UNIVERSITY OF CALIFORNIA, SAN DIEGO

Control and Steering of a Small 3D Printed Rover and its Applications to Engineering Education

A Thesis submitted in partial satisfaction of the requirements for the degree Master of Science

in

Engineering Sciences (Mechanical Engineering)

by

Talesa Rene Bleything

Committee in Charge:

Professor Thomas Bewley, Chair

Professor Maurício de Oliveira

Professor Michael Tolley

2016

The Thesis of Talesa Rene Bleything is approved and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California, San Diego

2016

Dedication

Thank you Tom. For everything.

Table of Contents

List of Figures

List of Tables

ABSTRACT OF THE THESIS

Control and Steering of a Small 3D Printed Rover and its Applications to Engineering Education

by

Talesa Rene Bleything

Master of Science in Engineering Sciences (Mechanical Engineering)

University of California, San Diego, 2016

Professor Thomas Bewley, Chair

This thesis presents both the control design used to stabilize a small rover on two wheels as a mobile inverted pendulum, as well as the steering design used to achieve a series of unique stable driving modes. Additionally, the rover is assessed as an educational platform for use in teaching various STEM topics at both the high school and undergraduate university levels. The vehicle, termed BeagleRover, uses four DC motors and four servo motors to independently steer each wheel and achieve a total of six different driving modes, four of which are inherently stable four-wheel drive modes while two are unstable two-wheel drives modes. Two of the four stable drive modes

implement Ackermann steering geometry to reduce side slip when driving around a turn. Experimental results using measurements from an onboard gyroscope suggest that side slip is indeed reduced by this method. Stability in the two unstable drive modes is achieved through classical control methods including lead and lag control. Complementary filtering of gyroscope and accelerometer measurements is used to derive accurate body position data for use in feedback. In support of current STEM education trends, this thesis provides a detailed solution set to the steering, control and filtering problems for potential use in STEM course material.

Chapter 1 Introduction

The purpose of this thesis is twofold. First, it is to apply classical and digital control theory to control a four-wheel steerable rover, referred to as BeagleRover, with the ultimate goal of achieving a unique combination of various four-wheel drive modes, two-wheel balance modes and the transition between four-wheel and two-wheel driving and back. BeagleRover is pictured in Figure 1.1. Second, it is to provide support material and lay groundwork for the design of Science, Technology, Engineering and Mathematics (STEM) curricula centered on BeagleRover and two other educational robots termed BeagleMiP, a miniature Segway-like mobile inverted pendulum, and BeagleMav, a small hexacopter. These three together comprise the EduLine, a line of educational robots designed for use in the classroom or as commercial products. The potential impact of this work lies mainly in furthering advances in engineering and other STEM fields by contributing to the advancement of science and engineering education. It will be demonstrated that BeagleRover is well positioned to further this advancement as an educational robotics platform.

1.1 Research Motivation

According to a recent study out of the Construction Engineering Department at the University of Central Florida, the number of students pursuing STEM disciplines in the United States has decreased by 18% over the last two decades, while the number of undergraduate students specifically pursuing engineering degrees continues to follow a decreasing trend [1]. Some

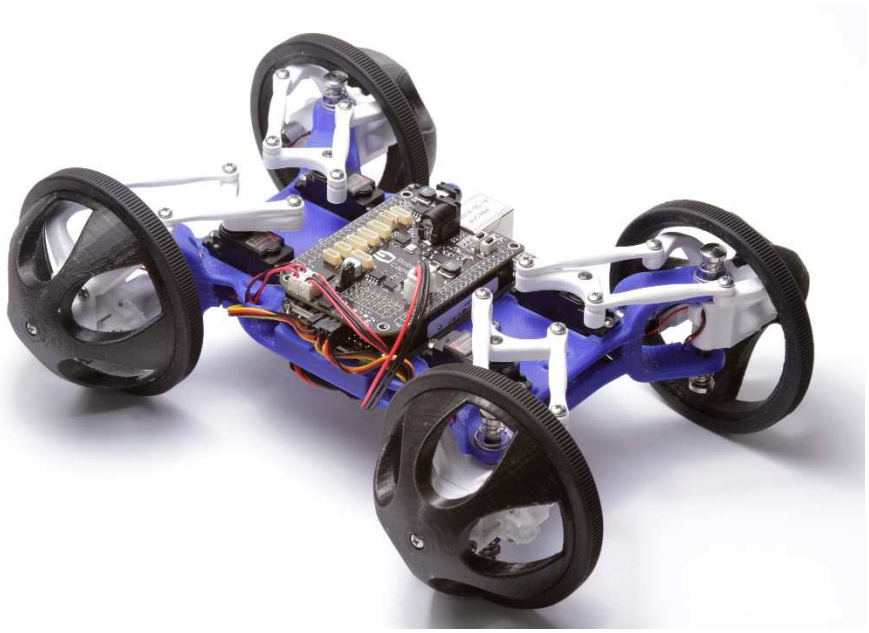

Figure 1.1: Image of the most current version of BeagleRover.

researchers believe the problem lies in attracting students to the field of engineering while others believe the bigger issue is retaining students throughout their studies and engaging them in the learning process, pointing to the fact that only 40% of students who choose STEM actually end up with a STEM degree [1]. Whether the root cause of the decline in STEM degrees is initial attraction or retention, it has educators motivated and a trend in the proposed solutions to this challenge has emerged. This trend can be quickly summarized as a transition away from traditional, lecture-based, instructor-focused classrooms toward more non-traditional, application-based, student-focused approaches where hands-on lessons are emphasized. Accordingly, this is creating the need for more educational platforms that can support these active learning, hands-on approaches.

Traditional approaches are defined by an instructor-focused setting in which information transmission is the primary goal and students take a largely

passive role. According to a literature review published in the Journal of Engineering Education in 2015, this results in surface learning approaches to studying and limited understanding [2]. In contrast, more and more evidence in the engineering and science education literature suggests a more student-focused, active environment, in which students engage physically with the activity in some way, leads to a deeper approach to studying and more thorough understanding [2]. This can potentially result in the student feeling more successful and improving retention. Consequently, teachers are being increasingly encouraged to implement a more hands-on and active curriculum in various STEM fields at the high school and higher education levels, where active learning approaches become increasingly rare as courses are transitioned to traditional lecture based techniques [3]. It is this understanding of the push toward and the efficacy of active learning approaches that motivates the application of BeagleRover to STEM education. This paper presents the technical development of an engaging physical platform and accompanying support material that together make the building blocks for compelling hands-on and active learning curricula at both the high school and undergraduate levels.

1.2 Research Scope

As previously stated, the ultimate goal of this research is to further the field of engineering by developing an engaging, dynamic, educational robotics platform that can be used to encourage student involvement in various areas of science and engineering, ultimately leading to an associated

career. This has resulted in a multidisciplinary project, integrating technical development and implementation while consistently documenting and designing from the educational perspective. The technical development of BeagleRover was accomplished using classical and digital control techniques, implemented exclusively in a multithreaded c-programming environment. These techniques were applied to an existing mechanical design, small changes being suggested along the way as the use case became more and more clear.

The major technical additions to a system previously capable of driving on four wheels in four simple drive modes include implementation of Ackermann steering geometry, control design to achieve balancing and driving on two wheels and the transition between driving on four wheels to driving on two wheels. The peak result being the ability of the vehicle to drive "up the wall," with sufficient friction between the tires and ground, in order to upright itself and drive away from the wall while balancing indefinitely on two wheels. This work provides the steering solution as well as the balance control solution for the maneuvers listed above. Additionally, switching from the use of Euler angles to implementation of a complementary filter for accurate state estimation is explained.

From the educational perspective, almost no support material written *exclusively* for teaching a dispersible course or lesson around the EduLine existed prior to the formal initiation of this thesis work. A major contribution of this work is a text we are currently calling BeagleBone Robotics (BBR), designed

to serve as the lab text for MAE 143C, Digital Control Systems, taught at UCSD and centering on BeagleMiP. BeagleBone Robotics is a three-part text. The first two parts include instructions for getting started with the hardware, robot assembly, and recommended homework exercises and solutions that range from getting started in Linux to PCB design using the free software Eagle. The third part is dedicated to control design. Currently the control design portion is written based on BeagleRover, as control design and implementation for the second robot in the EduLine family is the main technical focus of this paper.

1.3 Thesis Outline

This thesis is split into two parts in order to best illustrate its dual nature. Chapter two discusses related work while chapters three, four and five are mostly quantitative, dedicated to the technical development and implementation of the control and steering solutions. Chapters six and seven present the support material developed around the EduLine as a result of this thesis work. Chapters six and seven will be referred to as part two while chapters three, four and five will be referred to as part one. Chapters three, four and five also contain additional information and derivations that are not currently included in the support material but will be useful as such in the future.

Part one begins by explaining the steering mechanism used, although it should be made clear that the mechanical design existed prior to this work. The four different stable drive modes currently implemented on BeagleRover are then explained, demonstrating the highly engaging nature of the vehicle. This is followed by an explanation of Ackermann steering geometry which is

implemented in order to achieve a more efficient and maneuverable vehicle as well as to provide material that can be used in high school level coursework.

Following the discussion of the steering mechanism used and the different stable drive modes is the control solution for balancing BeagleRover on two wheels in its unstable drive mode. The control solution is the main theoretical focus of this paper. This section is mirrored in chapter six, where the theoretical portion of BeagleBone Robotics is presented, which is one of the primary contributions of this thesis work. Chapter six makes up the bulk of part two of this paper and is written to serve as a detailed solution set for the mobile inverted pendulum problem as taught in UCSD's Digital Control course, MAE143C.

The final chapter of part one details implementation of the steering and control solution in hardware and discusses BeagleRover's most dynamic maneuver, the transition between driving on four wheels to balancing on two wheels by driving up a wall. Finally, suggestions for future work and conclusions are given in chapters eight and nine respectively. The application of BeagleRover and the EduLine as a whole to high school level curricula is left to future work and as such is discussed in the future works section.

Chapter 2 Related Work

2.1 Current Trends in Science and Engineering Education

This section is not intended to be a comprehensive literature review of science and engineering education in itself but rather to highlight some meaningful examples of recent advancements in non-traditional classroom settings and set the stage for how BeagleRover may fit into these trends. However, all of the examples cited in section 2.1 have conducted comprehensive literature reviews and or specific case studies and therefor provide useful insight into how BeagleRover, and by extension the EduLine as a whole, may contribute to the advancement of engineering education. To this end, three example studies will be highlighted.

2.1.1 Example One: Flipped Classroom Model

There has recently been a surge in the popularity of the flipped, or inverted, classroom model [2]. The flipped classroom model is one in which technical knowledge is gained primarily through online videos prepared by the instructor and class time is used for problem solving through peer interaction, during which the instructor acts primarily as a guide. As opposed to the traditional classroom lecture model, the goal of the flipped classroom model is to free up class time for hands-on work and application of theory to real world problems.

In a study conducted by Butler, Zappe and Mahoney (2015), data from a comprehensive literature review on the current use of the flipped classroom

model in engineering education is inconclusive when using exam scores as the metric by which to measure success. Empirical evidence from a case study across three semesters of an undergraduate environmental engineering course (80-90 students per semester), during which the transition is made from a traditional lecture based format through two iterations of a flipped classroom design, similarly reports no significant improvement in cumulative exam scores. However, the students report a much higher satisfaction with the course via a student survey and 77% of students who engaged in version two of the flipped course agree that they would rather take a flipped course with the same instructor than a traditional course [2].

2.1.2 Example 2: Active Learning in a High School Pre-Calculus Course

As part of a National Science Foundation STEM grant, entitled Science and Technology Enhancement Program (Project STEP), graduate engineering students from the University of Cincinnati joined high school classrooms for 10 hours a week for the duration of a full academic year (Project STEP 2010). Under Project STEP, the graduate students created interactive, project-based high school lesson plans, meeting state science and math standards. The goal was to increase interest in STEM fields, stating that strong academic preparation in high school is more likely to lead to STEM degree completion [3].

The lesson of focus here, Shaking Up Pre-Calculus, written and led by Chelsea Sabo, a Robotics Postdoctoral Researcher at the University of Sheffield in the UK, teaches high school pre-calculus concepts using a miniature shake table. Students build structures to test using KNEX and analyze data using a Go!

Motion Sensor. The students were given pre and post assessments containing identical questions in order to assess whether or not the students learned the material. The results showed roughly 30% to 80% improvement on all questions from beginning of the class to end of the class, suggesting the material was successfully delivered.

Additionally, students were given feedback forms in order to get a clearer picture of the impact of Shaking Up Pre-Calculus. Results of this survey show that approximately 70% of students responded favorably to the question of whether or not the lesson made them want to learn more about engineering. Results were similar for increasing confidence in science and math. All three of these results were similar to those of the same survey conducted in year two of the course.

2.1.3 Example 3: MIT Introduces the Maker Portfolio

As of 2014, MIT Admissions began accepting a "maker portfolio" as an equally weighted part of their admissions process. In a white paper prepared by MIT Admissions for the inaugural White House Maker Fair in 2014 [4, 5], the maker portfolio is described as "a powerful tool for helping admissions officers to identify, understand, evaluate and admit exceptionally skilled applicants whom a conventional selective admissions process might undervalue or "overlook altogether" [4]. An official decision by one of the leading tech universities in the country to value conventional assessment methods equally with "high maker potential" represents a profound paradigm shift toward the formal acceptance of non-traditional approaches to technical education.

Other universities may begin to follow suit, potentially widening the market for platforms such as BeagleRover that can be used in the high school STEM classroom, especially in a flipped classroom type of model.

2.2 Educational Robotics

Robotics construction kits designed for education have an approximately 50-60 year old history, arguably beginning with the Logo programming language developed by Seymour Papert in 1967 and the associated programmable Turtle robots [6, 7, 8, 9]. Since this introductory step, robotics in education has been ever increasing in popularity. This is especially true within the past couple decades as computation has continued to become cheaper and cheaper, enabling programmable robotics platforms to reach a wide range of classrooms and schools [9, 10]. This has led to what's been termed a "robotic revolution" [9]. Accordingly, a multitude of robotics platforms marketed as educational exist today, some more legitimate than others. If the EduLine is to become a viable educational platform, an understanding of successful educational robotics platforms is a must. An overview of several of these platforms is given next.

2.2.1 FIRST Robotics

For Inspiration and Recognition of Science and Technology (FIRST) is a not-for-profit public charity youth organization specializing in robotics education. The mission of FIRST is to inspire youth to become science and technology leaders by engaging kids from kindergarten to high school in mentor-based robotics programs [11]. Reaching over 400,000 students in 2015-

2016, FIRST is widely recognized as the "leading not-for-profit STEM engagement program for kids worldwide" [11, 12].

The FIRST program is competition based and is most well-known for its flagship competition, FIRST Robotics. FIRST Robotics Competition (FRC) is for high school students grades 9-12 and is the FIRST program that is most comparable to the skill set potentially taught at this level by the EduLine, specifically BeagleRover. In order to participate in FIRST Robotics, a student must be part of a team of at least 10 and must be willing to commit to demanding time requirements as the program is designed to operate in an afterschool setting, similar to a school sports team. The season last between six weeks and four months depending on how far a team advances in the competition [13]. The cost per team is anywhere from 5,000.00$ - 6,000.00$ plus travel, food and team shirts, with costs naturally increasing as teams stay in the competition longer [14]. Although FIRST offers resources to help teams with funding, the number one concern of team leaders is affordability, according to a study conducted by Brandeis University [15].

The competition itself requires a range of skill sets both technical and non-technical, and students need not have any prior experience to participate [13]. Teams receive a standard kit of parts that includes the mechanical and electrical components necessary to build that year's robots. The robots are industrial sized requiring access to a large space and a variety of machine shop power tools [13]. The students are responsible for the mechanical design as well as the programing of the robots in order successfully perform in a variety of field

games. The current hardware used centers around the roboRIO from National Instruments, a student robotics controller designed specifically with FRC in mind that combines a Xilinx FPGA and dual-core ARM Cortex-A9 processor. It can be programmed using LabVIEW graphical programming tools or C text based language [16]. Both the reconfigurable I/O architecture (RIO), LabVIEW software and C are used in industry for a range of applications and give students the opportunity to work with professional grade tools.

2.2.2 Lego Mindstorms

Lego Mindstorms Education EV3 is Lego's most current solution to teaching STEM concepts in the classroom. Unlike FISRT Robotics, the Lego Mindstorms EV3 approach is designed to be implemented during the regular school day and is aligned with Common Core State Standards and Next Generation Science Standards [17]. Educators can choose between three different packages which include the EV3 Core Solution, EV3 Curriculum Solution and EV3 Comprehensive Solution. All packages can teach a range of classroom sizes and come with the same hardware and software. The differences lies in the curriculum support that is provided. The Core Solution includes no curriculum while the Curriculum Solution comes with the EV3 Design Engineering Projects Curriculum which includes lesson plans and over 30 hours of instruction [17, 18]. The most in depth option is the EV3 Comprehensive solution which includes the EV3 Design Engineering Projects Curriculum, EV3 Science Curriculum and EV3 Coding Activities, all of which are supported by C-STEM curriculum (explained further below) and include lesson plans at varying

numbers of total teaching hours [19]. For a classroom of 30 students, the kits cost 5,999.95$, 6,299.95$ and 10,499.95$ in order of increasing value of the curriculum as listed above, and are designed to provide one set of hardware and software per every two students [17, 18, 19].

The hardware provided with all variants of the Lego Mindstorms Education EV3 kits includes a range of common sensors, motors, battery, cables, build instructions, Lego Technic building bricks and the EV3 intelligent brick. The combination of the EV3 brick, which is built around a 32-bit Arm9 processor and a Linux based operating system, with Lego Technic building bricks allows for many different types of robots to be built within this one system [20]. The programming environment is designed to follow a drag and drop paradigm and is based on LabVIEW. Although a historically limited programming environment, the Mindstorms EV3 can be programmed using standard industry and college level tools such as C and Java [21, 22].

The focus of the Mindstorms EV3 program is at the k-12 level, especially the pre, elementary and middle school levels, as that is the level of study that the provided EV3 curricula address (FIRST Lego League for elementary school students uses Lego Mindstorms technology). That being so, there is room for use of the platform at undergraduate college levels and there are many examples of this being done [21, 22, 23, 24, 25]. Out of these examples, few exhibit the use of Lego Mindstorms as a powerful teaching tool for advanced college coursework, with many more pointing to the utility of the platform for teaching freshman level introductory computer science and engineering courses [21, 22,

23, 25]. One study out of Iowa State University reported great success with using Lego Mindstorms for an introductory computer science course for non-majors, but declining success even at the programming I level, with failure at the programming II level, leading the author to conclude that the Mindstorms platform is only viable for a few courses [24]. Limitations for use at the college level are reported elsewhere as well, pointing to technical limitations as well as student opinion on the effectiveness of the platform at teaching certain concepts [21, 24].

2.2.3 C-STEM and Linkbots

Computing, Science, Technology, Engineering and Mathematics (C-STEM) is a program that focuses on STEM education at the K-14 grade levels through the use of computing and robotics. It comes out of the UC Davis Center for Integrated Computing and STEM Education started by Dr. Harry H. Cheng, professor in the Mechanical and Aerospace Engineering Department at UC Davis. Although lacking the same name recognition as FIRST and Lego Mindstorms, millions of dollars in grants and funding from the National Science Foundation (NSF) and California Department of Education has been allocated to the center to continue their research into how integrated computing and robotics STEM education in formal and non-formal settings affects student engagement and learning, especially in underrepresented and at-risk groups [26]. C-STEM curriculum aligns with Common Core State Standards and Career and Technical Education Standards and is a UC approved educational preparation program, a category on all UC college application forms [26, 27].

At the high school level, C-STEM holds A-G program status which satisfies UC/CSU admission requirements [28].

Not all C-STEM curricula require use of the robotics platform, but it is made optional for all courses. The robotics platform used is Linkbot by Barobo, an educational robotics company co-founded by Dr. Harry Cheng and one of his graduate students, and is designed for use with C-STEM curricula [29, 30]. Linkbots are modular robots with two degrees of freedom that can be snapped together to create different systems. These bots are designed to allow anyone to begin working with robots out of the box while providing a research grade platform simultaneously [30, 31]. Each Linkbot is a mobile robot in itself with two rotating face plates, an on board three-axis accelerometer and rechargeable lithium-ion battery [32]. Many additional sensors are available, as well as freely available CAD files for extending the mechanical design, including an accompanying Computer Aided Design and 3D printing curriculum [32].

The C-STEM curriculum centers on the C-STEM Studio software and Ch, a C/C++ interpreter created by Dr. Harry Chang [33]. C-STEM Studio is a software platform that integrates Ch, Linkbot Labs, Ch Linkbot Controller, Ch Mindstorms Package and Robot Controller for Lego Mindstorms NXT/EV3, RoboSim, RoboBlockly and ChDuino [34]. RoboBlockly and ChDuino are both C-STEM designed graphical user interfaces for Ch and Arduino respectively. C-STEM Studio is claimed to be the "only program in existence that can control Linkbots, NXT and EV3 in a single program with only a few lines of C/C++ code" [34]. The primary goal of Ch and C-STEM Studio within the C-STEM curriculum is to teach

computing and STEM concepts through introductory C/C++ programming concepts and robotics. Research into the effectiveness of the C-STEM educational platform in teaching computing and STEM concepts and or engaging students in further scientific studies is still underway.

At first glance the C-STEM program seems very accessible in terms of cost as the C-STEM Studio software and RoboBlockly are both free of charge, while the Ch software package only costs 300$ [35]. However the complete picture of the C-STEM model reveals that it is much more of a commitment for a school to start a C-STEM program. The main reason for this is that a school must become a "C-STEM school" before classrooms can gain access to the full curriculum. Becoming a C-STEM school costs an annual subscription of 600.00$-1,000.00$ and requires teacher training that ranges anywhere from 3,000.00$-15,000.00$ [35]. What's more, including the robotics package for 32 students adds another 5,599.00$ plus C-STEM textbooks at 20.00-50.00 each sold in sets of 25 copies or more [35]. If the primary concern of schools participating in FIRST robotics is cost, then the cost to initiate a C-STEM program will most certainly be a concern and a barrier for many schools as well.

2.3 Conclusions

The three programs discussed above, FIRST Robotics, Lego Mindstorms and C-STEM differ in their approaches to reaching students. FIRST Robotics is solidified in the after school realm while Lego Mindstorms and C-STEM both align with Common Core State Standards for integration into the traditional school day curriculum. While Lego Mindstorms and C-STEM have this in common, C-

STEM requires teacher training and a school-wide adoption of the program rather than the more class by class approach of Lego Mindstorms. Additionally, C-STEM has a focus on underserved schools and at-risk students. In terms of group activity, FIRST Robotics is heavily group work focused, providing a single parts kit per team, while the Lego Mindstorms program is designed to provide a robotics kit for every two students. C-STEM programs are designed to provide a robotics kit to each individual student although group work is encouraged.

Although differing in their approaches to reaching students, all of these programs have some key things in common. First, they are all fairly expensive to implement. Second, they are all primarily focused on and marketed toward absolute beginners to programming and robotics. This is true for the skill level of the students as well as the teachers and mentors that would be implementing the programs. Although some work has been done with Mindstorms at the college level, most of the evidence leans toward it not being effective beyond the most introductory courses. All of these platforms attempt to introduce students to professional grade tools (as will the EduLine), however none of them have a truly clear path from high school level curriculum to more advanced college level work contained within the platforms themselves. There is potential for the EduLine to fill this gap. This paper will in part show that the addition of EduRover to the EduLine family provides the opportunity to teach high school level STEM concepts while clearly demonstrating the bridge to advanced college level study that is necessary to fully master the concepts introduced by EduMiP. This can potentially inspire students to continue the pursuit of STEM

disciplines, especially engineering degrees, into college.

Another similarity of FIRST Robotics, Lego Mindstorms and C-STEM is the high cost of implementation. As the EduLine is still in its nascent stages as far as development of a formal curriculum is concerned, it remains to be seen how expensive it will be to implement on a per classroom basis. However, based on the costs outlined for the above programs, there is opportunity for the EduLine to affect a great number of students by offering an economically lower tier solution at the high school level. As BeagleRover is based on the 35.00$ BeagleBone Black, and in turn supported by the open source Linux community, while also being almost completely 3D printable, offering a lower cost educational robotics solution is highly possible. And if achieved will thereby increase the accessibility to schools. What's more, for the same reasons, it is promising that each student in a classroom could be provided the opportunity to work with their own robot through the EduLine while still being more economically viable than FIRST, Mindstorms or C-STEM. The potential impact of working on shared robots being lowering the hands-on engineering work done by each individual student and their sense of accomplishment at the end of a course.

It should also be noted that BeagleRover is a very unique physical platform. The author was unable to find anything really comparable, educational or otherwise, to a vehicle equipped with a similar four-wheel steering mechanism that can also balance on two wheels as well as transition between four-wheel and two-wheel driving via remote control. The closest

platform found was an MS thesis project out of the MAE department at UC Irvine that addresses vehicle rollover through control experiments done on an RC car [36]. The vehicle presented in this study can temporarily balance on two wheels as well as transition between four and two-wheel modes using a prop but does not possess four-wheel steering capability. The goal of this project was to gain insight into controlling a vehicle during rollover by balancing it on its two side wheels in the event that rollover occurs, not to create a vehicle designed to be intentionally driven on two wheels. And as such the platforms are dynamically very different. A couple other platforms were discovered that present a four-wheeled vehicle capable of balancing on two wheels however they do not have nearly the dynamic range of BeagleRover and do not warrant further discussion here [37, 38]. The uniqueness of BeagleRover contributes to its innate appeal, potentially setting it apart from the other educational robotics platforms discussed in this section in its ability to attract and engage students in STEM curriculum.

Chapter 3 BeagleRover Steering

The basic goal of steering any vehicle or vessel is to achieve travel in the desired direction and the types of steering mechanisms and methodologies are numerous. Four wheeled ground vehicles such as BeagleRover determine the direction of travel by proper angling of the wheels, as opposed to tracked vehicles, for example, that use differential steering to induce change in direction. In commercial passenger vehicles this is done with a series of gears, rods, linkages and pivots that make up the steering mechanism [39]. The wheels are turned using a manually operated steering wheel placed in front of the driver via the steering column [40]. Remotely controlled four wheeled land vehicles have become very developed and sophisticated and often mimic various steering designs of commercial vehicles, in many cases mirroring the functional and structural capabilities thereof [41]. BeagleRover's steering mechanism is greatly simplified in comparison.

Rather than the common axle based design, BeagleRover has a servomotor attached to each wheel via a four-bar linkage mechanism, enabling active four-wheel steering with the ability to turn each wheel completely independently of the other three. The ability to independently drive and steer all four wheels is commonly known as "swerve-drive" within the FIRST Robotics Competition community [42]. The linkage mechanism is visible in Figure 3.1. The chassis is designed to allow each wheel to rotate a total of 120° while maintaining its contact patch in line with the pivot point of the motor. This steering design allows for much flexibility and various different modes of driving,

Figure 3.1: Close up of four bar steering mechanism.

resulting in a very dynamic and engaging platform. The different drive modes will be illustrated in section 3.3. Additions of this work to a preexisting mechanical design include implementation of active four wheel steering and Ackermann steering, both of which will be discussed in the following sections. Both features are implemented in software.

3.1 Active Four Wheel Steering

Active four wheel steering describes a steering methodology in which all four wheels turn simultaneously given a single steering input from the driver [43]. In many commercial passenger vehicles with four wheel active steer, the front two wheels are controlled manually by the driver via the steering wheel while the rear two wheels are controlled by a computer and actuators [40, 44]. Having two control inputs to steer the front and rear wheels independently allows for the optimization of both yawing of the vehicle and lateral motion, resulting in higher maneuverability at low speeds and greater stability at high speeds [43]. Higher maneuverability is accomplished at low speeds by turning

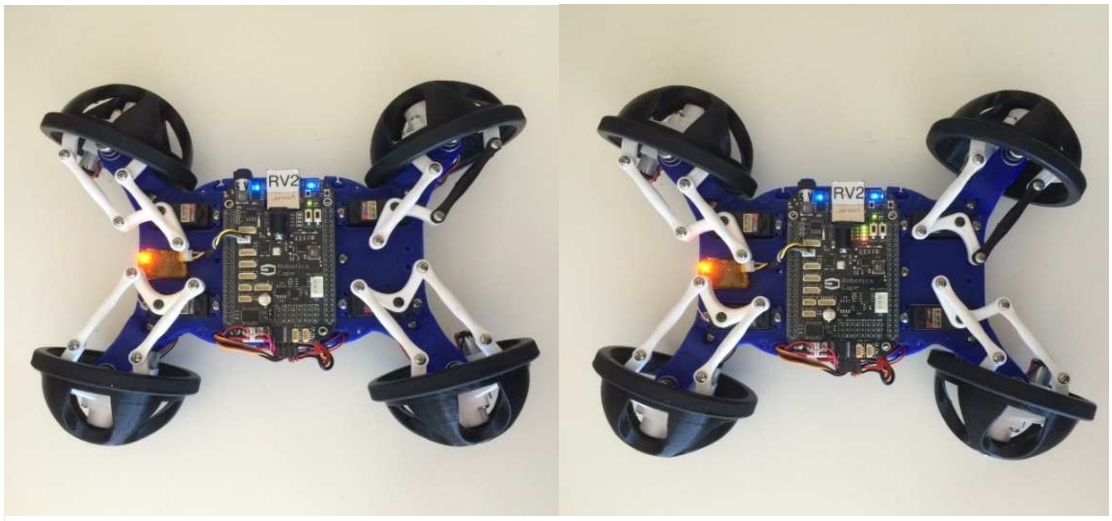

Figure 3.2: BeagleRover showing out of phase alignment of the wheels on the left and in phase alignment on the right. In the left image, the front wheels are rotated clockwise while the rear wheels are rotated counterclockwise. This is opposed to the right image where all wheels are rotated clockwise.

the rear wheels out of phase with the front wheels, thereby decreasing the turning radius of the vehicle. Conversely, applying in phase rotation at high speeds decreases yaw rotation and increases lateral stability [43, 44, 45, 46]. Many high performance vehicles today employ active four wheel steering [40]. In phase rotation and out of phase rotation of the wheels is demonstrated by BeagleRover in Figure 3.2.

Optimizing performance of active four wheel steerable vehicles relies on advanced dynamics and mathematics and results in the front and rear wheels being turned at different angles based on a number of factors. These factors include vehicle speed, steering angle and states of the system among others [46]. As an educational platform, BeagleRover is not designed for optimum performance and is not intended to be a high precision vehicle. Therefore active four wheel steering is demonstrated by toggling between different drive modes, where each mode either features in phase turning of the front and rear

wheels or out of phase turning. It is not speed dependent. This allows the user to experiment with both methods at low or high speeds, comparing the performance of each, which is a useful educational skew.

With four control inputs, one servo motor independently controlling each wheel, BeagleRover is capable of maintaining its simplified design while still achieving high maneuverability and providing an introduction to many of the advanced techniques and concepts used in commercial and competition vehicles. One such technique employed on BeagleRover in addition to active four wheel steering is Ackermann steering geometry, which will be described next.

3.2 Ackermann Steering Geometry

Ackermann steering geometry was patented in 1818 by Rudolph Ackermann [47]. It is a very well-known and well used concept with ample support material available to aid in understanding. Although implementation can become complex quickly, it is a very simple concept at its core and presents a great teaching platform for high school level geometry, trigonometry and related concepts. The goal of Ackermann geometry is to prevent slide slip of the wheels when going around a turn, preventing loss of energy in the direction perpendicular to motion as well as unnecessary wear and tear to the vehicle. The geometric solution to this is to have all four wheels rolling around a common point during a turn, each maintaining motion in the direction tangential to the circle created by connecting the center of curvature to the contact point of that wheel. The different circles traced by the inner and outer

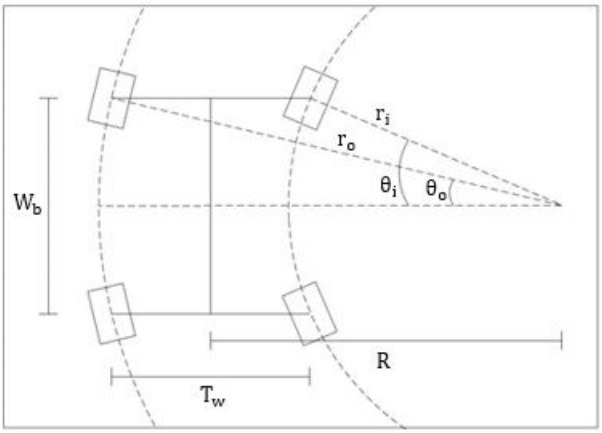

Figure 3.3: Ackermann steering geometry showing difference in radii between the circle traced by the inner wheel and the circle traced by the outer wheel in a right hand turn.

wheels of a four-wheeled vehicle are shown in Figure 3.3, illustrating the need for different angles of rotation.

Ackermann geometry is implemented many different ways for different steering designs. The goal for any vehicle implementing Ackermann steering is to have all wheels rolling around a common center point as shown in Figure 3.3. In front wheel steering vehicles, this common point is determined by the rear wheels as that is the limiting factor. On BeagleRover, all four wheels turn independently of each other so the common point is aligned with the center of the wheel base allowing for a much tighter turn radius. The comparison is shown in Figure 3.4.

Because all four wheels on BeagleRover are controlled completely independently, it is possible to map user input from the DSM2 radio directly to inner or outer wheel turn angle. It seems most intuitive from the user perspective to have the user turn input be mapped to the turn angle of the inner wheel. It was determined through experimentation that this gives the "expected" response from the vehicle. The amount of turn of the outer wheel is then

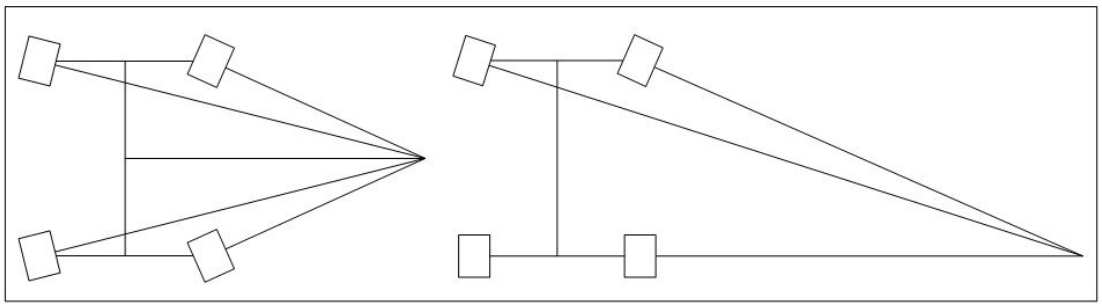

Figure 3.4: Comparing minimum turn radii of four wheel steering (left) vs. front wheel steering (right) on the same vehicle. Consider the front inner wheel in each drawing to be turned the maximum number of degrees as limited by the mechanical design. In the case of BeagleRover the maximum turn of the inner wheels is 17.5° before contact is made with the chassis.

calculated based on Ackermann geometry. Referring to Figure 3.3, define the following constants:

$\theta_i = angle\ of\ rotation\ of\ inner\ wheel,$

$\theta_o = angle\ of\ rotation\ of\ outer\ wheel,$

$T_w = track\ width,$

$W_b = wheelbase,$

$R = distance\ from\ center\ of\ curvature\ to\ center\ of\ vehicle.$

The quantities θ_i, T_w and W_b are known, where θ_i is the user input turn value from the DSM2 radio. Assuming pure rotation about the joint center of curvature, R and ultimately $\theta_o$can be calculated based on these values. Using the triangles depicted in Figure 3.5 and some basic trigonometry, we can solve for R using equation 3.1, plugging that value into equation 3.2 to ultimately solve for θ_o.

$$\tan(\theta_i) = \frac{\frac{W_b}{2}}{R - \frac{T_w}{2}} \tag{3.1}$$

$$\tan(\theta_o) = \frac{\frac{W_b}{2}}{R + \frac{T_w}{2}} \tag{3.2}$$

Equations 3.1 and 3.2 represent what is known as ideal Ackermann steering geometry [39]. In theory, this ideal is achievable by BeagleRover due to its simplified steering mechanism. Because all four wheels are mechanically independent of each other and controlled in software, there is nothing mechanically limiting the rotation angle of the outer wheel from complying with the ideal Ackermann steering criteria based on the rotation angle of the inner wheel. However, some inaccuracies are expected in practice due mainly to imperfections of 3D printed parts and by hand assembly. Servo resolution could also be a factor. Note θ_i refers to the angle of rotation of both inner wheels and accordingly θ_o refers to the angle of rotation of both outer wheels. The angles of the front wheels mirroring those of the rear wheels theoretically allows for the shortest turn radius possible when implementing ideal Ackermann geometry. The turn radius would eventually be limited by the four-bar linkage mechanism that attaches each servomotor to each wheel, however the wheels make contact with the vehicle chassis before the threshold of the four-bar linkage is reached.

As stated previously, implementation of four-wheel steering and Ackermann steering geometry quickly becomes very complex when attempting to optimize commercial and professional vehicle performance. However it has been demonstrated that through simplified versions of these techniques BeagleRover could be used to teach high school level mathematical concepts. This platform also enables many different hands on experiments through leveraging the on-board sensors of the Robotics Cape.

Such hands on experiments can be powerful in reinforcing theoretical concepts learned as well as potentially increase student engagement and interest in STEM [3]. Results of a few such experiments that would be accessible at the high school level are given in figures 3.5 through 3.8. See the figure captions for detailed descriptions of the experiments.

When looking at Figure 3.5 and Figure 3.6, notice the significant difference in amplitude range. Figure 3.6 displays much more dramatic peaks as compared to Figure 3.5, suggesting a potential increase in side slip of the vehicle without the implementation of Ackermann geometry. There also seems to be a low frequency component that is more pronounced in Figure 3.6 than in Figure 3.5. This could also suggest the presence of more side slip as the back tires periodically lose traction and recover. A more detailed analysis is required to prove that is the case, however the qualitative analysis shows clearly that Ackermann geometry changes the behavior of the vehicle. This gives tangible feedback to any student having previously studied the mathematical concepts used in implementation of ideal Ackermann steering geometry. These same results were reproduced with a wider turn radius in Figures 3.7 and 3.8. There are many, many more similar experiments that could be done with BeagleRover to compliment high school level curricula.

3.3 Drive Modes

Enabled by the independent four wheel steering design, four different stable driving modes are currently implemented on BeagleRover: normal four wheel steer, lane change, crab and spin. Ackermann steering geometry, as

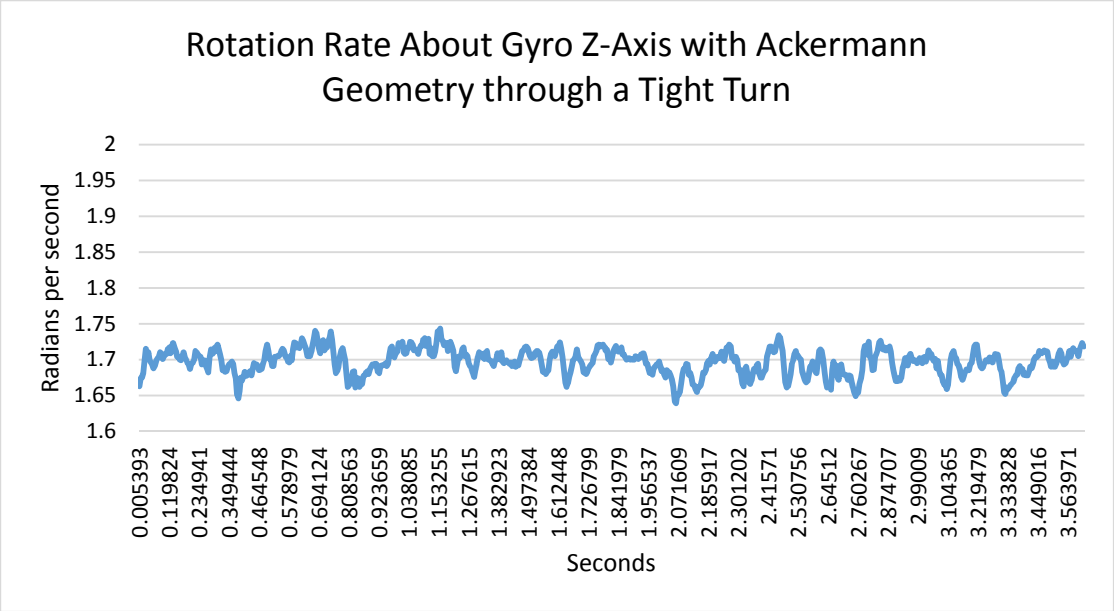

Figure 3.5: This plot shows the rotation rate about the z-axis of the gyroscope in radians per second through one full counterclockwise rotation of the vehicle with Ackermann geometry implemented. The motors were driven at 30% duty cycle as the vehicle traced out a circle of radius ≈0.318 m on a hard wood floor.

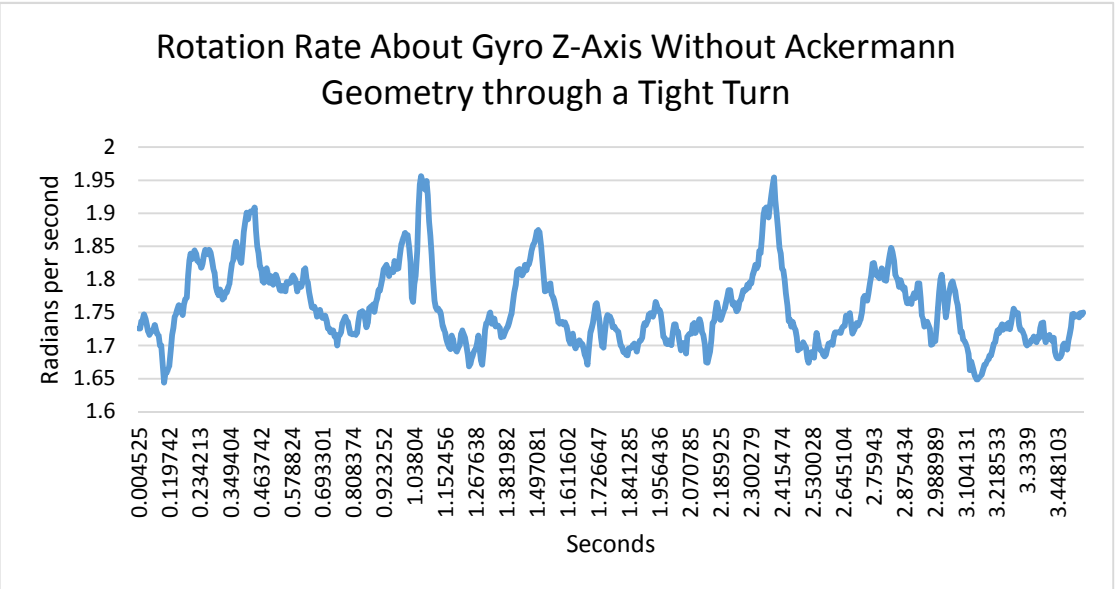

Figure 3.6: This plot shows the rotation rate about the z-axis of the gyroscope in radians per second through one full counterclockwise rotation of the vehicle without Ackermann geometry implemented. The motors were driven at 30% duty cycle as the vehicle traced out a circle of radius ≈0.318 m on a hard wood floor.

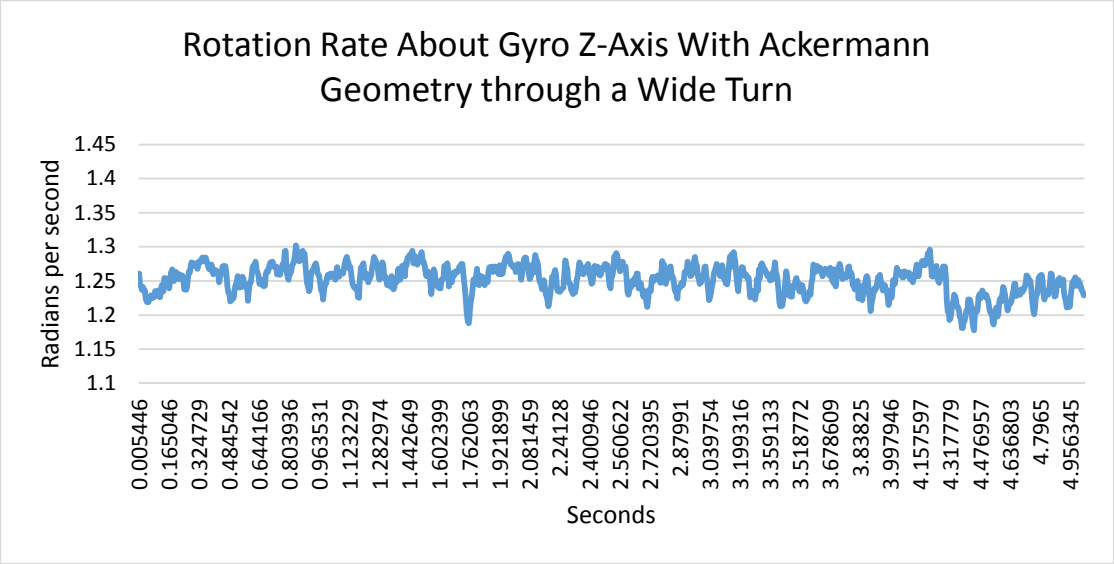

Figure 3.7: This plot shows the rotation rate about the z-axis of the gyroscope in radians per second through one full counterclockwise rotation of the vehicle with Ackermann geometry implemented. The motors were driven at 30% duty cycle as the vehicle traced out a circle of radius ≈0.481 m on a hard wood floor. This is a 51.2% wider turn than depicted in figures 3-5 and 3-6.

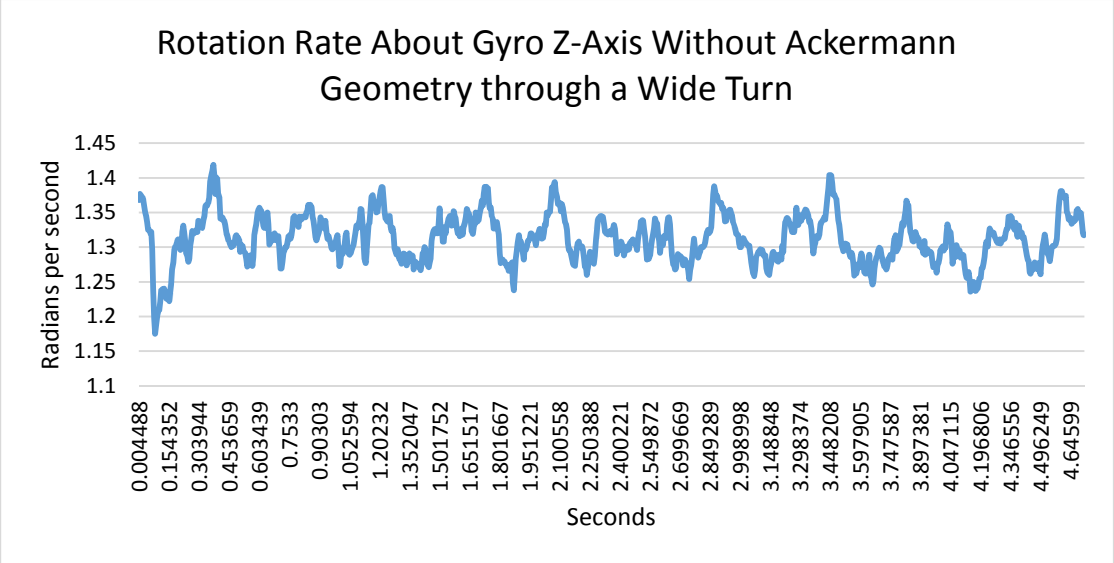

Figure 3.8: This plot shows the rotation rate about the z-axis of the gyroscope in radians per second through one full counterclockwise rotation of the vehicle without Ackermann geometry implemented. The motors were driven at 30% duty cycle as the vehicle traced out a circle of radius ≈0.481 m on a hard wood floor. This is a 51.2% wider turn than depicted in Figures 3-5 and 3-6. The comparison of Figures 3-7 and 3-8 shows similar results as that of Figures 3-5 and 3-6.

described in the above section, is implemented in normal and crab modes. In addition to these four stable modes of driving, BeagleRover is also capable of driving on two wheels in an unstable configuration called balance mode, which is the focus of chapter three. The vehicle is controlled with a DSM2 RC radio, using the two switches available to select between the four different drive modes. Other drive modes could be implemented, such as front wheel steering (two wheel steering) or steering without Ackermann, however another user input switch on the DSM2 radio would be required. Both of these additional modes present valuable educational skews, the latter of which is implemented in a separate example program included in the Robotics Cape codebase (discussed in detail in chapter four).

In normal four wheel steer mode, all four wheels face forward in a neutral position and the vehicle drives as would be expected. There is an arrow carved out of the chassis on the front of the vehicle to mark which direction is forward. When turning in normal mode, both inside wheels will rotate the same number of degrees according to user input, but out of phase with each other. In other words, the front inside wheel on a left hand turn will rotate counterclockwise from the user perspective and the rear inside wheel will rotate clockwise from the user perspective. The outer wheels will follow the same pattern, the front wheel rotating counterclockwise and the rear rotating the same number of degrees clockwise. The outer wheels will rotate the number of degrees calculated according to Ackermann steering geometry. The inner wheels can rotate a maximum of 17.5° before making contact with the body of the vehicle,

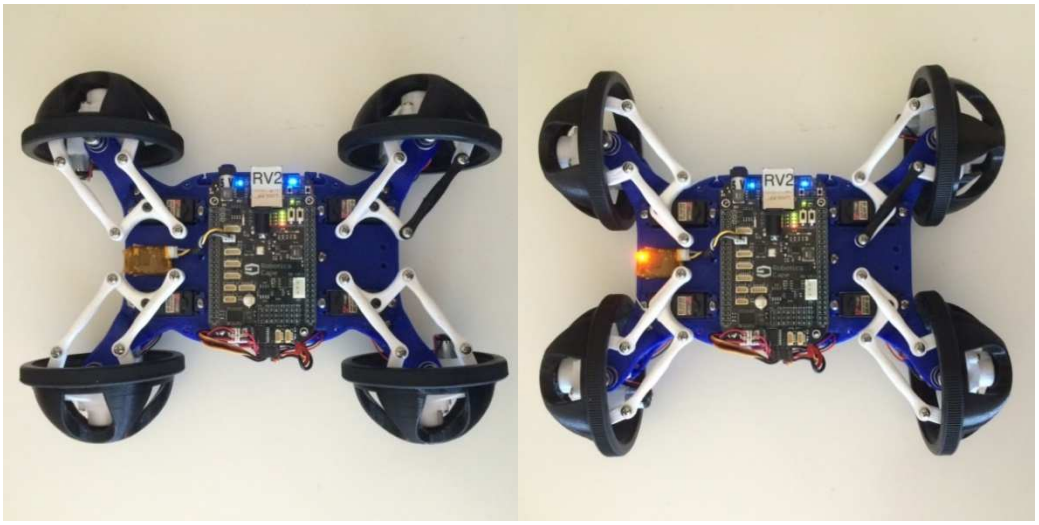

Figure 3.9: BeagleRover pictured in its Normal drive mode on the left and Crab drive mode on the right.

making for a minimum turn radius of ≈0. 28 m. Normal four wheel steer mode with zero steering input is pictured in Figure 3.9.

Crab mode operates exactly as normal four wheel steering mode, the inner wheels rotating according to user input and the outer wheels rotating based on the inner wheels and according to Ackermann steering geometry. In neutral position, the wheels are rotated either clockwise or counterclockwise by 90 degrees from normal mode and the "front" of the vehicle is now considered to be the Ethernet port side of the BeagleBone Black. This is also pictured in Figure 3.9. The maximum number of degrees the inner wheels can turn before making contact with the body of the vehicle is ≈12.5°, resulting in a wider minimum turn radius of 0.44 m.

One of the most fun drive modes BeagleRover displays is lane change mode or what is actually referred to as "crab" steering in the automotive industry (not to be confused with the definition of crab mode in this paper) [40]. As discussed in section 3.1 on active four wheel steering, many high performance

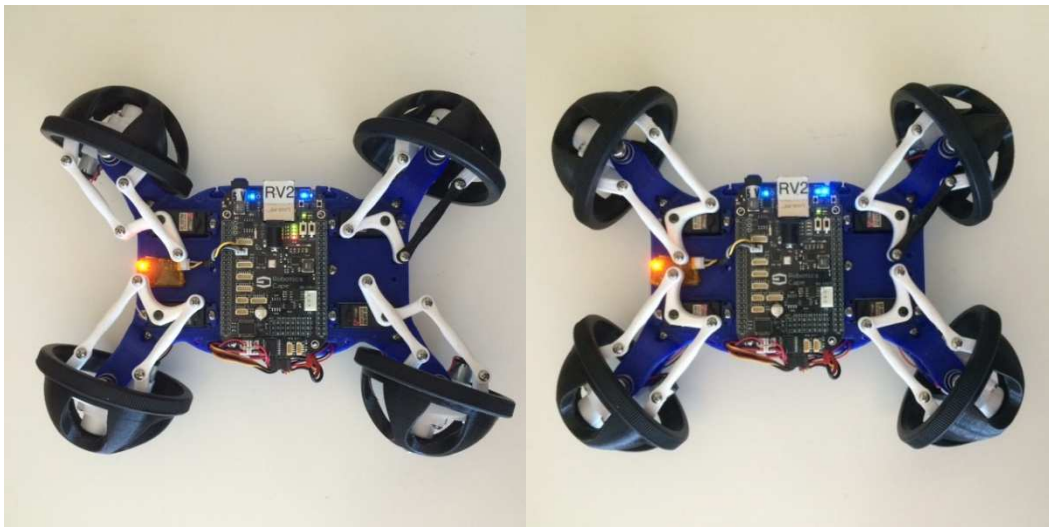

Figure 3.10: BeagleRover pictured in Lane Change drive mode on the left and Spin drive mode on the right.

vehicles employ this technique when turning or changing lanes at high speeds (hence why it is called "lane change mode" here) to improve stability of the vehicle. However this technique is not only used for improved handling. It is also found on vehicles where angled lateral translation is required such as camera dollies, and on many farm equipment vehicles to avoid unnecessary soil compaction [40]. As with normal mode, the maximum rotation angle of the wheels is 17.5° as limited by contact with the vehicle chassis. Therefore the maximum angle at which the vehicle can translate is 17.5° from straight. Lane change mode is pictured in Figure 3.10.

Last of the stable drive modes is spin mode. In spin mode, all wheels turn 54.78° so that the center of curvature of all wheels intersect at exactly the intersection of the two lines of symmetry of the chassis, aka its central axis. Therefore, the vehicle should spin in place exactly around this point, allowing for maneuvering in very tight spaces. In reality, the vehicle comes close to doing so but there is a slight amount of translation. This is likely caused by imperfections

due to 3D printed parts and or imprecise servos. Spin mode is pictured in Figure 3.10. Discussion of balance mode is left to the following chapter on control design.

Chapter 4 Control Design

This chapter will provide a solution to balancing the Beagle Rover on two wheels from the modeling and control design perspective. Originally, this chapter was written to be a detailed solution set for the mobile inverted pendulum problem as taught in UCSD's MAE143C, complete with all necessary derivations, and to be included in BeagleBone Robotics as a portion of a curriculum capable of distribution to other universities. That version of the control solution is presented in chapter six and is considered by the author to be one of the major contributions of this work. Here in chapter three however, the control solution will be presented in a fashion more traditional for a technical paper. Derivation of the dynamics describing the motion of Beagle Rover in balance mode is left to chapter six as that derivation is a detailed and slightly elaborated inclusion of the derivation previously done for BeagleMiP outside of this thesis work. The control solution, as that is unique to Beagle Rover and this work, is the primary focus of this chapter. Implementation in hardware and software, although alluded to here, is handled in detail in the following chapter.

4.1 Problem Statement

The plant is BeagleRover and will be modeled as an inverted pendulum on two wheels as depicted in Figure 4.1. Beginning with the equations of motion describing the system, the end goal is to design a discrete time control law to stabilize the body angle, θ, about its upright, unstable equilibrium position. The final product will be a difference equation ready for implementation in a

microcontroller, the BeagleBone Black in this case. In order to achieve this goal, classical control methods such as lead/lag control and pole placement will be used to stabilize the body of the Rover on two wheels about its unstable equilibrium position. Leveraging the discrete equivalent design approach, the controller is first designed in continuous time and later converted to discrete time for implementation in digital electronics. The primary tools and techniques used include the Laplace and Z-transforms, lead and lag control, bode and root locus plots, the closed loop step response and Tustin's approximation with prewarping, all of which will be discussed in more detail in the subsequent sections.

4.2 Equations of Motion

As discussed above, the full derivation of the equations of motion of the Beagle Rover system is left to chapter six as it is the derivation taught in MAE143C. In this section, the free body diagrams and equations of motion are briefly presented in order to provide supporting context for the control design section. Initially simplifying and considering a 2D representation of one wheel/motor and a simple rod, the free body diagrams of the wheel and rod are depicted in Figure 4.1.

By approximating the wheel as a thin, solid disk and the body of the vehicle as a simple rod the inertia of each can be calculated as

$$I_w = \frac{1}{2} m_w R^2, \tag{4.1}$$

$$\mathrm{I_r} = \mathrm{m_r L^2} \tag{4.2}$$

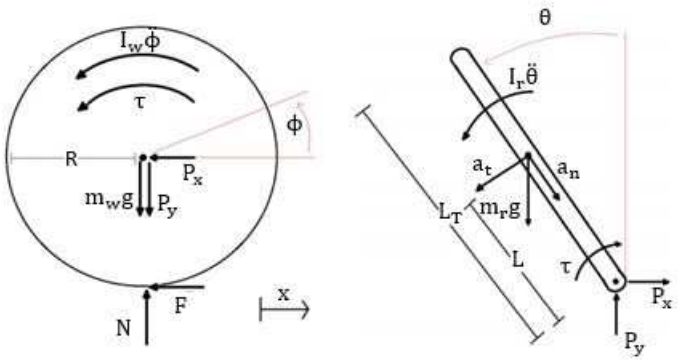

Figure 4.1: Free body diagrams of the wheel and rod.

where the constants are defined as follows:

$m_w = mass\ of\ wheel,$

$m_r = mass\ of\ rod,$

$I_w = inertia\ of\ wheel,$

$I_r = inertia\ of\ rod,$

$\tau = torque\ of\ motor,$

$R = radius\ of\ wheel,$

$L = length\ from\ end\ of\ rod\ to\ center\ of\ mass\ of\ rod,$

$P_x\ and\ P_y = reaction\ forces\ between\ wheel\ and\ rod\ in\ the\ x\ and\ y\ directions,$

$a_t = tangential\ force,$

$a_n = normal\ force,$

$g = gravity\ \left(9.81\frac{m}{s^2}\right).$

Whether balancing in crab mode or regular driving mode (short mode or tall mode respectively), take the center of mass to be at the intersection of the two axes of symmetry of the entire vehicle. This is an approximation as the "rod" in this model does not actually include the bottom two wheels. However, the mass of the wheels is negligible compared to that of the entire

vehicle so for simplicity's sake we will take the center of mass to be at the intersection when in reality it is very slightly higher. A more rigorous calculation could be done but would likely not result in any performance gain, especially because some error would still be expected [48].

Using the free body diagrams to first define the kinematic relations describing the position and acceleration of the center of mass of the rod, and assuming that both wheels are initially moving together without any slip between the wheels and ground, the full nonlinear equations of motion are derived as

$$(m_r RLcos\theta)\ddot{\phi} + (I_r + m_r L^2)\ddot{\theta} = m_r gLsin\theta - \tau, \tag{4.3}$$

$$(I_w + (m_r + m_w)R^2)\ddot{\phi} + (m_r RLcos\theta)\ddot{\theta} = m_r RL\ddot{\theta}^2 sin\theta + \tau \tag{4.4}$$

where τ is total input torque to the system. By the small angle approximation [49], equations (4.3) and (4.4) can be linearized about the body's inverted equilibrium point to yield

$$(\mathrm{m_r RL})\ddot{\phi} + (\mathrm{I_r} + \mathrm{m_r L^2})\ddot{\theta} = \mathrm{m_r gL}\theta - \tau, \tag{4.5}$$

$$(I_w + (m_w + m_r)R^2)\ddot{\phi} = (m_r RL)\ddot{\theta} = \tau \tag{4.6}$$

where all terms are now linear in theta, allowing for the application of linear control techniques.

Using the linearized equations of motion of our system, we could proceed through the control design process and design a stabilizing controller in simulation. In fact, this exercise is recommended as a first pass at designing a stabilizing controller and is gone through in detail in chapter six. However, if

attempting to implement this initial control design in hardware, the designer would find out that although stable in simulation, the robot would more than likely not balance in reality. This is due to the fact that the motors themselves have dynamics that must be included in the model. The motors used in BeagleMiP and BeagleRover have been previously characterized for use in MAE143C so consider the constants to be known quantities. The motor dynamics are described by

$$\tau = \bar{s}u + (b - \zeta)\omega \tag{4.7}$$

where the constants are defined as:

$\bar{s} = stall\ torque,$

$b = damping\ coefficient,$

$\zeta = viscous\ friction,$

$I_m = inertia\ of\ motors,$

$\omega = motor\ speed\ (\dot{\phi} - \dot{\theta}),$

$u = motor\ input\ (PWM\ value\ between - 1\ and\ 1).$

As before, the wheels are taken to be solid disks when estimating their inertia. The total inertia of one wheel and motor is

$$I_w = I_m + \frac{1}{2}m_w R^2 \tag{4.8}$$

and the final linearized equations of motion accordingly become

$$(\mathrm{m_r RL})\ddot{\phi} + (\mathrm{I_r} + \mathrm{m_r L^2})\ddot{\theta} = \mathrm{m_r gL}\theta - 2\tau \tag{4.9}$$

$$(\mathrm{I_w} + (\mathrm{m_r} + \mathrm{m_w})\mathrm{R^2})\ddot{\phi} + (\mathrm{m_r RL})\ddot{\theta} = 2\tau \tag{4.10}$$

where tau is defined above to be the torque from one motor, m_w now includes

the mass of both wheels and I_w now includes the inertia of both wheels and motor gearboxes.

4.3 Lag Control

Before beginning the control design process using classical control techniques, the transfer function describing the input/output relationship between PWM input to the motors, u, and body angle output, θ, must be defined. In order to derive the transfer function, the Laplace transform is performed on the final linearized equations of motion followed by algebraic rearrangement to get an equation of the form $\theta(s)/U(s) = numG1(s)/denG1(s)$. First apply the definition of the Laplace transform to equations (4.9) and (4.10), assuming zero initial conditions. Then making the appropriate substitutions for τ and ω defined in the previous section and rearranging algebraically, the transfer function is

$$G1(s) = \frac{\theta(s)}{U(s)} = \frac{2s(C4+C1)}{(-C4C2+C1^2)s^3+(2(\zeta+b)(-C4-C2-2C1)s^2+(C3+C4)s+2(\zeta+b)C3} \quad (4.11)$$

where:

$$C1 = m_r RL, \quad (4.12)$$

$$C2 = I_r + m_r L^2, \quad (4.13)$$

$$C3 = m_r gL, \quad (4.14)$$

$$C4 = I_w + (m_r + m_w)R^2. \quad (4.15)$$

Plugging in the constants yields the final transfer function as shown in equation 4.16.

$$G1(s) = \frac{\theta(s)}{U(s)} = \frac{-226.2s}{s^3+7.221s^2-136.7s-292.2} \quad (4.16)$$

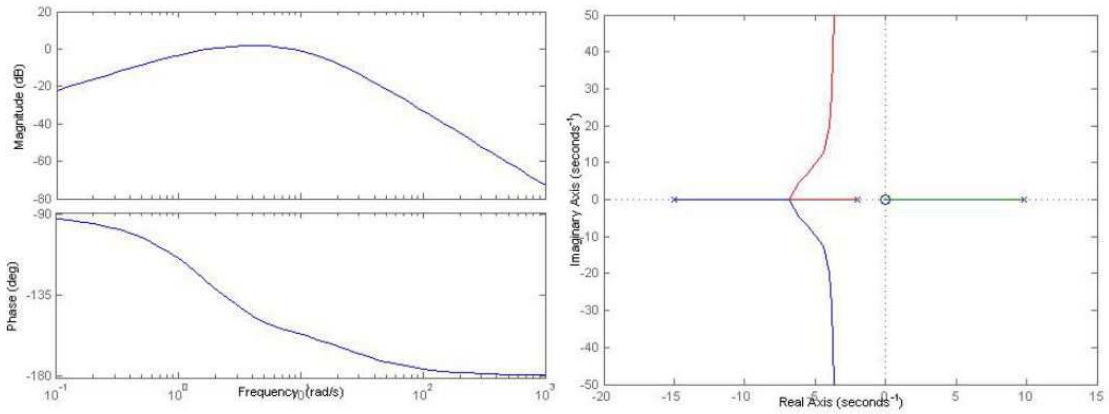

Figure 4.2: Bode (left) and root locus (right) plots of –G1(s).

The root locus and bode plots, with negative gain applied, are shown in Figure 4.2.

Looking at the root locus, no amount of gain can stabilize the system. In order to bring the locus into the stable left half plane using pole placement techniques, the controller must cancel the zero at the origin and replace it with a stable zero. Note this is generally done with caution as a pole-zero cancellation on the imaginary axis can lead to instability due to inaccuracies in the model [50]. This model has been proven to be accurate so the cancellation will be performed. Looking at the bode plot, higher magnitude at low frequencies is needed to achieve acceptable tracking. By applying lag control over the appropriate frequencies we can bump up the low frequency gain as well as cancel the zero at the origin, replacing it with a stable zero and bringing the locus into the LHP. The form of a lag controller is given in equation 4.17.

$$D_{lag}(s) = \frac{s+z}{s+p} \; for \; z > p \tag{4.17}$$

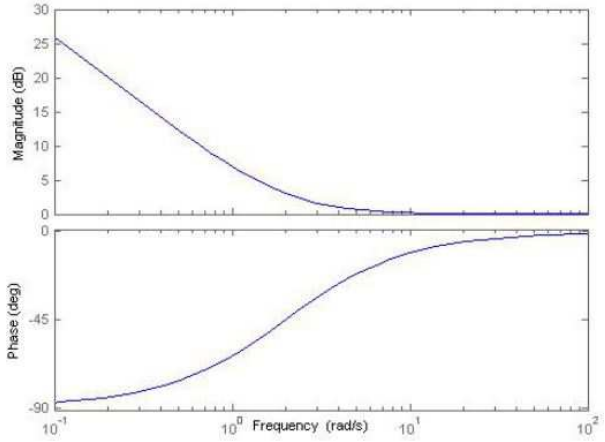

Figure 4.3: Bode plot of lag controller $D_{lag}(s)$.

The primary use of a lag controller is to increase the gain at low frequencies in order to achieve good tracking of the reference signal. The gain will increase by a factor of z/p. Because the pole is at zero, z can be chosen such that the lag control takes affect at frequencies well below the desired crossover frequency of 20 rad/s (which has been determined by choosing a crossover frequency an order of magnitude below the sample rate of the BeagleBone Black's ADC, avoiding significant phase loss at this critical frequency) [50]. This is necessary due to the phase lag caused by the lag controller as eroding the phase at crossover can potentially lead to closed-loop instability. After a few iterations, z=3 is chosen. The resultant bode plot of the lag controller is seen in Figure 4.3, illustrating the phase lag effect of a lag controller as well as the high gain at low frequencies. The lack of roll off of the gain at low frequencies is the result of having a controller pole at zero, acting as a pure integrator. Ideally the gain would be rolled off at very low frequencies to avoid integrator wind up, a phenomenon that can lead to instability in the case of motor saturation, however this is prevented due to the plant zero at the origin

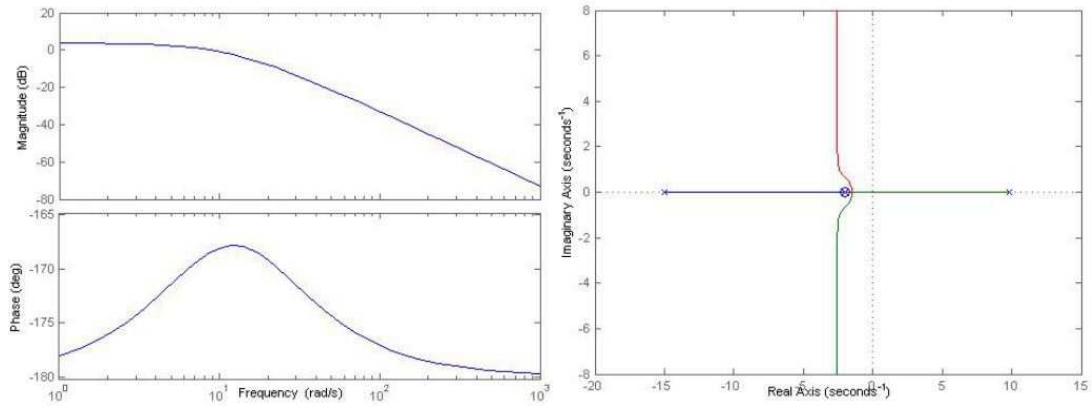

Figure 4.4: Bode (left) and root locus (right) plots of $-D_{lag}(s)G1(s)$.

[50]. The Bode and root locus plots of the plant with negative gain and lag control combined are given in Figure 4.4.

4.4 Lead Control

The bode plot with lag control shows that the low frequency gain has been increased but at the detriment of dangerously eroding the phase margin. Also, crossover at the desired frequency is not achieved. The locus is now brought over into the stable LHP with the appropriate gain applied, however this design will have an unacceptably high overshoot due to the low phase margin. In order to achieve crossover at the desired frequency and increase the phase margin, lead control will be applied using much of the same logic as is section 4.3. The following second order design guides will be used:

$$\omega_c = \frac{1.8}{t_r}, where\ t_r = rise\ time \tag{4.18}$$

$$\omega_c = \sqrt{pz}, where\ pz = p * z \tag{4.19}$$

As before, placing a zero at -15 to cancel the pole and replacing with a faster pole will help to achieve the desired rise time without applying too much

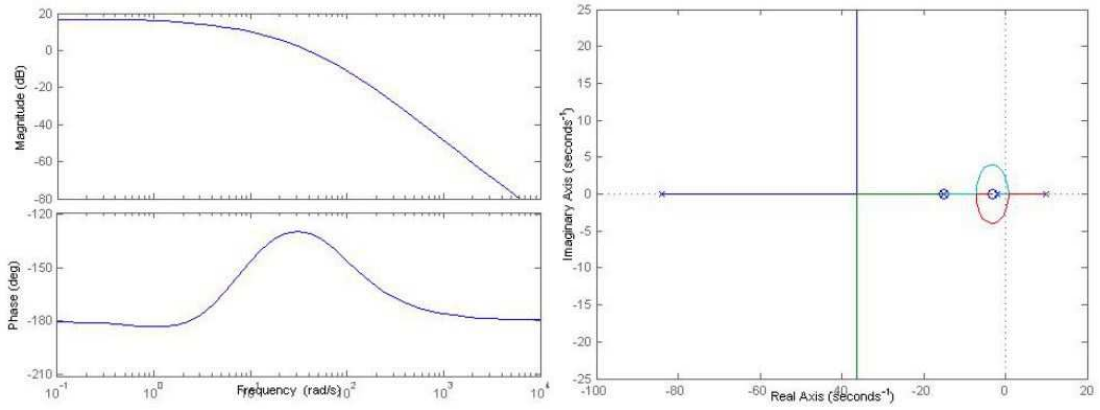

Figure 4.5: Final open loop Bode (left) and root locus (right) plots with lead and lag control where $D_{lag}(s) = (s + 15)/(s + 86.4)$.

overall gain to the system. Keeping in mind that we would like a rise time of 0.05-0.1 seconds, and a crossover frequency of ≈20 rad/s, the above equations can be leveraged to calculate the location of the pole that should achieve maximum phase increase at crossover [50]. Using equation (4.18), a rise time of 0.05s corresponds to a crossover frequency of 36. This yields a pole at 86.4 by equation (4.19). Although this is not quite a factor of 10 higher than the zero at 15 to achieve maximum phase increase [50], ample phase margin of about 52 degrees is still achieved. The design guides are approximate and not all will be met exactly. After adjusting the gain to again achieve crossover at the target frequency, the final values chosen are $\omega_c = 23\ Hz$, $t_r \approx 0.08s$, $PM = 52°$. The final open loop bode and locus plots with lag and lead control applied are shown in Figure 4.5.

4.5 Closing the Loop

The final step in continuous time is to check the closed loop step response to ensure that the design is stable and the design criteria have been

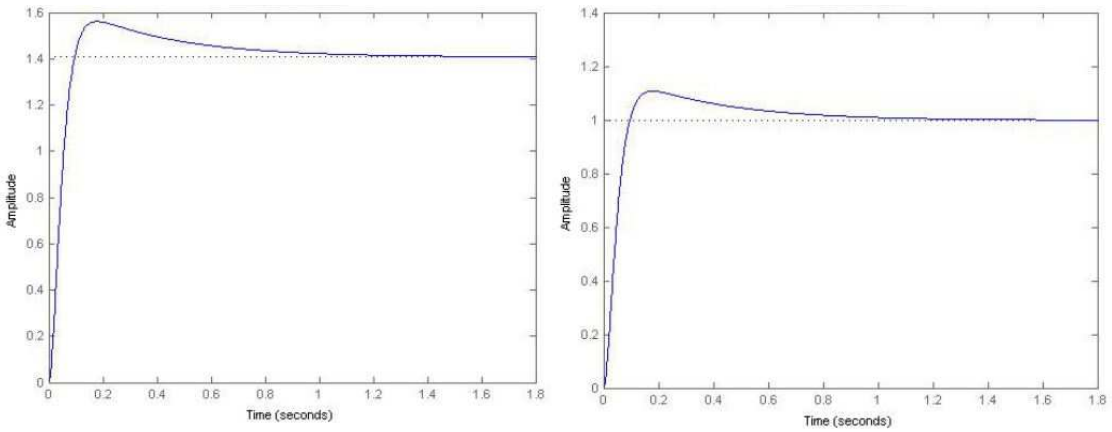

Figure 4.6: Closed loop step response of system without a loop prefactor on the left and with a loop prefactor of 1/1.4 on the right. The addition of the loop prefactor causes the step response to settle at one as desired.

met. Referring to $D_{lead-lag}(s)$ as $D(s)$, the form of the closed loop transfer function is

$$H(s) = \frac{K(s)D(s)G1(s)}{1+K(s)D(s)G1(s)} \tag{4.20}$$

Performing a few final adjustments, $D_{lag}(s)$ and $D_{lead}(s)$ become

$$D_{lag}(s) = \frac{s+2.5}{s} \tag{4.21}$$

$$D_{lead}(s) = \frac{s+15}{s+86.4} \tag{4.22}$$

with a closed loop gain K(s) = -10. Giving a step input to the system results in the step response plotted on the left hand side of Figure 4.6 with a rise time of 0.06 s, settling time of 0.79 s and an overshoot of 10%. Because the response settles at a value of ≈1.41 instead of 1 as desired, a loop prefactor of P = 1/1.4 can be incorporated, yielding the step response pictured on the right hand side of Figure 4.6.

4.6 Discrete Time Controller

In order to implement the controller in hardware, it must be converted from continuous time to discrete time to obtain the difference equation the

controller must obey. Tustin's approximation with prewarping is used to convert to a discrete time transfer function, applying prewarping around the crossover frequency to insure accurate conversion at this frequency. Applying the inverse Z transform, the final difference equation to be implemented in the microcontroller is given by equation 4.23.

$$u_k = 1.652u_{k-1} - 0.6525u_{k-2} - 8.626\theta_k + 16.52\theta_{k-1} - 7.902\theta_{k-2} \qquad (4.23)$$

4.7 Balancing in Normal Drive Orientation

All of the values presented in the control solution thus far have been for balancing BeagleRover in its short or "crab" orientation. However, the vehicle is capable of balancing on all four of its sides. In order to balance BeagleRover in its tall orientation, the change in the distance from the center of the wheel to the center of mass of the body, *L*, and the corresponding change in the body inertia, I_r could be accounted for. If so, the transfer function from input U(s) to output $\theta(s)$ becomes

$$G2(s) = \frac{-137.2s}{s^3+5.917s^2-96.48s-206.2} \qquad (4.24)$$

with poles at -12.4, -1.98 and 8.43 which are slightly "slower" than the poles of G1(s) when balancing in crab mode. The control design process would proceed exactly as stepped through in the preceding sections. In practice however, it was found desirable to simply increase the gain on the system when balancing in the taller orientation. This will be further addressed in the following section on implementation.

Chapter 5 Implementation

Chapter three presented the control solution to balancing BeagleRover in its two-wheeled unstable configurations while chapter two identified the stable four-wheeled drive modes and steering solution. Now that both solutions have been developed in theory and simulation, the next step is to present the implementation in hardware and software. This chapter will give an overview of the hardware and general software implementation strategy used in development of the EduLine as well as developments specific to BeagleRover. The codebase and programming environment will be discussed as well as the techniques used in state estimation. Additionally, the technique used to balance on all four sides as well as transition from driving on four wheels to balancing on two wheels will be explained.

5.1 Hardware and Codebase

All three robots in the EduLine use the BeagleBone Black (BBB), a "low-cost, community-supported development board for developers and hobbyists" [52]. The BeagleBone Black is an open hardware microprocessor development board that can fit in the palm of a hand. It has enough flash storage and volatile RAM to support a full-featured operating system and a custom build of the Debian Linux operating system comes pre-installed. The BBB also features a wide variety of connectivity options with two 2×23 pin header rows, USB client and host capabilities, as well as Ethernet and HDMI ports.

The BeagleBone Black benefits from an add-on board, or cape, to provide easy access to all of the functionality it provides. The solution to this

used by the EduLine, and therefore BeagleRover, is the Robotics Cape designed by James Strawson. The Robotics Cape is a $35.00 add-on board that provides the ability to drive up to four bi-directional DC motors via H-bridges and up to eight servomotors. BeagleRover uses four bi-directional DC motors and four servomotors. The cape also supports encoder counting on four channels for motor position feedback, three of which are broken out in hardware and a fourth added utilizing the BeagleBone Black's on-board PRU (programmable real-time unit). BeagleRover does not currently use encoders due to space restrictions of the mechanical design. However, BeagleMiP does use two encoders for position control. A 9-axis inertial measurement unit (IMU) is also included on the cape as well as a barometer for use in flight applications. The IMU is comprised of a 3-axis accelerometer, 3-axis gyroscope and 3-axis magnetometer. BeagleMip and BeagleRover both use the accelerometer and gyro for state estimation. The cape also supports DSM2 radio and Bluetooth for controlling the robots wirelessly. Development of BeagleRover was done using a DSM2 radio, further work being necessary to incorporate Bluetooth control.

In addition to the hardware, all EduLine robots come with an extensive codebase written to support the Robotics Cape. The codebase includes libraries for all of the functions used by BeagleMiP and BeagleRover for hardware interfacing, plus many functions not currently used by these two platforms. Additionally, approximately 30 example programs, ranging from blinking an LED to gyro calibration, are included. A subset of the current example programs written as part of the Robotics Cape codebase are shown

Table 5.1: A subsection of the example programs included in the Robotics Cape codebase.

Program Name	Program Description	Program Features
drive.c	Most advanced drive code for BeagleRover	Four stable drive modes, Ackermann steering, Balancing on all four sides, transition from drive to balance
drive_simple.c	Simplified drive code for BeagleRover	Four stable drive modes
rover_balance.c	Balance code for BeagleRover	Balance on all four sides of vehicle
test_orientation.c	Determine orientation of Robotics Cape	Example of using Euler angles to determine orientation
complementary_filter.c	Measure pitch angle of Robotics Cape	First order complementary filter
battery_monitor.c	Monitor charge level of 2S, 3S or 4S Lithium Ion or Polymer battery	Illuminate four LEDs on cape indicating charge level, under-voltage protection
center_servos.c	Put servos in their neutral position	No additional features
calibrate_gyro.c	Offset factory steady state error of gyro	Save steady state offsets so do not need to recalibrate each time program starts

in Table 5.1. The first three programs were written by the author as a direct result of this thesis work. The remaining programs are included as they are integrated into the drive code or were found useful in developing the drive code, illustrating the utility of the Robotics Cape codebase in writing more complex programs. Note this is not an exhaustive list. All code is written in the C programming language and available on GitHub. All instructions for installing and getting started with the Robotics Cape libraries on the BeagleBone Black are detailed in chapter three of BeagleBone Robotics and on the designer's website [52].

Notice there are two versions of the drive code, drive.c and drive-simple.c listed in Table 5.1. The drive-simple.c is a simplified version of the drive

code lacking Ackermann steering and balancing that is included for educational purposes, so that the student may compare the differences, especially in steering behavior, directly on the robot. The test-orientation example was also written specifically for drive.c in order to implement balancing on all four sides of the vehicle. Independently this serves as a great example of a simple multithreaded program to test operation of the motion processing unit on the Robotics Cape (MPU-9150).

5.2 Programming Environment

The BBB is capable of functioning as a low power desktop computer by connecting a USB keyboard, mouse, driving an HDMI display and rendering a graphical user interface (GUI). However this was not done as the robots are designed to be mobile and therefore used in a headless configuration (without a GUI). Instead, because the BeagleBone has an operating system on board enabling communication with the robot via standard network protocols, all programming was done on a host computer. Development was done in a Windows 8 environment, communicating with the BeagleBone via USB and generating a command line interface using the free software application PuTTy. All code was written in C using Notepad++ and compiled using standard Linux commands from the command line, all the while transferring files back and forth using the free SFTP (Secure File Transfer Protocol) program WinSCP. Getting started in this programming environment is explained in detail in the first few chapters of BeagleBone Robotics. As made apparent in chapter three, control design was done in MATLAB leveraging the control toolbox.

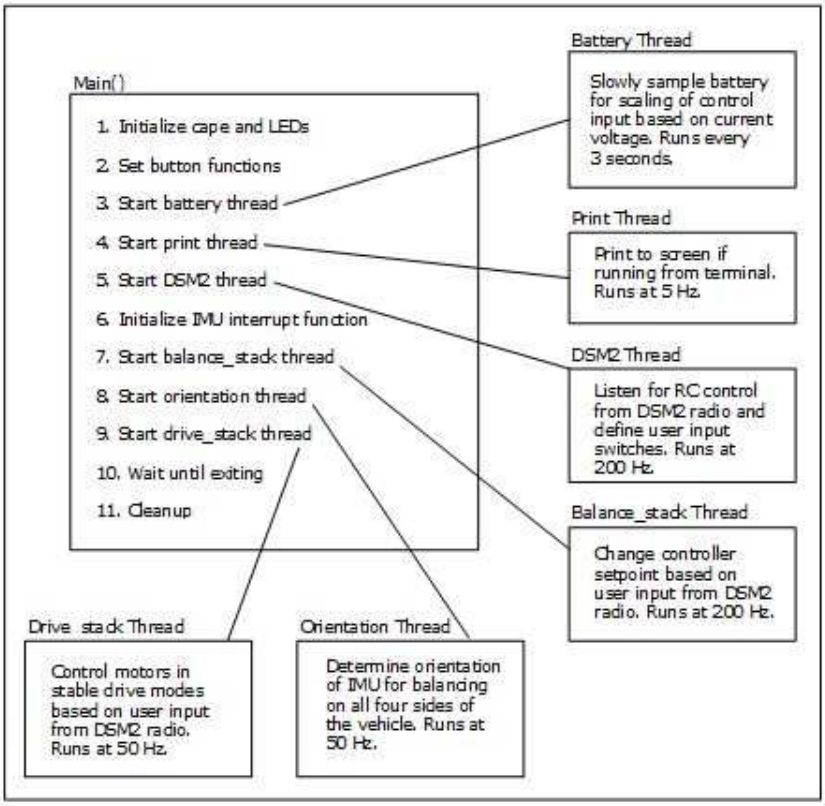

Figure 5.1: Visual depiction of the various threads running in the drive.c program and their respective tasks.

5.2.1 Multithreaded Programming and Drive.c Program Architecture

An advantage of having an operating system on board is the ability to execute multithreaded programs potentially increasing the functionality of the application. In the case of BeagleRover, multithreading is not strictly necessary, however it illustrates a powerful tool for use in more demanding applications. A high level view of the drive.c program organization is shown in Figure 5.1.

The main() function is responsible for the setup routine and starting all threads as well as cleanly shutting down the program if an exit condition is met. The order in which the threads are listed in Figure 5.1 is the order in which they appear in the program. Although it is not technically a separate thread, the balance_core() function deserves mention as it is the IMU interrupt function

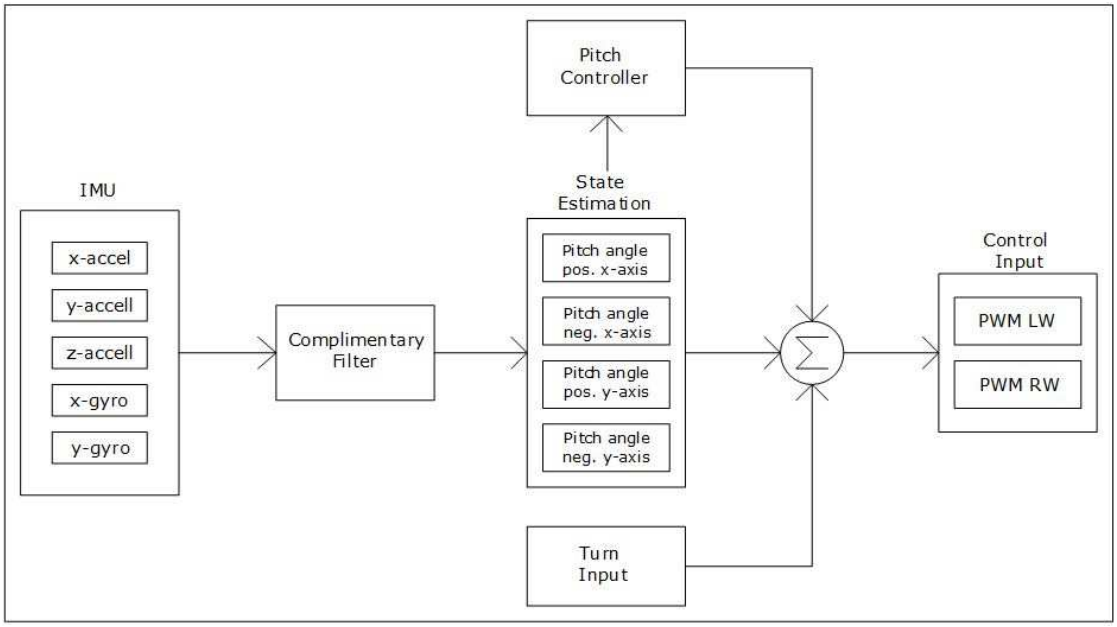

Figure 5.2: Software architecture of the balance function. PWM LW stands for PWM input to the left wheel and PWM RW is PWM input to the right wheel.

responsible for stopping, or interrupting, all programs running in order to retrieve sensor data at a rate of 200Hz. Once the raw sensor data is retrieved it is sent through a complimentary filter to calculate pitch angle in any orientation of the vehicle. The estimated state variable is then used by the control algorithm developed in chapter three to provide the necessary input to the motors to maintain stability in balancing, all within the balance_core() function. This is depicted graphically in Figure 5.2. Many of the example programs included in the Robotics Cape codebase are used by the various different threads. For example, test_orientation.c was directly ported over to drive.c to serve as the orientation_detector() thread with minor changes made for the application to drive.c.

Notice there is a block in Figure 5.2 labeled "turn input." This is the input given to the motors in order to steer the vehicle while in balance mode. This

input does not rely on the state variable. It is a simple constant applied via user input from the DSM2 radio. When turning in balance mode, the turn constant is applied equally to both motors, increasing PWM to one motor and equally decreasing PWM to the other motor. Therefore total input torque to the system is not changed due to turning input and the vehicle maintains stability.

5.3 State Estimation

As discussed at length in chapter three on control, BeagleRover is modeled as a mobile inverted pendulum with a single input of torque (ultimately PWM to the motors) and single output of pitch angle, θ. Because it is a SISO system there is only one state variable to control, pitch angle, or, tilt angle. For the control algorithm to be successful in maintaining stability the state variable must be reliably computed. As depicted in Figure 5.2 this is done leveraging the on board sensors of the IMU, specifically the accelerometer and gyroscope. It should be mentioned that state estimation is used for two different purposes in the drive code. Aside from being used to detect deflection from upright by the balance_core() function and control algorithm, state estimation is also used to detect which side of the vehicle is facing upward and which motors are in contact with the ground (or whatever surface on which the vehicle is balancing, we'll call it ground) and therefore which motors should receive the PWM signals. This function is performed by the orientation thread as depicted in Figure 5.1 and is accomplished using Euler angles as calculated by the Digital Motion Processor within the IMU. This differs from the complementary

filter used to estimate tilt angle for use in the balance controller as will be discussed further below.

5.3.1 A Note on Euler Angles

The current balance code for BeagleMiP, balance.c, uses Euler angles to estimate tilt angle when upright. For BeagleMiP this is readily achievable as it only balances in one orientation of the IMU. BeagleRover in contrast balances in four different orientations and using Euler angles becomes a more advanced process. An in depth discussion of Euler angles is outside the scope of this work, but briefly stated, the complication lies in the singularity of Euler rotation sequences causing inaccuracies in the calculation of theta in certain orientations of the vehicle [50]. However, Euler angles can more simply be used to yield a general determination of which side of the vehicle is facing upward and which side of the vehicle is in contact with the ground. In this case, a threshold value is detected that is far away from the point at which theta becomes unreliable. Furthermore, no feedback control is performed in this scenario therefore there is less need for absolute accuracy. Utilizing Euler angles to determine general orientation of the vehicle, as opposed to the complimentary filter approach used to determine theta for use in the balance controller, was done to illustrate the use of each as BeagleRover is intended to be an educational platform. Although using Euler angles to estimate theta for balance control as well is not an impossibility, that approach is more advanced than what is taught in MAE 143C and other typical undergraduate courses.

Therefore complimentary filtering was chosen as the primary solution to state estimation in order to remain consistent with the MAE143C curriculum.

5.3.2 Complementary Filter

The first order complementary filter used to estimate tilt angle is based on the complementary filter example code in the Robotics Cape codebase and adapted to be used to estimate theta in all four balance orientations of BeagleRover. The four balance orientations as named in drive.c are NOSE_UP, NOSE_DOWN, LEFT_DOWN and RIGHT_DOWN, corresponding to the negative x-axis, positive x-axis, positive y-axis and negative y-axis of the IMU respectively, according to the coordinate system depicted on the cape. The filter uses data from both the accelerometer and gyroscope combined to yield a much more accurate representation of theta across frequencies than could be achieved with either sensor alone. The reason lies in the type of noise to which each signal is most susceptible.

The accelerometer is used to determine angular position by measuring the position of the gravity vector. As the accelerometer rotates in space, the magnitude of the gravity vector in the direction of each axis changes, as depicted in Figure 5.3. Using the standard C math.h library function, atan2(), and two axes of the accelerometer, the angular position about the pitch axis can be easily calculated. For example, if BeagleRover was balancing perfectly in its NOSE_UP position, the negative x-axis of the accelerometer would experience almost exactly -1g of gravitational force while the z-axis would experience nearly zero. Passing these values to the atan2() function with proper

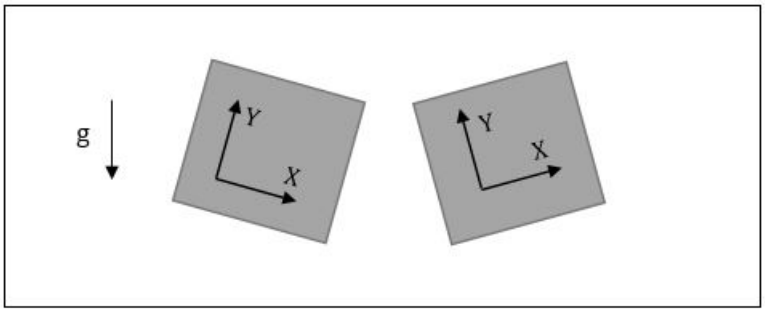

Figure 5.3: 2D representation of the axes of an accelerometer as it changes position in space. On the left hand side the x-axis reads slightly positive and on the right hand side the x-axis reads slightly negative.

attention to order of the arguments would yield an angle of nearly zero as expected. Note that care must be taken here with the signs of the vector components passed to the function in order to return the correct quadrant of the 2D x-z plane.

When used to measure angle of rotation, accelerometers can experience high frequency noise [53]. This is due to changes in acceleration caused by other factors such as horizontal motion. Considering the example given above of balancing BeagleRover in its NOSE_UP orientation, this would cause the z-axis to measure changes in acceleration not due to gravity, causing very significant errors in the estimation of angular position. This can be addressed by low-pass filtering the accelerometer signal, allowing only the low frequency signal due to gravity to pass through. It should be noted that low-pass filtering can cause phase lag [50]. As discussed in section 3.3 regarding lag control (essentially a low-pass filter), significant phase erosion at the crossover frequency can lead to instability of the system. If this is of concern, the phase lag of the low-pass filter should be accounted for during control design as depicted in Figure 5.4. As also discussed in chapter three, ample phase margin

was built into the control design for BeagleRover to account for this issue of phase erosion due to the low pass filter.

The gyroscope presents the opposite challenge, tending to output a signal corrupted by low frequency noise [55]. The gyro outputs measurement of angular velocity and is used to determine angular position by integrating over time. However, MEMS gyroscopes such as the one used in the IMU of the Robotics Cape, are subject to constant bias, which when integrated over time, quickly leads to unreliable measurements [53, 54, 55]. This is addressed by high-pass filtering the integrated signal from the gyroscope, eliminating the low frequency drift that occurs as the platform is held near stationary [53, 55].

The complementary filter takes advantage of the low frequency accuracy of the accelerometer and high frequency accuracy of the gyro, combining signals to achieve the best of both worlds [56]. In Laplace notation, the form of a low-pass filter is

$$F_{LP}(s) = \frac{\omega_c}{s+\omega_c} \tag{5.1}$$

where ω_c is the cutoff frequency above which the signal is attenuated [50]. Similarly, the form of a high-pass filter is

$$F_{HP}(s) = \frac{s}{s+\omega_c} \tag{5.2}$$

where ω_c is the cutoff frequency below which the signal is attenuated [50]. In the case of a complementary filter, the cutoff frequency for the low-pass and high-pass filters is the same, therefore $F_{LP} + F_{HP} = 1$, ensuring that all frequencies (ideally), minus the noise, are represented in the final reconstruction of the signal. The block diagram of the complementary filter is

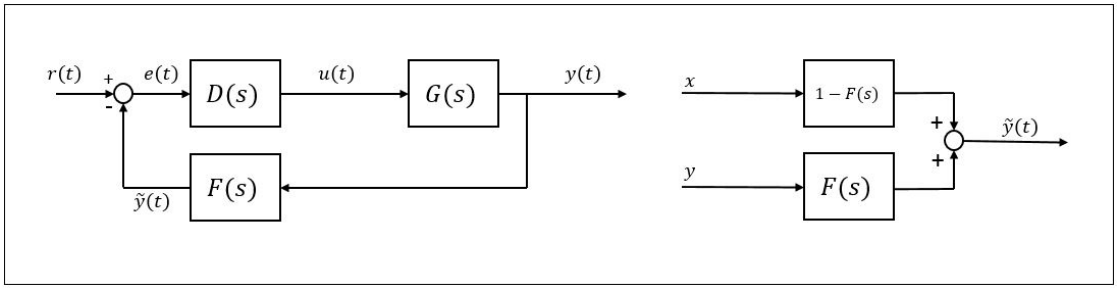

Figure 5.4: The block diagram on the left hand side illustrates integration of the complementary filter into the complete feedback system. Note disturbances and noise are not shown here. On the right hand side the block diagram of the filter itself is shown, where x and y in this case are the inputs to the low-pass and high-pass filters. Note the output of the filter is $\tilde{y}(t)$ because no filter is ideal and $y(t)$ will not be reconstructed exactly.

shown in Figure 5.4, illustrating the combination of the accelerometer signal and the gyro signal. The end effect is an estimation of theta that favors the accelerometer measurements at frequencies below ω_c and favors the gyro measurements at frequencies above ω_c.

When implemented in digital electronics, the complementary filter takes the form

$$\theta = gyro_{HP}(\theta + gyroData * dt) + accel_{LP}(accelData) \tag{5.3}$$

where $gyro_{HP}$ is the high-pass constant chosen for the high-pass filter and $accel_{LP}$ is the low-pass constant chosen for the low-pass filter. The accelerometer data represented by accelData has already been processed by the atan2() function and *gyroData*dt* accomplishes the integration of the angular velocity measurement performed at every time step. In the case of the platform at hand, data is collected at a rate of 200 Hz (dt = 0.005 seconds) as listed in table 5.1 and the high-pass and low-pass constants are approximately 0.99 and 0.01 respectively. These values are calculated based off the sample rate and the noise properties of the specific sensors used.

Note that in order to conserve resources, the filters are not calculated until the orientation of the vehicle is detected then the correct filter using the corresponding axes of the IMU is computed. However, it is important for accuracy that state estimation is constantly computed, initially relying on raw accelerometer data alone for immediate usage by the filter when needed. It was observed that if the filter was not initialized using the raw accelerometer data until a change in orientation was determined, a significant delay in reaching the correct value of theta resulted, causing an unacceptable delay in arming the controller. This resulted in instability when transitioning from four-wheel driving to two-wheel driving using a wall.

5.4 Code Optimization and Features

BeagleRover is designed to be an educational platform. Because of this, the code has been written and optimized for readability and user experience. It is thoroughly commented and organized in a way that promotes understanding. There are also multiple blocks of code that are commented out and currently unused but that provide either a potentially good teaching example or debugging assistance. Examples include print loops for printing various different values to the console as well as a mock yaw controller for future implementation. Aside from readability, the user experience has also been considered from the point of view of creating a dynamic and engaging vehicle that is easy and fun to use. This was consistently kept in mind throughout the development process and many features were implemented to this end. Table 5.2 shows some of the features of the drive.c code. The last feature

Table 5.1: Drive.c key program features and their descriptions.

Feature	Description
Easy to drive while balancing	The drive and turn inputs from the DSM2 radio are damped out by a constant factor of 0.5 to allow for easier driving of the vehicle while balancing
Balance in four orientations	BeagleRover can balance on all four sides of the vehicle whether transitioning from cape side up or cape side down
Single balance controller for all four orientations	To streamline the code, the same balance controller is used for balancing in the tall orientation (NOSE_UP, NOSE_DOWN) as in the short orientation with simply a difference in gain K
Option to disable controller	Push the throttle stick up to disable the controller for driving around in four wheel mode and doing flips
Ackerman steering	Ackerman steering implemented in two of the four stable drive modes
Scaling of min/max reference angles	Minimum and maximum reference angles are scaled so that the user is less likely to drive the vehicle unstable while controlling from a DSM2 radion
Scaling of control input based on battery voltage	Control input (PWM to the motors) is scaled up by the ratio of the nominal battery voltage to actual battery voltage to compensate for loss of voltage
Transition to balancing from four wheel drive	BeagleRover can drive up a wall until upright then come away from the wall balancing on two wheels

listed, transition to balancing from four wheel drive, is explained in greater detail in the following subsection.

As listed in feature three of table 5.2, the same balance controller is used to balance in all four different orientations of the vehicle with the exception of the proportional gain value, K. The controller was designed around balancing in the short, or crab, orientation and used to balance in the tall orientation by simply increasing the gain to compensate for the higher center of gravity. As shown at the end of chapter three, the same exercise was gone through to design a controller specific to balancing in the tall orientation but upon

implementation, no significant performance gain was seen. Therefore it was decided to take the approach described above for the sake of streamlining the code for accessibility to users.

5.4.1 Drive to Balance Transition

BeagleRover is capable of the unique maneuver of smoothly transitioning from driving on four wheels to balancing on two wheels by driving up a wall, provided there is sufficient friction between the wheels and ground. This is done by initializing the balance controller only after the vehicle has reached a certain angle of incline, or the start angle. In order to achieve a smooth transition some finesse was required in timing at what start angle and how long of a delay to implement before arming the balance controller. It was expected that a delay would be necessary in order to prevent the controller from outputting too high of a control value upon initialization and causing instability. However, after experimentation with different combinations of values it was discovered that choosing the correct start angle alone was sufficient in achieving stability in the transition. Adding an additional delay is useful in affecting the overall delay in arming the controller without affecting the start angle as the start angle also affects computation of the complementary filter and in turn accurate estimation of the state variable. However the additional delay was found to be superfluous.

Chapter 6 Introduction to BeagleBone Robotics

Chapter one introduced this thesis by explaining its dual purpose. The first was to present the control and steering solution of a small ground rover capable of four wheel steering, balancing on two wheels and transitioning between the two. This was handled in chapters two, three and four. The second, and primary motivation of the work, was to contribute to an educational platform through developing curriculum support. Chapter six marks a change of focus from the quantitative solutions of the previous chapters to the presentation of the curriculum support material written as a result of this thesis work. More work is needed in order to deliver a polished curriculum written around the EduLine, however the combination of BeagleBone Robotics plus the hardware and CAD files for 3D printing the robots is already a very strong starting point for a motivated instructor.

As explained in section 1.2 almost no material written ex*clusively* toward the end goal of developing a dispersible curriculum around the EduLine existed on paper prior to the initiation of this thesis work. The word dispersible is key here as material that is designed to be distributed as a model for educators to follow in teaching particular concepts (a curriculum) takes on a different form than material that is written to provide support material in teaching those concepts (a textbook). A curriculum not only requires explanations of necessary concepts and example problems as found in a textbook, but lesson plans and adherence to standards often mandated by governing parties. In short, a curriculum teaches educators how to teach the subject at hand. What is presented in this

chapter and the next is a precursor to a complete curriculum designed around what is taught in MAE143C. It is hoped that this is a first step toward creating the written material necessary for the EduLine to make an impact in classrooms outside of UCSD. BeagleBone Robotics is considered by the author to be the most significant contribution of this thesis work to furthering the EduLine as an educational product.

6.1 BeagleBone Robotics Outline

BeagleBone Robotics is co-authored by Talesa Bleything and James Strawson, the teaching assistant for MAE143C for three iterations of the course and the designer of the Robotics Cape used on all EduLine robots. The text has three main parts. Part one is what we are referring to as the lab text and is primarily authored by James Strawson, with the author of this thesis acting as editor and first user. Part two is a complete set of build instructions for BeagleMiP and BeagleRover. Part three is the complete control solution to balancing BeagleMiP or BeagleRover. We recommend that MAE143C or similar courses designed around the EduLine be taught in a lab format, where regular classes are reserved for development of the control theory and supporting concepts while a special lab section is reserved for hardware related material. The next section gives brief outlines of parts one and two. Part three is presented in full in chapter seven.

6.1.1 BeagleBone Robotics Parts One and Two

Part one of BeagleBone Robotics begins by stepping through the getting started process with the BeagleBone Black as well as a description of the

workflow that will be used throughout the text. This includes instruction on how to use the various communication options such as networking over USB, Ethernet and Wi-Fi, as well as an intro to the Linux command line and file transfer protocols. The last of the getting started chapters is focused on installing and using the Robotics Cape.

The following chapters deal with circuit design and controlling hardware through GPIO and SPI protocols. These sections use an additional lab kit containing LEDs, wires, breadboard and a seven segment display. The last few chapters discuss how to use various features of the Robotics Cape and BBB including battery management, on board sensors, H-bridges for driving DC motors, buttons and LEDs, and counting quadrature encoders, all through the use of the Robotics Cape library. All of these chapters and topics are accompanied by exercises designed to ensure success with the hardware and programming environment. By the time the student reaches the end of part one, he or she should be armed with the tools necessary to successfully implement the control solution to balancing BeagleMiP or BeagleRover that is the culminating result of MAE143C.

Part two of BeagleBone Robotics contains the complete instructions for assembling BeagleMiP and BeagleRover. This is largely self-explanatory, however it should be emphasized that all parts, with the exception of the electronics, tires and motors, are 3D printed. Although BeagleMiP and BeagleRover are both designed to be robust and durable, the fact that it is 3D printed makes replacing broken parts fast and cheap, both of which are

important to an educational platform. This is not to mention the educational value of learning 3D printing technology in itself. As 3D printers become more and more common, prices will drop, making the technology increasingly available to a range of schools at the high school level as well as college. The design files for printing both vehicles are publicly available and instructions on how to access them are included with the build instructions.

Chapter 7 BeagleBone Robotics Part Three

The following sections of chapter seven present part three of BeagleBone Robotics in its entirety. It is currently written around BeagleRover and contains the same control solution for balancing on two wheels as was presented in chapter three, only in much greater detail and including some material taught in MAE143C that is not absolutely needed to balance the vehicle. It should also be made clear that by no means is everything that's taught in MAE143C included in this text. The format is written in an exercise, solution style and the language is less formal to match that of BeagleBone Robotics parts one and two. The material, including key terms and definitions, is presented in the order in which the author finds it most comprehensible.

7.1 Introduction to BeagleBone Robotics Part Three

This section is intended to provide a comprehensive solution to balancing the Beagle Rover on two wheels from the modeling and control design perspective. It is written in semi-chronological order so that users may obtain a clear and thorough understanding of the workflow required when using the classical control techniques that will be presented. Implementation in hardware and software, although alluded to here, is handled in detail in another section of the text. The workflow required refers to the order in which necessary concepts build upon each other as well as to the iterative process of control design. This text is written for MAE 143C, the technical elective undergraduate/graduate level Digital Control course taught at UCSD. In its entirety, BeagleBone Robotics can be used to varying degrees to support a

range of curricula, topics including but not limited to, classical control, digital control, embedded systems, robotics, dynamics, multithreaded programming, 3D printing, ordinary differential equations, digital/analog circuits and board design among others. The focus of this chapter is the dynamics and control used to balance the Beagle Rover and supporting concepts.

7.2 Using this Chapter

All control related tools and concepts used in this solution are taught in UCSD's MAE 143C course, however not everything taught in the course is used here. Additionally students are expected to have a basic understanding of ordinary differential equations and exposure to statics/dynamics is a plus. Included along the way are sample exercises to which this text provides solutions. We start with the equations of motion governing our system and end with a discrete time control law ready to be implemented in a microcontroller. Each subsection is meant to serve as a derivation of the solutions presented in order to *aid* the curriculum design and lesson planning process. This is not a complete curriculum and no section is intended to stand alone in teaching a particular concept. To this end, the tools necessary to understand each subsection are listed at the beginning of that section. The supporting textbook used in MAE 143C is Numerical Renaissance by Dr. Thomas Bewley.

7.3 Problem Statement

Our plant is the Beagle Rover and will be modeled as an inverted pendulum on two wheels as depicted in Figure 7.1. Beginning with the equations of motion describing the system, our end goal is to design a discrete

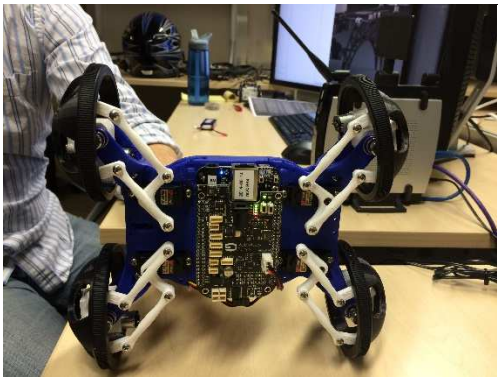

Figure 7.1: BeagleRover pictured on the left and 2D model of BeagleRover as a mobile inverted pendulum on the right.

time control law to stabilize the body angle, θ, about its upright, unstable equilibrium position. The final product will be a difference equation ready for implementation in a microcontroller, the BeagleBone Black in our case.

7.4 Equations of Motion

In order to derive the equations of motion for the Rover in balance mode, the system is modeled as a mobile inverted pendulum, an example of which is provided in Numerical Renaissance, Ex. 17.10 and followed closely here. The input to the system is torque from two motors and the output is body angle, θ. Initially, the equations are simplified by considering a 2D representation of one wheel/motor and a simple rod. The inertia and torque from the pair of wheels and motors will be integrated later. First, the free body diagrams of the wheel and rod are presented followed by the kinematic relations, dynamics and an integration of the two in order to derive the full nonlinear equations of motion.

7.4.1 Free Body Diagrams and Constants

Sample exercise: Sketch the 2D free body diagram of the wheel/rod

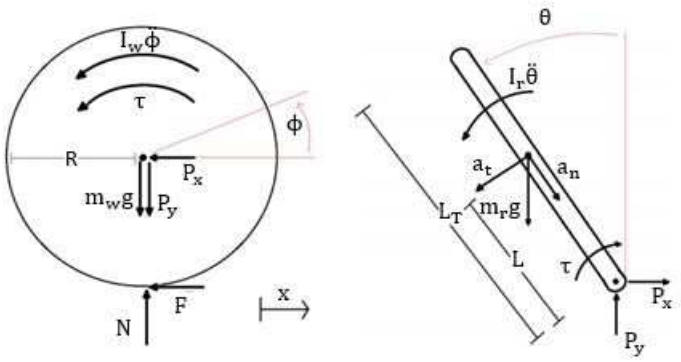

Figure 7.2: Free body diagrams of the wheel and rod.

system. Be sure to clearly mark the coordinate system(s) used. Calculate the inertia of the rod and wheels, approximating the wheels as solid discs.

Concepts and keywords: free body diagram, force, inertia, torque, perpendicular axis theorem, stationary vs. body coordinate systems, normal and tangential forces.

Solution: Where

$m_w = mass\ of\ wheel,$

$m_r = mass\ of\ rod,$

$I_w = inertia\ of\ wheel,$

$I_r = inertia\ of\ rod,$

$\tau = torque\ of\ motor,$

$R = radius\ of\ wheel,$

$L = length\ from\ end\ of\ rod\ to\ center\ of\ mass\ of\ rod,$

$P_x\ and\ P_y = reaction\ forces\ between\ wheel\ and\ rod\ in\ the\ x\ and\ y\ directions,$

$a_t = tangential\ force,$

$a_n = normal\ force,$

$g = gravity\ \left(9.81\frac{m}{s^2}\right),$

calculate the inertia of the wheels by approximating them as thin solid discs. By the perpendicular axis theorem, the inertia of one wheel is

$$I_w = \frac{1}{2} m_w R^2. \quad (7.1)$$

The inertia of the body approximated as a rod is

$$I_r = m_r L^2. \quad (7.2)$$

Whether balancing in crab mode or regular driving mode, take the center of mass to be at the intersection of the two axis of symmetry of the vehicle. This is an approximation as the "rod" in our model does not actually include the bottom two wheels. However, the mass of the wheels is negligible compared to that of the entire vehicle so for simplicity's sake we will take the center of mass to be at the intersection when in reality it is very slightly higher. A more rigorous calculation could be done but would likely not result in any performance gain, especially because some error would still be expected.

7.4.2 Kinematics

Sample exercise: Derive an equation describing the position of the center of mass of the rod in terms of x(t) and $\theta(t)$, and another describing the acceleration.

Concepts and key words: differentiation, vector, unit vector, center of mass, acceleration, basic trigonometry, basic algebra, stationary vs. body coordinate system.

Solution: Define **r(t)** as the position vector from a *stationary* coordinate system as defined in Figure 7.2 to the center of mass of the rod, x(t) as the horizontal position of the center of the wheel also measured from a stationary

coordinate system, and $\theta(t)$ as tilt angle of the rod measured counter clockwise from upright. Writing **r(t)** as a function of x(t) and $\theta(t)$ we get the kinematic relationship

$$\boldsymbol{r} = x\boldsymbol{e^1} - Lsin(\theta)\boldsymbol{e^1} + Lcos(\theta)\boldsymbol{e^2}. \tag{7.3}$$

Differentiating twice yields the acceleration

$$\ddot{\boldsymbol{r}} = \ddot{x}\boldsymbol{e^1} - \ddot{\theta}Lcos(\theta)\boldsymbol{e}^1 + \dot{\theta}^2Lsin(\theta)\boldsymbol{e}^1 - \ddot{\theta}Lsin(\theta)\boldsymbol{e^2} - \dot{\theta}^2Lcos(\theta)\boldsymbol{e^2}. \tag{7.4}$$

Define $\boldsymbol{e}^{\perp} = \boldsymbol{e^1}\,cos(\theta) + \boldsymbol{e^2}sin(\theta)$ as the direction perpendicular to the rod and $\boldsymbol{e}^{\parallel} = \boldsymbol{e^2}\,cos(\theta) - \boldsymbol{e^1}sin(\theta)$ as the direction parallel to the rod and plug into the above equation to get

$$\ddot{\boldsymbol{r}} = \left[cos(\theta)\,\ddot{x} - L\ddot{\theta}\right]\boldsymbol{e}^{\perp} - [sin(\theta)\,\ddot{x} + L\dot{\theta}^2]\boldsymbol{e}^{\parallel} \tag{7.5}$$

7.4.3 Dynamics

Sample exercise: Assuming that both wheels are initially moving together so that there is no turning, and that there is no slip between the wheels and the ground, derive the nonlinear equations of motion of the wheel/rod system in terms of body angle, θ, wheel angle, ϕ, and input torque from the motors τ. For now think of τ as a single input value to the system. How does the torque applied by the motors affect the wheel? The body?

Concepts and key words: Newton's 2nd law of motion/rotation, dot product, equations of motion.

Solution: Define P_x and P_y as the forces that the rod exerts on the wheels in the positive $\boldsymbol{e}^1$ and $\boldsymbol{e}^2$ directions and $\phi(t)$ as the rotation of the wheel measured counterclockwise from a reference position. As the motor spins it applies a torque to the wheel that spins the wheel in one direction and causes

the rod to rotate in the opposite. In order to write down the dynamics for our system we will make use of the following common equations:

Newton's Second Law of Motion

$$\Sigma F = ma, \tag{7.6}$$

Newton's Second Law for Rotation

$$\Sigma \tau = I\alpha, \tag{7.7}$$

Arc of a Circle (ϕ in radians)

$$arc\ length = r\phi, \tag{7.8}$$

Definition of Dot Product

$$\boldsymbol{A} \cdot \boldsymbol{B} = \|A\|\|B\|cos(\theta). \tag{7.9}$$

Using these equations and making two key assumptions, the first being that both wheels are initially moving together (no turning) and the second that there is no slip between the ground and the wheels, we can write down the following dynamic equations as well as the position of the wheel center:

Position of wheel center

$$x = R\phi, \tag{7.10}$$

Acceleration of rod in $\boldsymbol{e}^{\perp}$

$$m_r[\ddot{\boldsymbol{r}} \cdot \boldsymbol{e}^{\perp}] = m_r\left[cos(\theta)\,\ddot{x} - L\ddot{\theta}\right] = -mrgsin(\theta) - P_y\,sin(\theta) - P_x co\,s(\theta), \tag{7.11}$$

Acceleration of rod in $\boldsymbol{e^1}$

$$m_r\left[\ddot{\boldsymbol{r}} \cdot \boldsymbol{e^1}\right] = m_r\left[\ddot{x} - Lcos(\theta)\ddot{\theta} + Lsin(\theta)\dot{\theta}^2\right] = -P_x, \tag{7.12}$$

Acceleration of wheel center in $\boldsymbol{e^1}$ where F is friction force between wheel and ground

$$\mathrm{m_w}\ddot{\mathrm{x}} = \mathrm{P_x} - \mathrm{F}, \tag{7.13}$$

Rotational acceleration of rod

$$I_r\ddot{\theta} = -\tau - P_y L sin(\theta) - P_x L cos(\theta), \tag{7.14}$$

Rotational acceleration of wheel

$$I_w\ddot{\phi} = \tau - RF. \tag{7.15}$$

where m_w and I_w are the mass and moment of inertia of *both wheels.* We will account for the torque of both motors later when applying control. For now it's fine to think of τ as a single value representing total input torque to the system.

7.4.4 Nonlinear Equations of Motion

Our goal is to derive two equations that together describe the motion of the rod and the wheel as the torque from the motors is varied. Therefor we would like these equations to be in terms of θ, ϕ and τ. Begin by rearranging equation 7.14 to get

$$P_y\, sin(\theta) - P_x\, cos(\theta) = \frac{I_r\ddot{\theta} - \tau}{L}. \tag{7.16}$$

Plugging into equation 7.11, multiplying by L and distributing m_r yields

$$m_r L\ddot{x}\, cos(\theta) - m_r\ddot{\theta}L^2 = -m_r L g sin(\theta) + I_r\ddot{\theta} + \tau. \tag{7.17}$$

Rearrange to get **Equation of Motion 1**

$$-(m_r L cos(\theta)\ddot{x}) + (I_r + m_r L^2)\ddot{\theta} = m_r L g sin(\theta) - \tau. \tag{7.18}$$

Now rearrange equations 7.13 and 7.15 to get

$$P_x = m_w\ddot{x} + F \tag{7.19}$$

and

$$F = -(\frac{I_w\ddot{\phi} - \tau}{R}). \tag{7.20}$$

Plugging both 7.19 and 7.20 into equation 7.12 yields

$$m_r\left(-\ddot{\theta}Lcos(\theta)+\dot{\theta}^2Lsin(\theta)\right)=-m_w\ddot{x}+(\tfrac{I_w\ddot{\phi}-\tau}{R}). \quad (7.21)$$

Multiplying 7.21 by R and distributing m_r gives **Equation of motion 2.**

$$I_w\ddot{\phi}-(m_wR+m_rR)\ddot{x}+\left(m_rRLcos(\theta)\right)\ddot{\theta}=m_rR\dot{\theta}^2\,sin(\theta)+\tau \quad (7.22)$$

Finally applying the no slip condition of equation 7.10, we get equations 7.23 and 7.24 which are the final nonlinear equations of motion of our system.

$$\left(m_rRLcos(\theta)\right)\ddot{\phi}+(I_r+m_rL^2)\ddot{\theta}=m_rLgsin(\theta)-\tau \quad (7.23)$$

$$(I_w+(m_r+m_w)R^2)\ddot{\phi}+\left(m_rRLcos(\theta)\right)\ddot{\theta}=m_rRL\dot{\theta}^2sin(\theta)+\tau \quad (7.24)$$

7.4.5 Linearization

Exercise: Linearize the equations of motion about the body's inverted equilibrium point using small angle approximation.

Concepts and key words: Linearity, linearize, small angle approximation, Taylor series expansion, perturbation.

Solution: The equations of motion as they stand are nonlinear in θ. In order to apply linear control techniques, we must linearize the equations. Because our control algorithm will be designed to continuously correct the system back to zero error ($\theta=0$) we can use small angle approximation, considering very small perturbations to theta around its inverted equilibrium point. Making the substitution $\theta=\bar{\theta}+\theta'$ where $\bar{\theta}=0$ and extending this to ϕ and τ accordingly, the perturbation equations are:

$$\left(m_rRLcos(\theta')\right)\ddot{\phi}'+(I_r+m_rL^2)\ddot{\theta}'=m_rLgsin(\theta')-\tau', \quad (7.25)$$

$$(I_w+(m_r+m_w)R^2)\ddot{\phi}'+\left(m_rRLcos(\theta')\right)\ddot{\theta}'=m_rRL\dot{\theta}'^2sin(\theta')+\tau. \quad (7.26)$$

Applying the truncated Taylor series expansion resulting from the small angle

approximation

$$sin(\theta') \approx \theta' - \frac{\theta'^3}{3!}, \tag{7.27}$$

$$cos(\theta') \approx 1 - \frac{\theta'^2}{2!} \tag{7.28}$$

and neglecting all primed quantities that are quadratic or higher (since the square of a small number is an even smaller number), the linearized equations of motion are

$$(m_r RLcos)\ddot{\phi} + (I_r + m_r L^2)\ddot{\theta} = m_r Lg\theta - \tau, \tag{7.29}$$

and

$$(I_w + (m_r + m_w)R^2)\ddot{\phi} + (m_r RL)\ddot{\theta} = \tau. \tag{7.30}$$

7.5 Control

Classical control methods using lead/lag control and pole placement will be used to stabilize the body of the Rover on two wheels about its unstable equilibrium position. We will be leveraging the discrete equivalent design approach, designing a controller in continuous time and later converting to discrete time for implementation in a microcontroller. The primary tools and techniques we will use include the Laplace and Z transforms, lead and lag control, bode and root locus plots, the closed loop step response and Tustin's approximation with prewarping, all of which will be discussed in more detail below. The control process that follows is iterative in nature as the designer applies the aforementioned tools in a deliberate fashion in order to meet the desired performance specifications such as rise time, settling time and overshoot of the system. Approximate design guides used during pole placement are provided for assistance and it is useful to have them at one's

disposal before beginning the control design process. We will briefly list them next.

7.5.1 Approximate 2nd Order Design Guides

Concepts and key words: order of a system, 2nd order behavior, step input, s-plane, natural frequency, poles of a transfer function, lead control, crossover frequency, rise time, settling time, percent overshoot, pole placement.

As the title of this section suggests, the following design guides are most applicable to second order systems however will also provide helpful guidance for systems of higher order, especially if the system is characterized by 2nd order behavior. Some commonly used characteristics of the step response of a system and the corresponding design guides are below.

1. **Percent overshoot** of the system is defined as the maximum percent by which the output of the system in response to a step input exceeds its steady state value.

$$\zeta \geq 0.5 \rightarrow M_p \leq 15\% \tag{7.31}$$

$$\zeta \geq 0.7 \rightarrow M_p \leq 5\% \tag{7.32}$$

2. **Rise time** of the system is defined as the time it takes for the output of the system to a step input to reach 0.9 of the steady state response

$$t_r = {}^{1.8}/_{\omega_n} \tag{7.33}$$

where ω_n is the natural frequency.

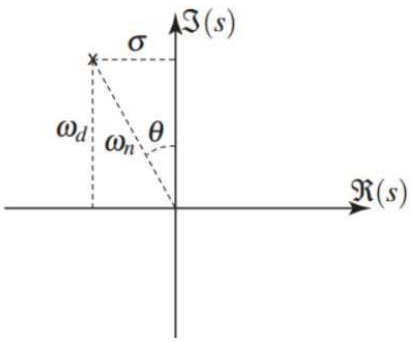

Figure 7.3: Graphical depiction of the s-plane showing how the natural frequency of the system is affected by pole location.

3. **Settling time** of the system is defined as the time it takes for the output of the system to a step input to settle to within $\pm 5\%$ of the steady state value.

$$t_s = {}^{4.6}/_{\sigma} \tag{7.34}$$

Some other useful guidelines are:

4. **Pole Location** of the plant transfer function in the s-plane and how it affects the natural frequency of the system is shown in Figure 7.3.
5. For good **phase bump** when implementing lead control

$$\varphi_{max} \to {}^{p}/_{z} = 10. \tag{7.35}$$

6. To **achieve crossover frequency** at an order of magnitude below the sample rate of the ADC

$$\omega_c = {}^{1.8}/_{t_r}, \tag{7.36}$$

$$\omega_c = \sqrt{pz}. \tag{7.37}$$

Note these guidelines are approximate and it is likely that the designer will not achieve all criteria exactly.

7.5.2 G1(s)

Exercise: By first taking the Laplace transform of the linearized equations of motion, derive the transfer function of the plant, $G1(s)$, from input $\tau(s)$ to output $\theta(s)$.

Concepts and key words: transfer function, Laplace transform.

Solution: For a first pass at designing a stabilizing controller, take the input to the system to be τ and output θ. Motor dynamics will be included later. To obtain the transfer function from input τ to output θ, take the Laplace transform of the linearized equations of motion and rearrange to get an input/output relationship of the form $\theta(s)/\tau(s) = num\ G1(s)/denG1(s)$. Applying the definition of the Laplace transform and assuming zero initial conditions, the transformed equations are

$$(m_r RLcos)s^2\phi(s) + (I_r + m_r L^2)s^2\theta(s) = m_r gL\theta(s) - \tau(s), \tag{7.38}$$

$$(I_w + (m_r + m_w)R^2)s^2\phi(s) + (m_r RL)s^2\theta(s) = \tau(s) \tag{7.39}$$

Solving equation (7.39) for $\phi(s)$, plugging into equation (7.38) and algebraically rearranging, we get the transfer function from input $\tau(s)$ to output $\theta(s)$.

$$G1(s) = \frac{m_r RL + I_w + (m_r + m_w)R^2}{(m_r{}^2R^2L^2 - (I_w + (m_r + m_w)R^2))(I_r + m_r L^2)s^2 + (m_r gL)(I_w + (m_r + m_w)R^2)} \tag{7.40}$$

From here on, assume all modeling and calculations to be done using MATLAB. For balancing in crab mode, referring to the constants depicted in and listed below the free body diagrams (Fig. 5.2), take $L = 0.06m$, which is at the intersection of the two axis of symmetry of the vehicle, moment of inertia for the rod and wheels to be $I_r = m_r L^2$ and $I_w = m_w L^2$ (inertia of both wheels) with $m_r = 0.612kg, m_w = 0.054kg$ (mass of both wheels) and normalizing so that

the highest power of s has a coefficient of 1 in the denominator, we get:

$$G1(s) = \frac{-913}{s^2 - 146.4} \tag{7.41}$$

Now that we have the transfer function, we may begin the control design process using the classical control techniques and design guides highlighted thus far.

7.5.3 Discrete Equivalent Design

Exercise: What is the Nyquist frequency and how does it affect the system? How can the control designer compensate for this effect?

Concepts and key words: block diagram, DAC, ADC, ZOH, microcontroller, bode plot, frequency domain, phase margin, Tustin's approximation with prewarping, Padé approximation, crossover frequency, Nyquist frequency, aliasing, low pass filter, sample time.

Solution: To design our controller, we will use the discrete equivalent design approach. The controller is designed in continuous time and later converted to discrete time for implementation in the discrete time electronics. The continuous time controller, D(s), is represented as a cascade of the analog to digital conversion of the error signal (obtained by comparing the output of the sensors to a reference signal), the discrete time controller, D(z), and the digital to analog conversion necessary to provide analog signals to the motors. When designing a controller in continuous time, the designer must be cognizant of the h/2 time delay that results from the use of a zero-order-hold in the microcontroller's DAC, where h the sample time of the ADC. The ADC, D(z), DAC cascade and the Laplace transform of the resulting delay are depicted in the

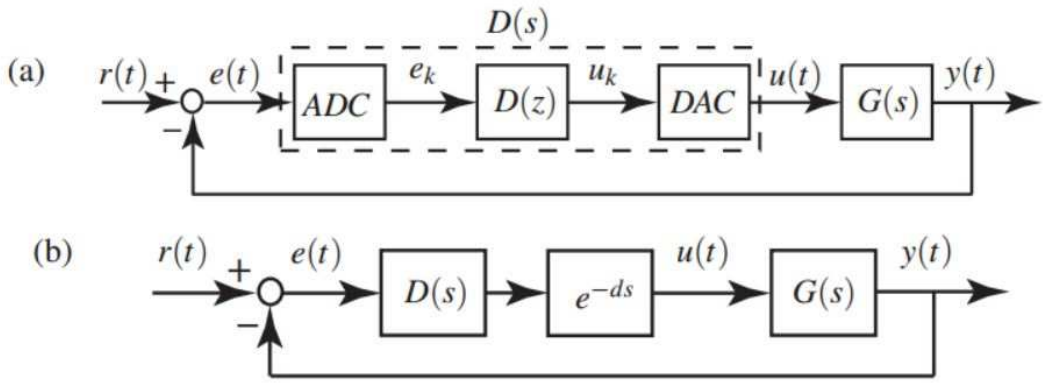

Figure 7.4: Block diagram of the controller/plant system. The bottom diagram shows the series connection of the controller and plant with the delay that arises from the digital-to-analog conversion of the discrete time controller output. The top diagram shows the discrete time version of the controller including the necessary data conversion for communicating with the sensors and motors.

block diagrams of Figure 7.4.

The h/2 time delay results in potentially significant phase loss if occurring within an order of magnitude of the $\boldsymbol{Nyquist\ frequency} = \pi/2$, defined as the frequency above which aliasing occurs and the output of the ADC into the discrete time controller can no longer be trusted as accurate. To compensate for this, most ADC's incorporate a low pass filter above the Nyquist frequency which also results in phase loss. The combined phase loss can be problematic if the phase at the crossover frequency is eroded enough to potentially cause unacceptably high overshoot and or closed loop instability of the system. To account for the phase loss, the designer can either represent the time delay resulting from the DAC's ZOH with a Padé approximation built directly into the continuous time representation of the plant, G(s), or simply build enough of a phase margin in at crossover to insure stability even with the h/2 phase loss. After designing D(s) to meet the design specifications such as the desired rise time and settling time, we will use Tustin's approximation with prewarping to

obtain the discrete time controller, D(z), that the microcontroller is to obey. We will first go through the process of incorporating a Padé approximation of a delay into G1(s) and designing a stabilizing controller so that the user of this text may have it as an example. Later it will be omitted.

7.5.4 Padé Approximation

Exercise: Comment on the phase margin and how this can be affected. What type of control can be used to stabilize the system?

Concepts and key words: Padé approximation, rational function, IMU, sample time, root locus, bode plot, s-plane and stability, "speed" of a system.

Solution: As depicted in the above block diagram, the Laplace transform of the delay function is e^{-ds} which is not a rational function of s. Implementation of an irrational function in a discrete time microcontroller is problematic, therefor we will use a Padé approximation in its place given by $e^{-ds} \approx F_n(s)$ where $F_n(s)$ increases in accuracy with higher values of n. For our purposes $n = 2$ is sufficient and the approximation is

$$F_2(s) \approx \frac{1-ds/2+(ds)^2/12}{1+ds/2+(ds)^2/12}. \tag{7.42}$$

We must now specify exactly the h/2 time delay to be used in MATLAB's Padé function. The IMU on the Robotics Cape has a sample time of 200Hz which translates to a time delay of $d = 0.005/2$ seconds. The resulting Padé approximation is

$$P(s) = \frac{s^2-2400s+1.92e^{0.06}}{s^2+2400s+1.92e^{0.06}}. \tag{7.43}$$

Combining $\mathrm{P(s)}$ with $\mathrm{G1(s)}$ and applying a negative gain results in the root locus

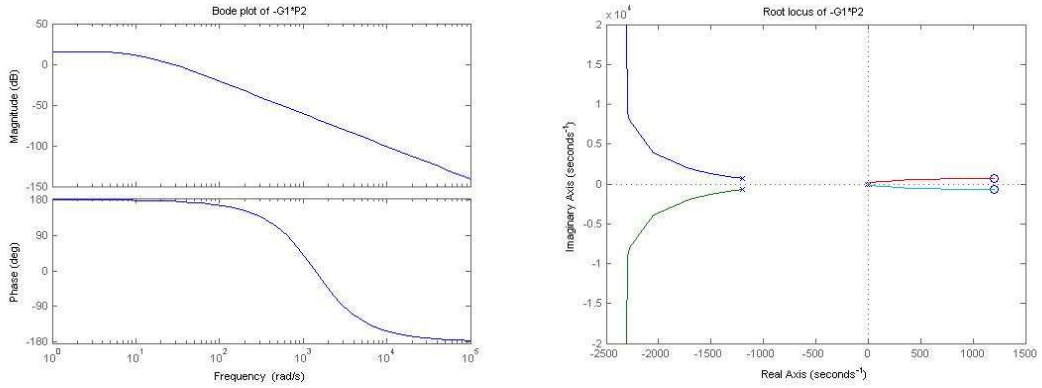

Figure 7.5 Bode (left) and root locus (right) plots of the plant, G1(s), combined with a second order Padé approximation of the delay introduced to the system by the BeagleBone Black's DAC.

and bode plots of the "new plant," as pictured in Figure 7.5.

The n=2 Padé approximation added two stable, very fast poles which will decay quickly and not affect the plant dynamics in a significant way. It can be seen from the root locus that no amount of gain can be applied to stabilize the system. Also, looking at the bode plot, there is no phase margin which is defined as the amount of phase the open-loop system is away from 180 degrees at the frequency where the magnitude crosses 1, aka the crossover frequency. In order to bump up the phase margin and achieve stability, a lead controller will be applied.

7.6 Lead Control

Exercise: Design a lead controller to stabilize the G1(s)P2(s) system. Clearly explain the reasoning behind your choice of the pole and zero locations of the controller as well as the overall gain, K, of the system. Provide the root locus and bode plots. What can you say about tracking of the reference signal? How can this be affected?

Concepts and key words: Gain, lead control, reference signal, tracking, lag control.

Solution: The form of a lead controller is

$$D_{lead}(s) = K * \frac{s+z}{s+p} \; where \; z < p. \tag{7.44}$$

The primary goals of applying lead control here are to bring the locus into the stable left half plane while simultaneously speeding up the system and increasing the phase margin. If we choose z = 12.1 to cancel the stable pole and replace with a faster pole, we can decrease the rise time and settling time of the step response without increasing the gain too high and potentially risking instability. By design guide 4, choosing p=120 should result in an appropriate phase margin. Then adjust the gain K to achieve crossover at the desired frequency, increasing it to bump up the magnitude of the bode plot and vice versa. We shoot for a crossover frequency of about 20 rad/s as that is an order of magnitude below the sampling frequency of the ADC, another design guide. After iterating to achieve close to the desired crossover and an acceptable phase margin, a potential lead controller is

$$D_{lead}(s) = K * \frac{s+12.1}{s+120} \; with \; K = 2.75. \tag{7.45}$$

After combining the lead controller and plant (with Padé approximation) in series, the root locus and bode plots are shown in Figure 7.6.

This is only one potential solution, yielding a phase margin of $\approx 45°$ at a crossover of $\approx$ 17 rad/s with a damping of 0.6. Notice that the magnitude at

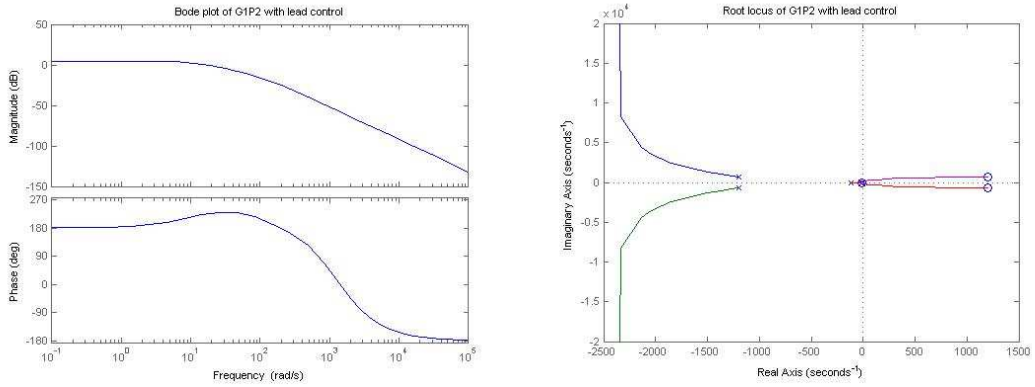

Figure 7.6: Bode (left) and root locus (right) plots of the plant, G1(s), and n=2 Padé approximation of the delay function, P(S), with lead control applied.

low frequencies is not very high and could result in poor tracking of the reference signal. This could be affected by adding lag control which will be exemplified after incorporating the motor dynamics.

7.7 Including Motor Dynamics

Exercise: Recalculate the inertia of the wheels including the inertia of the motors/gearboxes using $I_e = 3.6E-8\ Kgm^2$ and the given information below.

Exercise: Given the equation for motor torque, incorporate the motor dynamics into the original linearized equations of motion of the system, apply the Laplace transform and rearrange as before to derive the new transfer function of the system from input U(s) to output $\theta(s)$. Plot the root locus and bode plots. Comment. What is the format of a possible stabilizing controller?

Solution: Although we have now achieved a stabilizing controller in simulation, the robot itself would very likely not balance in reality. This is due to the fact that the motors themselves have dynamics that we have hitherto neglected. We will include them now and follow a similar control design approach, utilizing the root locus and bode plotting tools, this time with the

addition of the closed loop step response and lag control. The motor dynamics can be modeled as

$$\tau = \bar{s}u + (b - \zeta)\omega \tag{7.46}$$

where:

$\bar{s} = stall\ torque,$

$b = damping\ coefficient,$

$\zeta = viscous\ friction,$

$I_e = motor\ armature\ inertia,$

$\omega = motor\ speed\ (\dot{\phi} - \dot{\theta})\ or\ (\dot{\theta} - \dot{\phi})\ depending\ on\ motor\ polarity,$

$u = motor\ input\ (PWM\ value\ between -1\ and\ 1).$

Additionally, when calculating the inertia of the wheels and motors combined, we must multiply the motor armature inertia by the square of the gearbox ratio prior to summing with the wheel inertia. As before, the wheels are taken to be solid disks when estimating their inertia. The total inertia of one wheel and motor is

$$I_w = I_e + \frac{1}{2}m_w R^2 \tag{7.46}$$

and the final linearized equations of motion accordingly become

$$(m_r RL)\ddot{\phi} + (I_r + m_r L^2)\ddot{\theta} = m_r gL\theta - 2\tau \tag{7.47}$$

$$(I_w + (m_r + m_w)R^2)\ddot{\phi} + (m_r RL)\ddot{\theta} = 2\tau \tag{7.48}$$

where τ is defined above to be the torque from one motor, m_w now includes the mass of both wheels and I_w now includes the inertia of both wheels and gearboxes. Making the proper substitutions, taking the Laplace transform, and rearranging algebraically exactly as before yields the new transfer function

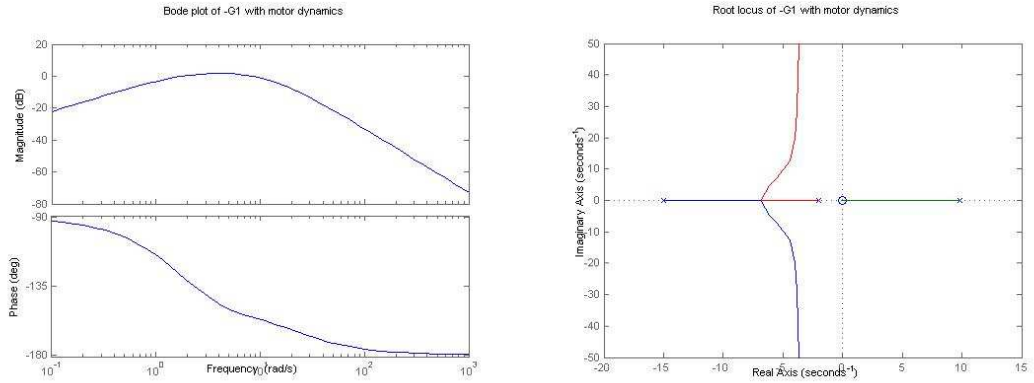

Figure 7.7: Bode (left) and root locus (right) plots of the plant G1(s) incorporating motor dynamics.

$$G1(s) = \frac{\theta(s)}{U(s)} = \frac{2\bar{s}(C4+C1)}{(-C4C2+C1^2)s^3+(2(\zeta+b)(-C4-C2-2C1)s^2+(C3+C4)s+2(\zeta+b)C3} \quad (7.49)$$

where:

$$C1 = m_r RL, \quad (7.50)$$

$$C2 = I_r + m_r L^2, \quad (7.51)$$

$$C3 = m_r gL, \quad (7.52)$$

$$I_w + (m_r + m_w)R^2. \quad (7.53)$$

Plugging in the constants for these particular motors gives the final transfer function.

$$G1(s) = \frac{\theta(s)}{U(s)} = \frac{-226.2s}{s^3+7.221s^2-136.7s-292.2} \quad (7.54)$$

The root locus and bode plots with negative gain applied are given in Figure 7.7. Looking at the root locus, no amount of gain can stabilize the system. In order to bring the locus into the LHP using pole placement techniques, our controller must cancel the zero at the origin and replace it with a stable zero. Note this is generally done with caution as a pole-zero cancellation on the imaginary axis can lead to instability due to inaccuracies in the model. We are confident in our model and will perform this cancellation. Looking at the bode

plot, we do not have a high enough magnitude at low frequencies to achieve acceptable tracking. By applying lag control over the appropriate frequencies we can bump up the low frequency gain as well as cancel the zero at the origin, replacing it with a stable zero and bringing the locus into the LHP as desired.

7.8 Lag Control

Exercise: Design a lag controller to stabilize the system and plot the root locus and bode plots. Comment.

Concepts and key words: Lag control, phase lag, integrator windup.

Solution: The form of a lag controller is

$$D_{lag}(s) = \frac{s+z}{s+p} \; for \; z > p. \tag{7.55}$$

The primary use of a lag controller is to increase the gain at low frequencies in order to achieve good tracking. The gain will increase by a factor of z/p. Because our pole is at zero, we are able to choose z such that our lag control takes affect at frequencies well below our desired crossover frequency of 20rad/s. This is necessary due to the phase lag caused by the lag controller. We do not want to erode the phase at crossover and risk instability. After a few iterations, we choose z=3 and the resultant bode plot of our lag controller is given in Figure 7.8, illustrating the phase lag effect of a lag controller as well as the high gain at low frequencies. Notice the lack of roll off of the gain at low frequencies, which is the effect of having a controller pole at zero, acting as a pure integrator. Ideally we would roll off this gain at very low frequencies to

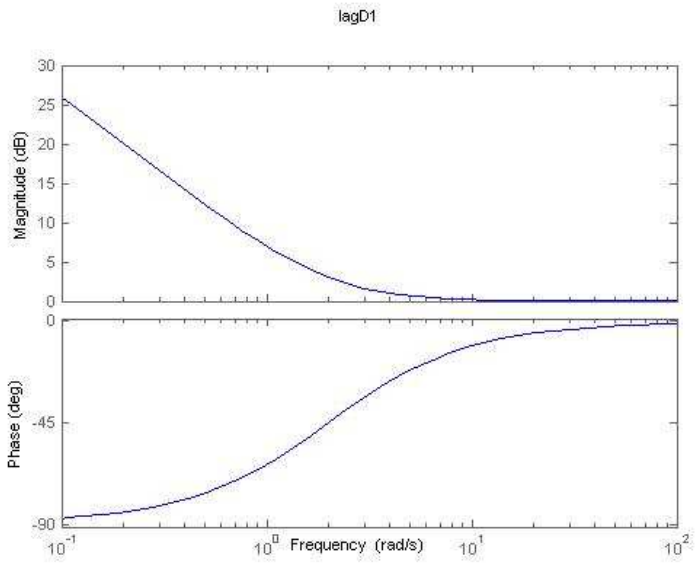

Figure 7.8: Bode plot of lag controller D_lag (s).

avoid integrator wind up, a phenomenon that can lead to instability in the case of motor saturation, however in this case we cannot. Combining this lag control with the plant and a negative gain yields the root locus and bode plots given in Figure 7.9.

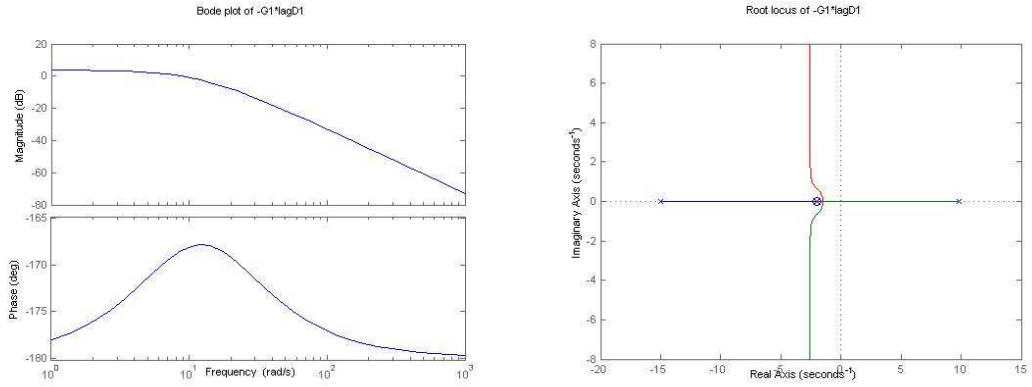

Figure 7.9: Bode (left) and root locus (right) plots of lag control combined with the plant G1(s).

7.9 Lead Control with Motor Dynamics

Exercise: Add lead control and plot the root locus and bode plots.

Identify design specifications. Comment.

Concepts and Key words: Lead control, phase margin, crossover frequency, pole zero cancelation, rise time.

Solution: The bode plot with lag control shows that we have increased the low frequency gain but have indeed dangerously eroded our phase margin. Also, we do not have crossover at the desired frequency. The locus is now brought over into the stable LHP with the appropriate gain applied, however this design does not meet the necessary specifications such as damping, etc. In order to achieve crossover at desired frequency and increase phase margin, we will add lead control using much of the same logic as we did previously. In order to design our lead controller and achieve crossover where we desire we will use the previously introduced design guides.

$$\omega_c \approx {}^{1.8}/_{t_r} \; where \; t_r = rise \; time \tag{7.56}$$

$$\omega_c = \sqrt{pz} \; where \; pz = pole * zero \tag{7.57}$$

As before, placing a zero at -15 to cancel the pole and replacing with a faster pole will help to achieve the required rise time without applying too much overall gain to the system. Keeping in mind that we would like a rise time of 0.05-0.1 seconds, and a crossover frequency of ≈20 rad/s, we can use the above equations to calculate the location of the pole that should achieve close to our desired specifications. A rise time of 0.05 seconds corresponds to a crossover frequency of 36 which gives us a pole at 86.4. This is not quite a factor of 10 higher than the zero at 15 to achieve max phase bump, but we still have a phase margin of about 52 degrees which is sufficient to account for any future

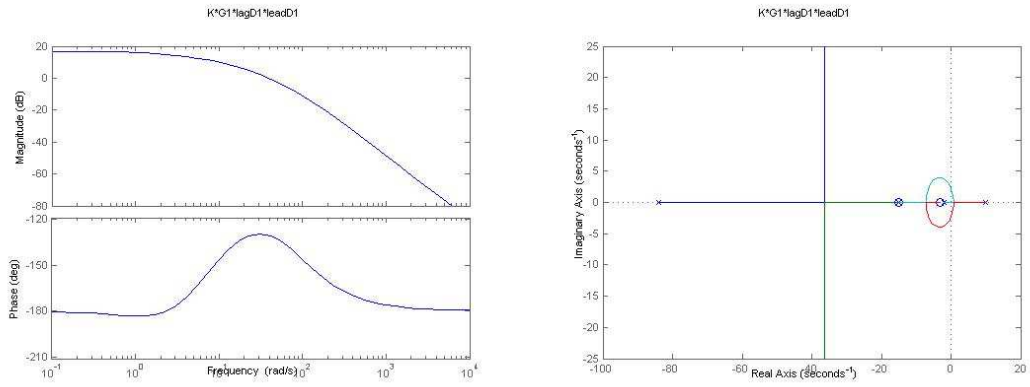

Figure 7.10: Bode (left) and root locus (right) plots of lead and lag control applied to the plant, G1(s), showing high gain at low frequencies for good tracking of the reference signal and ample phase bump at the crossover frequency.

phase loss associated with implementation in digital electronics. Now adjust the gain to get crossover at the target frequency. Cannot meet all approximate design guides exactly so compromise at $\omega_c = 23$, $t_r \approx 0.08$, $pm = 52$. The final open loop bode and root locus plots with lead control applied are given in Figure 7.10.

7.10 Closing the Loop

Exercise: Plot the closed loop step response. Comment on whether or not design specifications were met.

Concepts and Key words: closed loop, step input, step response, rise time, settling time, overshoot, loop prefactor.

Solution: Referring to $D_{lead-lag}(s)$ as $D(s)$, the form of the closed loop transfer function is

$$H(s) = \frac{K(s)D(s)G1(s)}{1+K(s)D(s)G1(s)} \tag{7.58}$$

Performing a few final tweaks, we end with $D_{lag} = (s + 2.5)/s$ and $D_{lead}(s) = (s + 15)/(s + 86.4)$ with a gain of -10. Giving a step input to this system

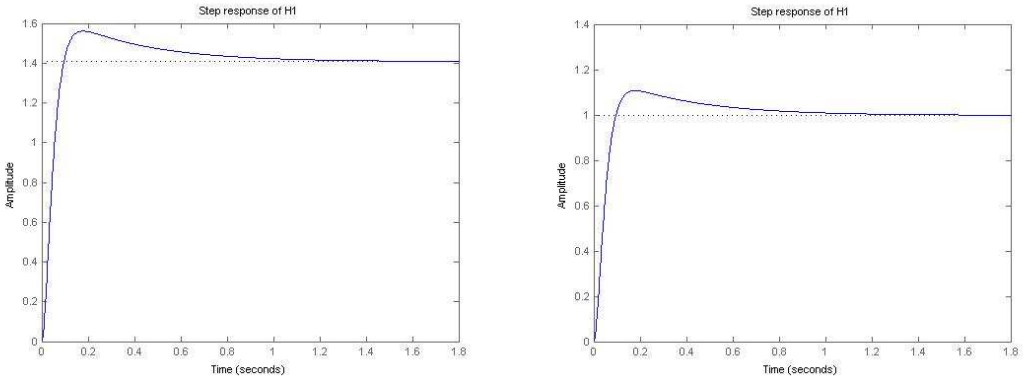

Figure 7.11: Step response of closed loop system without a loop prefactor on the left and with a loop prefactor on the right. Addition of the loop prefactor causes the response to settle at one as desired.

results in the step response plotted on the left hand side of Figure 7.10 showing a rise time of 0.06 s, settling time of 0.79 s and an overshoot of 10%. Notice that the response settles at a value of ≈1.41 instead of 1 as desired. This can be fixed by adding a loop prefactor of P = 1/1.41 as shown in equation (7.59). The step response now becomes that pictured on the right hand side of Figure 7.11.

$$H(s) = P * \frac{K(s)D(s)G1(s)}{1+K(s)D(s)G1(s)} \tag{7.59}$$

7.11 Discrete Time Controller

Exercise: Convert the continuous time controller to discrete time and derive the corresponding difference equation to be implemented in hardware.

Concepts and Key Words: discrete time, difference equation, transfer function, Tustin's approximation, prewarping, Z-transform, inverse Z-transform.

Solution: In order to implement the controller in hardware, we need to convert from continuous time to discrete time and obtain the difference equation the controller must obey. To convert to a discrete time transfer function we use Tustin's approximation with prewarping. We apply prewarping

around the crossover frequency to insure accurate conversion at this frequency. Then applying the inverse Z transform gives us the difference equation to be implemented in the microcontroller.

$$u_{k+2} - 1.652u_{k+1} + 0.6525u_k = -8.626\theta_{k+2} + 16.52\theta_{k+1} - 7.902\theta_k \qquad (7.60)$$

7.12 Balancing in Normal Drive Orientation

Exercise: Derive the transfer function of the system for BeagleRover in its tall orientation.

Concepts and Key Words: No new concepts or key words.

Solution: In order to balance BeagleRover in its tall orientation, we must account for the change in the distance from the center of the wheel to the center of mass of the body, L, and the corresponding change in the body inertia, I_r. Changing these values and recalculating, the transfer function for the new orientation becomes

$$G2(s) = \frac{-137.7s}{s^3+5.917s^2-96.48s-206.2} \qquad (7.61)$$

with poles at -12.4, -1.98 and 8.43 which are slightly "slower" than the poles of G1(s) when balancing in crab mode.

Chapter 8 Conclusions and Future Work

The purpose of this work is twofold. First, it is dedicated to the technical development of BeagleRover, a small RC car capable of numerous different drive modes including balancing and driving on two wheels. Second, it is to lay groundwork for how BeagleRover, as part of a larger platform currently being called the EduLine, can be used to affect STEM education. The technical portion of this paper centers on the control algorithm designed to achieve balance on two wheels as well as the smooth transition between driving on four wheels to driving on two using a wall. This is accomplished through classical and digital control methods, namely lead and lag control techniques. Implementation in a microcontroller is considered including the use of a complementary filter to adjust for sensor noise characteristics. Lastly, four-wheel steering is augmented by the addition of Ackermann steering geometry.

Successful implementation of the classical control and filtering techniques mentioned above has resulted in a vehicle capable of balancing on all four of its sides in two different unstable configurations. These consist of a "tall" configuration and "short" configuration, referring to a higher and lower center of mass respectively. The difference in location of the center of mass was handled by increasing the gain while balancing in tall mode. This proved to be sufficient in achieving balance however the vehicle does seem to display better disturbance rejection in its short configuration. While driving in Normal mode (on all four wheels with the front of the vehicle being what one would expect) Ackermann steering geometry was implemented to reduce side slip of

the vehicle while moving through a turn. Some simple tests leveraging the on-board gyroscope showed side slip does seem to be reduced by Ackermann geometry when going through a tight as well as a wide turn on a hard wood floor. The implementation of Ackermann geometry was also shown to provide a strong educational skew that is accessible at the high school level, leading into the educational portion of this thesis work.

Throughout the technical developments of this work, applications to STEM education are considered, with each major section including a discussion of its particular relevance to that topic. The education centric portion of this work culminates in a text we are currently calling BeagleBone Robotics (BBR), a portion of which is given in chapter seven. BeagleBone Robotics currently consists of a hardware section that is focused on such topics as getting up and running with the BeagleBone Black and programming in Linux, as well as a theoretical section on control design. The theoretical portion is written in a question and answer format designed to be used as a complete solution set for balancing BeagleRover on two wheels. This section is not currently augmented to be BeagleMiP specific (Beagle MiP being the first robot in the EduLine that is currently used to teach MAE 143C, Digital Control Systems at UCSD). Although not included in this text, BBR does include build instructions for both BeagleMiP and BeagleRover. As a precursor to a formal curriculum written around the EduLine, BeagleBone Robotics is considered to be a major contribution of this thesis. That being said, there is still ample room for augmentation of this work, both from the educational perspective as well as the technical.

From the educational perspective, some specific ways to augment BeagleRover as an educational platform are adding a graphical programming option, incorporating Ackermann steering geometry and complementary filtering into the theoretical portion of BBR, designing complete lesson plans for use by instructors, adding more examples of hands on experiments achievable at the high school level. From a high level view, the EduLine has the potential to reach a wide audience by offering a low enough barrier to entry to engage high school aged students while offering a direct path to college level curricula. A high school level course, or Volume 1 of BeagleBone Robotics, that has a direct counterpart at a well-respected university such as UCSD, or Volume 2 of BeagleBone Robotics, is potentially very powerful. In order to impact the largest number of students possible, BeagleBone Robotics should be brought into the classroom rather than focusing exclusively on extracurricular programs. What's more, the most ambitious goal is to adapt Volume 1 for use in public high schools by appealing to state standards for science and math education. To adapt the EduLine material to state standards for use in public school systems is a very in depth project that will take years to complete. But if done successfully, it could potentially impact the downward trend in students pursuing STEM degrees that was highlighted in chapter one, *especially if it is kept affordable*. Out of the EduLine, BeagleRover is the platform best suited for adaptation to high school level coursework for a number of reasons that have been explained throughout this paper.

From the technical perspective, some work could be done to improve

balancing of the vehicle as it has the tendency to "run off" when not given any user input from the DSM2 radio. A sure way to do this would be to add position control by leveraging frequency separation techniques. Other sensors such as wheel encoders would be required to accomplish this reliably. However, this method is implemented by BeagleMiP and it may be desirable from the educational and commercial product perspective to allow for this difference between the two robots. More compelling than this however, are the potential improvements to the vehicle handling while driving in Normal mode. This could include more advanced versions of Ackermann steering geometry as well as the addition of torque vectoring which refers to spinning the wheels at different speeds through a turn to compensate for differences in distance traveled. Optimizing the handling of BeagleRover through these techniques could be a separate master's thesis project in itself.

Although this paper focused on BeagleRover as an educational product, there are many ways this project can be extended including to applications outside of education. In fact, the ability to customize and extend is a key component of the foundation of the project. One example is inspired by Tactical Electronics' Under Door Camera [57], a wireless camera designed to be slid under a door and then operated from a place of cover, there is a current effort in the Robotics Lab to design a mechanism that can be attached to the Rover and deploy a camera in a similar fashion. As seen in the second image, Tactical Electronics' Under Door Camera is meant to be placed in the desired location by a human hand, at best by an attached pole. If the same

result was to be achieved by a remotely controlled vehicle that can maneuver in very tight spaces, leveraging the different drive modes such as spin mode to turn around without lateral motion in any direction, the operator would benefit from maintaining a greater distance of cover. The applications to hostage situations and or exploration of a burning building for example are apparent, enhanced by the presence of an IR light source and IR camera to illuminate the inhabitants of a dark room without alerting attention.

A second example of extension beyond educational applications takes the form of an additional theoretical problem. While working on the steering code for BeagleRover, it got stuck in a loop while rotating in place on two wheels, similar to a spinning top. This begs the question, can it spin fast enough to be open loop stable in this orientation? By drawing parallels to the problem of a spinning top, one could attempt to calculate the necessary angular velocity of the vehicle about the vertical axis and therefor the necessary velocity of the motors. The physical limitations of the Rover were tested by writing code to disable balance control while driving on two wheels so that maximum angular velocity could be achieved prior to disabling feedback. The result was instability at high velocities even with the balance control enabled, let alone disabled. The tendency of the vehicle to rotate about one wheel rather than about the vertical axis quickly causes instability at higher speeds, even if a high enough speed for open loop stability was achievable. It is left to future study to move forward with this challenge.

Overall BeagleRover works well and the addition of this robot greatly

enhances the appeal of the EduLine as an educational robotics platform. The combination of BeagleRover and BeagleMiP along with BeagleBone Robotics provides a strong starting point for formal STEM curricula at both the high school and university levels. With future development, the EduLine stands to impact STEM education in a very real and meaningful way.