ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/265227587

SELF BALANCING ROBOT

Article

CITATIONS
2

READS
9,683

2 authors, including:

Nabil Lathiff
Microsoft
2 PUBLICATIONS **15** CITATIONS
SEE PROFILE

All content following this page was uploaded by Nabil Lathiff on 12 November 2016.

The user has requested enhancement of the downloaded file.

SELF BALANCING ROBOT

SIDDHARTHA BALASUBRAMANIAN
MOHAMED NABIL LATHIFF

PROJECT SPONSOR:
STEPHEN SWIFT
SOC ROBOTICS INC.

PROJECT 1116
APPLIED SCIENCE 459
ENGINEERING PHYSICS PROJECT LABORATORY
THE UNIVERSITY OF BRITISH COLUMBIA
APRIL 04TH 2011

Executive Summary

The objective of this project was to design and implement a self- balancing algorithm using the Cricket embedded processor. The implementation utilized both an accelerometer and a rate-gyroscope built into the micro-controller in order to achieve a vertical balance. The fusion of both sensor data into a single usable value was achieved through a complementary filter. Consequently, the output of the complementary filter was designed to be primarily dependent on the gyroscope data, to which a fraction of the accelerometer data was added to compensate for the gyroscopic drift.

An 8-inch robot with a single plate aluminum chassis was powered through a high-current H-bridge circuit connected directly to the Cricket board. The control loop, which included both the software implementation of the complementary filter and the PID controller, was measured to run at 530 Hz (±20Hz). Additionally, a pulse-width-modulation signal generator was implemented in software using the interrupt service routines of the micro-controller. Consequently, this resulted in a robust code-base which was able to achieve a self-balance with an oscillatory amplitude of 1 cm (± 0.3 cm) and a balance time of about 15 seconds.

Contents

Executive Summary iii

List of Figures iv

List of Tables v

1. Introduction 1

2. Discussion 2

2.1 Physical Model 2

2.2 Control Algorithm Design 3

2.2.1 Using an IR Tilt Sensor 3

2.2.2 Using the Accelerometer 4

2.2.3 Using a Gyroscope 5

2.2.4 Using both an Accelerometer and a Gyroscope 6

2.3 Hardware Design 8

2.3.1 Drive Mechanism 8

2.3.2 Power Supply Unit 8

2.3.3 Chassis 9

2.4 Software Implementation 9

2.4.1 Control Algorithm 10

2.4.2 Software PWM 11

2.4.3 Serial Communication 13

2.4.4 Discrete PID Controller 14

2.5 Testing Methods 15

2.5.1 Complementary Filter 15

2.5.2 PID Controller 16

- 2.5.3 Robot Hardware ... 16
- 2.6 Results ... 16
  - 2.6.1 Control Loop Performance ... 16
  - 2.6.2 Complementary Filter Accuracy ... 17
  - 2.6.3 Balancing Performance ... 19
  - 2.6.4 Software Code ... 19
- 2.7 Discussion of Results ... 20
- 3. Conclusion ... 21
- 4. Project Deliverables ... 22
  - 4.1 List of Deliverables ... 22
    - 4.1.1 Completed Self-Balance Code ... 22
    - 4.1.2 Robot Hardware ... 22
  - 4.2 Financial Summary ... 22
- 5. Recommendations ... 23
  - 5.1 Hardware ... 23
  - 5.2 Software ... 23
- Appendix A: The Physics Model ... 24
- Bibliography ... 29

List of Figures

Figure 1: The 3 degrees of freedom of the system 2

Figure 2: Control System for accelerometer only algorithm 4

Figure 3: Control system for the gyroscope only algorithm 5

Figure 4: Graph showing the effects of gyroscopic drift 6

Figure 5: Complementary filter 7

Figure 6: A typical control loop 10

Figure 7: Blocking PWM 12

Figure 8: Non-blocking PWM 13

Figure 9: PID controller 15

Figure 10: Complementary filter results 18

Figure 11: The free body diagram of the two wheels 24

Figure 12: The free body diagram of the chassis 26

List of Tables

Table 1: Financial Summary 22

1. Introduction

There are many projects on building self-balancing robots - ranging from simple DIY analog balancing bots to sophisticated self-balancing scooters. Self-balancing electro-mechanical systems have various uses, including the ability to showcase the computational performance of new and emerging embedded processors. This project is intended to be a showcase of the capabilities of the "Cricket" board which is based on an 8-bit Atmel AVR flash microprocessor. The robot will be used by the manufacturers of the Cricket board, SOC Robotics Inc., to market the product among hobbyists and professionals.

The prime objective of the project is to design code and implement an efficient self-balancing algorithm using the "Cricket" embedded processor. The project would primarily focus on the software implementation of the algorithm by incorporating a physical model of the robot and the available sensors onboard the Cricket controller. The software itself is intended to be portable and can be used on any self-balancing platform based on the Cricket controller by tuning a few parameters.

The Cricket controller, equipped with a 3-Axis accelerometer and a 3-Axis rate gyroscope, is a compact embedded microcontroller powered by the ATmega644 processor. The built-in dual 1.5A H-bridges and the external 32.756 KHz crystal, which enables a real time clock for timed software execution, make the "Cricket" controller a highly suitable platform for creating self-balancing applications.

Many relatively simple implementations of self-balancing robots by hobbyists use two IR reflectance sensors to determine the tilt angle of the robot. Almost all high-end self-balancing applications including the self-balancing scooter – the Segway, use either an accelerometer or a gyroscope or even a combination of both to maintain the robot vertical. Hence this project would focus on utilizing both available sensors on the Cricket controller. There are various software implementations of filters including the Kalman filter and the complementary filter which can be used to combine multiple sensor readings into a single usable value. The performance characteristic of each filter is analyzed and weighed according to their individual pros and cons in the Discussion section of this report.

This report discusses the theoretical considerations made at the start of the project, the steps that were taken to implement the self-balancing code, the analysis of the final results, the interpretation of the final results and finally the recommendations for future continuation of the project.

2. Discussion

The theory, including the methods and considerations behind the physical model, the filtering algorithm, and the software design are all discussed in this section.

2.1 Physical Model

The three degrees of freedom of the robot is shown in Figure 1.

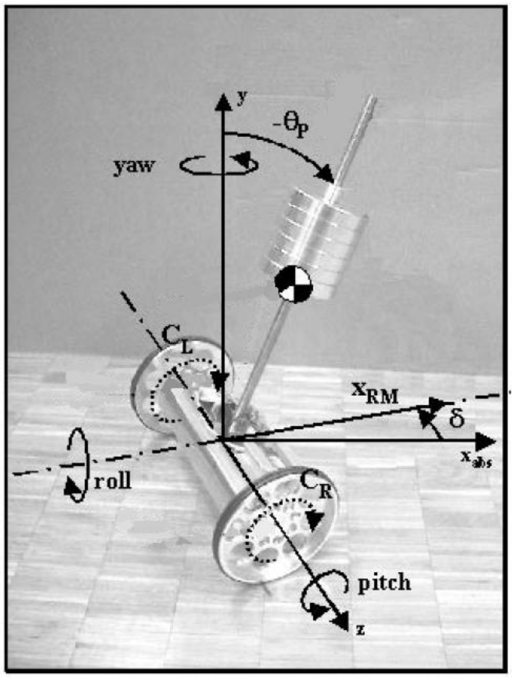

FIGURE 1: THE 3 DEGREES OF FREEDOM OF THE SYSTEM (SOURCE: ANDERSON, 2003)

The robot has the ability to rotate around the z-axis (pitch) by an angle θ_p with a corresponding angular velocity ω_p. The linear movement of the robot is characterized by the position x_{RM} and the velocity v_{RM}. Furthermore, the robot can rotate about the vertical axis (yaw) by an angle δ with a corresponding angular velocity $\dot{\delta}$.

The control system of the robot would be based on two state controls implemented through software: one controlling the stability around the lateral axis (pitch) and a second one about the vertical axis (yaw). Both state controls' outputs would then be combined and the robot's motion would be controlled by applying torques C_L and C_R to the corresponding wheels.

Equations R_1 and R_2 can be used to calculate the individual torques required for the robot's motion.

$$A_1 C_L + A_2 C_R = -\left(B_1 \ddot{\theta}_P + B_2\, \theta_P + B_3\right) \quad \textbf{(1)}$$

$$A_3 C_L + A_4 C_R = -\left(B_4 \ddot{\delta} + B_5\right) \quad \textbf{(2)}$$

Where R_1 models the robot when it is balanced at a stationary position and R_2 models the robot when it is turning about its y-axis while maintaining a balanced posture (see Appendix A. for the derivation and the coefficient values).

2.2 Control Algorithm Design

Since calculating the pitch angle θ_p accurately is crucial for the self-balancing apparatus, a few methods were considered in order to measure the pitch angle and its rate of change.

2.2.1 Using an IR Tilt Sensor

A method that is very popular with hobbyists is to use two infra-red sensors placed on either side of the robot. The control algorithm uses the sensors to measure the distance to the ground on either side of the robot. By comparing the values from the two sides, it is able to determine the pitch angle accurately.

This method uses very little hardware and requires a very simple control algorithm. However, since the purpose of the project is to advertise the features of the Cricket board, especially the integrated gyroscopes and accelerometers, this method was not used.

2.2.2 Using the Accelerometer

A Freescale Semiconductor MMA7260 three-axis accelerometer located on the cricket board can be used to get acceleration values in three-dimensions. Accelerations oriented in the y (upwards) and x (forward) directions would provide a way to measure the direction of gravity and subsequently calculate the pitch angle as per Equation (3). This system is illustrated in Figure 2.

$$\theta_p = \arctan \frac{a_x}{a_y} \tag{3}$$

However, when the robot is on the move, the horizontal acceleration would be added to the accelerometer readings. Due to this, the pitch angle can no longer be determined accurately. Hence this method, used alone, will not be ideal.

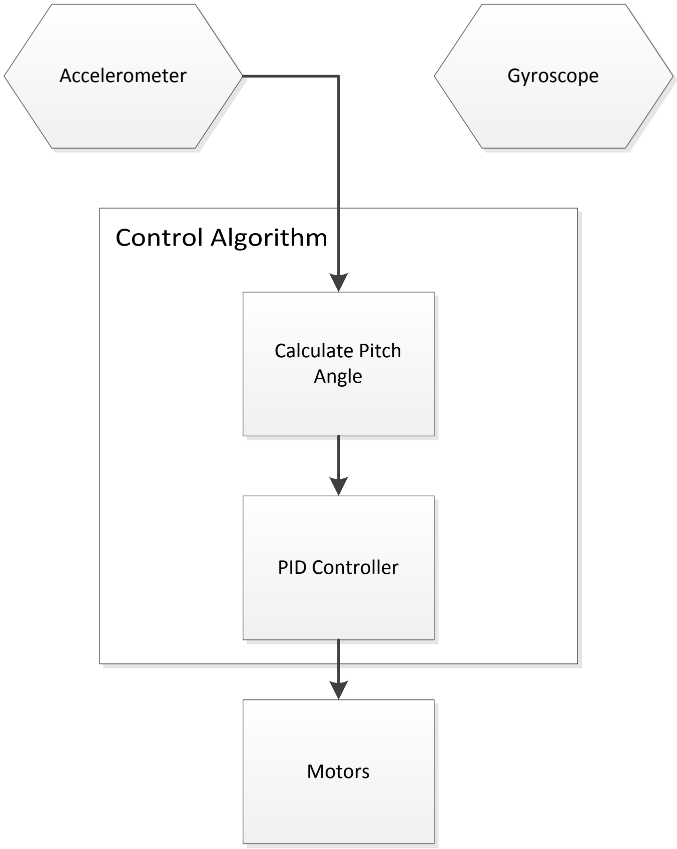
Accelerometer
Gyroscope
Control Algorithm
Calculate Pitch Angle
PID Controller
Motors

FIGURE 2: CONTROL SYSTEM FOR ACCELEROMETER ONLY ALGORITHM

2.2.3 Using a Gyroscope

The cricket board is also equipped with three Epson X3500 rate gyroscopes. A rate gyroscope measures the angular velocity and the signal can be integrated numerically (by using a simplified Runga-Kutta method) as in equation (4) to obtain the pitch angle.

$$\theta_p\,(i) = \theta_p\,(i\text{-}1) + {}^{1}\!/_{6}\,(\,val_{i-3} + 2\,val_{i-2} + 2\,val_{i-1} + val_i) \qquad (4)$$

An illustration of this system is shown in Figure 3.

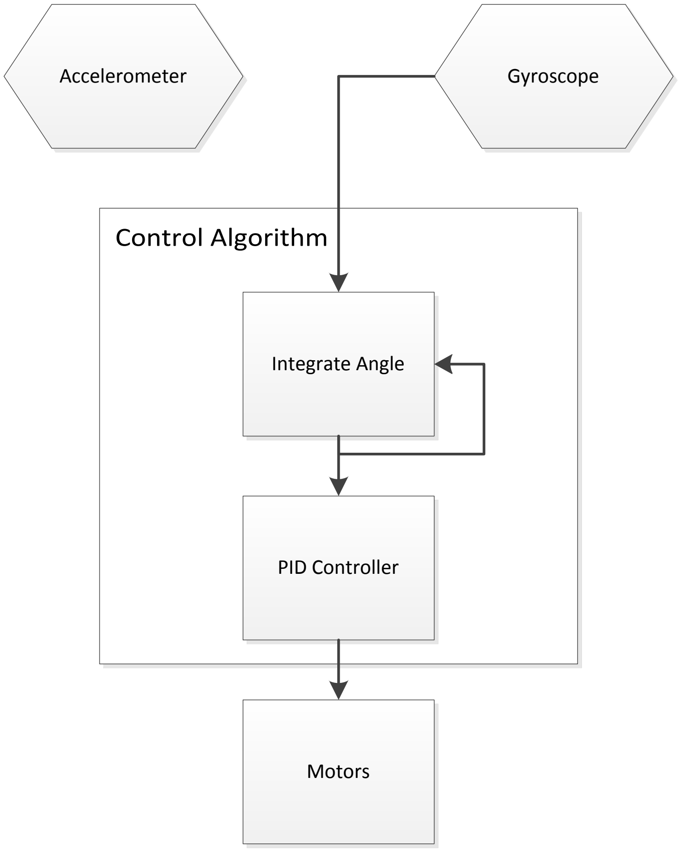

FIGURE 3: CONTROL SYSTEM FOR THE GYROSCOPE ONLY ALGORITHM

A common drawback of gyroscopes is that there exists a small DC bias which, upon integration, would cause the zero point to drift overtime. Hence, a balancing robot based solely on a gyroscope would be vertical for a few seconds and eventually fall over due to the drift of the zero point. The effect of gyroscopic drift can be clearly seen in Figure 4.

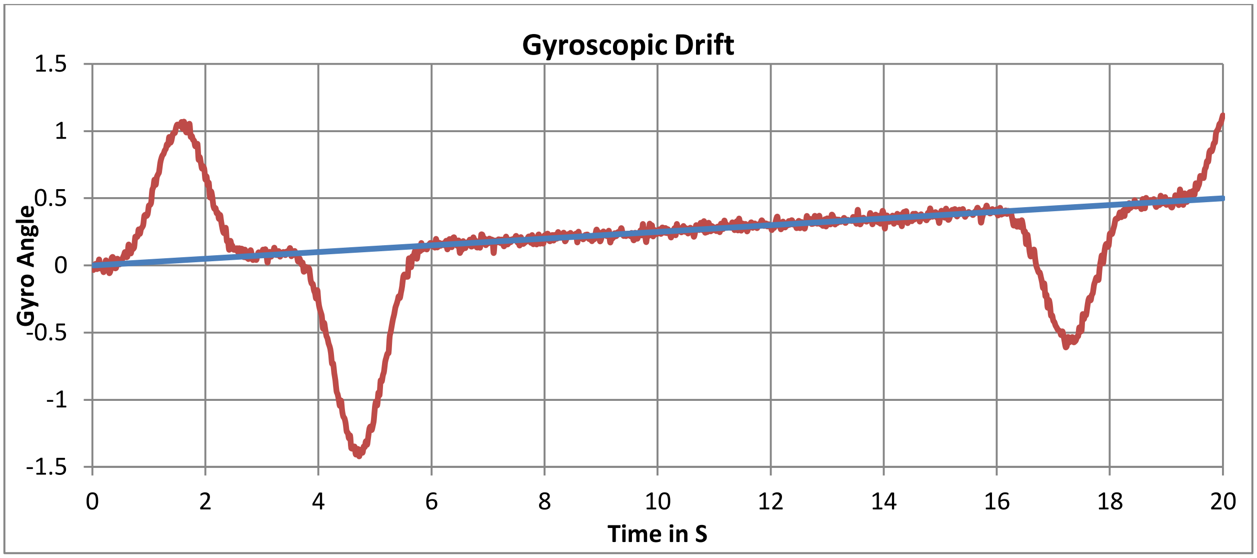

FIGURE 4: GRAPH SHOWING THE EFFECTS OF GYROSCOPIC DRIFT

2.2.4 Using both an Accelerometer and a Gyroscope

An accurate measurement of the pitch angle could be obtained by combining the outputs from the gyroscopes and the accelerometers through the use of a sensor fusion algorithm.

The most sophisticated algorithm is to use a discrete Kalman filter to combine both the gyroscope and accelerometer readings. A Kalman filter is a set of mathematical equations that provides a means to fuse multiple sensor readings into one by considering only the strengths of each individual signal. It also provides a good degree of noise cancellation. However, the Kalman filter is quite complex to implement, especially on the limited hardware of a microcontroller.

Another widely used solution is a filter known as the complementary filter. The complementary filter primarily uses the gyroscope output and integrates it to determine the pitch angle. However, a small component of the accelerometer output is also added to the result to compensate for the gyroscope drift (equation (5).

$$\theta = C_g \times Gyro + C_a \times Accel \tag{5}$$

The weights are determined by the constants C_g and C_a. These parameters are tunable and will need some adjusting for each hardware configuration. Typically, the gyro will have a weight of around 98%

and the accelerometer will have one around 2%. This allows the algorithm to utilize the fast and accurate gyroscope while also correcting for the drift using the accelerometer.

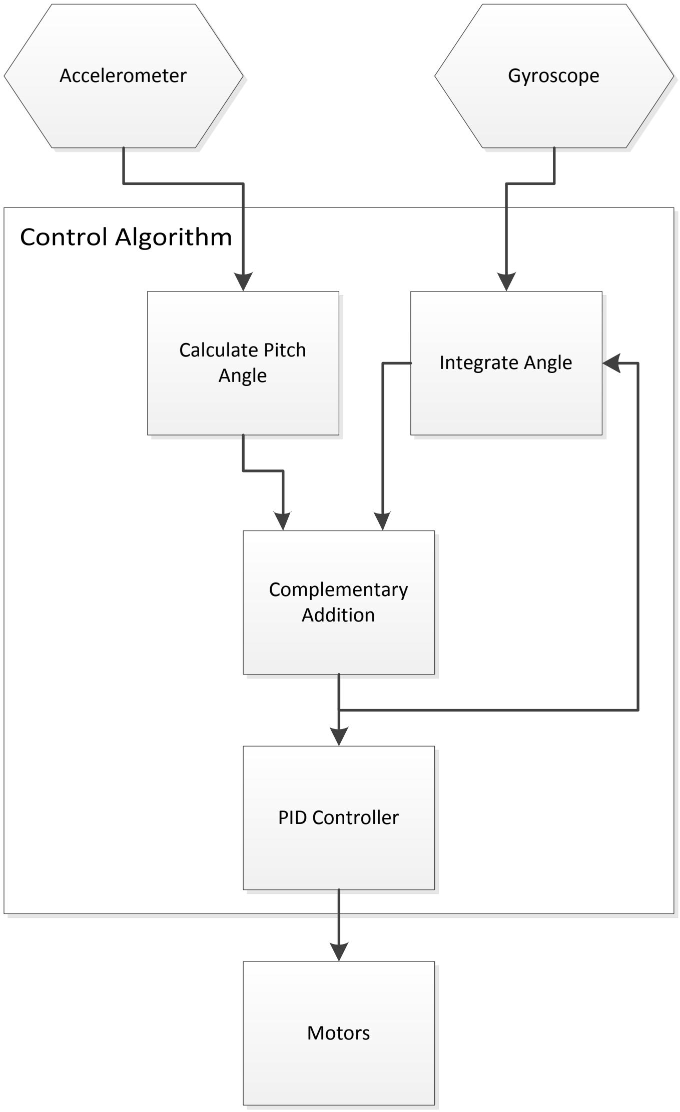
Accelerometer
Gyroscope
Control Algorithm
Calculate Pitch Angle
Integrate Angle
Complementary Addition
PID Controller
Motors

FIGURE 5: COMPLEMENTARY FILTER

This solution, shown in Figure 5, very nearly matches the output of a Kalman filter while using very little processor power. We found this algorithm to be the ideal balance between accuracy and speed. This is the algorithm that we used in our implementation of sensor filter.

2.3 Hardware Design

There were a variety of steps taken to implement the algorithm. Although the main deliverable was the control algorithm, we also had to design and build a hardware platform for testing.

2.3.1 Drive Mechanism

The motors provided by our sponsor were too slow and had a large response time. They were unsuitable for this application.

A Tamiya dual gearbox containing twin 3V motors was ordered and it remained our main drive mechanism. The gearbox had a gearing ratio of 58:1 which provided the right amount of torque and speed.

However, using these motors meant that we would also have to build a secondary H-bridge to supply the required 4+ Amps of current. We built a compact dual H-bridge using power MOSFETs. These H-bridges could easily supply around 8A – 10A of current without heat sinking.

We also ordered a pair of rubber wheels that provided adequate grip while still being light. The wheels attached directly onto the gearbox shafts, with no slip.

Overall, the drive mechanism was acceptable for this type of robot. However, this setup was far from ideal. Ideally, we would use a direct drive mechanism with stepper motors that allowed very precisely controlled movements. However, due to the additional complexity of controlling the stepper motors, and the fact that the hardware was not the main focus of the project, we decided to go with the standard gear motor drive mechanism.

2.3.2 Power Supply Unit

We initially planned to use a battery to power the entire robot. This battery would be mounted on top of the robot making its center of gravity high. However, we quickly realized that this made the robot very heavy and difficult to balance. We removed the batteries and built a compact voltage regulator unit to connect to a power supply and regulate the voltage to the levels required by the microcontroller and the motors.

We ran the robot at 6V. Voltage regulators stepped this voltage down to the 3.3V required by the Cricket board. The motors were run at the full 6V that was supplied.

2.3.3 Chassis

The chassis was built of a single plate of 1/8” aluminum. This made the robot very light. The aluminum was also very easy to cut and modify. This was especially useful as the robot was very much a prototype and required many modifications to the chassis.

Holes were cut out on top of the chassis to allow for balancing screws to be mounted. These were screws with weights attached to one end that, by screwing or unscrewing could be used to shift the center of mass to one side or the other.

2.4 Software Implementation

The software was our main deliverable for the project. We spent a considerable amount of time on this component of the project. The main part of the software is the control loop. The control loop is an infinite loop that reads data, processes it and then sets the motor voltages. The loop then repeats this process indefinitely. A typical control loop is shown in Figure 6.

The software was coded in plain C. The ICC AVR integrated environment was used for development. The sponsor’s provided programmer was used to load code into the microcontroller. Windows HyperTerminal was used to communicate with the robot over USB.

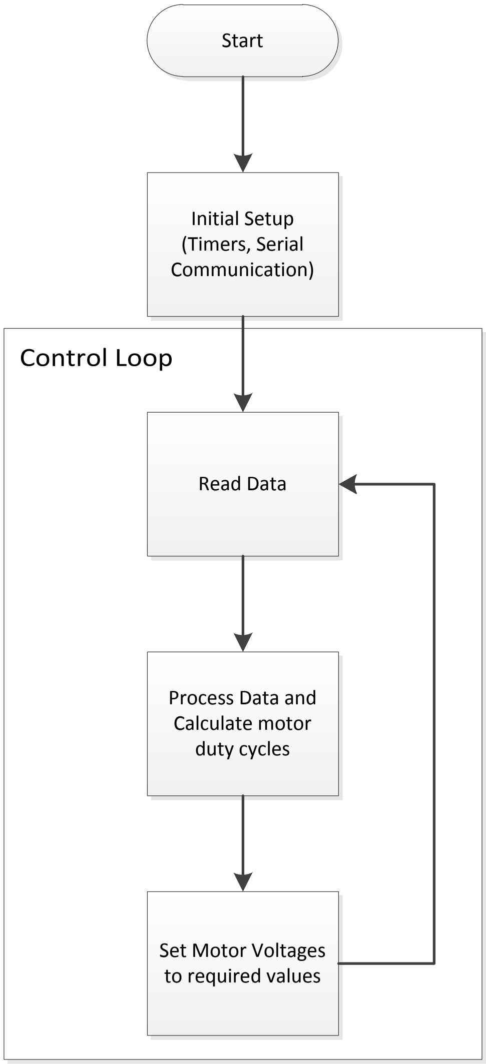
Start
Initial Setup (Timers, Serial Communication)
Control Loop
Read Data
Process Data and Calculate motor duty cycles
Set Motor Voltages to required values

FIGURE 6: A TYPICAL CONTROL LOOP

2.4.1 Control Algorithm

The sensor fusion method chosen was the complementary filter described in the previous section. Both the gyroscopes and the accelerometers are connected through 10-bit ADCs. This allows 1024 steps of measurement. This is more than enough precision for this specific purpose. Although there are 6 sensors present on the Cricket board, only two accelerometers and one gyroscope were used.

Single precision floating point variables were used for all the calculations involved. The cost of using integer variables is much lower. However, even with the floats, the performance was quite adequate due to the computational simplicity of the complementary filter.

One trigonometric function was required- Arctan - to determine the pitch angle from the accelerometer readings. Initially, a look up table was considered. However, due to limited memory, the standard math library provided was used for this function. We only used this function once per iteration of the control loop; hence the performance hit associated with it is negligible.

No dynamic memory is allocated. All variables are local variables stored on the stack. This allows the algorithm to achieve high performance.

The performance of the control loop was the main priority in designing the algorithm. The faster the loop, the more accurate the gyro integration will be. The loop needed to run at least 100Hz to offer an accurate tracking of the pitch angle. After some optimization, we could easily get the control loop to run at around 500Hz which was quite sufficient for the balancing application.

2.4.2 Software PWM

The very first module we designed was a software PWM system. The cricket board did not have two independent hardware PWM lines; hence we had to implement them in software. This meant keeping track of time and then turning the motors on and off depending on the required duty cycle. We initially implemented a blocking PWM system, where the PWM system would become part of the control algorithm itself as shown in Figure 7.

This meant that the algorithm would have to “wait” until a PWM cycle completed. This method would thus block execution till the cycle completed. This situation was not ideal, especially since the gyroscope readings need to be integrated continuously.

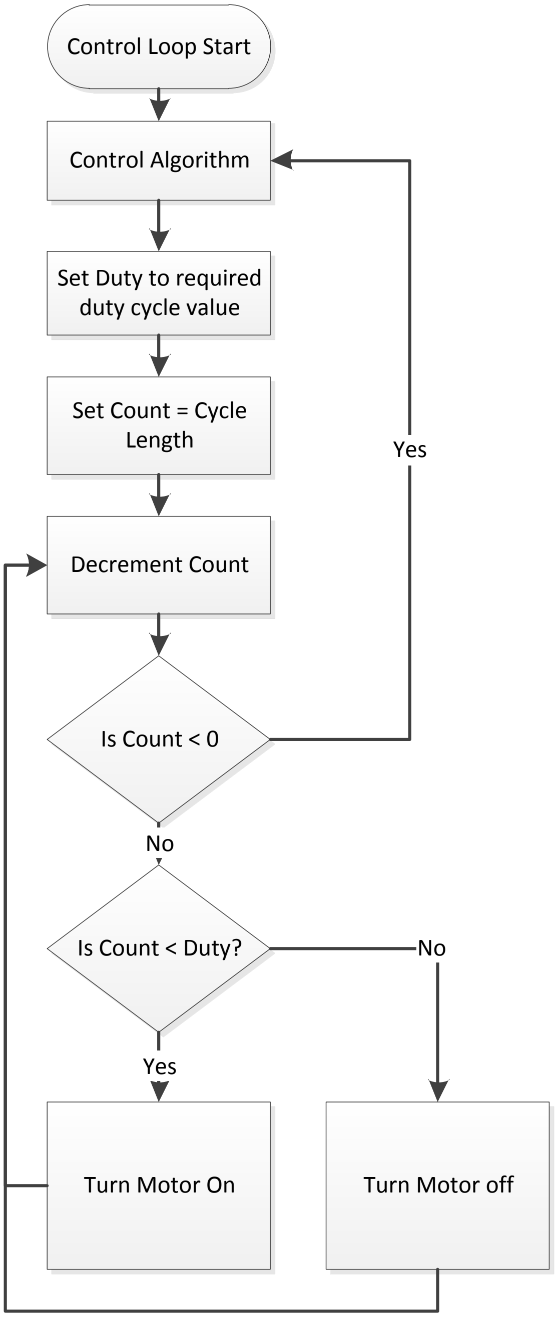
Control Loop Start
Control Algorithm
Set Duty to required duty cycle value
Set Count = Cycle Length
Decrement Count
Is Count < 0
Yes
No
Is Count < Duty?
No
Yes
Turn Motor On
Turn Motor off

FIGURE 7: BLOCKING PWM

An alternative solution is to use interrupt timers to do the PWM control. The cricket board has a 1ms timer that we can use to keep track of time. An interrupt service routine (ISR) would be called by the microcontroller at regular 1ms intervals. This ISR then keeps track of the duty cycle and appropriately switches the motors on or off. The ISR and the main control loop are illustrated in Figure 8.

The main advantage of this method is that the control algorithm can now operate independently of the PWM loops. We used a cycle period of 16ms with duty cycles steps of 6.25%. This allowed plenty of control over the speed of the motors.

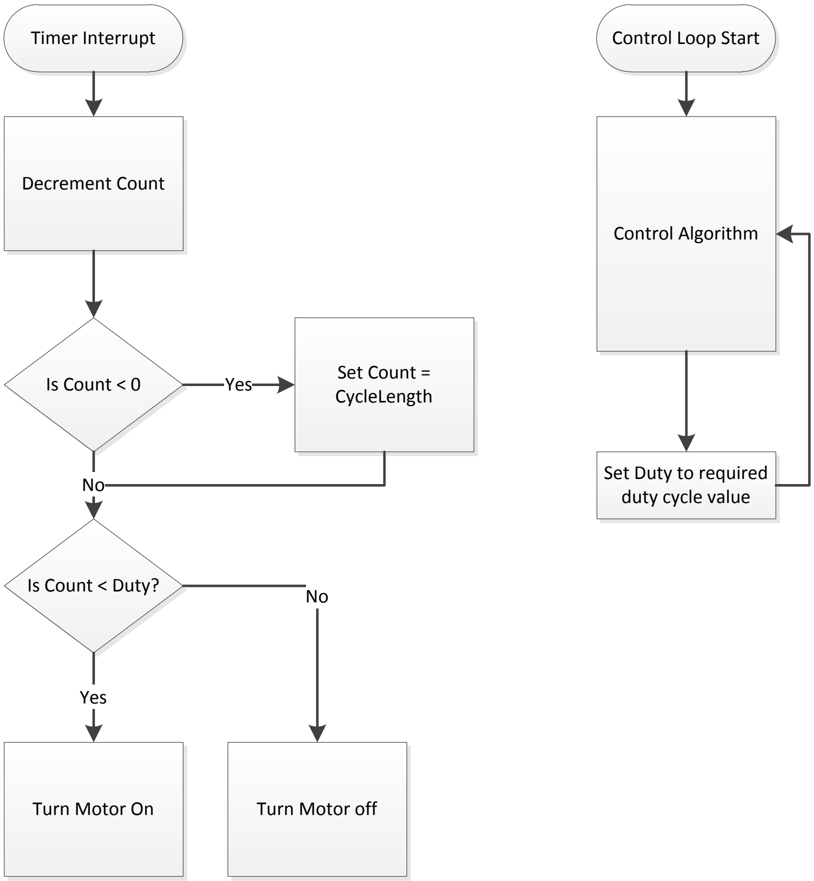
Timer Interrupt
Decrement Count
Is Count < 0
Yes
Set Count = CycleLength
No
Is Count < Duty?
No
Yes
Turn Motor On
Turn Motor off
Control Loop Start
Control Algorithm
Set Duty to required duty cycle value

FIGURE 8: NON-BLOCKING PWM

2.4.3 Serial Communication

To debug the algorithm, we had to devise a way to communicate with the robot. The cricket board has a serial communication interface that we could send debug output to. We used the code provided by the sponsor to implement a communication mechanism.

This mechanism scanned the serial interface for input at the end of the control loop and then displayed the state variables such as current duty cycle, pitch angle, etc. We later expanded this interface to allow real-time tuning of the PID values.

2.4.4 Discrete PID Controller

The output of the complementary filter is the pitch angle. Using this pitch angle, an error value was calculated and fed into a PID controller (Figure 9) to output a motor duty cycle. The PID controller has 3 components:

Proportional gain

This gain directly relates to the current error to the current motor output. This forms the bulk of the gain in the controller and reacts instantly to any error.

Integral gain

This gain compensates for any steady state error caused by drift or the offsets in the center of mass of the robot. It works by integrating the values of error over a period of time and hence reacts to steady state error after it accumulates.

Derivative gain

This gain uses the derivative of the error to reduce the overshoot of the output.

The three gains, kP, kI and kD are tunable parameters and must be adjusted according to the specific hardware used. There are very many ways to tune these parameters to an optimal value. We used the manual method to do this. First, we set all the gains to 0. Then kP is increased until the response oscillates. Then kP is set to half its value. kI is now increased until the steady state error is corrected in sufficient time. Finally, kD is increased to minimize overshoot.

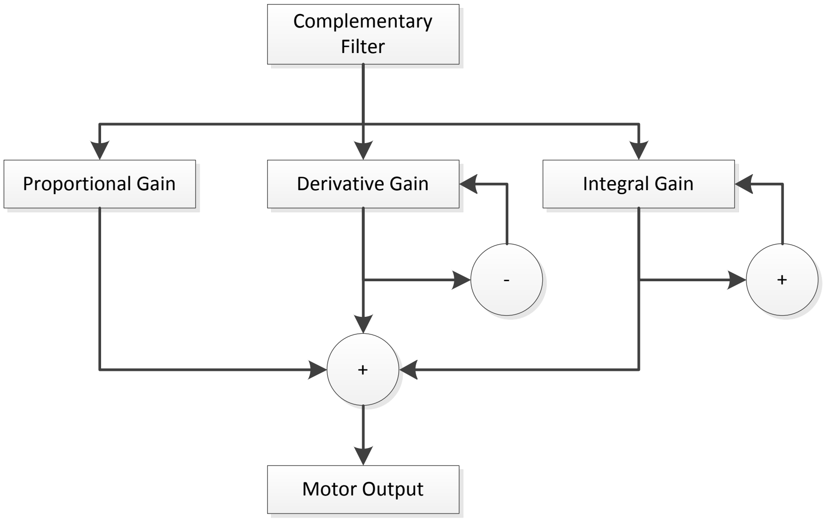

FIGURE 9: PID CONTROLLER

2.5 Testing Methods

Testing was mostly done on an ad-hoc basis. Any code change was immediately tested and modified. Data was also occasionally collected and then analyzed on the computer using Excel to reveal any obvious problems with the algorithm.

2.5.1 Complementary Filter

There was no real good way to get live data from the robot while it was running. The only way was to output the data through the serial communication interface and this too was quite limited. The Cricket Board used a baud rate of 38400. This means that we could send 4800 characters or 1200 floating point numbers across to the computer in a second. However, with our control loop running at around 500Hz, we could only send over 2 numbers in real time. This also meant that our control loop would be effectively blocked while the data was being transmitted.

The code for the control algorithm was written in standard C. Hence we could easily develop a desktop version of the same code. We opted to simply record only the raw gyroscope and accelerometer data from the cricket board and then run the data through the desktop version of the control algorithm to see how it performed. This way, we could judge the accuracy of the sensor algorithm. The outcome of this analysis is discussed in the results section.

2.5.2 PID Controller

The PID controller testing was manual. After each retune of the controller, we would subjectively assess the stability of the robot. Since the tuning procedure was quite simple, this was not a problem. However, we did note down occasionally, the jitter, oscillation frequency and the amplitude by using a ruler and a stopwatch. These measurements were just to provide a general indication of the performance of the algorithm for tuning it even further. Good PID settings were logged in the log book before being changed.

2.5.3 Robot Hardware

Since the hardware was not the focus of the project, we did not do much testing of it. Basic checks, using a voltmeter and an oscilloscope, were done to ensure proper functionality of the H-bridges and the power supply units. The motor and gearboxes came in as pre-made parts; hence there was no requirement to test those. The motor's stall and no-load current draws were measured to be 0.8A and 4A respectively. This was done to ensure that they did not burn out the H-Bridge that was being used.

2.6 Results

This section describes in detail the results of the project. It provides a qualitative assessment as well as quantitative data to support that assessment.

2.6.1 Control Loop Performance

The control loop frequency is the largest number of times the control loop can run in one second. The higher this value is, the better the accuracy of the sensor integration and the better the balancing.

Our initial target for the control loop frequency was 100Hz. This target allows 100ms time to receive sensor data, process it, and output the motor duty cycle. Upon implementing the code, we discovered that the algorithm could easily run at over 500Hz. The maximum that we recorded was 530Hz. The control loop frequency varies by ±20Hz depending on the interrupt mechanisms used by the software PWM.

This was much better than expected and it allowed us to add even more complexity to the loop. We added in a serial communications interface that allowed us to tune the PID values in real-time via a serial cable. Adding this interface brought the control loop frequency to around 490Hz – which was still much higher than the required target.

2.6.2 Complementary Filter Accuracy

The complementary filter was very successful at combining the readings from the gyroscope and the accelerometers while eliminating drift.

Data was collected from the robot at a frequency of 50Hz. The robot was rotated on either side. The robot was also moved horizontal to the ground to simulate acceleration without rotation. The data collected was then run through the complementary filter on the desktop machine and the results were plotted using Excel. As can be seen from Figure 10, the filter successfully eliminates all gyroscopic drift. The filter handles horizontal accelerations very well. A fair degree of noise cancellation is also achieved due to the 4th order Runga-Kutta integration technique.

As can be seen from the plots, the accelerometer records a lot of jitter and vibration, even when the robot is not rotating. When these readings are processed, the estimate for the angle has quite a bit of variation within a short period.

The gyroscope, however, records accurately the rotation but has the problem of the drift as shown in the plot.

Combining the two sensors yields the filtered estimate which has none of the problems of the two individual sensors.

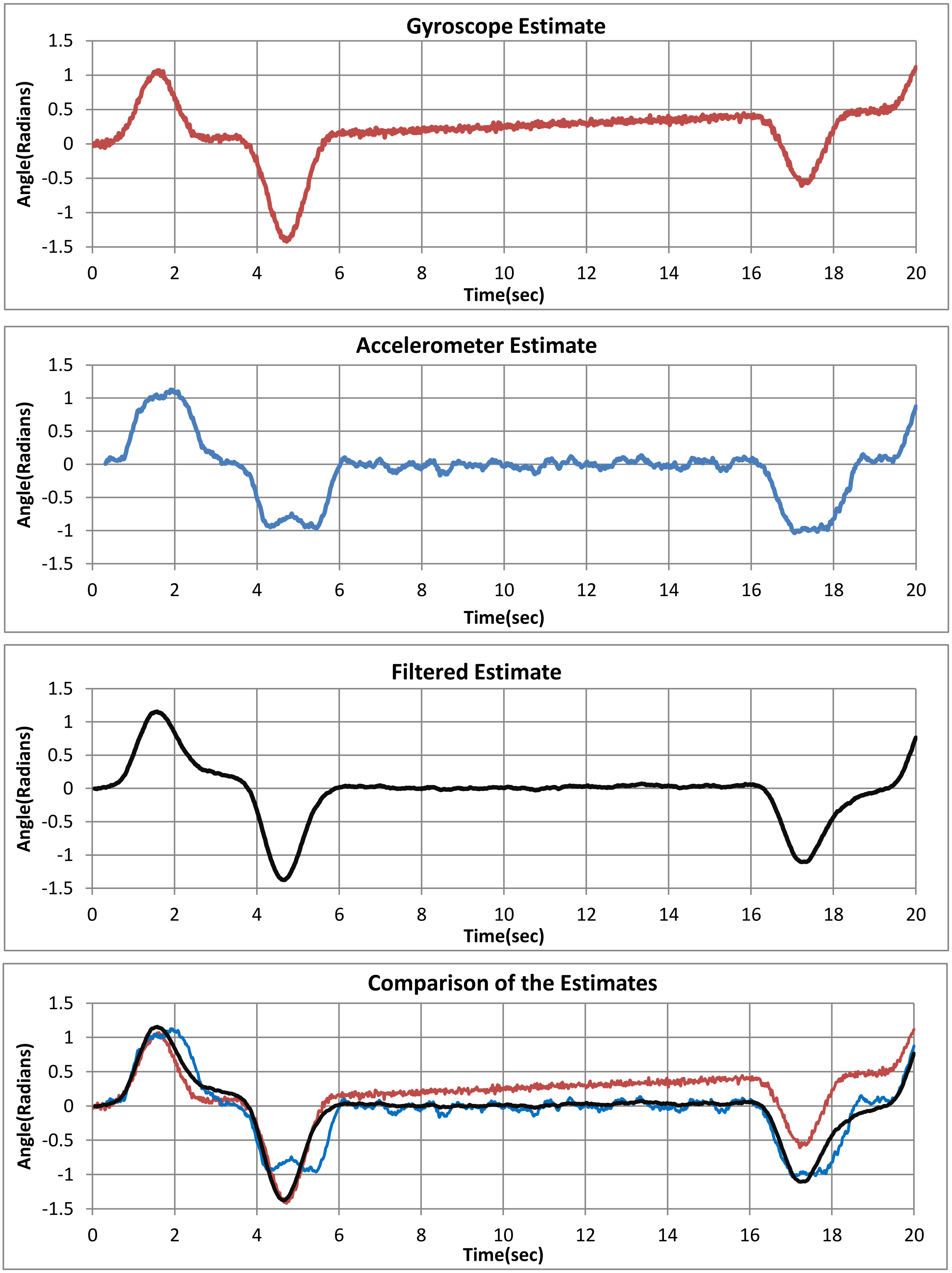

FIGURE 10: COMPLEMENTARY FILTER RESULTS

2.6.3 Balancing Performance

We set about tuning the PID controller to achieve the best balance. With only the proportional gain, we managed to get the robot to oscillate about the vertical with a jitter of around 3cm ± 1cm measured at the top of the robot. The robot could balance completely on its own for about 15 seconds. After this time, the steady state error and oscillation amplitude grew exponentially which would make the robot fall on one side.

Reducing the proportional gain and adding in the integral gain allowed the oscillation amplitude to be dampened. The robot now would now balance with minimal jitter of less than 1cm. The robot also responded to external disturbances (pushing with a finger) very well. The robot regained balance very quickly (within 1 second). However, an excessive disturbance caused it to oscillate wildly and then fall over. Adding in the derivative gain reduced the overshoot. At this state in the tuning, one effect that was very noticeable was that the robot could follow a finger with minimal force.

Further tuning would allow the robot to balance indefinitely. However, there was a lot of play in the gear motors we were using. Hence further tuning did not affect the balance performance. Moreover, the sponsor had plans to build a proper, large scale version of the robot, with precisely controllable stepper motors. This robot will need a complete retune of the PID parameters anyway; hence we decided not to spend more time on tuning our prototype version.

2.6.4 Software Code

The main deliverable of the project, the source code was commented clearly indicating all the locations of major modules, variables and tunable parameters. Changes to the software or the hardware can be easily made since the code was designed to be fairly abstracted and modularized. This is especially important as the Cricket board will undergo several revisions before it goes to the market and will almost certainly require changes to the code. The source code is packaged as a zip file with a readme document attached to it.

2.7 Discussion of Results

Overall, we were satisfied with the results of the project. We had completed our deliverable successfully. The performance of the complementary filter and the control algorithm exceeded our initial expectations.

Further improvements would require a better hardware platform to test and tune the parameters on. A direct drive system with precise control would be an ideal drive mechanism. The sponsor has indicated plans towards building a larger robot with stepper motors with a new revision of the Cricket board. This new revision will have more accurate gyroscopes, accelerometers and also an additional magnetometer for absolute positioning. This move would definitely allow more tweaking and tuning of the algorithm.

3. Conclusion

A precise and accurate sensor filtering mechanism was designed based on a complementary filter to determine the pitch angle. The filter successfully eliminated gyroscopic drift and sensor noise.

The implementation of the control loop on the Cricket performed very well running at 530 Hz (± 20 Hz). The software PWM, implemented through interrupt service routines of the Cricket controller, allowed PWM control with a period of 16 milliseconds and duty cycle steps of 6.25%.

The control loop and the PID controller acted as a closed loop feedback mechanism, in order to balance the robot with a maximum oscillation amplitude of 1 cm (± 0.3 cm) and a balance time of approximately 15 seconds.

4. Project Deliverables

The following are the deliverables of the project.

4.1 List of Deliverables

There is one main deliverable - the source code for the control algorithm. The secondary deliverable is the completed robot prototype.

4.1.1 Completed Self-Balance Code

The completed code-base includes all the implementations of the self-balancing algorithm as described in the Discussion section of this report. The code can be re-used on any self-balancing platform which uses the Cricket controller. The code would be submitted to the sponsor electronically via a zip file.

4.1.2 Robot Hardware

The robot that was built during the project period to test the self-balancing algorithm would be submitted to the sponsor along with the newly-built H-bridge circuit, the Cricket board, the USB10 connector board, and the parallel port programming board.

4.2 Financial Summary

Most of the materials were provided by either the sponsor or by the project lab. However, a few parts had to be ordered over the internet.

TABLE 1: FINANCIAL SUMMARY

#	Description	Quantity	Vendor	Cost(CAD)	Purchased by:	To be funded by:
1	GM3 Robot Motors	2	SolarBotics	$50.29	Sid	Sponsor
2	Tamiya Sport Wheel and Tire Sets	2 Sets				
3	LM298 Hbridge Chip	1				

5. Recommendations

A final set of recommendations is listed below:

5.1 Hardware

Hardware PWM control is a must-have in these types of robots. The software PWM performs well but with its long cycle length, it is far from ideal. The software PWM had a cycle frequency of about 60Hz. A hardware PWM line would allow a cycle frequency of many KHz, offering much more precise and efficient control. Dual hardware lines would allow independent control of both motors to turn the robot.

The onboard H-bridges need to be upgraded to higher current versions. A current output of 4A is recommended. Many commercially available robot kit motors such as the GM2 and GM3 have current demands in this range. The non-necessity to build separate H-bridges will be very attractive to hobbyists and professionals.

A single power port powering both the motors and the Cricket board is also highly recommended.

5.2 Software

Simplified programming and serial communication to and from the PC is a must-have. The parallel port programmer we used was quite cumbersome and prone to timeouts and other failures. Serial communication using HyperTerminal worked well. However, a custom developed application that allows data collection as well would be very ideal. A single USB cable programming and serial communication solution would be optimal.

Use of a standardized development platform such as Arduino and Wiring is highly recommended. Currently, there exists a large community of developers for Arduino. Leveraging this would be greatly beneficial to developers seeking to make use of the Cricket.

A shift to a much more powerful environment such as .Net should also be investigated. There exist a few boards on the market now, such as the Netduino and the Fez, which allow development in C#/.Net. C# is a very powerful language and supporting it will allow developers to single-handedly develop very complex applications.

Appendix A: The Physics Model

The following 6 state variables are used to model the motion of the robot:

x_{RM}	straight line position	[m]
v_{RM}	straight line speed	[m/s]
θ_n	pitch angle	[rad]
ω_n	pitch rate	[rad/s]
δ	yaw angle	[rad]
$\dot{\delta}$	yaw rate	[rad/s]

The free body diagrams of the two wheels are shown below

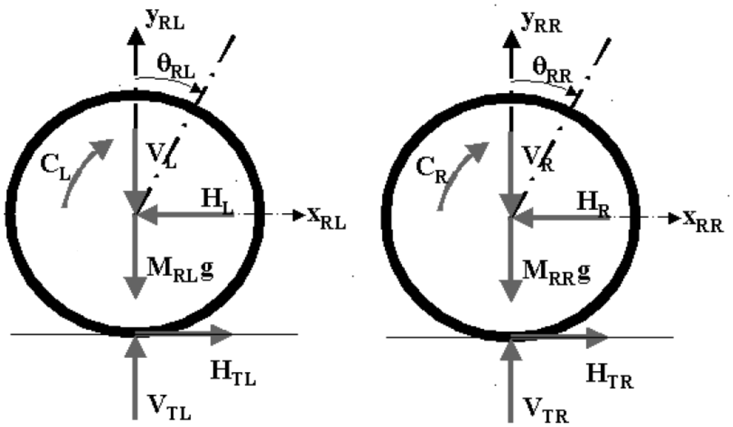

FIGURE 11: THE FREE BODY DIAGRAM OF THE TWO WHEELS (SOURCE: ANDERSON, 2003)

Where,

M_{RL} , M_{RR}	Mass of the rotating wheels	[kg]
$V_L, V_R, V_{TR}, V_{TR}, H_L, H_R$	The reaction forces	[$kgms^{-2}$]
θ_{RL}, θ_{RR}	Rotation angle	[rad]
$C_{L,} C_R$	Wheel torques	[kgm^2s^{-2}]
H_{TL}, H_{TR}	Friction forces	[$kgms^{-2}$]
R	Radius of the wheel	[m]

J_{RL}, J_{RR}	Moment of inertia of the wheels with respect to the z-axis	[kgm^2]

The equations of motion derived for the left wheel are shown below,

The forces about the x-axis:

$$\ddot{x}_{RL}M_{RL} = H_L + H_{TL}$$

$$\therefore\ H_L = -\ddot{x}_{RL}M_{RL} + H_{TL} \tag{1}$$

Similarly for the right wheel,

$$H_R = -\ddot{x}_{RR}M_{RR} + H_{TR} \tag{2}$$

The forces about the y-axis:

$$\ddot{y}_{RL}M_{RL} = V_{TL} - M_{RL}g - V_L$$

Since we assume that the wheels would always stay in contact with the ground and there would be no slip during the wheel's rotation (i.e no movement in the z-direction and no rotation about the x-axis), then $\ddot{y}_{RL} = 0$.

$$\therefore\ V_L = V_{TL} - M_{RL}g \tag{3}$$

Similarly for the right wheel

$$V_R = V_{TR} - M_{RR}g \tag{4}$$

Rotation equations:

$$\ddot{\theta}_{RL}J_{RL} = C_L - H_{TL}R$$

$$\ddot{\theta}_{RL} = (C_L - H_{TL}R)/J_{RL} \tag{5}$$

$$\ddot{\theta}_{RR}J_{RR} = C_R - H_{TR}R$$

$$\ddot{\theta}_{RR} = (C_R - H_{TR}R)/J_{RR} \tag{6}$$

The free body diagram of the chassis is shown in the figure below.

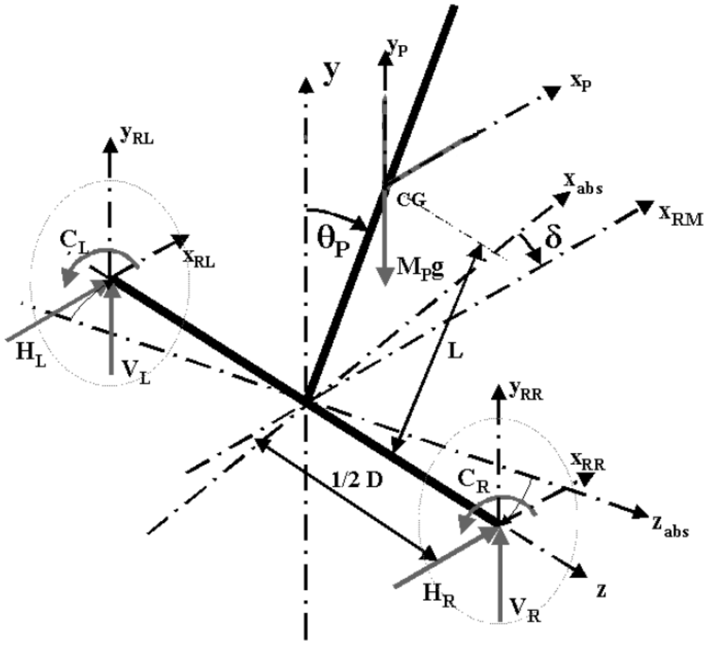
y_{RL}
C_L
x_{RL}
H_L
V_L
y
y_P
x_P
CG
θ_P
M_Pg
x_{abs}
δ
x_{RM}
L
y_{RR}
x_{RR}
C_R
1/2 D
z_{abs}
H_R
V_R
z

Figure 12: The free body diagram of the chassis (Source: Anderson, 2003)

Where,

M_P	Mass of the chassis	[kg]
D	The horizontal distance between the two wheels	[m]
CG	The center of gravity of the chassis	
L	The distance between the z-axis and CG	[m]
$J_{P\theta}$	Moment of inertia of the chassis with respect to the z-axis	[kgm^2]
$J_{P\delta}$	Moment of inertia of the chassis with respect to the y-axis	[kgm^2]

The forces about the x-axis:

$$\ddot{x}_P M_P = H_R + H_L \tag{7}$$

The forces about the y-axis:

$$\ddot{y}_P M_P + M_P g = V_R + V_L \tag{8}$$

Rotation at CG about the z-axis:

$$\ddot{\theta}_P J_{P\theta} = (V_R + V_L)L\sin\theta_P - (H_R + H_L)L\cos\theta_P - (C_L + C_R)$$

By substituting equations 1, 2, 3 and 4 and by rearranging the equation:

$$(C_L + C_R) = (V_{TR} + V_{TL} - (M_{RR} + M_{RL}))gL\sin\theta_P - \big(H_{TR} + H_{TL} - (\ddot{x}_{RR}M_{RR} + \ddot{x}_{RL}M_{RL})\big)L\cos\theta_P - \ddot{\theta}_P J_{P\theta}$$

Since θ_P would be small around the operating point of the robot, linearizing the above equation using the following first order small angle approximations - $\sin\theta_P \approx \theta_P$ and $\cos\theta_P \approx 1$ and by rearranging:

$$(C_L + C_R) = -\big(J_{P\theta}\ddot{\theta}_P - (V_{TR} + V_{TL} - (M_{RR} + M_{RL}))gL\,\theta_P + \big(H_{TR} + H_{TL} - (\ddot{x}_{RR}M_{RR} + \ddot{x}_{RL}M_{RL})\big)L\big)$$

Substituting $\ddot{x}_{RR} = R\ddot{\theta}_{RR}$ and $\ddot{x}_{RL} = R\ddot{\theta}_{RL}$ and equations 5 and 6 into the above equation and by rearranging, equation R$_1$ is obtained. Since equation R$_1$ models the robot when it's balanced at a stationary position the coefficient of static friction μ_S can be used to relate V_{TR}, V_{TL} and H_{TR}, H_{TL} by the equations $H_{TR(\max)} = \mu_s V_{TR}$ and $H_{TL(max)} = \mu_s V_{TL}$.

$$\boldsymbol{A_1 C_L + A_2 C_R = -\big(B_1\ddot{\theta}_P + B_2\,\theta_P + B_3\big)} \tag{R$_1$}$$

Where,

$$A_1 = \left(1 + \frac{M_{RR}RL}{J_{RR}}\right)$$

$$A_2 = \left(1 + \frac{M_{RL}RL}{J_{RL}}\right)$$

$$B_1 = J_{P\theta}$$

$$B_2 = (V_{TR} + V_{TL} - (M_{RR} + M_{RL}))gL$$

$$B_3 = \mu_s L\left(V_{TR}\left(1 + \frac{R^2 M_{RL}}{J_{RL}}\right) + V_{TL}\left(1 + \frac{R^2 M_{RR}}{J_{RR}}\right)\right)$$

Rotation at CG about the y-axis:

$$\ddot{\delta} J_{P\delta} = (H_L - H_R)\frac{D}{2}$$

By substituting equations 1 and 2,

$$\ddot{\delta} J_{P\delta} = (\ddot{x}_{RR}M_{RR} - \ddot{x}_{RL}M_{RL} + H_{TL} - H_{TR})\frac{D}{2}$$

Substituting $\ddot{x}_{RR} = R\ddot{\theta}_{RR}$ and $\ddot{x}_{RL} = R\ddot{\theta}_{RL}$ and equations 5 and 6 into the above equation and by rearranging, equation R_2 is obtained. Since equation R_2 models the robot when it's turning about its y-axis while maintaining a balanced posture, the coefficient of friction μ would have a value between μ_K (coefficient of kinetic friction) and μ_S (coefficient of static friction), which is used to relate V_{TR}, V_{TL} and H_{TR}, H_{TL} by the equations $H_{TR} = \mu V_{TR}$ and $H_{TL} = \mu V_{TL}$.

Even though the moment of inertia $J_{P\delta}$ depends on the angular position θ_P of the chassis, since we are only considering small deviations of θ_P around $\theta_P = 0$ we can assume $J_{P\delta}$ to be constant at $\theta_P = 0$.

$$\ddot{\delta}J_{P\delta} = (R\ddot{\theta}_{RR}M_{RR} - R\ddot{\theta}_{RL}M_{RL} + H_{TL} - H_{TR})\frac{D}{2}$$

$$\boldsymbol{A_3C_L + A_4C_R = -\left(B_4\ddot{\delta} + B_5\right)} \qquad \textbf{(R}_2\textbf{)}$$

Where,

$$A_3 = \frac{RDM_{RL}}{2J_{RL}}$$

$$A_4 = -\frac{RDM_{RR}}{2J_{RR}}$$

$$B_4 = J_{P\delta}$$

$$B_5 = \mu\left(V_{TR}\left(1 + \frac{R^2DM_{RR}}{2J_{RR}}\right) - V_{TL}\left(1 + \frac{R^2DM_{RL}}{2J_{RL}}\right)\right)$$

Bibliography

Anderson, D. P. (2003, May 19). *nBot Balancing Robot*. Retrieved March 6, 2011, from SMU Geology: http://www.geology.smu.edu/~dpa-www/robo/nbot/

Colton, S. (2007, June 25). *The Balance Filter.* Retrieved April 2, 2011, from MIT Website: http://web.mit.edu/scolton/www/filter.pdf

View publication stats