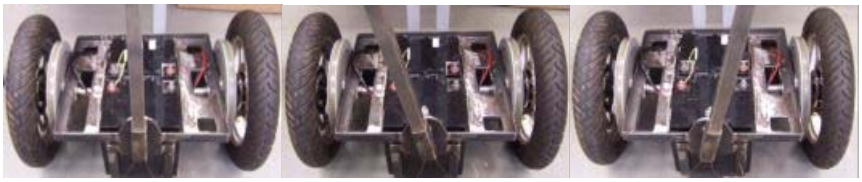


Fig 2.1 View of the scooter

The handle bar can be swung to left side and right side. This is to allow testing of one ergonomic style of steering. See the pictures in *Fig 2.2*.



(a) (b) (c)

(a) Going straight status

(b) Right turn status

(c) Left turn status

Fig 2.2 Steering method of the scooter

When the handle bar is moved to left/right, the scooter will turn left/right; when it is up straight, it will go straight. The height of the T-bar can be adjusted from 80 cm to 110 cm.

Other possible methods for ergonomic include steering twisting handlebar or rocking handlebar on top for a rigid column.

2.2 Electrical hardware

The simple diagram following shows the configuration of the motor drive hardware part. Each wheel is controlled by its own microcontroller. Therefore there are two identical but independent driving systems to ensure two motor can be controlled separately for implementing steering.

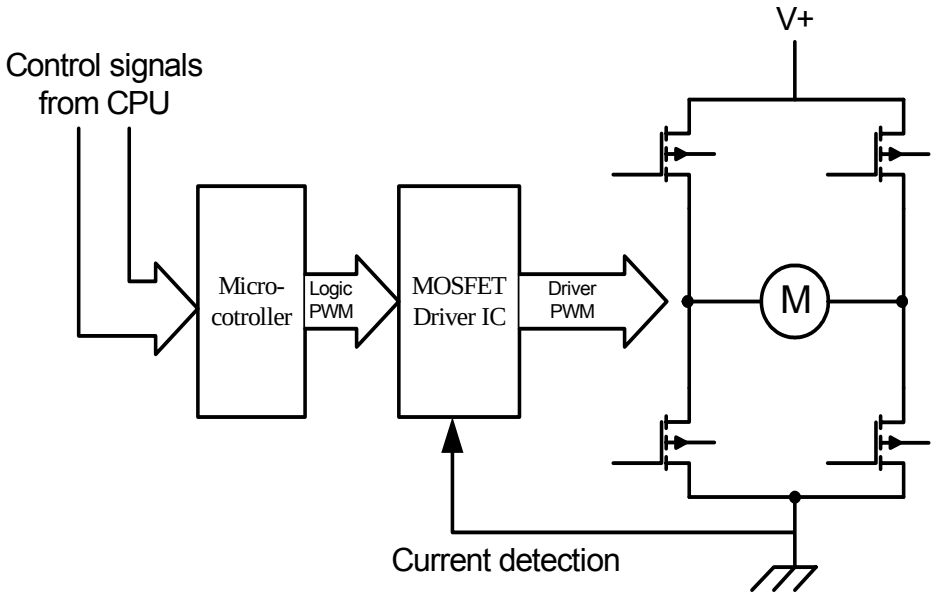


Fig 2.3 Motor driver circuit block

2.2.1 Power.

Power of the scooter is provided by two common 12 V lead acid car batteries. Each of them has a dimension of H * W * D = 19 cm * 20 cm * 16 cm. The batteries are in series to get a 24V power supply. 24V is applied to the H-Bridge MOSFETs directly and it is also regulated into 5V and 12V to supply the control circuit and driver circuits. These two lead-acid batteries are heavy (about 16 - 17kg each), but

they can provide plenty of current for two driving motors, especially when motors are working in overload. They also store energy when the H-Bridge pumps power back to line—so called regenerative braking.

2.2.2 Driving motors.

Two *MY1016* motors are used for driving. They are normally used for electric bikes. View as *Fig 2.4*.

Fig 2.4 View of the scooter motor MY1016

The rated current of *MY1016* has a specification of rated 19 A under 24V, rated speed 2750 RPM and maximum output power 350 W. Because of the motor's low resistance (1Ω armature resistance), it will has a huge inrush current (up to 120A) at the moment when the motor switches from fully off to on or alters direction.

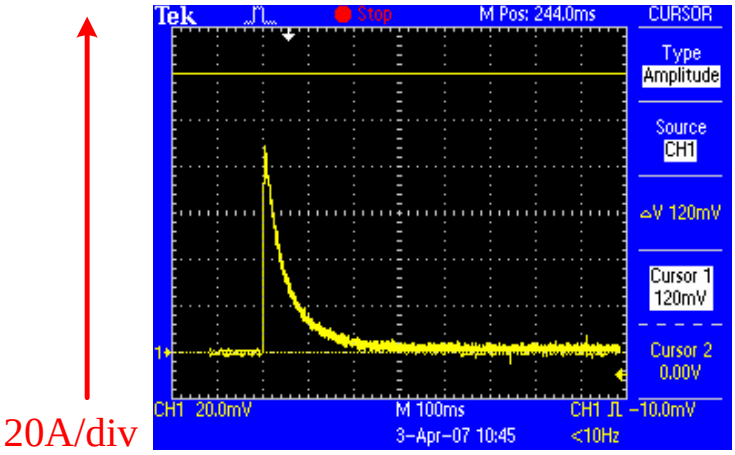


Fig 2.5 Motor current curve when started from off to on

Fig 2.5 shows when motor MY1016 is started from fully stop to on state, the curve of current in the armature. Note that vertical scale is 1:1000 since a 1000X probe had been applied when it is measured.

This current will last for few nanoseconds to few microseconds, and it is enough to blow up the H-Bridge in certain circumstance. Therefore very careful design is required otherwise the driver circuits can be damaged. Many MOSFETs were burnt out in development of this circuit by the author.

2.2.3 Balancing sensor.

The centre of gravity of the Segway and rider is not easy to sense and signal to the control system. Segway use a gyro, which is a device for measuring or maintaining orientation, based on the principle of conservation of angular momentum. [3]

A gyroscope is a sensor which comprises a spinning mass mounted within a gimbal system. In the absence of friction, the spinning mass would remain stationary in inertial space and ideally act as a portable reference direction. [3] Segway use a gyroscope and other tilting sensors to monitor the centre of gravity about 100 times per second.

Instead of using Gyroscope, the scooter here uses an accelerometer (Eval-ADXL330Z) as a balancing sensor. See *Fig 2.6*. This is a silicon chip with a "diving board" cantilever. The "diving board" bends a tiny amount in the direction of gravity and electronics detects how much it bends. With two of these "diving boards" arranged perpendicular to each other, it can figure out the angle of gravity by computing the arctangent of the ratio of the bending measurements. As the result, this device gives the tilting angle in XOY plane as outputs and it feeds back a 1.8~3.2Vdc from -90° to +90° for each axis. However, note this signal will be given only when the sensor is still, so actually it measures the force of acceleration and also the acceleration of gravity, because gravity is constant acceleration on the surface of the earth.

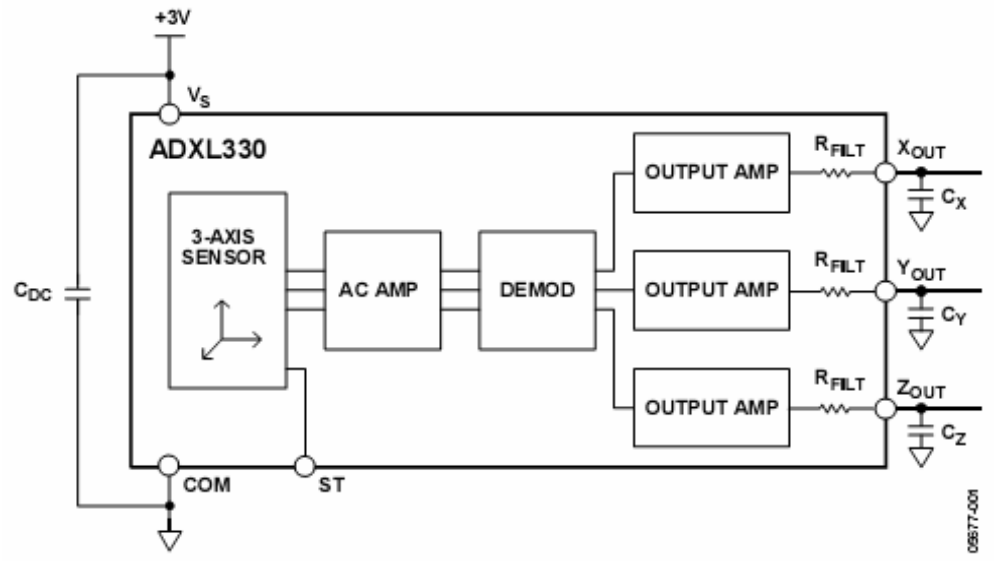


Fig 2.6 Functional block diagram of ADXL330

When the scooter is moving, the vibration of the chassis and handle bar look like noise and the sensor (an accelerometer) will give an output of the vibration acceleration which looks like noise. To get the real tilting angle, the signals from the accelerometer should have the constant of gravity subtracted and then be integrated to give angular velocity and then integrated again to give tilting angle.

3 CIRCUIT & SOFTWARE DESIGNS

3.1 Motor driver design

In this section, a high performance is needed because the driver system of the scooter has to deal with a relatively heavy load—a total weight of around 120kg for the rider and the scooter itself. The driver circuit should have the capability to cope with even higher a current when some situations happen. For example, when the motors are jammed, the driver has to deal with more than 120A peak current and a more than 40A continuous current.

3.1.1 H-Bridge MOSFETs control.

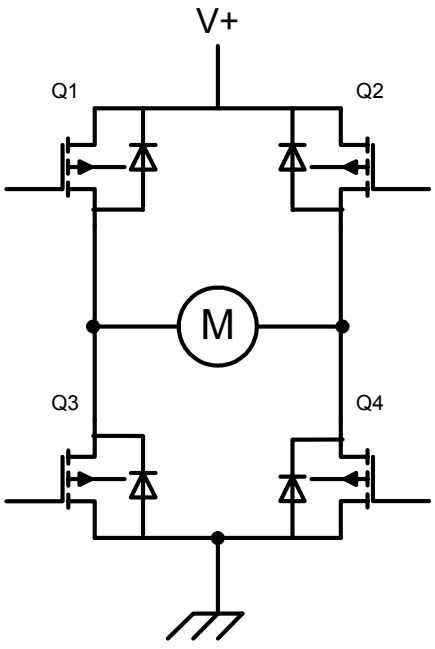


Fig. 3.1 Configuration of bidirectional PWM H Bridge

Fig 3.1 shows the basic layout of H-Bridge BDC motor (Brushed DC motor) bidirectional operation and variable speed control.

A four-MOSFET array forms an "H"; V+ is applied to the drains of both high side MOSFETs (Q1 & Q2), and ground connects to the sources of low side MOSFETs (Q3 & Q4). By switching these four MOSFETs ON and OFF, bidirectional current flow through the load can be produced. For example, when Q1, Q4 are on and Q2, Q3 are off, the current path is V+→ Q1→ Load→ Q4→ GND, current flows through the load from left hand side to right hand side; when Q2, Q3 are on and Q1, Q4 are off, the current path is V+→ Q2→ Load→ Q3→ GND, and current flows through the load from right hand side to left hand side.

Note that two FETs on one arm (Q1 and Q3 or Q2 and Q4) turned on simultaneously should be strictly avoided, because it would short circuit the power V+ to ground directly and produce a huge current which would kill the FETs. Furthermore, a dead time should be applied to prevent FETs on one arm switch on at the same time, because MOSFET need few nanoseconds to fully turn off from on-status. [5] The figure bellow illustrates how dead time control avoids this potential risk.

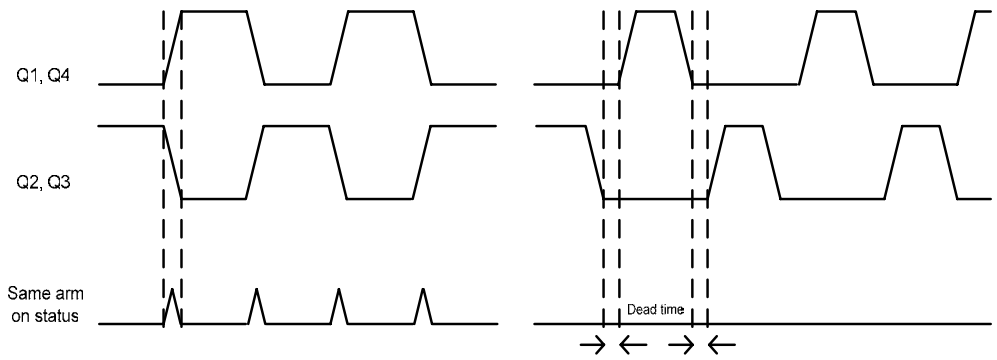


Fig. 3.2 Dead time control of H Bridge

3.1.2 Variable motor speed control.

Motor speed is controlled by the method which is called PWM (Pulse-width modulation). Basically it involves switching the FET on and off in relatively high frequency (normally more than 500 Hz, below this frequency may result resonance noise on motor), modulating the on/off ratio in every switching cycle. [6] This ratio is defined as duty cycle. Since this is implemented in a very short time (500 Hz means it has 500 on/off cycles per second), the applied voltage on the switching FET will be chopped into discrete periods. The average DC voltage equals the supply voltage times the duty cycle. It means if a 10V voltage is applied, changing the duty cycle from 0%~100% will get a 0~10V variable RMS DC voltage on the load.

Realize that there are two basic techniques in the H-Bridge PWM control. (1) Keep the high FET active and modulate the low FET or (2) Modulate the high FET and

keep the low one active. For example, when HL (high left) FET Q1 is being modulated, LR (low right) FET Q4 is active (turned on), Q2, Q3 keep off. In the other direction, the opposite control status would be used.

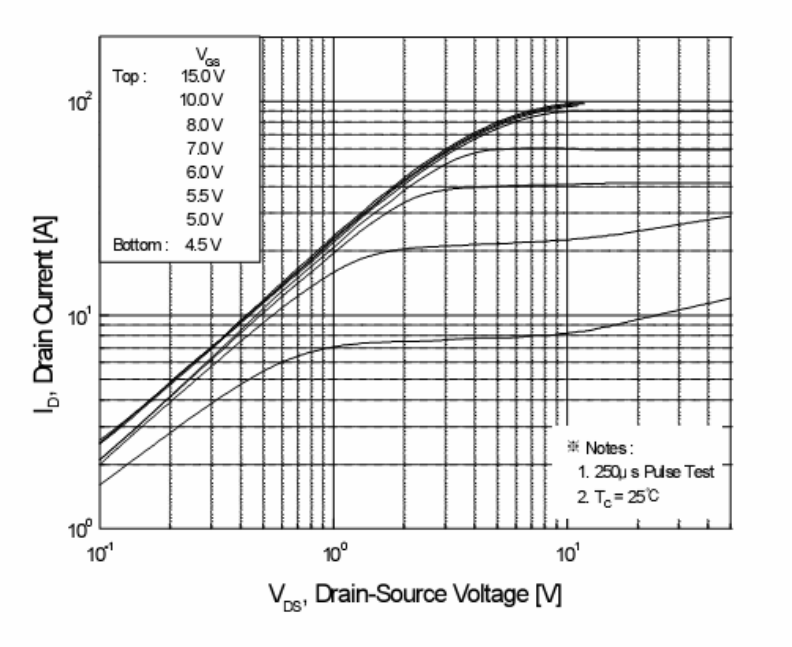


Fig. 3.3 MOSFET FQP32N12V2 on-region characteristic

An N-Channel MOSFET is fully turned on by meeting the requirement of V_{GS} is around 8V.[7] *Fig. 3.3* is the MOSFET FQP32N12V2 on-region characteristic from its datasheet. It shows that if 20A D-S current is needed (in this case, the rated current of the scooter's driving motor is 19A), V_{GS} should be more than 7V. This V_{GS} is to ensure the FET to turn on fully. However, top FETs of the H-Bridge can hardly do this when the load is inductive, especially the back electromotive force (Back-EMF) changes anytime when the motor is running.

Using P-channel MOSFET on the top side is an easy way to solve the problem above. Because the P-FET is turned on by giving a V_{GS} less than a certain minus value and setting V_{DD} constant (e.g. 12V, 24V…). When the FET is fully on, its R_{DS} is very small, so V_S is about equal to V_{DD}. However, normally P-channel MOSFET has a larger dissipated power (larger switching loss) than N-channel one. Also it has a limited rated power specification and a higher cost against N-channel MOSFET at the same rated power spec.

Therefore, in this project, four MOSFETs in the H-Bridge are all N-Channel. They are driven by an N-Channel MOSFET driver chip (Intersil ISL83202).[8] This driver chip provides a circuit to switch the N-MOSFET on for limited period of time (set by choosing different value of the bootstrap capacitor) on the top side of the H-Bridge. Fig 3.4 shows how this works.

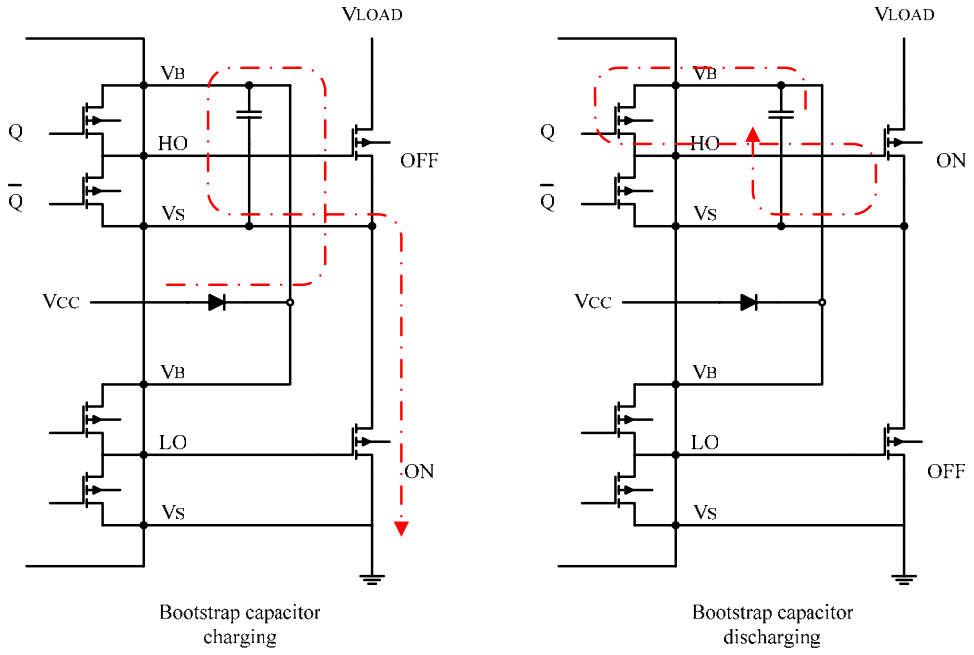


Fig. 3.4 Bootstrap capacitor charge and discharge routes

When the low side MOSFET is turned on, VCC charges the bootstrap capacitor (see the dotted-lines).

For the above reason, in this H-Bridge driver circuit two high side MOSFETs are always modulated because they can not be turned on constantly and both two bottom side MOSFETs are always turned either on or off when the motor is running in one direction, or the other.

Actually, the bootstrap cap is charged to V_{CC} ($V_+ = V_{CC}$) at the beginning, then the high side FET is turned on so the negative terminal of the capacitor is set to $V_- = V_{LOAD}$. Note that voltage across capacitor can not be changed in a short period, so the positive terminal of the capacitor is boosted up to $V_+ = V_{CC} + V_{LOAD}$. This voltage is enough to fully turn on the high FET.

3.1.3 Inrush and transient voltage.

A Bushed DC motor is an inductive load, and besides this, power cords from power supply to the circuit, wires on PCB board, etc., have inductance. In some circumstances this inductance can be fatal to MOSFETs.

We know the equation of voltage related to inductance is $V = L\frac{di}{dt}$.

H-Bridge FETs switch in relatively high frequency, so the main load drive circuit will be broken thousands of times in few seconds. While the circuit is broken, the

inductance of wires and motors do not allow the current stop, so the voltage would go higher and higher, if this voltage higher than the MOSFET limitation of V_{DS}, the FET will be blown up. Therefore current loops should be added for these voltage-sensitive MOSFETs when they switch from on state to off.

There are equivalent capacitors between MOSFET drain gate, gate source and drain source. Their capacitance are about 500pF.[6] Because this equivalent capacitance is small so even a small amount charge will result in a high voltage across the terminals($V=Q/C$). High voltage spikes will go across them and affect on the other side. No matter what V_{DS} specification of the MOSFET, the bearable V_{GS} is just ±20 V (some Gate Zener protected MOSFET has ±30 V rated, but it is still a low specification), so voltage peak come across the capacitor from drain will easily get the PN junction of gate-source broken down. Hence extra care must be taken in these cases. See *Fig 3.5* below.

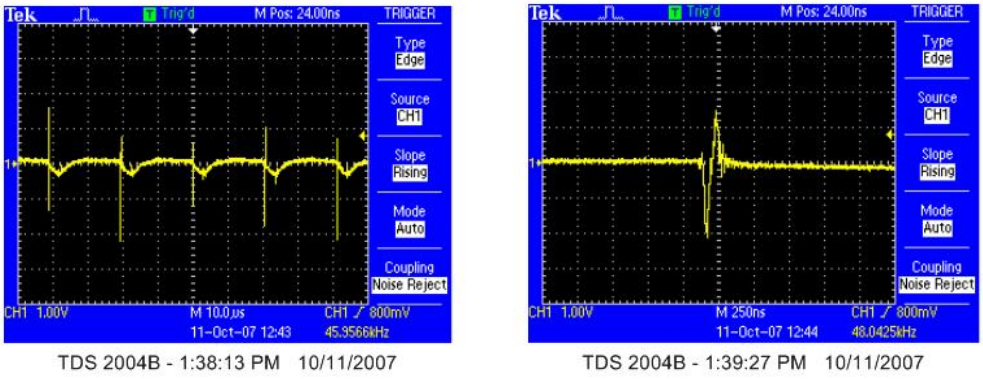


Fig 3.5(a) Spike captured by oscilloscope at the power line

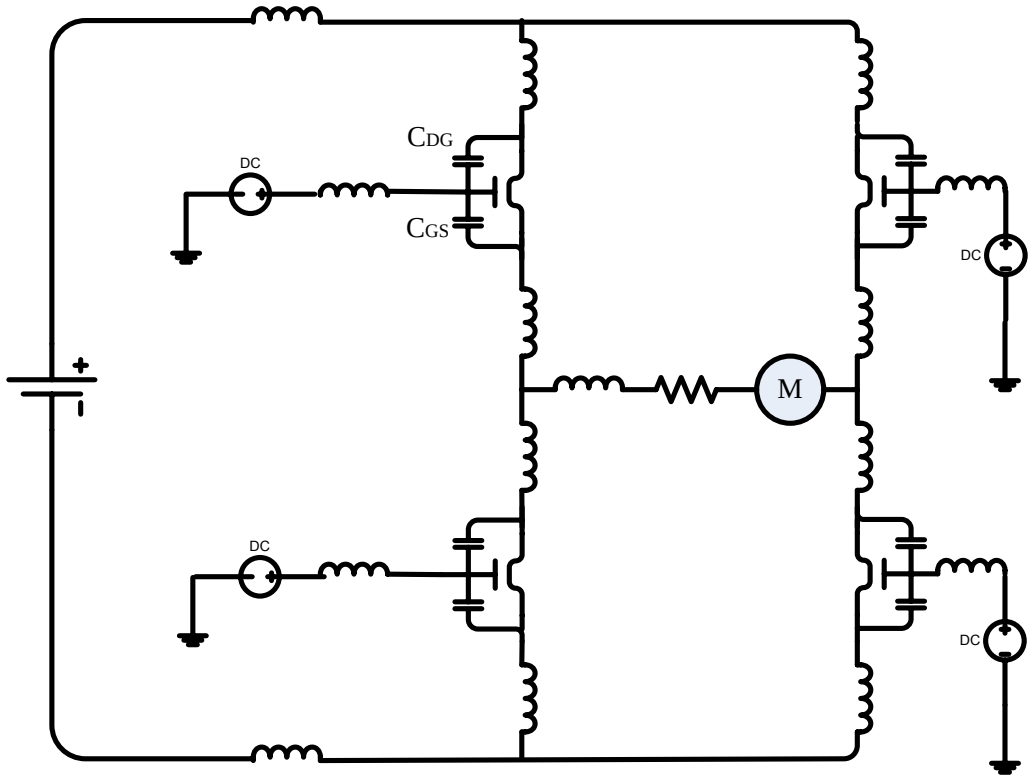


Fig 3.5(b) Equivalent circuit diagram considering

wiring inductance and MOSFET terminal capacitances.

Sudden changes in supply current as the motor current switches to freewheeling cause the spikes on supply at the Drain of the upper FETs. Spikes on Drain travel through C_{DG} to cause V_{GS} to exceed the allowed limit. Meanwhile, Gate inductances arise from the wiring to the FET; supply inductances arise from the wiring to the battery.

In the actual circuit, we use a tantalum capacitor to solve the high frequency spikes and another big capacitor (2200 µf) to deal with the low frequency spikes across the

power terminals. Both these two caps need to keep their leads short to reduce their series inductance. An RC snubber is used for the motor and RDC snubbers are used for D-S of MOSFETs. To protect the G-S junction of the MOSFETs, ensure V_{GS} < ±20 V, 15V Zener diodes are added across G-S terminals as well. (Refer to the circuit figure – See *Figure 3.10*)

3.1.4 BDC motor 4-quadrant operation.

A BDC motor has an inductance, and it stores motion energy when it is running. This makes the operation a bit complicated.

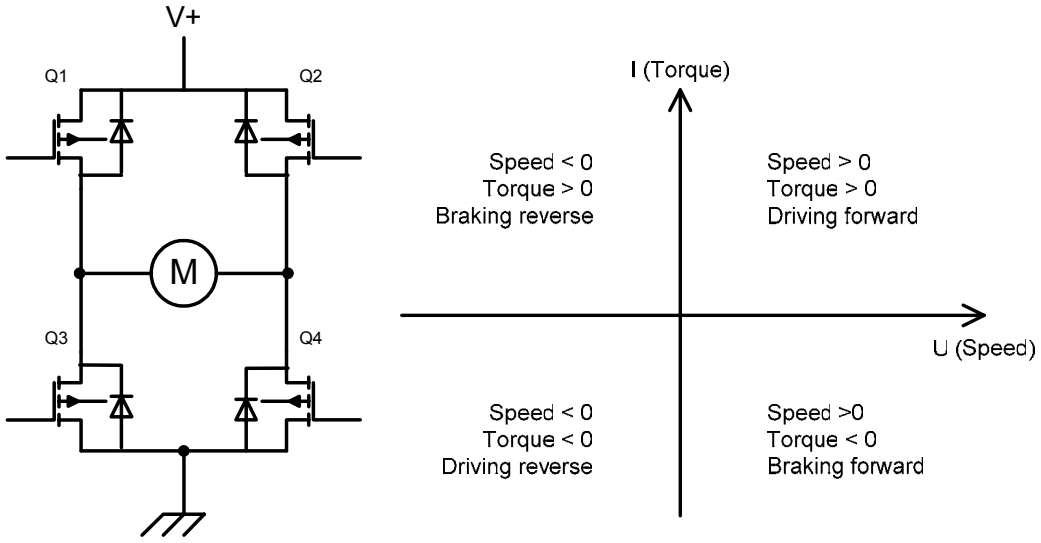


Fig 3.6 4 quadrants plane of Motor's operation

An H-Bridge bidirectional motor driver allows the motor to work in all 4 quadrants – forward driving mode, reverse driving mode, forward braking mode and reverse

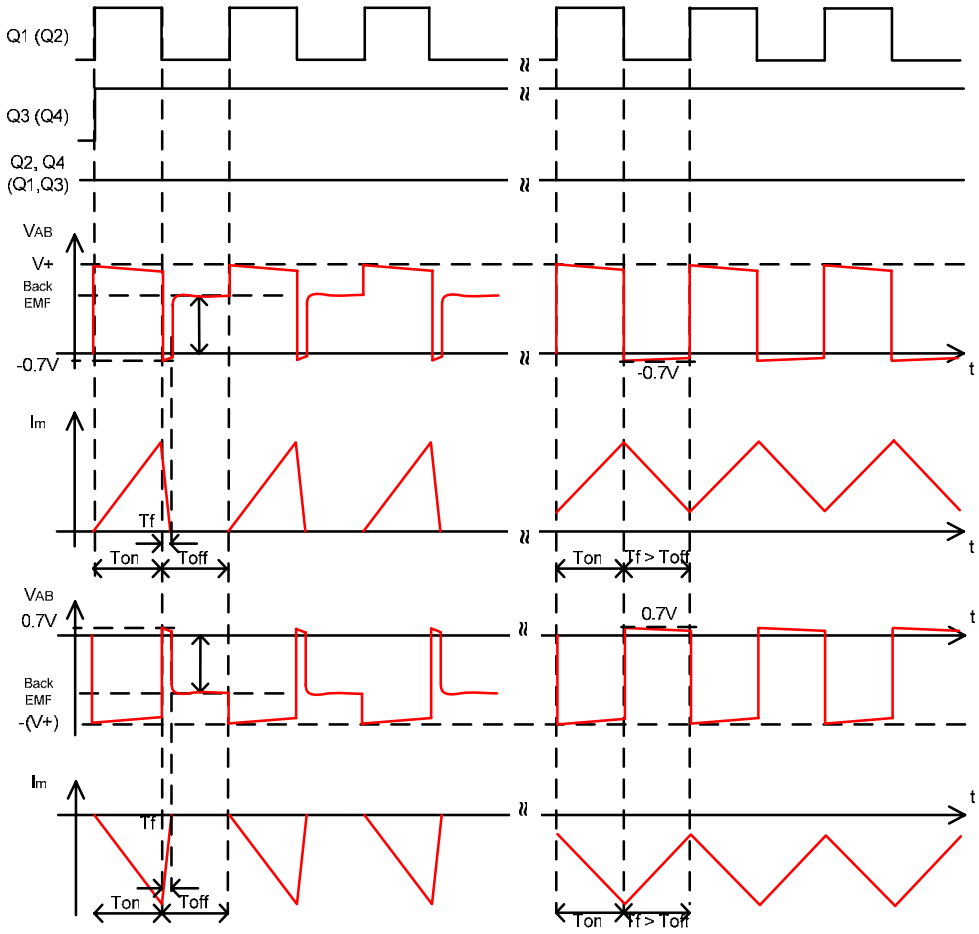
braking mode. These four quadrants are shown in the U-I (Speed-Torque) plane above (Fig 3.6).

In the 1st quadrant, the voltage across the motor's armature $V_m > 0$, the current goes through the armature $I_m > 0$, the motor is running in forward driving mode. When Q1 is on, current path is Q1—Armature—Q4; when Q1 is off, inductance of the armature does not allow the current to change suddenly, so the current will keep flowing in the same direction from the path: freewheel diode inherent in Q3—Armature—Q4 until the Q3 drain voltage is higher than -0.7 V (current discontinuous mode) or Q1 switches on again (current continuous mode). In current discontinuous mode, after the cathode of freewheel diode is higher than -0.7 V, since the motor is still spinning, a back EMF voltage can be seen across the motor's armature. However, in the current continuous mode, because freewheel diode is still in positive bias (the cathode is clamped at -0.7 V), Q1 switches on before this procedure finishes. This situation happens when the armature inductance is high or the Q1 switching frequency is high. Note the back EMF voltage can not be measured in this case.

In the 4th quadrant, $V_m > 0$, $I_m < 0$, and the motor is running in forward braking mode. The armature voltage is positive, meaning the motor is still running forwards (speed > 0), because of the rotating inertia of the motor. If at this moment the motor is told to change running direction, it would have a current going through

the opposite direction to brake the motion to slow down until the motor fully stops and then starts to go in the other direction.

In the 3rd quadrant, $V_m < 0$, $I_m < 0$, the motor is running in reverse driving mode; in the 2nd quadrant, $V_m < 0$, $I_m > 0$, the motor is running in reverse braking mode. They are quite similar to the circumstances of the first & the second quadrant respectively but the directions of current and voltage are opposite.



Discontinuous Mode **Continuous Mode**

Fig 3.7 Voltage and current waveforms of driving modes

(current discontinuous mode and current continuous mode)

Fig 3.7 illustrates how the motor works in the continuous mode or the discontinuous mode.

In the discontinuous mode, during T_{ON}, V_{DD} is applied to the motor directly, but due to the inductance of the motor, the current arises slowly; at the moment the MOSFET switches, the voltage of the motor is clamped at -0.7V because of the freewheeling period (in this period the energy store in the inductor is released as freewheeling current); while freewheeling period finishes, the MOSFET has not been switched on again, so the back EMF voltage is shown on the motor terminals.

In the continuous mode, while the MOSFET switches off, the motor voltage is clamped at -0.7V because of the freewheeling period, since the current of the motor can not be changed suddenly (due to the motor armature inductance), so it is still flowing, but the MOSFET switches on again before this period finishes, so V_{DD} is applied on the motor, and the motor current arises again. The motor works much more smoothly under the current continuous mode rather than the current discontinuous mode.

There are two more “different circumstances” while the motor is working in braking mode. (a) Braking with dissipating power And (b) Braking with generating power.

Table 3.1 shows different combinations in a clear way.

Octant	Description	Torque (armature current)	Speed (armature voltage)	Energy
I	FDD	+	+	-
II	FDG	+	+	+
III	FBD	-	+	-
IV	FBG	-	+	+
V	RDD	-	-	-
VI	RDG	-	-	+
VII	RBD	+	-	-
VIII	RBG	+	-	+

* Octant possibilities in shading blocks are impossible based on the law of conservation of energy.

Description abbreviation guide:

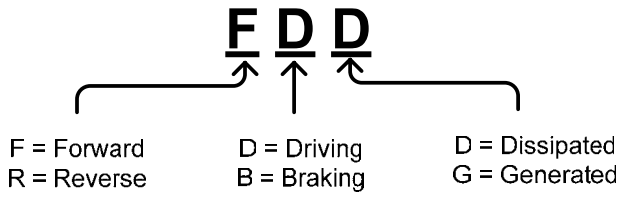


Table 3.1 Octant possibilities table for the motor running mode

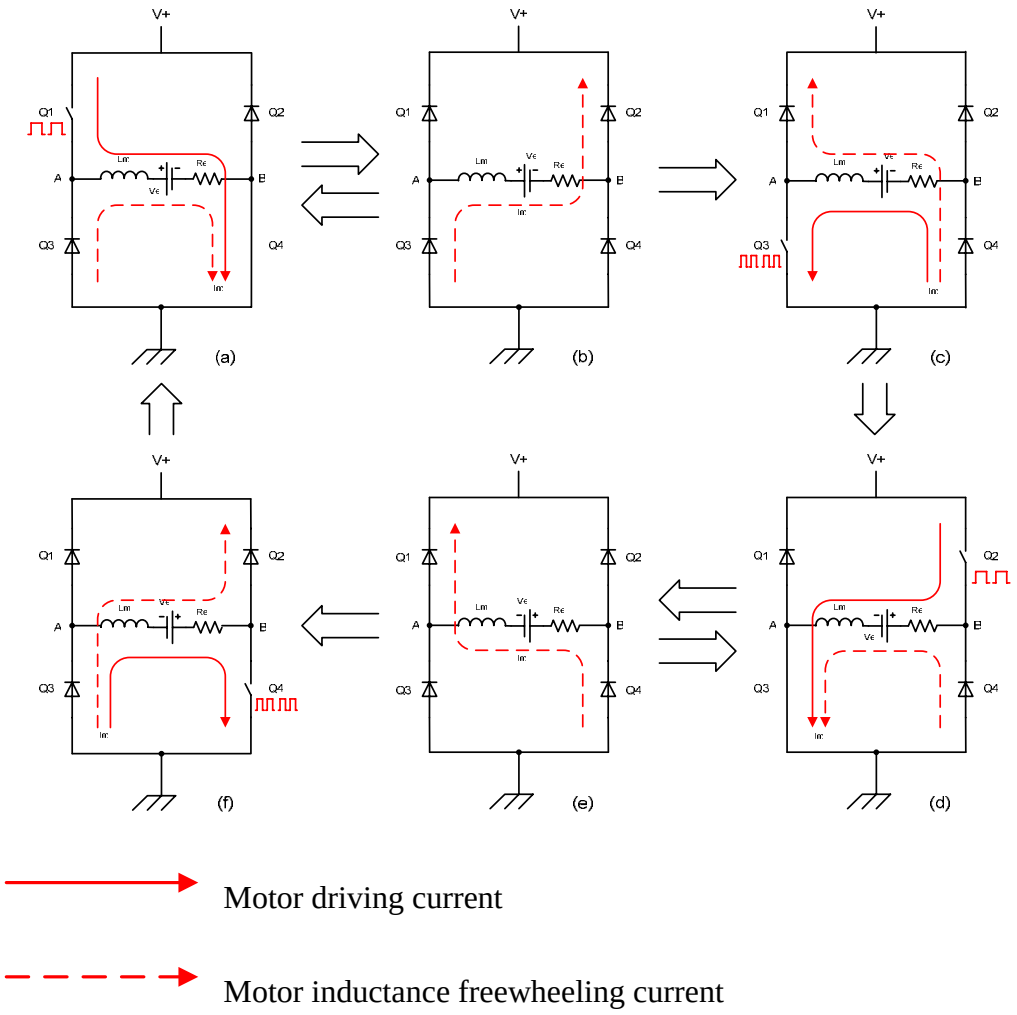


Fig 3.8 Motor current I_m descriptions in different modes

The motor drivers described in this paper follow the mode-shift as shown in *Fig 3.8*. *Fig 3.8 (a)* is the system running in FDD mode, it has driving current and freewheel current in every switch cycle, the motor stores energy as motion and dissipates power from the battery in this mode. If a stop command is sent in FDD mode (all 4 FETs are cut-off) then the system will shift to *Fig 3.8 (b)*, because the motor still has energy in the form of spinning, it will slow down gradually until it stops or

another signal tells it to go either back to forward or to backward before fully stopping.

It is very safe when the system shifts from *Fig 3.8 (b)* to *Fig 3.8 (a)* no matter how fast the motor is still going, because in the FDD mode it will speed up again (keep storing energy in the motor again). However, going to reverse is not that easy because of the motion inertia of the motor. If now switched into RDD mode directly, the energy the motor had stored will release suddenly and might blow the FET up because a huge current flows. To avoid this, FBG mode is added between these two.

There are two current paths in this state (See *Fig 3.8 (c)*). When Q3 switches on (Q1,Q2 & Q4 always keep off), a motor current is formed through inherited diode of Q4 – motor – Q3, energy stored in the motor now is dissipated as heat by the motor's equivalent resistor (R_m), and now the motor is running in FBD mode; when Q3 turns off this current continues because of the motor inductance and will go through Q4 diode – motor – Q1 diode, the energy now is pumped back to the power supply, the motor works as a generator, so it is running in FBG mode. The switching frequency of Q3 should be strictly fixed by the motor's characteristic (V_{BEMF}, L_m and R_m). If Q3 turns on too long, the current will increase dramatically (because low R_m) and might exceed the rated current of Q3, and Q3 would be damaged. Typically T_{ON} and T of Q3 are both μS class. Therefore here Q3's switching is controlled by PFM (Pulse Frequency Modulation) instead of PWM.

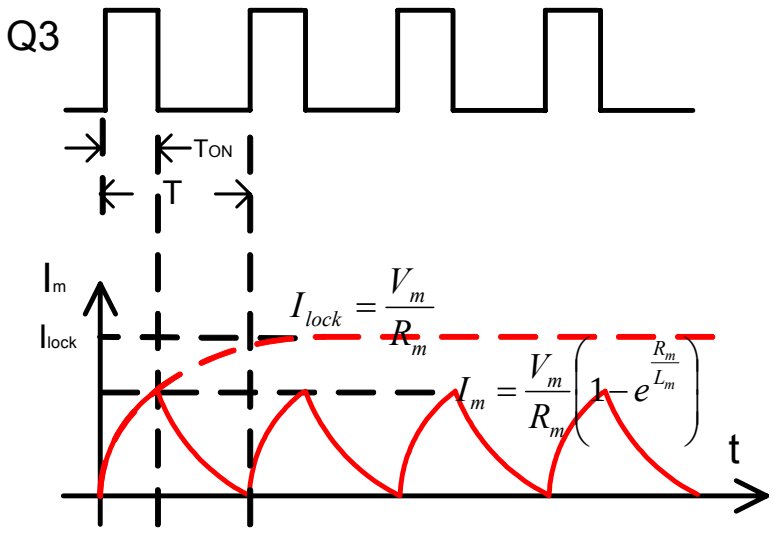
When the power is not enough to damage the FET directly, it is safe to shift the system from FBG/FBD to RDD mode (*Fig 3.8 (c)* to *Fig 3.8 (d)*).

The changes among RDD, RBG and FDD are similar to the situation above but in reverse.

Theoretically, there are another two situations will put the system in FBG and RBG modes. While the scooter is running down a big slope, because of gravity, it will run faster and faster, so the back EMF of the motor will be higher and higher and will eventually exceed the power supply voltage. At this moment, the motor will pump the energy it generated by gravitation back to the power supply. In *Fig 3.8 (a),* the current will go from the path Q4 diode – motor – Q1 diode to charge the battery. The same thing will happen in the state of *Fig 3.8 (d)*.

However, FDG and RDG modes will not happen because they break the law of conservation of energy.

Current waveform of the DC motor in FBG & RBG is shown in Fig 3.9.



$I_m = \frac{V_m}{R_m}\left(1 - e^{\frac{1}{\tau}}\right)$, time constant $\tau = \frac{L_m}{R_m}$,

where I_m *is motor's current,*

V_m *is motor's voltage,*

R_m *is motor's resistance and*

L_m *is motor's inductance.*

Fig 3.9 FBG & RBG current waveform

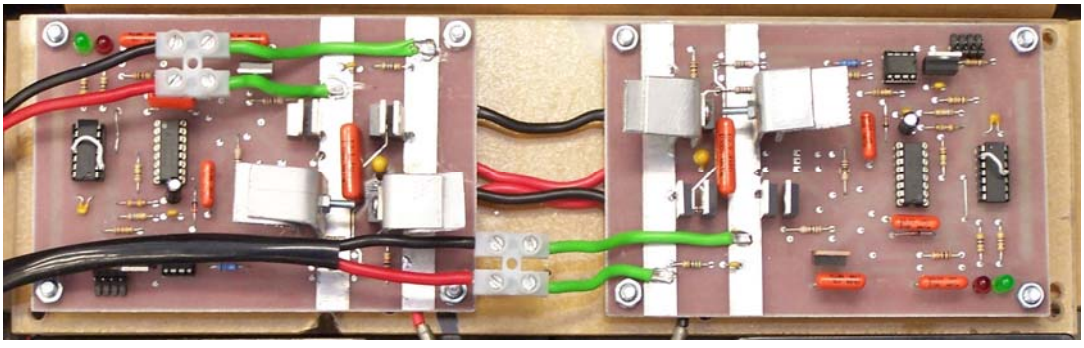


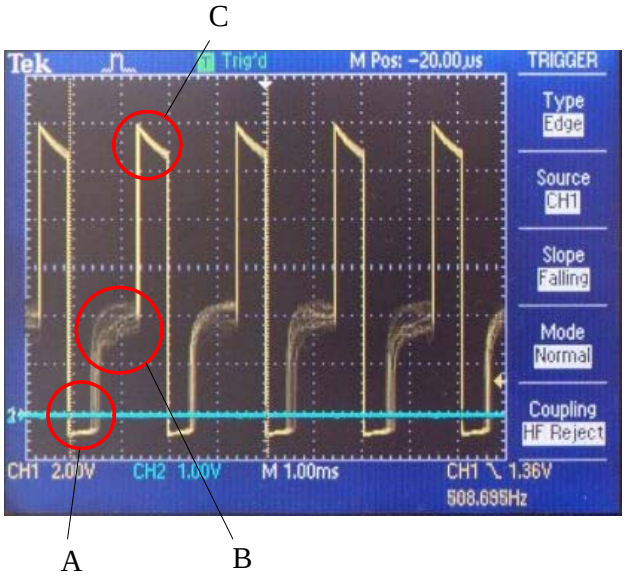
Fig 3.10 View of motor driving boards

Motor driving boards are shown in Fig 3.10 above.

3.2 Software

3.2.1 Motor drive and speed sample.

Each wheel is driven by a pulse-width modulated (PWM) H-Bridge which contains four MOSFETs. The switching frequency of these MOSFETs is 500 Hz (In this case). At this frequency the noise of the motors is not so obvious and the back EMF can still be measured. This is due to the motor's parameters—L_M, R_M. Different motors will have different characteristics, so the freewheeling periods will be different.



A – Freewheel time period

B – Back EMF shown period

C – MOSFET switch-on period

Fig 3.11 PWM output and back EMF voltage

Fig 3.11 shows the PWM output on the drain of the MOSFET which is being modulated. When the MOSFET is switched off, the inductance of the motor and wires maintains the current. This current is conducted by freewheel diodes in the MOSFETs between Drain and Source, therefore the voltage of the Drain is clamped at -0.7 V until the freewheeling period (450µS, part A in *Fig 3.4*) is over. At this

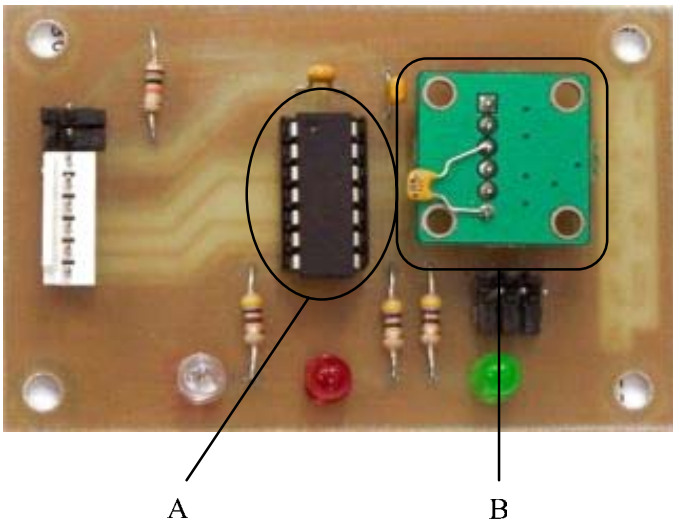
moment the motor is still running because of the motion inertia, so it is generating back EMF voltage. This back EMF voltage reflects the actual speed of the motor. If the MOSFET has not been switched on, the back EMF voltage can be detected by sampling through an ADC channel of the microchip; because the voltage appears on the motor terminals provided the system runs in a discontinuous mode.

Logic driver signals of the H-Bridge MOSFETs are provided by a microcontroller (Microchip PIC16F684). It receives signals from a Central processor as to how far and how fast the motor should go and generates driver pulses according to this signal. A driver IC deals with the pulses from the microchip and provides Gate drive level to MOSFETs and prevents two MOSFETs in the same side from turning on simultaneously. This driver IC also samples the current which is going through the motor and shuts down all MOSFETs if a serious situation occurs (overcurrent for a certain time period) to protect the MOSFETs and the motor. The microcontroller samples the speed signal (back EMF) of the motor and feeds this signal back to the central processor using a UART which will be described in *subsection 3.3*.

3.2.2 Central controller.

The central controller is another microchip PIC16F684. It processes the balancing signals from the accelerometer through the analogue to digital conversion channel, and sends instruction of how fast, how far the motors go (to keep balance) to the motor driver processor. It also deals with the steering signal from the handle bar and decides what the speed difference is between two motors (to steer). To avoid an

accident, the central controller would shut all the motor running outputs down if the scooter tilts over 45°. Meanwhile, it takes care of the speed sample and limits the highest speed.



A – Central microprocessor

B – Accelerometer

Fig 3.12 Accelerometer and Central processor

Fig 3.12 shows the board which holds the microprocessor and the tilting sensor, a three axis accelerometer.

3.3 Algorithm

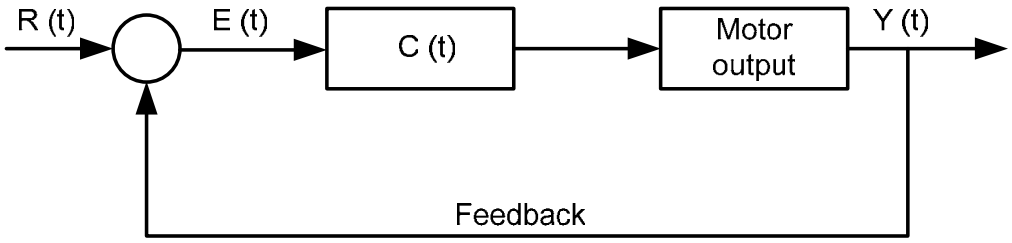
3.3.1 PID control.

The scooter and rider system is similar to a single stage inverted pendulum model. This is a complicated multi-variable, fast-responding, nonlinear and unstable system. In this case the scooter is expected to work with three inputs (tilting angle, tilting angular velocity and motor speed) and one output (the balance of the scooter—controlled directly by the motor torque).

Instead of keeping the centre of gravity upright to the wheels, the scooter keeps the tilting angle of the chassis to zero.

Once the centre of gravity moves forward or backward and the scooter is out of balance, the tilting angle will not be zero anymore. If the wheels move in the corresponding direction to reduce the angle at this moment, the tilting angle will go back towards 0°.

The loop is closed using a Proportional-Integral-Derivative (PID) feedback control is applied to this control loop. This is a relatively easy way to get the system work. The system block is shown as below.



$R(t) = Reference\ of\ desired\ set\ point$

$Y(t) = Measured\ response$

$E(t) = Error$

$C(t) = Controller\ response$

Fig 3.13 System block diagram

In order to keep the system stable, the desired set point R(t) (which is the difference between tilting angle and 0°) should always be supposed to be zero, so the controller's transfer function can be written:

$$C(t) = K_P E(t) + K_I \int_0^t E(t)dt + K_D \frac{dE(t)}{dt} \qquad (1)$$

After the linearization of the differential equation with *Laplace* transform, setting the constant of K_P, K_I and K_D (especially K_P & K_D) appropriately, the system will stay stable even in dynamic situations.

Furthermore, Y(t) is the tilting angle sampled from the accelerometer through an ADC channel. The sampled result is digital. Therefore, the transfer function must

be converted into a digital form which can be implemented in the microchip. In order to do this, some approximations of integral and derivative terms are done.

Based on the approximate slope of the tangent line at E (t) (rise/run), the approximations regarding the integral and derivative terms can be written as follow:

$$\int_0^t E(t)dt \approx T_S \sum_0^N E(n) \qquad (2)$$

$$dE(t)/dt \approx [E(n) - E(n-1)]/T_S \qquad (3)$$

where $E(n)$ is the current error, $E(n-1)$ is the pervious error and T_S is the sampling period.

Now we can rewrite equation (1):

$$C(n) = K\{E(n) + (T_S/T_I)\sum_0^N E(n) + (T_D/T_S)[E(n) - E(n-1)]\} \qquad (4)$$

where $K_P = K, K_I = \frac{K}{T_I} and K_D = KT_D.$

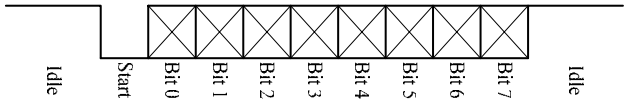
Equation (4) is the digital transfer function to be used for the PID control of the system.

Besides the three inputs which are mentioned above, there is another input in this balancing system – steering control signal. This signal is from the handle bar's swing. It is out of the main control loop so it does not affect the system balance.

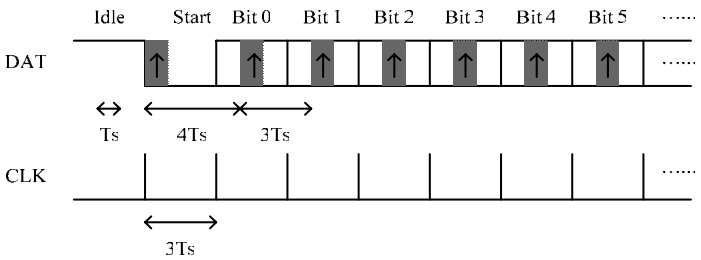
Steering signal can be delivered by an offset VR from a range of 0~5V, then collected by the microchip via an ADC channel. In this case, the Y-axis of the accelerometer is used for sensing the steering signal. Stick the accelerometer on the handle bar and swing it left and right, a 1.8~3.2V DC range signal can be got. When the handle bar is fully upright, the reading of this signal is 2.5V DC. Sampled by the ADC channel and scaled to -1~+1, proportioned to the left and right wheel drivers in the microprocessor, so two wheels would have different speed, then the steering can be controlled.

3.3.2 Universal asynchronous receiver/ transmitter (UART).

Microchip PIC16F684 is a low cost micro processor, it does not provide any hard ware PCI or UART communication functions, nor has any digital to analogue conversion channel. In order to establish the communication between the central processor and two motor driver processors, building a software universal asynchronous communication is necessary. Triple sampling method is a popular solution which is prompt and reliable.



RS data structure



Triple sampling method

Fig 3.14 RS data string structure

Fig 3.14 shows the data string structure and how the triple sampling method works. Data string is constructed by 1 start bit and 8 data bits. Idle status is a normal high level, start bit is active low and it will last for *3*Ts* by the transmitter; whenever the receiver captures the start bit, it begins to time, after *4*Ts* it will shift into data transferring status and sample the data bit every *3*Ts* until all data bits were sent. This is to ensure the receiver collects data exactly on the 1/3 of every data bits and minimize the chance of getting error data.

Using state machine can easily implement this method in microprocessor. (See the state machine in *Fig 3.15*)

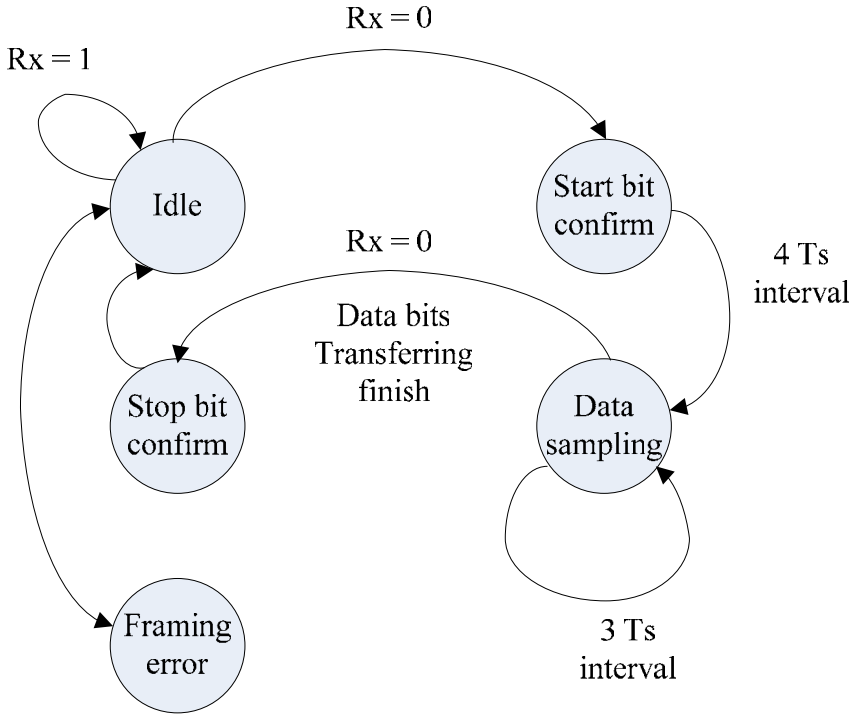


Fig 3.15 State machine of the RS transferring

In this project I managed to complete this communication by using two digital I/O pins (data pin & clock pin) and 1 internal timer. This part of the code only takes 12~15% of the CPU running resources and the baud rate can be easily promoted to 14.5K bit/S (CPU running at 8MHz).

3.3.3 Steering control

The steering of the scooter is implemented by making one wheel goes faster than the other. For example, if the left wheel goes at 60% full power forward and the right goes at 40% full power forward, the scooter will turn right; when it is not moving, if the left wheel goes 10% full power forward and the right wheel goes 10% full power backward, the scooter will quickly turn around at the same place.

4. SYSTEM MATHEMATICAL MODELLING

4.1 Linear BDC mathematical model

The linear space state model of the brushed direct current motor is derived in this section. This model is used to get the relationship of the applied voltage of two motors (system input from the balancing sensor) and the applied torque of two wheels (system output).

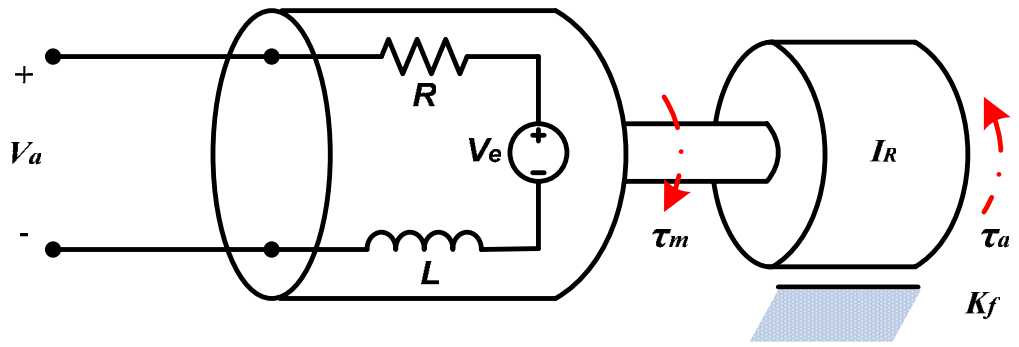


Fig 4.1 Equivalent circuit of a BDC motor

A BDC motor can be shown as its equivalent circuit (Fig 4.1).

The torque which is generated by the motor τ_m is proportional to the current going through the armature.

$$\tau_m = k_m i \tag{4.1}$$

where K_m is the constant of the motor torque.

The EMF voltage V_e of the motor is proportional to the angular velocity of the motor spin.

$$V_e = k_e \omega \tag{4.2}$$

where k_e is the constant of the back EMF

ω is the angular velocity of the motor

According to Ohm's Law & Kirchoff's Voltage Law, the sum voltage of the circuit loop must be zero, so it can be expressed as

$$V_a - iR - k_e \omega - L\frac{di}{dt} = 0 \tag{4.3}$$

so

$$\frac{di}{dt} = \frac{V_a}{L} - \frac{R}{L} i - \frac{k_e}{L} \omega \tag{4.4}$$

Newton's law of motion states that the sum of all forces applied on the shaft is linearly related to the acceleration of the shaft multiplied by the inertia load I_R. Therefore the preceding statement can be written as

$$\sum M = I_R \theta'' = \tau_m - k_f \omega - \tau_a \tag{4.5}$$

where I_R is the motor inertia constant

τ_a is the applied torque of the motor

k_f is the friction constant

Rewrite equation (4.5) as

$$\frac{d\omega}{dt} = \frac{k_m}{I_R} i - \frac{k_f}{I_R} \omega - \frac{\tau_a}{I_R} \tag{4.6}$$

To get the simplified linear function of this model, assume that the motor friction and motor inductance is considered negligible. Hence $L = 0$, $K_f = 0$. Equation (4.3) and (4.6) can be rewritten as

$$i = \frac{V_a}{R} - \frac{k_e}{R} \omega \tag{4.7}$$

$$\frac{d\omega}{dt} = \frac{k_m}{I_R} i - \frac{\tau_a}{I_R} \tag{4.8}$$

Substituting equation (4.7) & (4.8), the relationship of the motor current speed, applied voltage and applied torque is obtained as the function below:

$$\frac{d\omega}{dt} = \frac{k_m}{I_R R} V_a - \frac{k_m k_e}{I_R R} \omega - \frac{\tau_a}{I_R} \tag{4.9}$$

The motor's dynamic model can be represented with a state space expression.

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{k_m k_e}{I_R R} \end{bmatrix} \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{k_m}{I_R R} & -\frac{1}{I_R} \end{bmatrix} \begin{bmatrix} V_a \\ \tau_a \end{bmatrix} \tag{4.10}$$

where θ is the angular position, ω is the angular velocity.

Applied voltage and applied torque of the DC motor are inputs of this model. This is a first order differential equation.

4.2 Model of two-wheeled inverted pendulum

4.2.1 Model of two-wheels

The scooter system is very similar to a two wheeled inverted pendulum, which behaves similarly to pendulum on a cart. In this section two motion equations are derived by the pendulum and wheel dynamics separately.

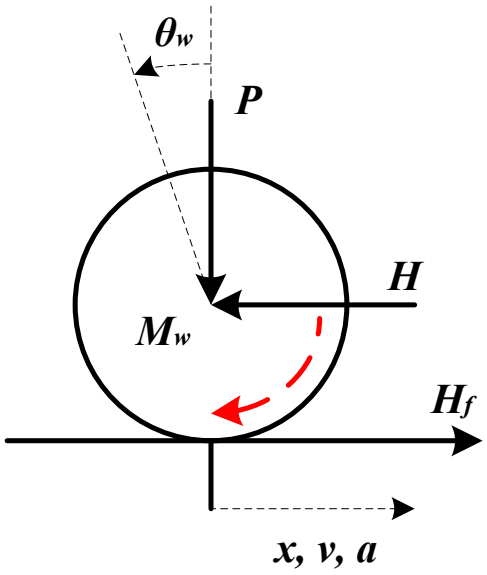


Fig 4.2 Free body diagram of the wheel

Due to Newton's law of motion, the sum of forces on the horizontal x direction is

$$\sum M = M_w a = M_w \ddot{x} = H_f - H \qquad (4.11)$$

where M_w is the mass of the wheel

H is the horizontal force and

H_f is the horizontal friction.

Sum of forces around the wheel centre is

$$\sum M = I_w a = I_w \ddot{\theta}_w = C - H_f \cdot r \tag{4.12}$$

where I_w is the inertia of the wheel

r is the radius of the wheel and

C is the applied torque from the motor to the wheel.

The motor torque is expressed as equation (4.13) from DC motor dynamics.

$$\tau_m = I \frac{d\omega}{dt} \tag{4.13}$$

According to the equation (4.9) form the DC motor derivation section, the output torque of the wheel is obtained.

$$C = I \frac{d\omega}{dt} = \frac{-k_m k_e}{R} \dot{\theta}_w + \frac{k_m}{R} V_a \tag{4.14}$$

Combining (4.12) and (4.14) can be rewritten as

$$I_w \ddot{\theta}_w = \frac{-k_m k_e}{R} \dot{\theta}_w + \frac{k_m}{R} V_a - H_f \cdot r \tag{4.15}$$

Thus,

$$H_f = \frac{-k_m k_e}{Rr}\dot{\theta}_w + \frac{k_m}{Rr}V_a - \frac{I_w \ddot{\theta}_w}{r} \tag{4.16}$$

Substitute equation (4.16) into (4.11), assume that two wheels and two motors are identical, dynamics of the left wheel and right wheel are acquired.

For the left wheel,

$$M_w \ddot{x} = \frac{-k_m k_e}{Rr}\dot{\theta}_w + \frac{k_m}{Rr}V_a - \frac{I_w \ddot{\theta}_w}{r} - H_L \tag{4.17}$$

For the right wheel,

$$M_w \ddot{x} = \frac{-k_m k_e}{Rr}\dot{\theta}_w + \frac{k_m}{Rr}V_a - \frac{I_w \ddot{\theta}_w}{r} - H_R \tag{4.18}$$

Transfer the angular system to the X-Y plane system,

$$\ddot{\theta}_w r = \ddot{x} \Rightarrow \ddot{\theta}_w = \frac{\ddot{x}}{r}$$

$$\dot{\theta}_w r = \dot{x} \Rightarrow \dot{\theta}_w = \frac{\dot{x}}{r}$$

By this linear transformation, equation (4.17) and (4.18) can be rewritten as:

For the left wheel,

$$M_w \ddot{x} = \frac{-k_m k_e}{Rr^2}\dot{x} + \frac{k_m}{Rr}V_a - \frac{I_w \ddot{x}}{r^2} - H_L \tag{4.17}$$

For the right wheel,

$$M_w\ddot{x} = \frac{-k_m k_e}{Rr^2}\dot{x} + \frac{k_m}{Rr}V_a - \frac{I_w\ddot{x}}{r^2} - H_R \qquad (4.18)$$

Add equation (4.17) & (4.18) to acquire the system total horizontal force by two wheels,

$$2\cdot\left(M_w + \frac{I_w}{r^2}\right)\ddot{x} = \frac{-2k_m k_e}{Rr^2}\dot{x} + \frac{2k_m}{Rr}V_a - (H_L + H_R) \qquad (4.19)$$

4.4.2 Model of the inverted pendulum

The scooter's chassis and the rider can be modeled as an inverted pendulum. *Fig 4.3* shows the free body diagram of this.

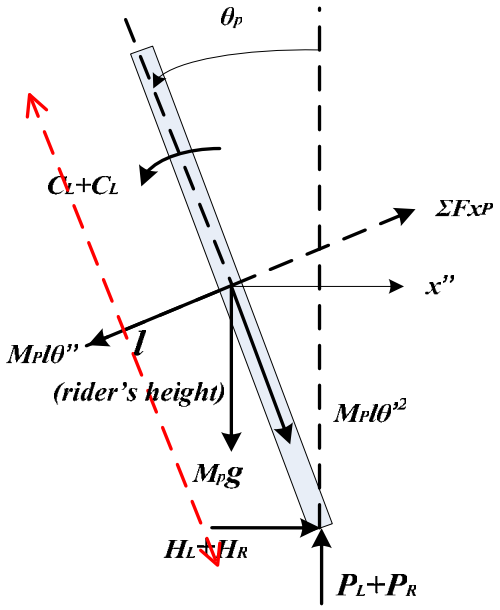


Fig 4.3 Free body diagram of the inverted pendulum

Applying Newton's Motion Law again, the sum of the horizontal direction forces of the pendulum is

$$\sum F_x = M_P \ddot{x}$$

$$M_P \ddot{x} = (H_L + H_R) - M_P l \ddot{\theta}_P \cos\theta_P + M_P l \dot{\theta}_P{}^2 \sin\theta_P \qquad (4.20)$$

thus,

$$H_L + H_R = M_P \ddot{x} + M_P l \ddot{\theta}_P \cos\theta_P - M_P l \dot{\theta}_P{}^2 \sin\theta_P \qquad (4.21)$$

where M_P is the mass of the scooter and the rider system,

l is the pendulum height (rider's height) from chassis

θ is the angle of the chassis and the horizontal line

The sum of the perpendicular direction forces of the pendulum is

$$\sum F_{xP} = M_P \ddot{x} \cos\theta_P$$

$$M_P \ddot{x} \cos\theta_P = (H_L + H_R)\cos\theta_P + (P_L + P_R)\sin\theta_P - M_P g \sin\theta_P - M_P l \ddot{\theta}_P \qquad (4.22)$$

The sum of the motion around the centre of mass of the pendulum,

$$\sum M_C = I\alpha$$

$$I_P \ddot{\theta}_P = -(H_L + H_R) l \cos\theta_P - (P_L + P_R) l \sin\theta_P - (C_L + C_R) \qquad (4.23)$$

where I_P is the inertia of the system (scooter + rider),

P_L, P_R are reaction forces between wheels and chassis,

C_L, C_R are applied torques from motors.

From equation (4.14), the applied torque from motors after linearization is

$$C_L + C_R = 2C = \frac{-2k_m k_e \dot{x}}{Rr} + \frac{2k_m V_a}{R} \tag{4.24}$$

Substituting equation (4.24) into (4.23),

$$I_P \ddot{\theta}_P = -(H_L + H_R) l \cos\theta_P - (P_L + P_R) l \sin\theta_P + \frac{2k_m k_e \dot{x}}{Rr} - \frac{2k_m V_a}{R} \tag{4.25}$$

thus,

$$I_P \ddot{\theta}_P - \frac{2k_m k_e \dot{x}}{Rr} + \frac{2k_m V_a}{R} = -(H_L + H_R) l \cos\theta_P - (P_L + P_R) l \sin\theta_P \tag{4.26}$$

Multiply equation (4.22) by $-l$,

$$-M_P \ddot{x} l \cos\theta_P = -(H_L + H_R) l \cos\theta_P - (P_L + P_R) l \sin\theta_P + M_P g l \sin\theta_P + M_P l^2 \ddot{\theta}_P \tag{4.27}$$

Substituting equation (4.27) into (4.26),

$$I_P \ddot{\theta}_P - \frac{2k_m k_e \dot{x}}{Rr} + \frac{2k_m V_a}{R} = -M_P \ddot{x} l \cos\theta_P - M_P g l \sin\theta_P - M_P l^2 \ddot{\theta}_P \tag{4.28}$$

Substituting equation (4.21) into (4.19) to eliminate (H_L+H_R),

$$2\left(M_w + \frac{I_w}{r^2}\right)\ddot{x} = \frac{-2k_m k_e}{Rr^2}\dot{x} + \frac{2k_m}{Rr}V_a - M_P\ddot{x} - M_P l\ddot{\theta}_P\cos\theta_P + M_P l\dot{\theta}_P{}^2\sin\theta$$

(4.29)

Rearrange equation (4.28) & (4.29), then the non-linear equation of the system motion is given as below,

$$(I_P + M_P l^2)\ddot{\theta}_P - \frac{2k_m k_e \dot{x}}{Rr} + \frac{2k_m V_a}{R} + M_P\ddot{x}l\cos\theta_P = -M_P gl\sin\theta_P$$

(4.30)

$$\left(2M_w + M_P + \frac{2I_w}{r^2}\right)\ddot{x} + \frac{2k_m k_e}{Rr^2}\dot{x} + M_P l\ddot{\theta}_P\cos\theta_P - M_P l\dot{\theta}_P{}^2\sin\theta = \frac{2k_m}{Rr}V_a$$

(4.31)

Assume that $\theta_P = \pi + \varphi$, where φ represents a small angle from vertical upright direction, these two equations can be linearised.

$$\cos\theta_P = -1, \quad \sin\theta_P = -\varphi \quad and \quad \left(\frac{d\theta_P}{dt}\right)^2 = 0.$$

This simplification was used to enable a linear model to be contained so linear state space controllers could be implemented.

Therefore, the linear equations of the motion are:

$$(I_P + M_P l^2)\ddot{\varphi} - \frac{2k_m k_e}{Rr}\dot{x} + \frac{2k_m}{R}V_a - M_P gl\varphi = M_P l\ddot{x} \qquad (4.32)$$

$$\left(2M_w + M_P + \frac{2I_w}{r^2}\right)\ddot{x} + \frac{2k_m k_e}{Rr^2}\dot{x} - M_P l\ddot{\varphi} = \frac{2k_m}{Rr}V_a \tag{4.33}$$

Rearrange equation (4.32) and (4.33) to get the state space representation of the system,

$$\ddot{\varphi} = \frac{M_P l}{(I_P + M_P l^2)}\ddot{x} + \frac{2k_m k_e}{(I_P + M_P l^2)Rr}\dot{x} + \frac{-2k_m}{(I_P + M_P l^2)R}V_a + \frac{M_P gl}{(I_P + M_P l^2)}\varphi \tag{4.34}$$

$$\ddot{x} = \frac{2k_m}{\left(2M_w + M_P + \frac{2I_w}{r^2}\right)Rr}V_a + \frac{-2k_m k_e}{\left(2M_w + M_P + \frac{2I_w}{r^2}\right)Rr^2}\dot{x} + \frac{M_P l}{\left(2M_w + M_P + \frac{2I_w}{r^2}\right)}\ddot{\varphi} \tag{4.35}$$

Combine and substitute equations (4.32), (4.33), (4.34) and (4.35), to obtain the system state space equation is obtained:

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\varphi}\\ \ddot{\varphi}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \frac{2k_m k_e(M_P lr - I_P - M_P l^2)}{Rr^2\alpha} & \frac{M_P{}^2 gl^2}{\alpha} & 0\\ 0 & 0 & 0 & 1\\ 0 & \frac{2k_m k_e(r\beta - M_P l)}{Rr^2\alpha} & \frac{M_P gl\beta}{\alpha} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \varphi\\ \dot{\varphi}\end{bmatrix} + \begin{bmatrix}0\\ \frac{2k_m(I_P + M_P l^2 - M_P lr)}{Rr\alpha}\\ 0\\ \frac{2k_m(M_P l - r\beta)}{Rr\alpha}\end{bmatrix}V_a \tag{4.36}$$

where

$$\alpha = I_P\beta + 2M_P l^2(M_w + \frac{I_w}{r^2})$$

$$\beta = 2M_w + \frac{2I_w}{r^2} + M_P$$

This model of the system assumes that the wheels of the scooter will always stay in contact with ground and there is no slip at the wheels. Cornering forces are also considered negligible.

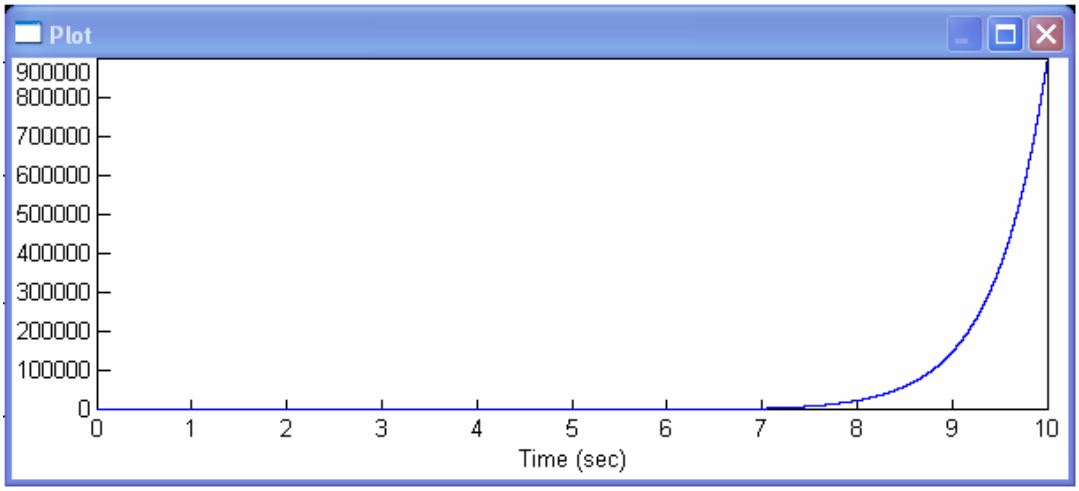
In this model, the definitions of parameters are shown in *Table 4.1*. All parameters are calculated and measured based on *Appendix I.*

Parameter	**Definition**	**Value in this case**
x	*Distance*	-
$\dot{x}$	*Speed*	-
$\ddot{x}$	*Acceleration*	-
φ	*Error angle*	-
$\dot{\varphi}$	*Error angular speed*	-
$\ddot{\varphi}$	*Error angular acceleration*	-
k_m	*Constant of the motor torque*	*0.069N/A*
k_e	*Constant of the motor's back-EMF*	*0.083V/rad.*
l	*Length of the pendulum*	*1.8 m*
r	*Wheel radius*	*200 mm*
R	*Resistance of the Motor*	*1.0 Ω*
M_P	*Mass of the pendulum*	*120 kg*
I_P	*Moment of inertia of the pendulum*	*87.67kg·m²*
M_w	*Mass of the wheel*	*3.5 kg*
I_w	*Inertia of the wheel*	*0.07kg·m2*
g	*Acceleration of gravity*	*9.8 m/S2*

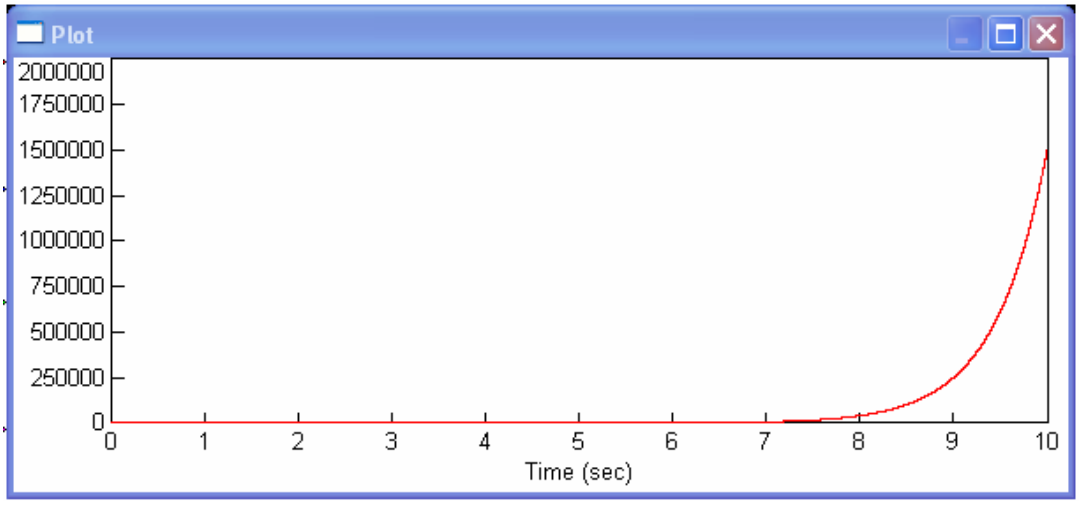
Table 4.1 Parameter definitions

5. CONTROLLER SIMULATION

An inverted pendulum is an inherently unstable system, so its open-loop system will result in tilting angle and velocity going unbounded when an input is applied. *Fig 5.1* shows the response to a small input of the uncontrolled system.



Velocity Plot



Tilting angle

Fig 5.1 Response to a small input of open-loop system of the scooter

To make a simple feedback control: Get the tilting angle, θ, compare it with the reference value (inverted pendulum straight up, tilting angle $\theta = 0°$), get the error and compensate the system input. The system root locus, is shown in *Fig 5.2*.

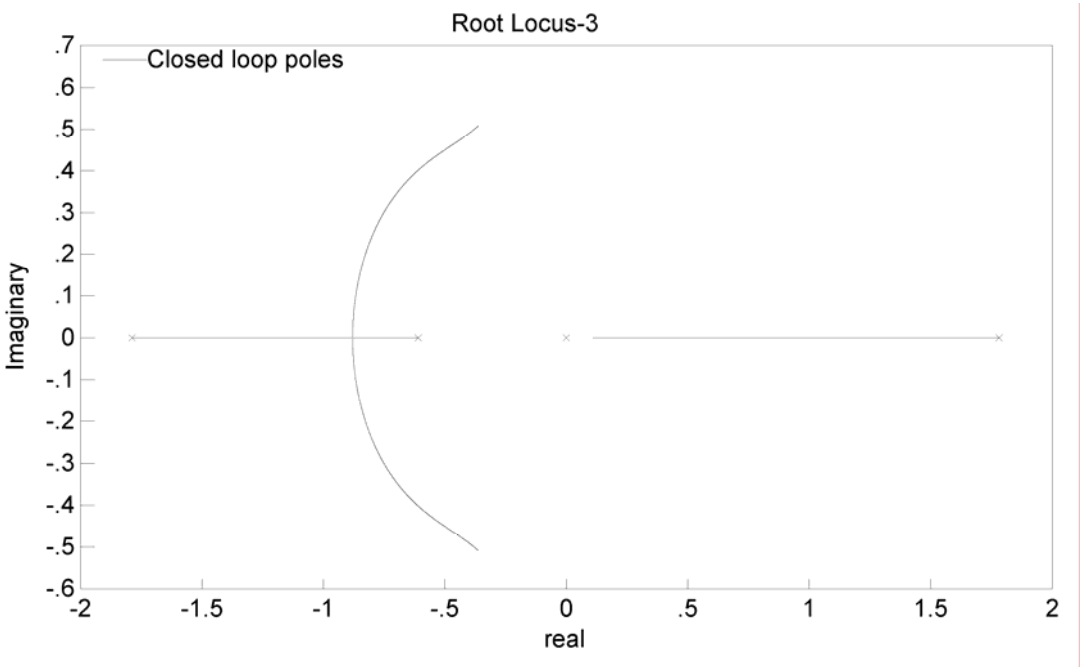


Fig 5.2 Uncontrolled system root locus

We can see 4 poles in the figure, two are in the left hand side of the imaginary axis, one is about zero and the last one (around +1.8 at real axis) is in the right hand side of the s-plane which makes the system unstable.

Now we add a PID controller to the system. PID coefficients are set to $K_P = 200$, $K_I = 0$ and $K_D = -50$. The closed loop root locus is plotted in *Fig 5.3*.

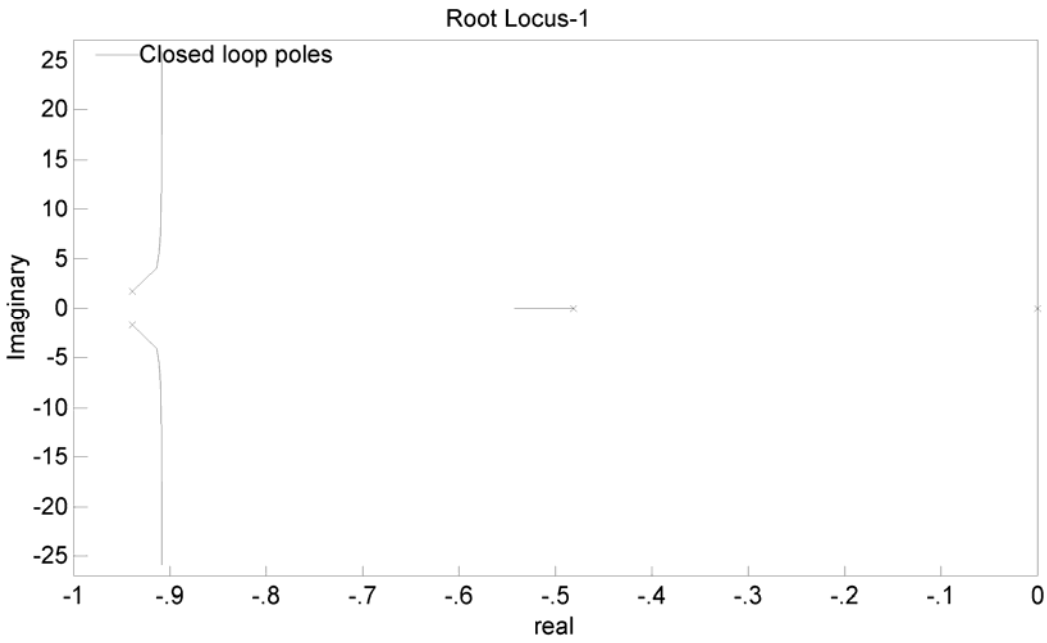
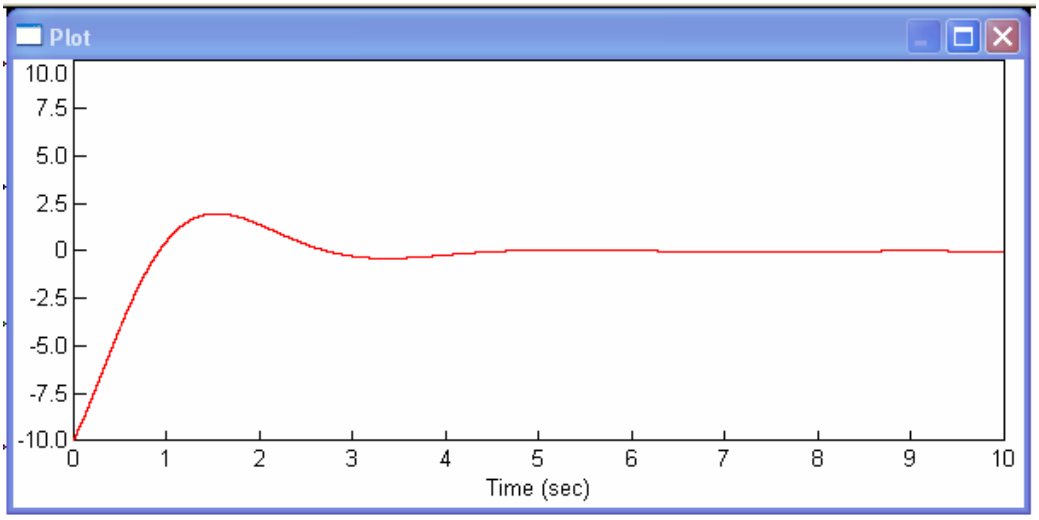
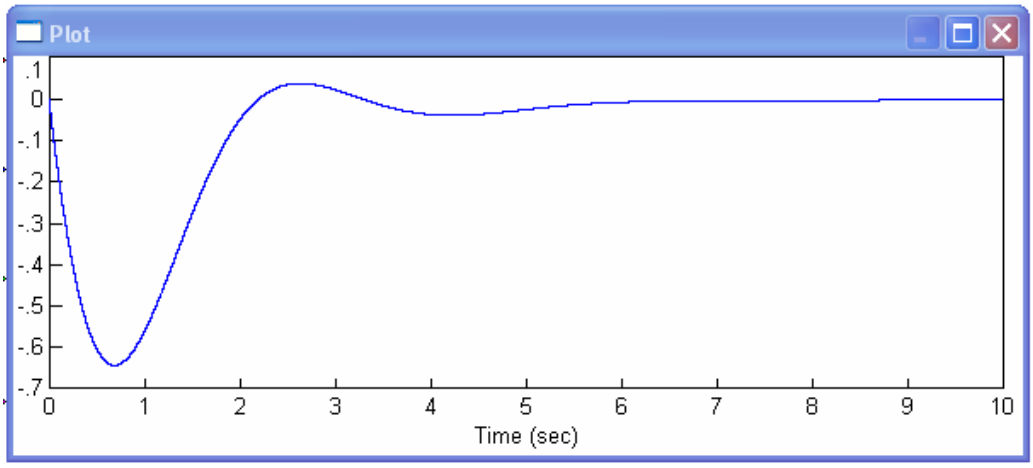


Fig 5.3 System root locus after PID controller added

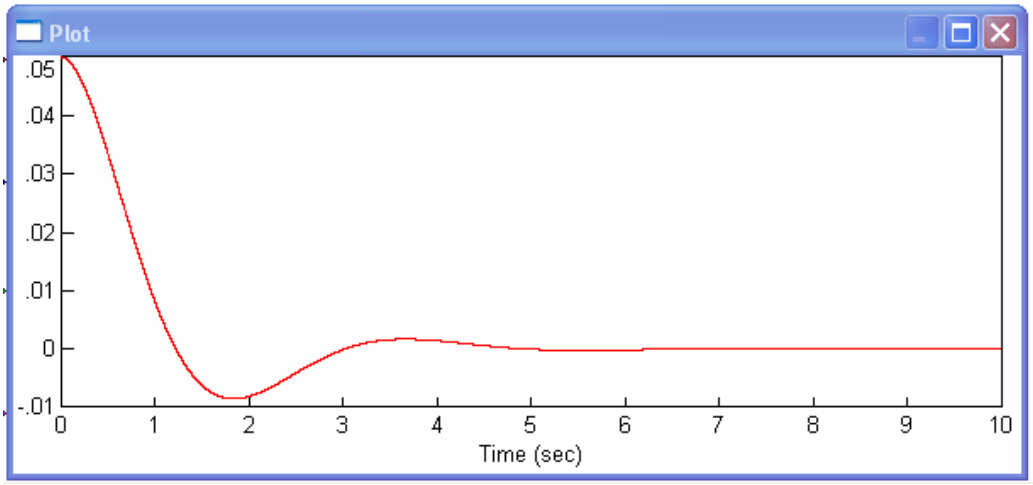
Now the system has 4 poles and one is at 0, the other three poles are all in the left hand side of the s-plane including a couple of conjugate complex (-9.4±0.2i & -4.7). So the system is stable and controllable.



Voltage response



Velocity response



Tilting angle response

Fig 5.4 Output curves of the system under PID control

6. CONCLUSION

The scooter system simulation (using Vissim) shows that this model can be controlled and keep balance well. However, the simulation is given—the rider's weight and height. This input affects the result. We assume that the rider is 1.8m height and 80kg. If a 1.5m height and 40kg person rides the scooter, the scooter will go uncontrollable; the same situation will happen if the scooter has no rider on it. To cure this problem, the PID coefficients must be recalculated. It means a different rider will have different PID parameters to keep the system working. Therefore, PID controller may not be a best choice for this kind of self-balancing system.

Due to the time limit, I have not completed the practical test yet. So far by setting the PID coefficients, the scooter stands and keeps balance within a small range of titling angle. It can go forward and backwards smoothly (at slow speed), Proved I lean no more than a angle (see *Fig 6.1*), but the response time is too slow to stop falling if the rider leans too much either forward or backwards. The problem is the signal from the sensor, the accelerometer, giving an acceleration measurement but not a tilting angle θ. The calculation of tilting angle is implemented by the central controller (a microprocessor PIC16F684), where a software LPF is applied. This filter does not get the result very accurately, so it has an error between the tilting angle input of the practical system and the tilting angle input of the simulation.

Fig 6.1 Working scooter with a rider

To improve the controller and the system, FUZZY or optimal control is recommended instead of the PID controller. Also, in the software sampling the tilting angular acceleration and calculating the tilting angle, a more complicated filter should to be considered.

APPENDIX I: CALCULATING & MEASURING PARAMETERS FOR MATHEMATICAL MODEL

Constant of the motor torque k_m

According equation (4.1) $k_m = \frac{\tau_m}{i}$. To get a relatively accurate value of torque, an extension stick is attached to the wheel (this extension is also to get rid of the friction of wheel and the pulley). Based on the experimental measurement, when the current of the motor armature is *i = 12.5 A*, the force of the stick at the far end *F = 9 N*.

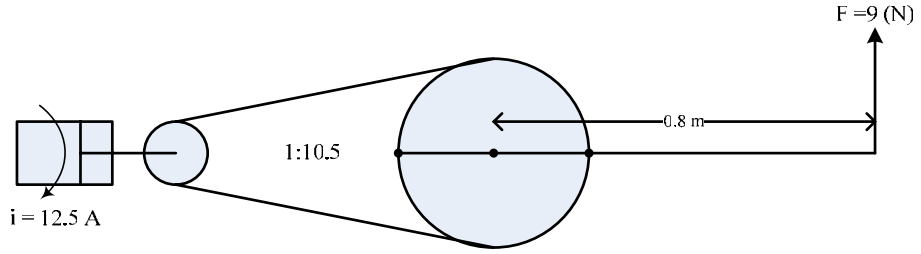


Fig AI.1Motor and pulley model diagram

Thus, $k_m' = \frac{\tau_m'}{i} = \frac{9N}{12.5A} = 0.72N/A$.

where k_m' is the constant of the torque at the edge of the pulley.

Since the pulley has a 1:10.5 ratio, so

$$k_m = k_m' \frac{1}{10.5} = \frac{0.72N/A}{10.5} = 0.069N/A$$

Constant of the motor's back EMF, k_e

According equation (4.2), $k_e = \frac{V_e}{\omega}$. When the motor is running under rated specification, rated voltage is 24V, and rated speed is 2750 rpm. Therefore,

$$k_e = \frac{V_e}{\omega} = \frac{24V}{2750r/m} = \frac{24 \times 60}{2750 \times 2\pi} = 0.083V/rad.$$

Length of the pendulum l

Assume that the rider of the scooter has a 1.8 m high. So,

l=1.8 m.

Wheel radius r

r=0.2 m.

Resistance of the armature of motor R

R = 1.0 Ω.

Mass of the pendulum M_P

Assume that the rider (80kg) and the chassis (40kg) system is an ideal pendulum which has all mass at the middle at the centre of gravity. This system weights

M_P = 120kg.

Mass of the wheel M_w

The mass of the wheel including pulley and axle is M_w=*3.5kg*.

The moment of inertia of the pendulum I_P

The following equations are obtained from Wikipedia on web.

(http://en.wikipedia.org/wiki/List_of_moments_of_inertia)

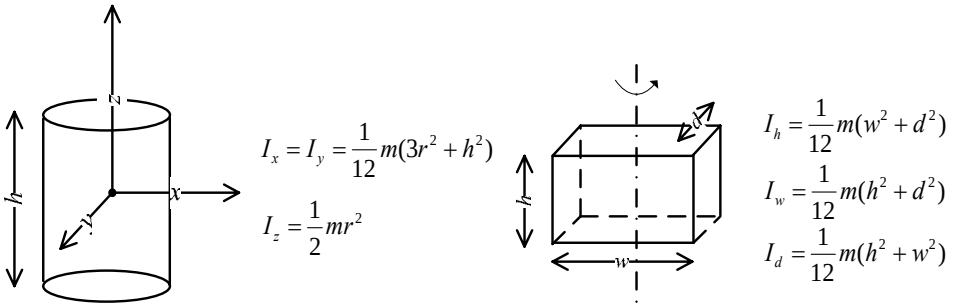


Fig AI.2 Moment of inertia of cylinder and cube

Since the pendulum (The rider and chassis system) rotates against the bottom of the cylinder, not against the centre of gravity, so the equation $I_x = \frac{1}{12}m(3r^2 + h^2)$ has to be transformed by shifting x-axis to the bottom of the cylinder.

According to the parallel-axis theorem, the rotational inertia of a body about any axis is equal to the rotational inertia *(*$= Mh^2$) it would have about that axis if all its mass were concentrated at its centre of mass, plus its rotational inertia (= I_{CM}) about a parallel axis through its centre of mass. [21]

Therefore, $I = I_{CM} + Mh^2$. (M is the mass of the cylinder and h is the perpendicular distance between the two parallel axes.)

$$I_{P1} = \frac{m}{12}(3r^2 + h^2) + m(\frac{h}{2})^2 = \frac{M_{P1}}{12}(3r^2 + h^2) + M_{P1}(\frac{h}{2})^2$$

$$I_{P1} = \frac{80kg}{12}\left[3\times(0.15m)^2 + (1.8m)^2\right] + 80kg\times(0.9m)^2 = 86.85kg\cdot m^2$$

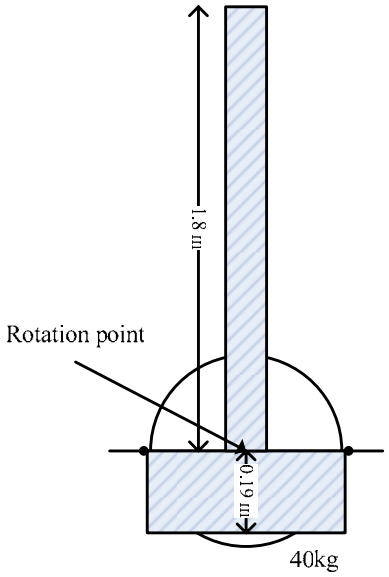


Fig AI.3 Wheeled inverted pendulum

$$I_{P2} = \frac{m}{12}\left[h^2 + d^2\right] + m\times(\frac{h}{2})^2 = \frac{M_{P2}}{12}\left[h^2 + d^2\right] + M_{P2}(\frac{h}{2})^2$$

$$I_{P2} = \frac{40kg}{12}\left[(0.32m)^2 + (0.19m)^2\right] + 40kg\times(\frac{0.19}{2}m)^2 = 0.823kg\cdot m^2$$

$$I_P = I_{P1} + I_{P2} = 86.85kg\cdot m^2 + 0.823kg\cdot m^2 = 87.67kg\cdot m^2$$

Inertia of the wheel I_W

Base on *Fig. AI.2*,

$$I_w = I_z = \frac{m}{2} r^2 = \frac{M_w}{2} r_w^{\ 2} = \frac{3.5kg}{2} \times (0.2m)^2 = 0.07kg \cdot m^2$$

Acceleration of gravity g

$g=9.81m/S^2$

APPENDIX II: SYSTEM CIRCUIT DRAWINGS

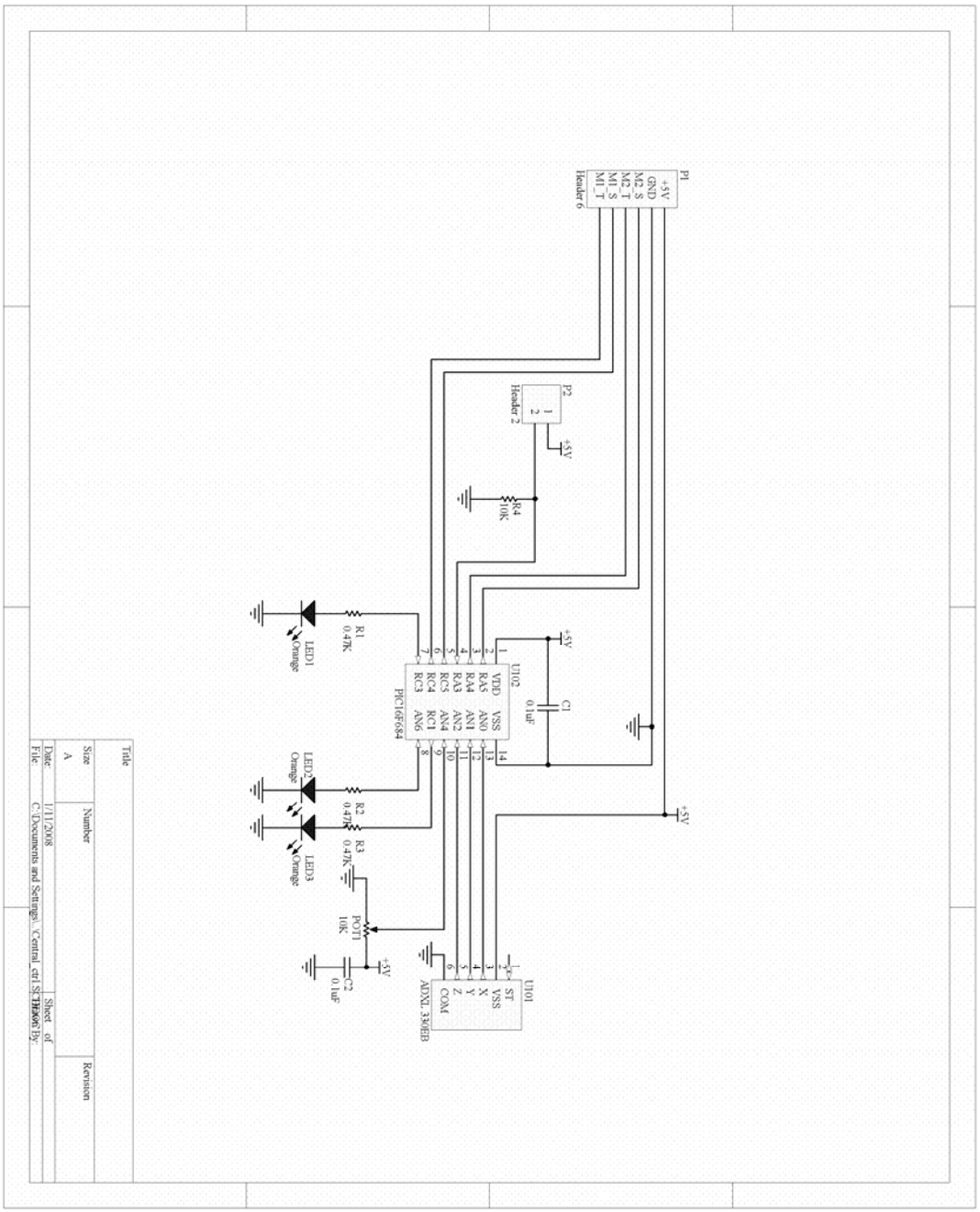


Fig AII.1 Central controller circuit schematic

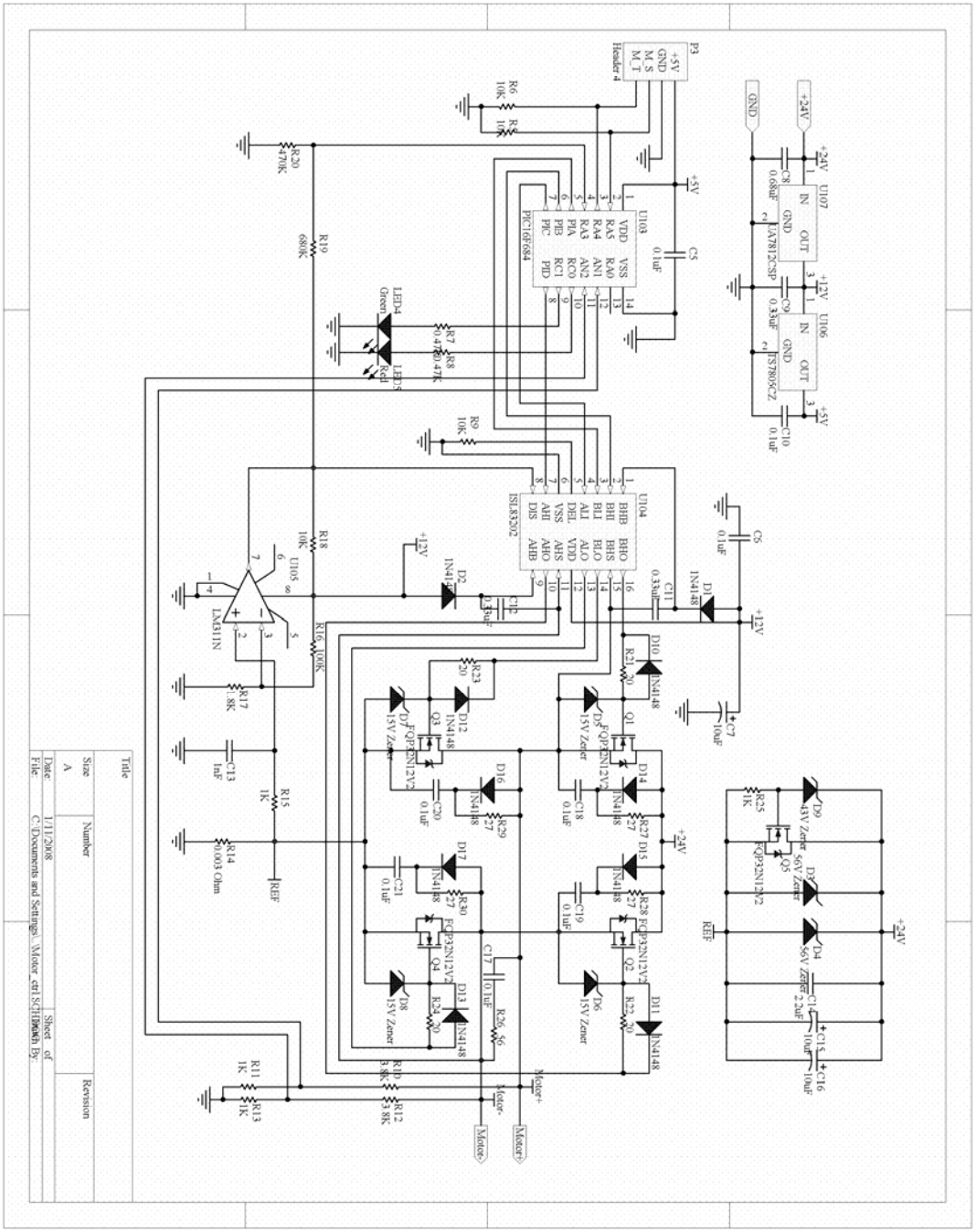


Fig AII.2 Motor controller circuit schematic

APPENDIX III: C CODE FOR MICROCHIPS

Motor driver code:

```
#include <pic.h>

// Watchdog Timer, MCLR and Fail-Safe Clock Monitor disabled.  Internal Oscillator selected.
__CONFIG(WDTDIS&MCLRDIS&INTIO&FCMDIS);

#define Forward  CCP1CON = 0xCC
#define Reverse  CCP1CON = 0x4C
#define Brake    CCP1CON = 0x0C
#define Stop     CCP1CON = 0x00

#define P1A      RC5
#define P1B      RC4
#define P1C      RC3
#define P1D      RC2

#define G_LED   RC1
#define R_LED   RC0

#define  FDD     0
#define  FBG     1
#define RDD      2
#define RBD      3

#define  RX_PIN RA4
```

```
#define TMR0CONST 117

#define  RS_IDLE          0
#define  RS_DATA_BIT  1
#define  RS_STOP_BIT   2
#define  RS_STOP_END  3

struct {
        unsigned char state;
        unsigned char sliceCount;
        unsigned char shiftBuff;
        unsigned char shiftCount;
        unsigned char dataBuff;
} rsRx;

unsigned char Motor_status;
signed char Torque;
unsigned short int i = 0;

//-------------------------------------------------------------------------------------------
void main()
{
        OSCCON = 0b01110001;            // 8 MHz internal oscillator
        OPTION = 0b11000000;            // Timer0 running at 1 MHz
        INTCON = 0b10010000;            // Open globle interrupt, Timer0 INT enable
        PIE1 = 0x01;                    // Enable Timer2 interrupts
        PIR1 = 0x00;                    // Clears interrupt flags
        TRISA = 0b00111111;             // All inputs
```

```
TRISC = 0x00;                   // All outputs
ANSEL = 0x00;                   // Disable all ADC channels
CMCON0 = 0x07;                  // Disable comparator
T2CON = 0b00000100;             // Enable TMR2 and set prescaler 1:1
PR2 = 100;                      // Set PWM frequency as 8 uS, F = 125k Hz
CCPR1L = 0;                     // Set PWM duty-cycle = 0
rsRx.state = RS_IDLE;           // Set rsRX status
Motor_status = 1;


while (1)
{
        if (RA3){
                CCPR1L = 0;
                Torque = 0;
                for (i=0; i<2500; i++) {}
                Motor_status = RBD;
        }
        else{
        switch (Motor_status) {
                //-------------------------------------------------------------
                        case FBG:
                                Brake;          // Alter PWM mode
                                P1B = 0;
                                P1C = 0;
                                P1D = 0;
                                CCPR1L = 50;    // Set duty cycle as 45%
                                R_LED = 1;      // Red LED ON
                                G_LED = 1;      // Green LED ON
```

```
                    if (Torque > 0){
                            CCPR1L=0;
                            Motor_status = FDD;
                    }
                    if(Torque < 0){
                            CCPR1L=0;
                            Motor_status = RDD;
                    }
                    break;

//-------------------------------------------------------------
            case FDD:

                    if ((CCPR1L == 0)&&(Torque <= 0)) {
                            Motor_status = FBG;
                            break;
                    }
                    Forward;            // Alter PWM mode

                    if (CCPR1L<Torque) CCPR1L=CCPR1L+1;
                    if (CCPR1L>Torque) CCPR1L=CCPR1L-1;
                    if (CCPR1L<0) CCPR1L = 0;
                    if (CCPR1L>60) CCPR1L = 60;

                    R_LED = 0;          // Red LED OFF
                    G_LED = 1;          // Green LED ON
                    break;
```

```
//-------------------------------------------------------------
        case RDD:
                if ((CCPR1L == 0)&&(Torque >= 0)){
                        Motor_status = RBD;
                        break;
                }
                Reverse;// Alter PWM mode

                if (CCPR1L<-Torque) CCPR1L=CCPR1L+1;
                if (CCPR1L>-Torque) CCPR1L=CCPR1L-1;
                if (CCPR1L<0) CCPR1L = 0;
                if (CCPR1L>50) CCPR1L = 50;

                R_LED = 1;      // Red LED ON
                G_LED = 0;      // Green LED OFF
                break;

//-------------------------------------------------------------
        case RBD:
                Stop;           // Quit PWM mode
                P1B = 0;// Turn off high FETs and turn
                // on one low FET to
                P1D = 0;        // circulate the current.
                P1A = 0;        //

                R_LED = 1;      // Red LED ON
                G_LED = 1;      // Green LED ON
```

```
                                        P1C = 1;


                                        if (Torque > 0){
                                                CCPR1L=0;
                                                Motor_status = FDD;
                                        }
                                        if (Torque < 0){
                                                CCPR1L=0;
                                                Motor_status = RDD;
                                        }
                                        break;


                                //-----------------------------------------------------------------
                                }
                        Torque = rsRx.dataBuff;         // Get data from central micro.
                        for (i=0;i<250;i++){}
                        }
                }
}

//---------------------------------------------------------------------
void interrupt isr(void)
{
        if (INTF&&INTE){


                switch (rsRx.state){
                        //-----------------------------------------------------------
                        case RS_IDLE:
```

```
  if (RX_PIN==0){
                rsRx.sliceCount = 4;
                rsRx.shiftCount = 8;
                rsRx.state = RS_DATA_BIT;
        }
  break;
//-------------------------------------------------------------
case RS_DATA_BIT:
        if (--rsRx.sliceCount==0){
                rsRx.shiftBuff >>= 1;
                if (RX_PIN)
                        rsRx.shiftBuff|=0x80;
           rsRx.sliceCount = 3;
                if (--rsRx.shiftCount==0){
                        rsRx.state = RS_STOP_BIT;
                }
        }
        break;
//-------------------------------------------------------------
case RS_STOP_BIT:
        if (--rsRx.sliceCount==0){
                rsRx.dataBuff = rsRx.shiftBuff;

        rsRx.state = RS_IDLE;
                INTE = 1;          // Enable RA2 interrupt
                T0IE = 0;          // Disable Timer0 interrupt
        }
        break;
```

```
                        //-------------------------------------------------------------
                        default:
                                rsRx.state =        RS_IDLE;
                }
                INTF = 0;                       // Clear RA2 interrupt flag
        }
}
```

Central controller code:

```
#include <pic.h>

// Watchdog Timer, MCLR and Fail-Safe Clock Monitor disabled.  Internal Oscillator selected.
__CONFIG(WDTDIS&MCLRDIS&INTIO&FCMDIS);

#define  TX_PIN1 RC4
#define  M1_CLK          RC5
#define  TX_PIN2 RA4
#define  M2_CLK          RA5

#define  RS_IDLE         0
#define  RS_DATA_BIT  1
#define  RS_STOP_BIT   2
#define  RS_STOP_END 3

#define G_LED RC1
#define R_LED RC2
#define W_LED RC3

bit rsTxBusy = 0;
struct {
        unsigned char state;
        unsigned char sliceCount;
        unsigned char shiftBuff1;
        unsigned char shiftBuff2;
        unsigned char shiftCount;
} rsTx;
```

```
signed short int Knob = 0,X_Axis = 0,Y_Axis = 0;
unsigned short int  i = 0, K = 0;
signed short int Angle0 = 0, Angle1 = 0;
signed char Torque = 0, Torque1 = 0, Torque2 = 0;
signed short int Steer = 0;
signed short int Theata = 0, Theatasum = 0;

//-------------------------------------------------------------------------------
void main()
{
        OSCCON = 0b01110001;  // 8 MHz internal oscillator
        OPTION = 0b10000000;            // Timer0 prescaler 1:2; running at 1 MHz
        INTCON = 0b11100000;            // Globle interrupt, PEIE enable, Timer0 INT enable
        PIE1 = 0b00000001;              // Enable Timer1 interrupt
        T1CON = 0b00010001;             // Enable Timer1, prescaler 1:2 2uS/bit
        T2CON = 0b00000100;
        TRISA = 0b00001111;             // output: RA4,RA5 inputs: RA0,RA1,RA2,RA3
        TRISC = 0b00000001;             // RC0 input, others output
        CMCON0 = 0x07;                  // Disable comparator
        CCP1CON = 0b00001100;           // ECCP function closed
        PR2 = 200;
        ANSEL = 0b00010111;             // AN0,AN1,AN2,AN4 enable
        ADCON0 = 0x0D;                  // Left justified, Vref = Vdd, AN0, ADC enabled
        ADCON1 = 0x10;                  // ADC clock = Fosc/8
        rsTx.state = RS_IDLE;           // Set rsTX status
        rsTxBusy = 0;                   // Clear rsTxBusy bit
```

```
while(1)
{
        if (Angle0<1 && Angle0>-1) Angle0 = 0;
        if (Angle0>1) Angle0 = Angle0-1;
        if (Angle0<-1) Angle0 = Angle0+1;
//-------------------------------------------------------------
        CHS1=0; CHS0=1; CHS2=0;         // Select channel 1
        GODONE = 1;                     // Start ADC conversion
        while(GODONE){;}                // Wait for the conversion to finish
        Y_Axis = ADRESH;                // Put ADC value into variable
        Steer = Y_Axis-127;
        if (Steer>5) Steer = 5;
        if (Steer<-5) Steer = -5;
        if (Steer>0) {R_LED = 0; G_LED=1;}
        if (Steer<0) {R_LED = 1; G_LED=0;}

        CHS1=0; CHS0=0; CHS2=1;         // Select channel 4
        ;
        ;
        GODONE = 1;                     // Start ADC conversion
        while(GODONE){}                 // Wait for the conversion to finish
        Knob = ADRESH;                  // Put ADC value into variable
        Knob += -127;
//---------------------------------------- ----------------------

        if (RA3) {
                Torque = 3*Theata;      //Knob/3; + 8*(Angle0-Angle1);
```

```
                    if(Torque == 0) {G_LED = 1; R_LED = 1;}
                    if(Torque> 0) {G_LED = 1; R_LED = 0;}
                    if(Torque< 0) {G_LED = 0; R_LED = 1;}

                    if(Torque>50) Torque = 50;
                    if(Torque<-50) Torque = -50;
                }
                else    Torque = 0;

        //---------------------------------------------------------------
                if (rsTxBusy == 0) {
                    rsTx.shiftBuff1 = Torque; // + (Torque+1)*(Steer/500);
                    rsTx.shiftBuff2 = Torque; // + (Torque+1)*(Steer/500);
                    rsTxBusy = 1;
                }
        }
}

//---------------------------------------------------------------
void interrupt isr(void)
{

        if (TMR1IF) {
                CHS1=0; CHS0=0; CHS2=0;         // Select channel 0
                W_LED = !W_LED;
                Angle1 = Angle0;
                TMR1H = 0xFB;                   //833*2uS=1.666mS
                TMR1L = 0x7B;
```

```
        GODONE = 1;                     // Start ADC conversion
        while(GODONE){}                 // Wait for the conversion to finish
        X_Axis = ADRESH;                // Put ADC value into variable
        X_Axis += -127;

        if(i<10) {
                Theatasum += X_Axis;
                if(Theatasum>20000) Theatasum = 20000;
                if(Theatasum<-20000) Theatasum = -20000;
                i++;
        }
        else {
                Theata = Theatasum/10;
                i = 0;
        }

        Theatadot += X_Axis;
        if (Theatadot>25000) Theatadot = 25000;
        if (Theatadot<-25000) Theatadot = -25000;
        Theata += Theatadot;
        if (Theata>30000) Theata = 30000;
        if (Theata<-30000) Theata = -30000;

        if(Theata>0) CCPR1L = Theata;
        if(Theata<0) CCPR1L = -Theata;
        TMR1IF = 0;
}
```

```
        if (T0IF){
                M2_CLK = 1;     // Sent CLK raising edge to Motor2.
//              M1_CLK = 1;     // Sent CLK raising edge to Motor1.
                switch (rsTx.state){
                //-------------------------------------------------------------
                        case RS_IDLE:
                                if (rsTxBusy){
                                        TX_PIN1 = 0;
                                        TX_PIN2 = 0;
                                rsTx.sliceCount =3;
                                        rsTx.shiftCount =8;
                                        rsTx.state = RS_DATA_BIT;
                                        }
                                else {
                                        TX_PIN2 = 1;
                                        TX_PIN1 = 1;
                                }
                                break;
                        //-------------------------------------------------------
                        case RS_DATA_BIT:
                                if (--rsTx.sliceCount==0){
                                        if (rsTx.shiftBuff1 & 0x01)
                                                TX_PIN2         = 1;
                                        else
                                                TX_PIN2 = 0;
                                        if (rsTx.shiftBuff2 & 0x01)
                                                TX_PIN1         = 1;
                                        else
```

```
                    TX_PIN1 = 0;
                rsTx.shiftBuff1 >>= 1;
                rsTx.shiftBuff2 >>= 1;
                rsTx.sliceCount = 3;
                if (--rsTx.shiftCount==0)
                    rsTx.state = RS_STOP_BIT;
            }
            break;
        //-------------------------------------------------------
        case RS_STOP_BIT:
            if (--rsTx.sliceCount==0){
                TX_PIN2 = 1;
                TX_PIN1 = 1;
                rsTx.sliceCount = 18;       // Set stop bit length.
                rsTx.state = RS_STOP_END;
                }
            break;
        //-------------------------------------------------------
        case RS_STOP_END:
            if (--rsTx.sliceCount==0){
                rsTxBusy = 0;
                rsTx.state = RS_IDLE;
            }
            break;
        //-------------------------------------------------------
        default:
            rsTx.state = RS_IDLE;
}
```

```
                TMR0 = 117;                 // Baud rate setting 10K
                M2_CLK = 0;                 // Sent CLK falling edge to Motor2.
//              M1_CLK = 0;                 // Sent CLK falling edge to Motor1.
                T0IF = 0;                   // Clear Timer0 interrupt flag
        }
        return;
}
```

APPENDIX IV: SIMULATION OF THE SCOOTER

Balancing simulation

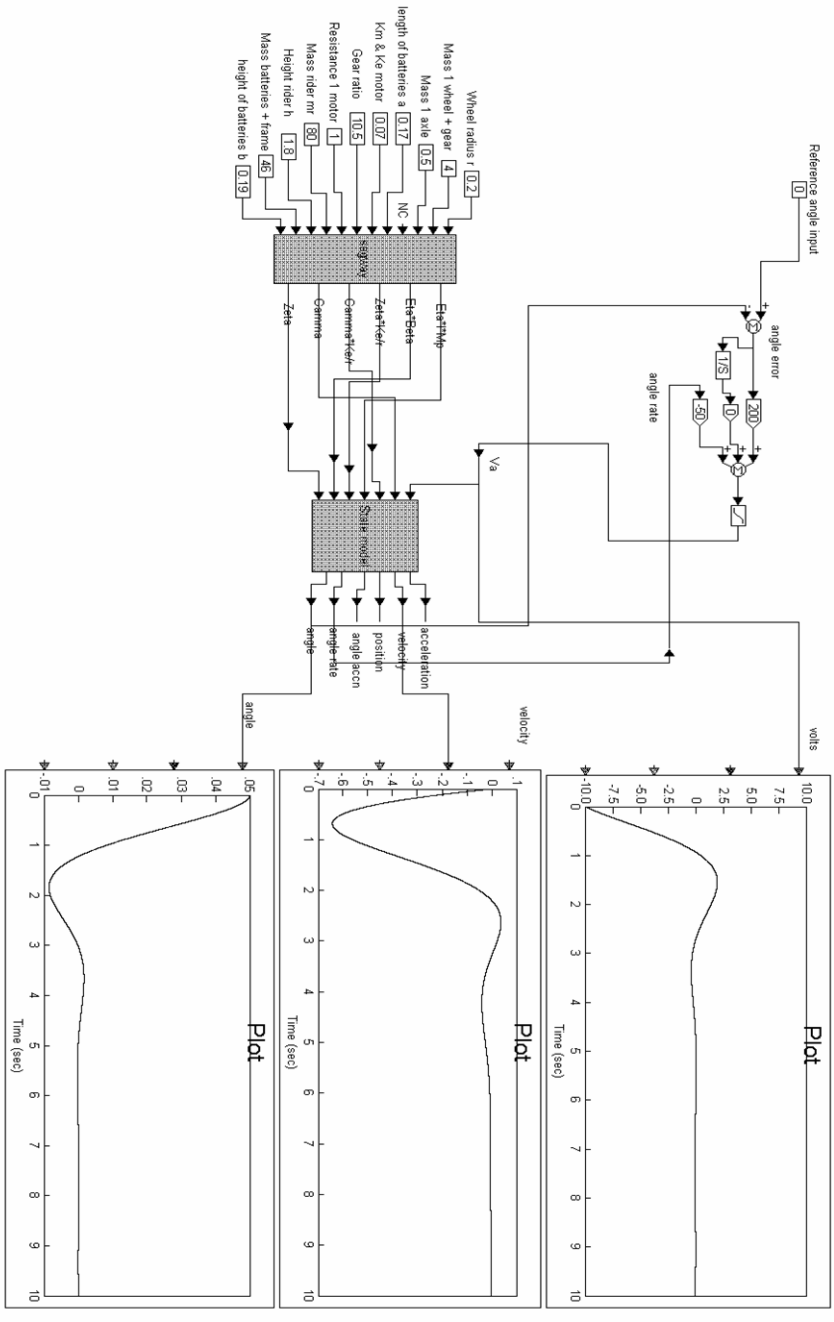


Fig AIV.1 Simulation result when PID coefficients are fixed

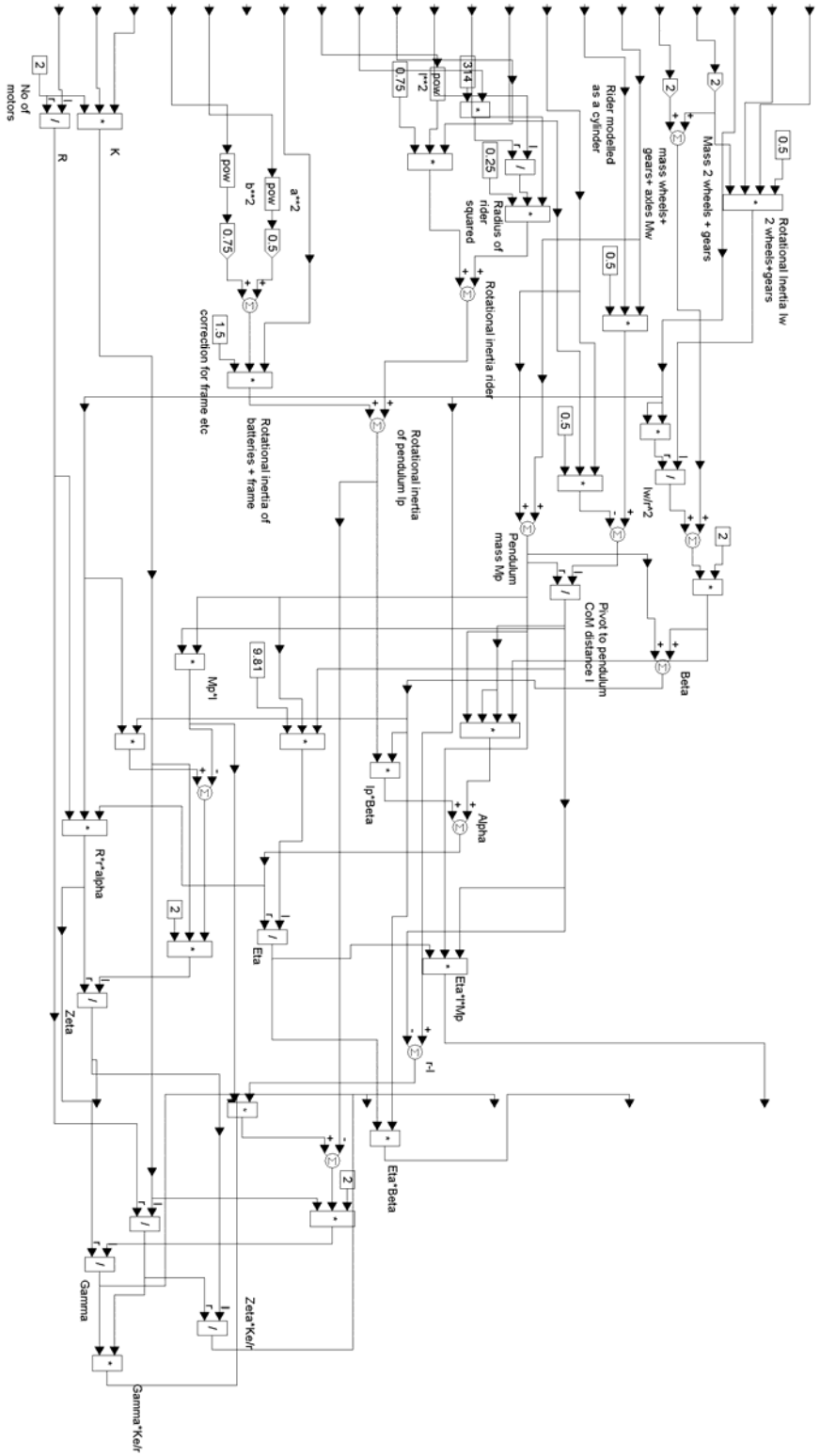


Fig AIV.2 System block model in software

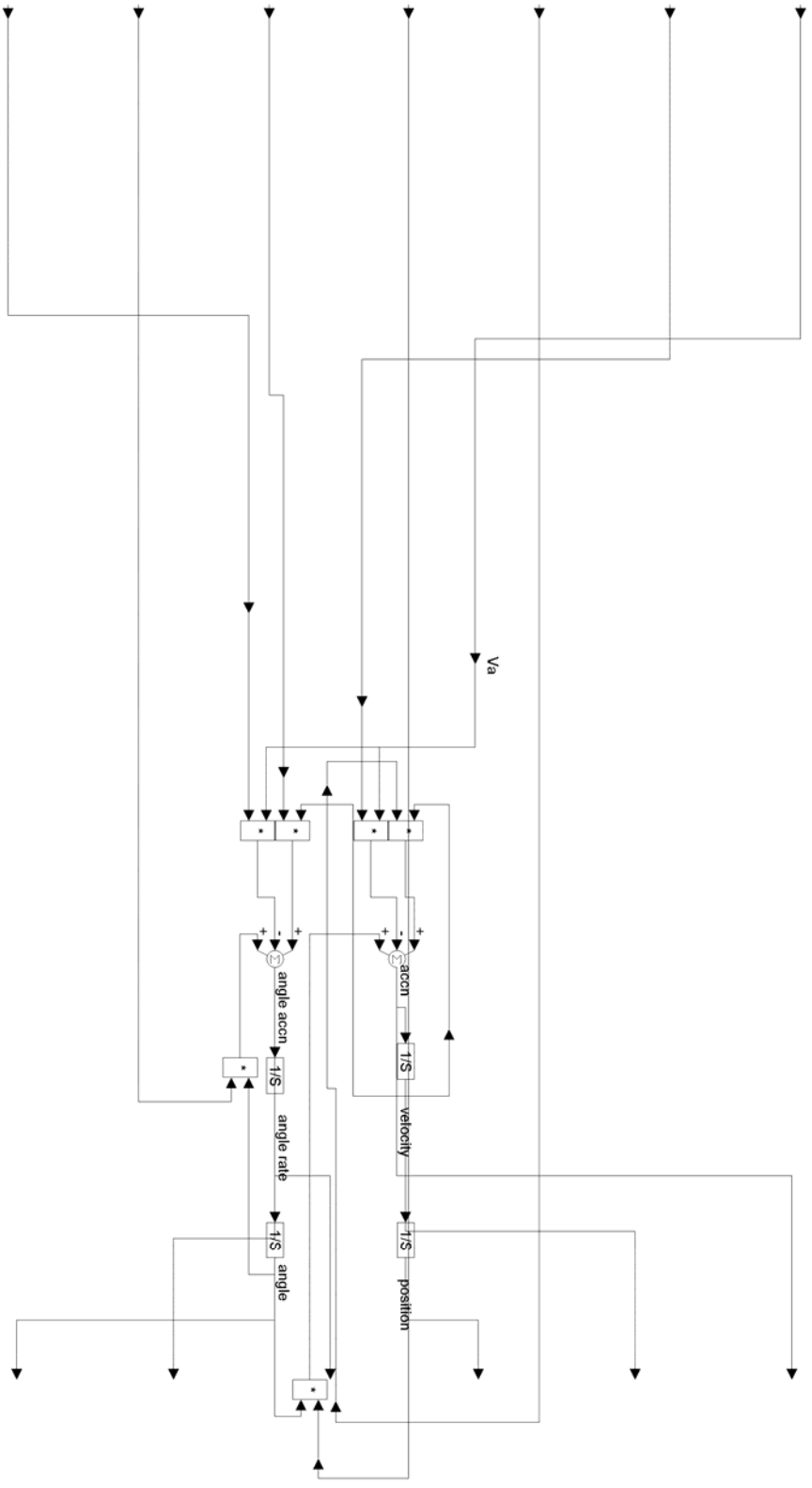


Fig AIV.3 State space block model in software

REFERENCES

[1] Segway Inc., Bedford, NH, USA. See http://www.segway.com/. Retrieve on 25th Aug. 2007.

[2] Ego Vehicles LLC, MA, USA. See http://www.egovehicles.com/. Retrieve on 25th Aug. 2007.

[3] *Gyroscope -- Wikipedia, the free encyclopedia*. See http://en.wikipedia.org/wiki/Gyroscope, Retrieve on 14th Nov. 2007

[4] T. Blackwell, *Building a balancing scooter*. http://www.tlb.org/scooter.html/. Retrieve on 25th Aug. 2007.

[5] *MOSFET basic*, R&D2 Group, Fairchild Korea semiconductor, Nov, 1999.

[6] *Pulse-width modulation.* See http://en.wikipedia.org/wiki/Pulse-width_modulation.

[7] *FQP32N12V2 120V N-channel MOSFET datasheet,* Fairchild semiconductor corporation, 2003.

[8] *ISL83202 55V, 1A peak current H-bridge FET driver datasheet,* Intersil corporation, Dec, 2006.

[9] *Lego Segway*. See http://www.youtube.com/watch?v=V40ScvJeFxg&mode=related&search=. Retrieve on 25th Aug. 2007.

[10] J. Charais & R.Lourens, *PID control of an inverted pendulum*, Microchip technology Inc., 2004.

[11] *PIC16F684 datasheet*, Microchip technology Inc., 2007.

[12] Rich Chi Ooi, *Balancing a two-wheeled autonomous robot,* The university of western Australia, 2003.

[13] *Small, low power, 3 axis ± 3g iMEMS accelerometer,* Analog device corporation, 2006.

[14] J.B. Forsythe, *Paralleling of power MOSFETs for high power output,* international rectifier, EI Segundo, California.

[15] M. Rylee, Low-cost bidirectional brushed DC motor control using PIC16F684, Microchip technology Inc., 2003

[16] G. Joos & T. H. Barton, Four-quadrant DC variable-speed drives—design considerations, Proceeding of the IEEE, Dec, 1975.

[17] B. Rosenthal, Appling power MOSFET drivers, Application note, Intersil Americas Inc., Sep, 2003.

[18] *Snubber circuits suppress voltage transient spikes in multiple output DC-DC flyback converter power supplies,* Application note, Maxim corporation, Dec, 2001.

[19] P. C. Todd, *Snubber circuits: theory, design and application,* Unitrode Corporation, May, 1993.

[20] List of moments of inertia, see http://en.wikipedia.org/wiki/List_of_moments_of_inertia, Retrieve on 11th Jan. 2008.

[21] D. Halliday, R. Resnick & J. Walker, Fundamental of physics, John wiley & sons, INC. 1997.