Self-Balancing Unicycle Final Report

To
Andy Chang, National Instruments
Dr. Robert Throne, Rose-Hulman Institute of Technology

From
Spencer Carver
Kevin Collins (Leader)
Ander Solorzano
Ruffin White-Magner

Report Date
5/13/2013

Executive Summary

The goal of this project is to design and construct a robotic self-balancing unicycle for National Instruments (NI). Using NI hardware, such as the Compact RIO real-time controller, and their LabVIEW software, this project is an effort to demonstrate NI's real-time solutions and their Robotics Toolkit as a viable solution for advanced robotic systems and controls. In addition to the physical device, NI has also asked the team to utilize the Robotics Simulator within LabVIEW to model and program the unicycle, and to debug the robotics development environment to ensure appropriate operation. This robotic prototype is being designed to serve primarily as a demonstration for NI Week 2013, a convention held from August 5th through August 8th in Austin, TX where National Instruments showcases uses for their current and newly revealed products.

The project was divided into three main stages over a period of thirty weeks. The first stage included researching past projects and brainstorming new and modern solutions. The second stage consisted primarily of the construction of the prototype and integration of the electrical and mechanical subsystems. The final stage was deploying the control system, theorized during the first stage, to the real time system for testing and debugging.

The final goal of this project was an ambitious attempt at an autonomous two-dimensional self-balancing robotic unicycle. Although the nonlinearity of inverted pendulums has been well documented and studied, our goal would remain challenging but still realistic. Throughout the project we have encountered obstacles, many unforeseen, such as negotiating and ordering custom engineered equipment from a foreign manufacturer across the globe in a foreign language. We also continuously cooperated with customer support as well as lead Research and Development teams within large corporations such as Micro-Strain and National Instruments, tackling technical issues and discovering system bugs in their own products well outside our own areas of expertise. Despite these continuous challenges, we have been able to demonstrate significant progress in constructing a one-dimensional robotic unicycle.

1. Introduction

1.1 Project Overview

Representing National Instruments as the client for this project, Andy Chang serves as the primary point of contact for status reports and updates on the project. This project in particular is designed to demonstrate how NI products (especially LabVIEW) can be applied to solve challenging controls problems. Within the scope of the project, the final goal was to deliver to the client a physical prototype of a unicycle capable of balancing on its own. The prototype has to correct for its own motions and correct for slight external disturbances.

In order to accomplish this, we divided the time spent on this project into three parts. The first part of this project involved accurately describing the unicycle problem within a simulated environment, specifically the National Instruments Robotics Environment Simulator. Once the simulated model showed promises of achieving steady-state while balancing, the team constructed a physical prototype. The difficulties in getting the prototype to function should have been mitigated by the use of the simulator. Nevertheless, the team found that implementing a real LabVIEW control algorithm on a physical system was much more difficult than simply using the simulator due to the unaccounted assumptions, errors in calculations used to model the plant system accurately, and both systematic and non-systematic errors.

The final product presented to the client resulted in a complete physical unicycle robotic platform with a deployable real-time controller and real-time data acquisition. All main electrical components, sensors, the real-time controller, and a wireless router are securely mounted and installed on the platform. At this stage the robot has the plant system model for a one-dimensional case installed (i.e. forward and backward balancing). To compensate for unwanted responses, the robot also has two attached training wheels to prevent it from falling to a side. The robot can have updated versions of the controller deployed without having it tethered to a computer. A user interface (UI) for tuning, testing, and simulating the controller was also created to facilitate in the debugging and deployment stages of a working real-time controller.

1.2 Technical Background

This project is a robotic application of a two-dimensional inverted pendulum problem and is an application of high-precision, real-time controls on a robotic platform. This problem is far more complex than one-dimensional inverted pendulums and as a result far less work has been done on this variant than the simpler system.

1.2.1 Control Systems Description

In order to construct a proper controller scheme for this project, the team had to identify the inputs and outputs of the system as well as to have an idea of how to couple the states in question. Although several approaches to this type of control systems has already been done [1][2][3][4][6], after research and careful considerations the team decided to create an original and innovative design that would model a human riding a unicycle by applying torque to systems to change the pitch. The team decided to model the plant system using a non-linear systems approach that would then be linearized and discretized by the real-time controller. Using a state-variable feedback approach, the status of the various dynamic states can be tracked and a control effort can be computed to achieve a steady-state behavior [5].

State variable feedback is a commonly used method in modern control systems since it allows the placement of poles anywhere for the system to reach steady-state [5]. After modeling the plant system using Lagrangian equations as discussed in section 1.2.5 to describe the non-linear states of the system, the states, inputs, and outputs of the system would then be parsed and linearized using the LabVIEW control and simulation toolkit.

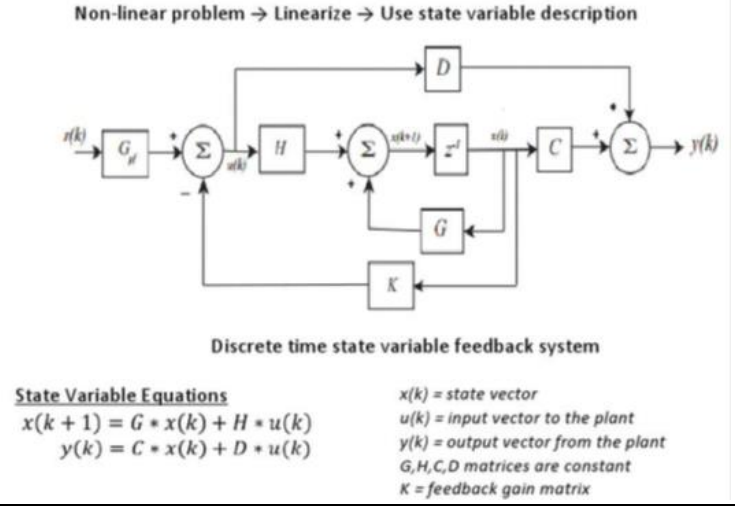

Figure 1. **Flow diagram of a basic discrete-time state variable system.**

In the flow diagram, the feedback gain matrix K is used to place the closed loop poles while the pre-filter G_{pf} is used to reduce steady-state oscillations which may lead to errors [5]. The state vector *x(k)* contains all the states for our system. The states in question are the lateral position of the system, the lateral velocity of the system, the angular position of the system, and the angular velocity of the system.

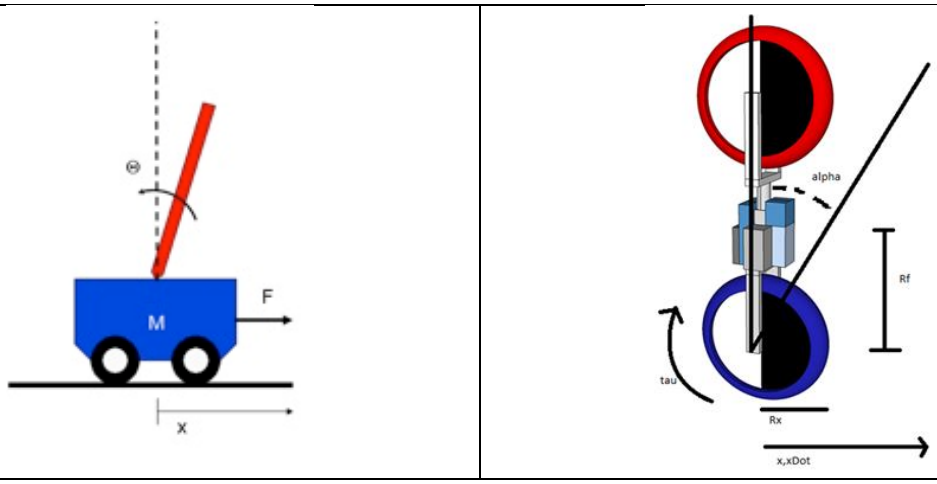

Figure 2. **Comparing the modeling of a linear inverted pendulum cart (left) and the robotic unicycle (right).**

In modern control systems, the steady-state time, the controllability, and number of states are deciding factors when choosing the type of controller for a non-linear system. The equations of motion for a unicycle are highly non-linear and contain couplings between different motions [1]. After linearizing the non-linear system in LabVIEW, the state-space model contains only states and not a non-linear set of state equations. Unfortunately, linearizing the system of equations gets rid of the couplings which reduce the robustness of the system [1]. For the unicycle control system, a Linear Quadratic Regulator (LQR) control and an Ackermann pole placement control were implemented as viable solutions for the controller.

1.2.1.1 Linear Quadratic Regulator (LQR) Controller Theory

The LQR control is an optimality procedure that minimizes the quadratic cost function of the inputs and outputs to return the optimal state feedback vector that stabilizes the system by assigning a weighting mechanism to the states [3]. The LQR algorithm assigns weights on the control effort, not matching the reference signal at the final time, and not matching the reference signal during the transient time [5]. For this type of controller, the system must be linearized and the weights, or penalties, are quadratic [5]. The cost function that minimizes the penalties on the states is:

$$J = \int_0^\infty (x^T Q x + u^T R u) dt$$

Where x is the state vector and u is the input vector, in this case the torque applied to the motors [3]. Finally, Q and R are state vectors that set the relative weights of the various states and the input [3]. Since the unicycle system contains four states as mentioned earlier, Q is then a 4x4 matrix while R is a scalar. The cost function has to be set so that the weights have a greater magnitude towards the angular states than the lateral translation of the system.

1.2.1.2 Ackermann Pole Placement Controller Theory

Ackermann pole placement is an approach that allows us to identify the controllability matrix assuming that the system is controllable [5]. By finding the controllability matrix, the state-feedback gain matrix can then be found which will enable the system to reach a controlled steady-state response [6].

Starting from a discrete-time state variable description of a plant in a state variable model, some procedures from the Cayley-Hamilton theorem are taken to then form a controllability matrix *K* [5].

$$x(k+1) = G * x(k) + H * u(k)$$

For a system with a state variable feedback where the input u(k) is a scalar, the input then becomes scaled. A new description of the plant can then be derived from the new input [6].

$$u(k) = -K * x(k)$$

$$x(k+1) = G * x(k) + H\big(-K * x(k)\big) = (G - HK) * x(k) = \tilde{G} * x(k)$$

Now utilizing the same approaches as the Cayley-Hamilton theorem and rearranging terms the $\Delta(\tilde{G})$ then becomes the following:

$$\Delta(\tilde{G}) = \Delta(G) - [H \quad GH \quad G^2H] \cdot \begin{bmatrix} bK + aK\tilde{G} + K\tilde{G}^2 \\ aK + K\tilde{G} \\ K \end{bmatrix}$$

The controllability matrix from the above equation is then given as

$$[H \quad GH \quad G^2H]$$

From this, the equations can then be rearranged to find *K* and compute the controllability matrix from the known *G* and *H* state matrices.

1.2.2 Inertial Measurement Unit (IMU)

The Inertial Measurement Unit (IMU) is a sensor that can calculate the bearing, orientation, heading, tilt, roll, yaw, pitch, linear acceleration, angular rate and/or can provide some localization feedback via Kalman filter. This internal Kalman filter helps reduce sensor noise and measurement error by aggregating measurement data from the gyroscopic, accelerometer, and magnetometer sensor. With this extended Kalman filter, angular orientation and velocities can be measured with greater confidence as elements within our state feedback control. In addition, the devices and filters are also used in aircraft, spacecraft, watercraft, guided missiles, UAVs, and other autonomous or remotely operated vehicles where precise orientation and localized feedback is mission-critical [7].

1.2.3 Compact Reconfigurable Instrument Output (cRIO)

The cRIO from National Instruments is at the heart of the robotic unicycle as it is the real-time implementation of our nonlinear control scheme. The cRIO possesses many similarities to that of other modular computational units such as microcontrollers, embedded systems, or mobile computers. The cRIO however is somewhat of a compromise between the spectrums of systems, as it possesses both the real-time performance and reliability of an embedded architecture as well as the flexible high level interface of a modern computer. This allows for rapid code development and deployment. An appropriate market they are suitable for is the development of automated manufacturing and industrial data acquisition applications.

The cRIO is networked and program over an Ethernet interface via a mobile Wi-Fi bridge powered through the cRIO's USB port. This makes programming and monitoring the robotic unicycle's real-time status convenient, allowing the balancing structure to be untethered and free from external forces such as lengthy network cabling.

The cRIO also extends an industrial serial interface protocol called RS232. This serial interface is used to communicate with both the IMU and the brushless motor controller. One particular note is that although both the IMU and the controller use the same serial protocol, the connections with the cRIO are not over the same serial interface. By definition RS232 is a serial interface, not a parallel one, so additional RS232 devices require separate serial busses for full-duplex communication so that both devices may communicate with each other simultaneously as well as asynchronously. An additional FPGA serial interface is used because there is only a single internal serial interface on the cRIO. Also, since the IMU data streaming and binary packet protocol is so complex, using pre-existing function libraries and callbacks compatible with the cRIO internal interface is a great advantage over redeveloping the same code for FPGA integration.

1.2.4 Brushless Motor Controller

Using a brushless motor controller from RoboteQ allows the cRIO to actuate the hub-motors to displace the frame and maintain balance. The RoboteQ brushless motor controller has two independent channels to control each motor, with each channel deriving from three phase interconnects that power each brushless motor. Because brushless motors are more sensitive to the way the current is induced in each phase, feedback is required to allow the motor controller to sense the pole alignment between the rotor and stator. This is done by embedding three hall-effect sensors within the assembly to continuously monitor the pole placement.

In addition to rotary feedback, the motor controller is also capable of being configured to interpret torque commands thereby simplifying the necessity to derive the proportional amount of current given the current position and speed of the motor necessary to induce an intended torque oneself. If this was the mission-critical device however, proper modeling of both the inertial frame as well as the complex properties of rotary actuators would be better suited than making this simplified approach.

1.2.5 Lagrangian Mathematical Modeling

The system was modeled mathematically using a Lagrangian approach. Lagrangian equations are very useful because they allow us to calculate the equations of motion without considering constraint forces. Given kinetic and potential energies of the system, the equations of motion can be calculated using generalized coordinates as opposed to traditional Newtonian mechanics that would have relied on summing vector constraint forces. The full details of this approach can be found in the document *System Description and Modeling*.

2. Accomplishments

2.1 System Description

To begin, we developed the robot within the LabVIEW Robotics Environment Simulator. This resulted in a basic model that we could test possible control algorithms on. The team was able to write a simple PID controller that worked very effectively within the environment simulator.

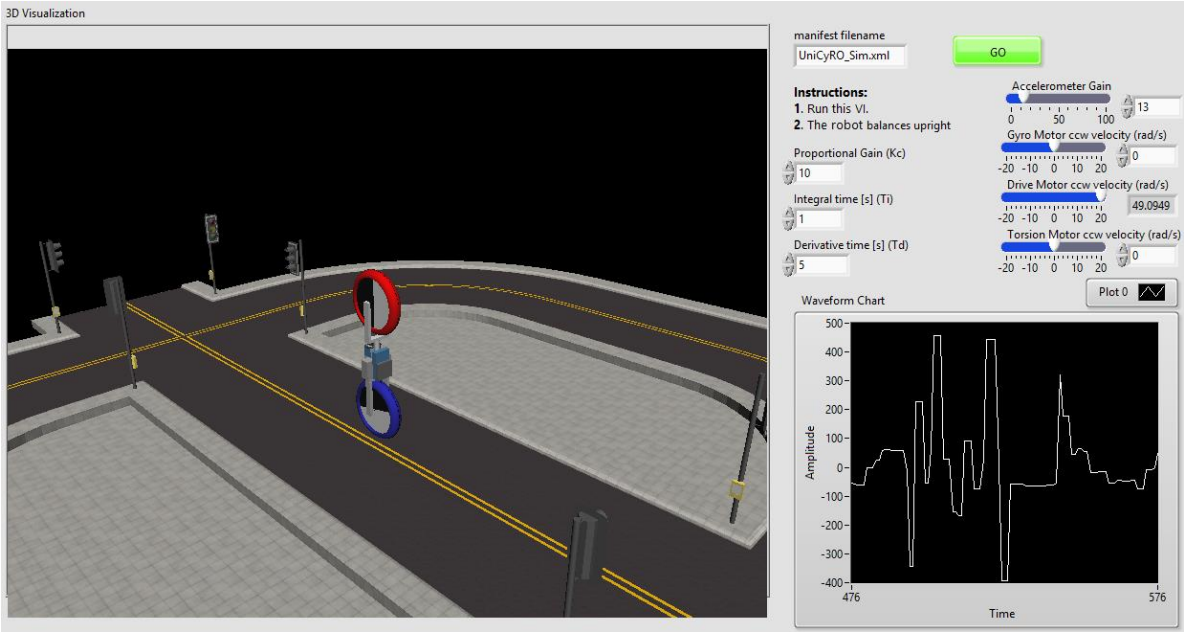

Figure 3. **A screenshot of our model in the robotics environment simulator.**

We then began constructing a physical prototype. First, the team had several important decisions to make regarding the controller that we wanted to use for the project. All decision matrices regarding this choice are found in the project binder.

Our client being National Instruments, the obvious choice is their compactRIO line of real-time controllers. We originally selected the cRIO 9082 and the cRIO 9025 as our two top selections, but included the 9024 at the request of our client, as it was practically identical to the 9025 model, but cost $500 less and was more optimized for a 'laboratory environment' such as the senior design workspace.

While a quick glance at our original comparison ruled out the 9082, it was more of a toss-up between the 9025 and 9024. As such, we constructed a weighting matrix in order to assign numeric values to different features we felt relevant. It should be noted that these weight ranges were constructed by evenly distributing the range of values available to us, not weighting actual controller values (for example, no cRIO has a CPU speed of less than 500MHz, but this would be a 3 on our weights due to the even distribution).

The 9024 turned out to slightly edge out the 9025, both in cost and in performance, while consuming slightly more power (a feature which, according to our client, is a non-issue due to the time the robot will be active).

Once we had a controller, many of the components purchased and added to the robot revolved around the requirements for the controller. When all of the components required for the project were determined, we went to work designing a frame to support them. We decided to go with standard weight T-slotted aluminum extrusion sold by 80-20. The main advantage of this was that the T-slots gave us a very modular approach to adding components to the frame. If we went with a solid frame, we would have to finalize the placement of each component before we made permanent modifications to the frame to attach them (machining, tapping). With the T-slotted extrusion, we could add, remove, or adjust the location of

components as we saw fit without any modification to the frame. This was ideal for the prototype nature of the project. Our initial model of system is shown below.

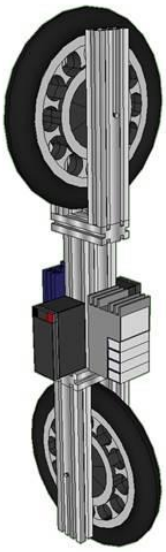

Figure 4. **Final Design of the Physical Prototype**

After the layout of the robot was determined, we split into two main tasks: building the frame of the robot, and wiring all of the components together.

On the mechanical side, the pieces for the frame were ordered with machining already done to make sure that the frame would be compatible with other 80-20 parts. We then did some machining to fit our own needs, and assembled the frame. A mechanical drawing of the frame is shown below.

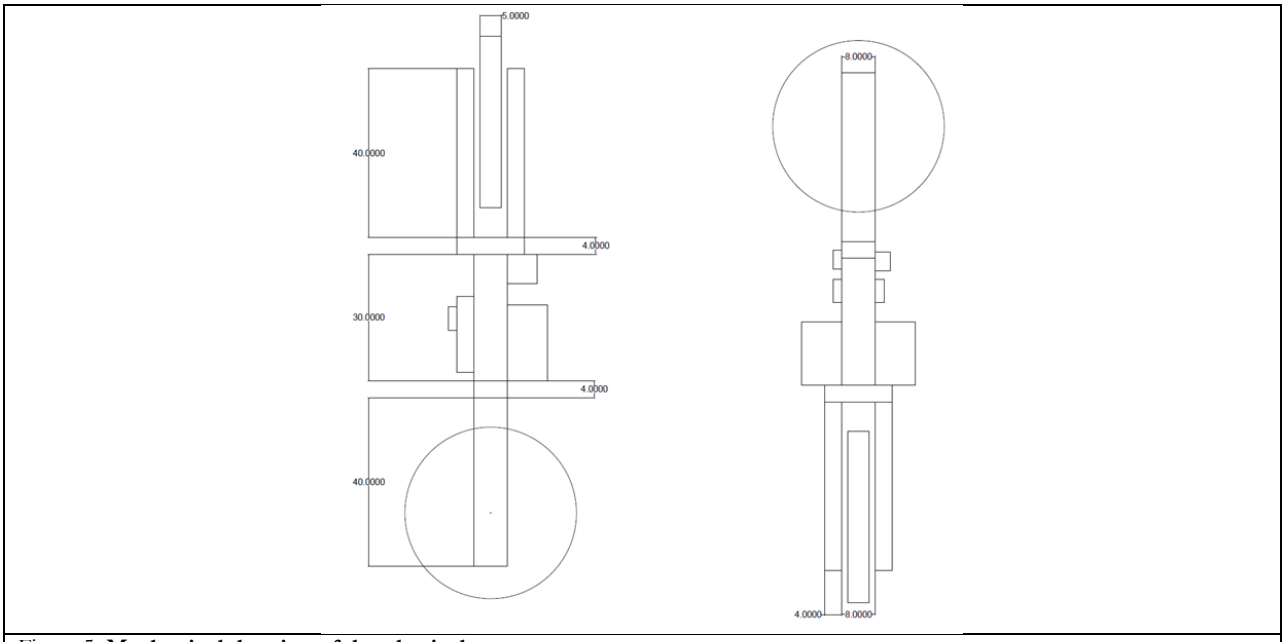

Figure 5. **Mechanical drawing of the physical prototype**

On the electrical side, the basic components of the robot were wired together. We added some extra features, including a kill switch/kill cord to cut the power to the wheels immediately in case the controller became unstable. The circuit diagram is shown below:

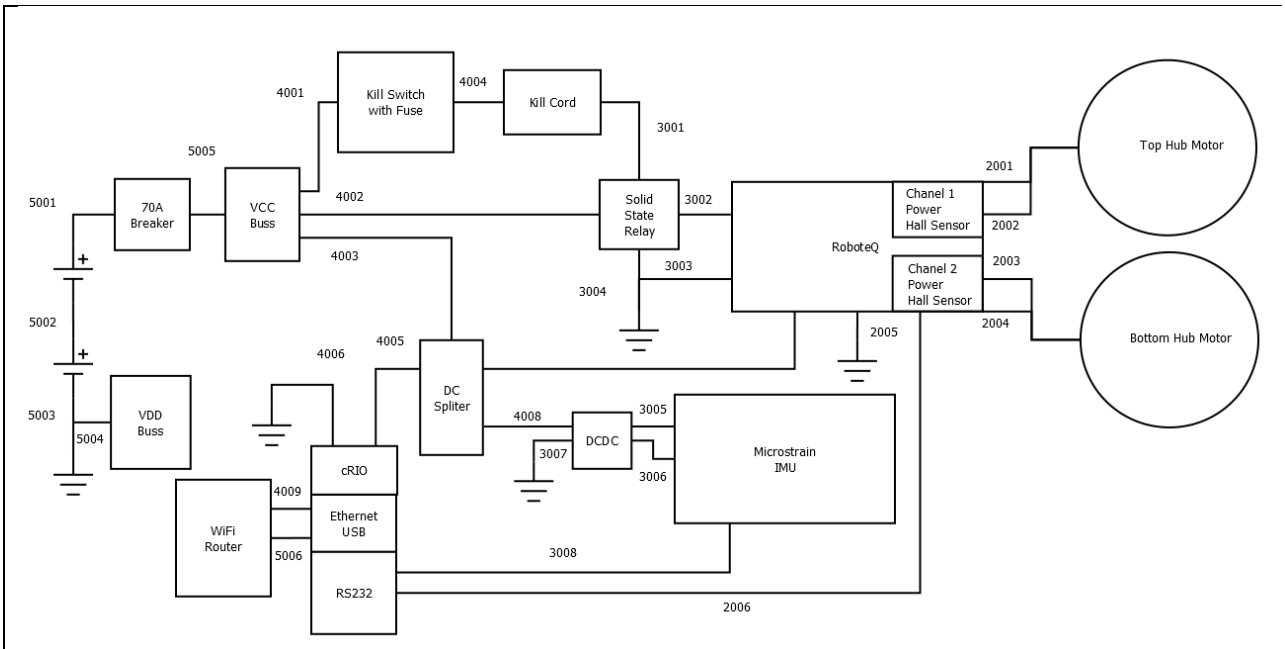

Figure 6. **Electrical wiring diagram for the physical prototype**

Once the layout of the robot was finalized, a mathematical model of the system was derived. A preliminary model had been created earlier, but now all of the parameters to reflect the actual robot were known. A detailed analysis of the creation of the mathematical model can be found in the document *System Modeling and Identification*.

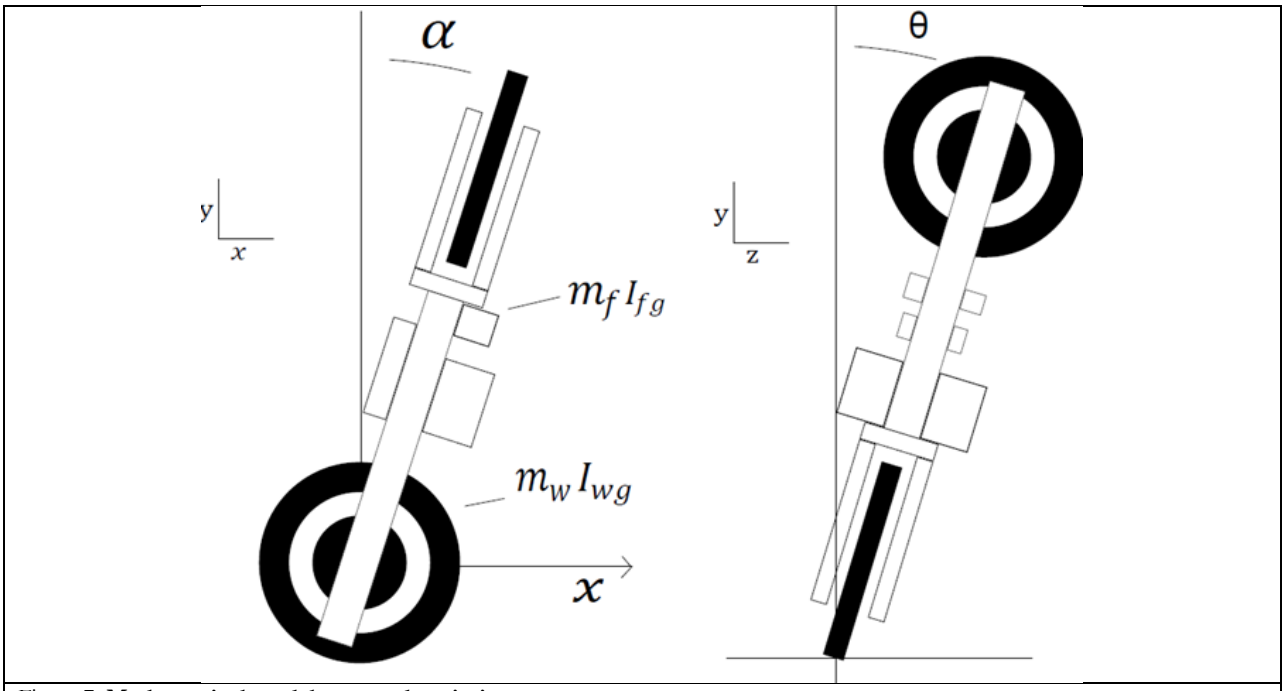

Figure 7. **Mathematical model system description**

With this mathematical model created, the control algorithm could be tailored to our model. For this, we determined that the following project diagram included everything that we needed to create a robot that would be capable of fulfilling the requirements laid out in our PDS. In doing so, we were able to determine 3 key processes that would need to be implemented that were done for us in the simulator: the reading of the robot's state via an Inertial Measurement Unit (IMU), the processing of these values, and finally plugging these values into the model and sending the resulting directions to the wheels of the robot.

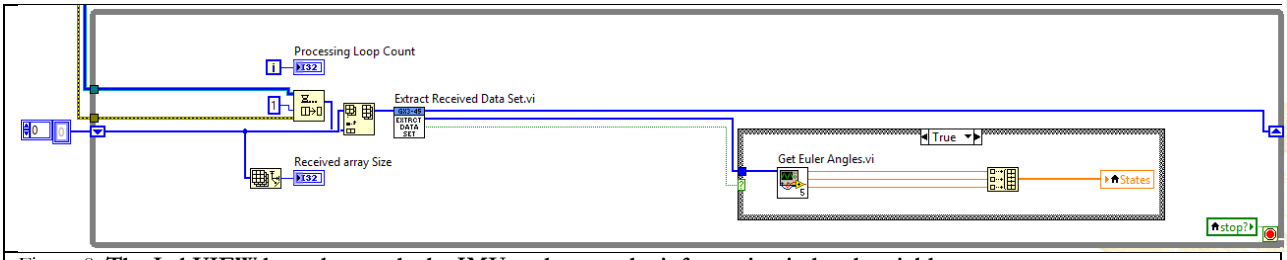

Figure 8. **The LabVIEW loop that reads the IMU and stores the information in local variables.**

The first of our three subsections handles acquiring the data from the IMU and then processes it into the state local variable and the queue of previous IMU readings. This is a while loop structure in LabVIEW, and as such the state local variable always has the latest information for use in the other control loops.

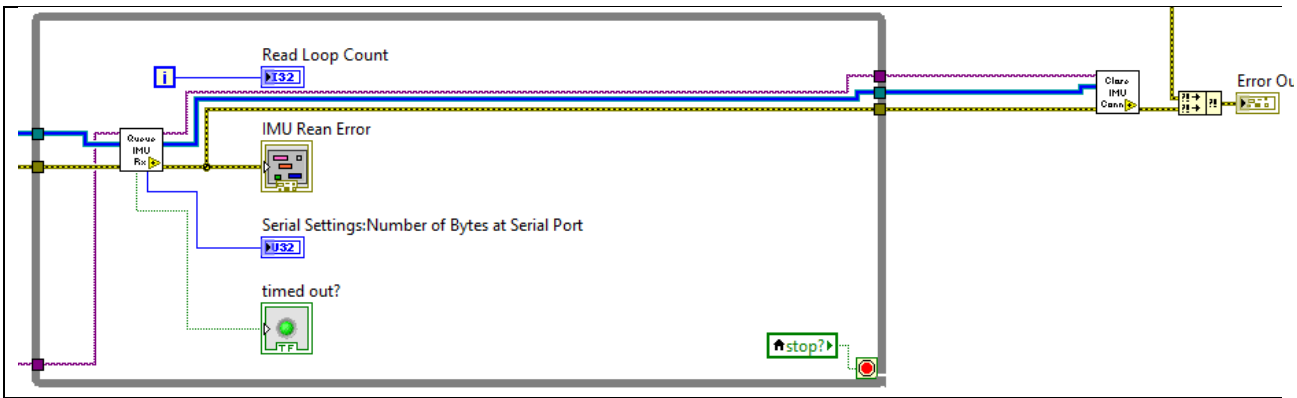

Figure 9. **This LabVIEW code determines if the IMU has timed out.**

The second of our control loops passes the queue of IMU readings and the current error state along to a sub-VI that looks to see if our IMU has timed out. If it has, the indicator is enabled (visible to the user on the LabVIEW front panel), and the error line is also enabled, so that the robot does not attempt to make adjustments off of faulty data.

While LabVIEW natively supports many different controllers and sensors, the Microstrain IMU we used for this project is currently not one of them. We were able to work around this through communication with Adam Amos, a representative of Rescue Robotics, who uses the same model IMU for an outback rescue UAV competition in Australia. Without their assistance, implementing Microstrain's newly released and unsupported MIP binary packing from scratch would have been a significant setback [9].

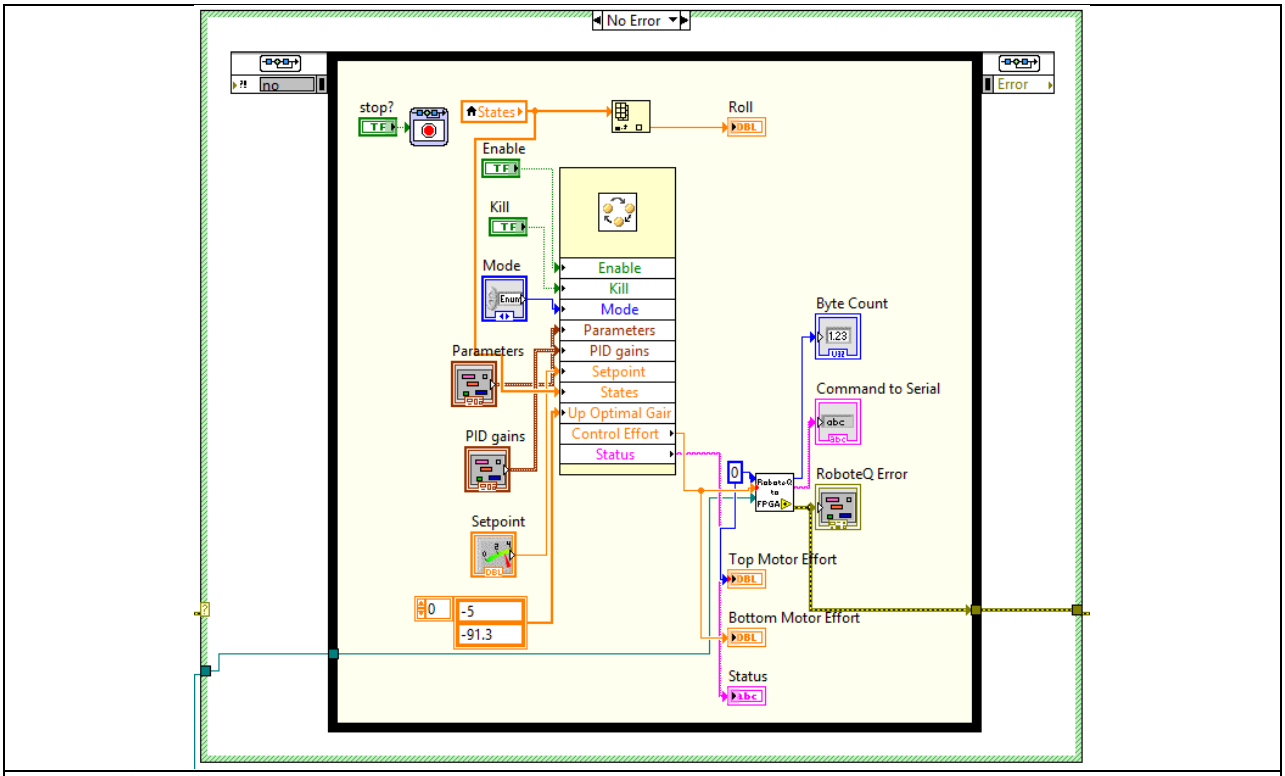

Figure 10. This LabVIEW code passes state information to our control model and then sends a control effort to the motors.

The last of our 3 sections houses the control model (the large rectangle in the center of the image), and the VI that sends the motor controls off to the wheels, as well as a few indicators to express the values on the front panel to an operating user. In the event the IMU has timed out, or another error has occurred, this entire section will not run and instead simply wait for a user to clear the error.

2.2 Comprehensive Testing Plan

There were four primary tests planned for the development of our unicycle; one during the design of the LabVIEW code in the Robotics Environment Simulator, and three physical balance tests. The simulated test took place at the beginning of the winter term of the project, when we had finalized our design. We calculated the moment of inertia for all of our components based on listed values and added them to the Environment Simulator. The test then consisted of taking our model and loading it into the Environment Simulator and adjusting parameters to produce a steady balancing state. It was noted during these trials what kind of control bandwidth and loop speeds would be required to execute a similar controller on the real-time platform.

The first of our physical tests planned was scheduled for the first few weeks of the Spring Term. This test was a one-dimensional balance test, designed to allow us to tune several of the control model's parameters while the unicycle was wearing training wheels, and then later tune the side-to-side motion of the robot. Unfortunately, due to problems with the controls of the robot, this was never realized, and instead we tuned a PID controller using this test. While originally slotted for a single week's worth of trials, this test wound up taking the remaining few weeks of the spring term. Despite the additional amount of time required, we were able to determine several key features about our robot that will help once the remaining controls have been completed, such as a necessary dead band to move the robot when small changes are required.

Our third test was also an isolated one-dimensional balance test, this time with the bottom fixed and the robot able to move from side-to-side. This balance test was originally planned for week 6, but was never reached due to the control issues hindering the previous test from being completed.

The final balance test planned was the first full two-dimensional test, with motion in both the drive wheel and the top wheel. Upon completion of this test, the robot would have met all requirements originally put forth by our client. Again however, this test was not completed due to issues with our controller.

2.3 Data Verification of PDS

TODO

3. Project Planning

In order to actually complete this project within the three quarter time limit of the capstone design course, the team decided (with our client's recommendation) to pursue an ambitious modeling plan for the fall and beginning of winter quarters, and to use the remainder of winter quarter to get the robot's simulation

functioning in the LabVIEW Robotics Environment Simulator. The spring term was primarily set aside to work on the physical prototype of the robot.

3.1 Plan of Approach and Timeline

Our initial approach to the project is outlined as follows:

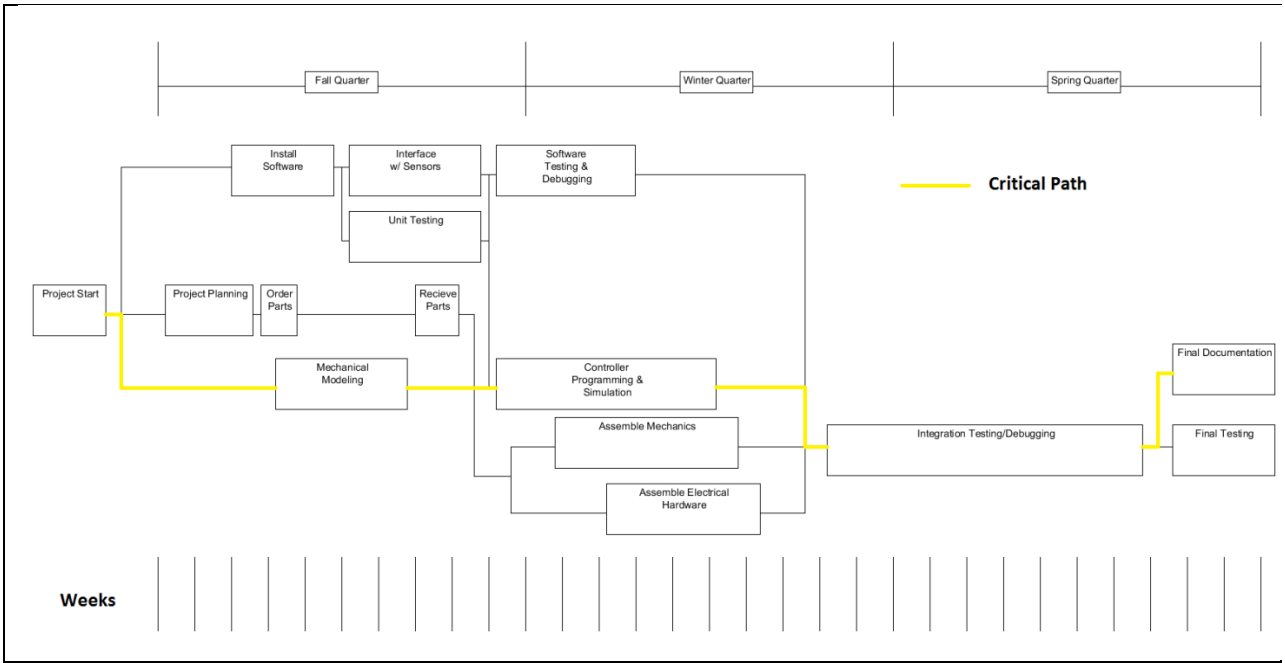

Figure 11. **Initial timeline of the project**

As a team, the entire project was broken down into several stages for each of the quarter. They are listed as follows:

Fall Quarter

- Model Design
- Model Controller
- Order Parts

Winter Quarter

- Physical Construction
- Model and Controls Adjustment
- Robot Programming

Spring Quarter

- Tuning and Debugging

During the fall, we wanted to design a very detailed model of our planned physical robot and import it into the LabVIEW Robotics Environment Simulator. We would then create a control algorithm in LabVIEW to control our model. While we were working on this, we would order parts for the physical robot to be built in the winter.

During the winter, we would build our robot. Then, we would adjust our model within the Environment Simulator to more closely represent the actual robot and adjust our control algorithm so that it works for the new model. Once we were confident in our algorithm, we would program it to our physical robot.

The entire spring quarter was initially reserved for adjusting our control algorithm and debugging the system. We anticipated that it will take quite a bit of work to tune the controller to a point that it would achieve acceptable results.

This plan was good in theory, but there were several unforeseen challenges that put us behind schedule.

Our first obstacle was in ordering the hub motors. The hub motors were a crucial part for testing that would have been great to have during the entire fall quarter to become familiar with how they worked. Since they were being ordered from China, it was a complicated process to order just two motors, and the process took a lot longer than anticipated due to communication and transaction barriers when using a foreign manufacturer and supplier.

Another obstacle we encountered was using the FPGA on the cRIO, as it did not seem to work at all right out of the box. While during our original evaluation of the cRIOs we valued serial ports, the model we selected and our client recommended only natively contained one. Since we needed two total (one for the IMU and one for the motor controllers), our second serial port was forced to be done through the FPGA expansion ports, via a NI 9070 expansion. Unfortunately for us, LabVIEW 2012 shipped with a bug when using the FPGA features of the cRIO, and we were unaware of this until NI support responded to our continuous inquiry. Once the provided hotfix had been applied, the code suddenly worked as expected. Additionally, the FPGA VIs seemed overly complicated for simply relaying serial commands, but several referenced examples aided the team in overcoming this obstacle.

On the controls side, we originally based our model off of a paper that we later found out included some simplifications that were not applicable to our project. Rather than risk a similar problem by finding a new model, we derived a new mathematical model (described in *System Modeling and Identification*) to accurately describe our system. Deriving another model for the system and then adapting the controller to the new model took some time that we were not originally anticipating, and set us back a few weeks.

3.2 Division of Labor

Control Systems - Controls systems consists of writing a control algorithm within LabVIEW to control our simulated robot and eventually the physical robot. Ander is our primary controls engineer since he has had the most experience with controls schema. Ander was tasked with developing, implementing, and deploying the plant model description from Kevin and imported it to a LabVIEW scenario. Additionally, Ander was also tasked with creating the User's Manual description for tuning the controller using the UI in LabVIEW. Since this is a control-driven project, additional references and descriptions of the control systems can be found in the *Control Systems Description.* Spencer also helped develop our control algorithm and is in charge of making sure that our controller output interfaces well with our mechanical and analytical models. Finally,

Ruffin then complied this into a flexible finite state machine for use in a demonstration environment along with software limits for personnel and equipment safety.

Robotics Environment Simulator – Ruffin was in charge of creating a detailed model within the Robotics Environment Simulator and creating a PID controller within LabVIEW for this model.

Parts Purchasing - Spencer and Ruffin were involved in purchasing components. They have dealt with lots of sensors through their experience on the robotics team and know where to look for components.

System Modeling - Kevin was in charge of creating a mathematical model of the system. The original controls algorithm used a mathematical model that was discovered during our initial research. However, it was later discovered that this model was not sufficient for our purposes, so a new model was derived. The details of this model can be found in the attachment *System Modeling and Identification*.

Frame Construction - Kevin designed and built the frame of the robot. He machined the extrusion to meet the requirements of the project and created mechanical drawings of the resulting frame. He also created mechanical drawings of placement of many of the major components, but some of these placements have shifted to accommodate new components.

Electrical Wiring Assembly- Ruffin designed and wired the electrical circuit and safety measures for handler operation. This included circuit specification for motors, controllers, batteries, regulators, relays for operation performance, power requirements, as well as compatibility and replaceability. He also created the schematic documentation and a bill of materials.

3.3 Budget and Key Expenditures

Since National Instruments is our client for this project, we used NI hardware and software wherever possible. Our main expense for this project was the LabVIEW Robotics Suite 2012, which is nearly $15,000 on its own. Our second largest expense was the cRIO 9024 real time controller, which cost just under $4,000. National Instruments provided us with both the robotics suite and the cRIO, thus our actual expense was drastically reduced. However, if someone wanted to replicate our project, they would need to take these expenses into account.

- Additional expenses include our Inertial Measurement Unit (IMU) that we were able to get a student discount on but it was still $2,660. This IMU is necessary due to its high level of precision in order to maintain a large control bandwidth. For this controls system, this is important because the faster the platform can collect data about its imbalance, the faster it can correct and maintain the balanced state with minimal output.

- Costing $625, the RoboteQ HBL2350 dual-channel motor controller with integrated Hall Effect sensors to provide feedback, served as another major component for our unicycle since it controlled each of the hub-motors by applying the necessary torque to stabilize the system.

- To provide locomotion and system stability, two brushless Hall Effect hub motors were purchased from Alibaba Electronics at approximately $100 per motor. The motors were ordered and customized directly from a manufacturer in China.

- Most of the hardware for the frame was purchased from 80-20, a popular manufacturer of T-slotted aluminum extrusion and accessories. The material was expensive, around $300 altogether, but the result is a very modular frame that can easily be adjusted and accommodate new parts very easily.

The remaining components purchased are summarized in the document *Bill of Materials*.

4. Manufacturability and Safety

An important part of this project was to create a robot that could be easily reproduced if another team wanted to try to further our progress. Overall, the design requires very little custom machining. In fact, the entire robot could be constructed from off-the-shelf components other than our method of attaching the wheels to frame, and even this requires very little machining. However, if this robot were to be mass produced, we would recommend that the layout of components be finalized and a frame be created without using T-slotted aluminum extrusion. Although this material is well suited to prototyping, it is not the most permanent way of constructing a frame, and it is one of the more expensive options.

Safety was one of our primary concerns when designing the system, which is evident in our implementation of several safety devices. The most prominent safety device is the addition of a large, red kill switch on the side of the robot. This kill switch, once depressed, will cut all power to the motors, stopping the robot in its tracks. The kill switch is also key activated, requiring a key to be inserted before the robot can be reset. This helps prevent unauthorized use that could result in injury. In addition to the kill switch, we have implemented a kill cord, a rope that, when pulled, will act just as the kill switch. This way, the user can maintain a safe distance from the robot, but still be able to kill power at a moment's notice.

Another issue is the safety rig of the robot. There are a lot of fragile components mounted to the robot. Considerable damage would be done to the system components and electrical connections if the robot falls uncontrollably to the ground. Due to its heavy yet needed weight to add sufficient inertia and high output speeds, considerable safety precautions such as a safety rig, should be used on the robot. Unfortunately, the safety rig that we constructed is too heavy, and has too high of a moment of inertia for us to use without having a noticeable effect on the overall moment of inertia of the robot. Furthermore, since we kept the testing to a one-dimensional case, the team decided that controller testing and tuning runs would be performed without the use of the safety rig. Nevertheless, the test runs conducted always involved a team member paying high attention to the behavioral response of the robot in order to catch it. In order to remove the human element as a safety barrier, an emergency mode was implemented in the software so if the robot ever went past 10 degrees in either direction from its steady-state configuration (i.e. 0 degree being upright), the power to the motors would be cut-off to prevent uncontrolled operation and prevent unwanted damage.

Even though the testing runs of the physical platform were conducted without the use of a mounted rig, it is highly recommended to design and install a rig with a lower moment of inertia so that all testing can be conducted with minimal alterations to the system model. However, the safety rig would not only serve a protective barrier to prevent damage to the sensitive and expensive components attached to the robot but also remove the human element as a safety mechanism.

5. Recommendations

One of the main problems with our model is the inaccuracies of how we calculated the moment of inertia of each element. In our mathematical model, all of the elements were modeled as a rectangular prism and the wheels were modeled as solid, uniform cylinders. Although the rectangular prisms are most likely 'close estimates' for our purposes, the moment of inertia of the wheel is a very critical parameter and we think that such a solid is an oversimplification.

Depending on the dynamic moment of inertia, the control algorithm sets control efforts for the wheels to counteract the moment of inertia of the entire system. Thus, if there is one component of the system that needs to be modeled as accurately as possible, it is the wheel and hub-motor. Unfortunately, this is our least accurately modeled component. Given enough time to determine the moment of inertia of the wheel experimentally instead of mathematically, could result in a more accurate model description, thus more accurate and stable controller could be implemented and deployed.

A subtle detail unforeseen when ordering the solid state relays is that it is best to use a trigger voltage that is well below the nominal voltage of your supply, rather than the exact rated voltage of the supply. This is especially true when using a mobile supply such as sealed lead-acid batteries that have non-ideal VI characteristics over internal capacity. Currently, to prevent the SSR from opening from a low signal voltage, we use an additional 9V battery to supply the voltage step up to the signal reference from the kill switches. However, due to the leakage current into the SSSR signal in, the 9V battery source must be replaced or recharged after a few days of operation. This adds an unnecessary complexity to the maintenance and operation of the robot. Thus it is suggested that should the SSR ever need to be replaced, a different model rated for the same current but lower trigger voltage (say 16V to allow for moderate discharge and usage time using the current 24V system) for the switching signal should be selected.

6. Conclusions

Overall, we are very pleased with our accomplishments this year. We will present National Instruments with a fully assembled, mobile robot that is capable of being programmed wirelessly from within LabVIEW. Although the robot does not balance in both directions as specified in the original PDS, we have shown that a control algorithm can be implemented on the robot, and that with some tuning, even a simple PID controller can be effective at balancing the robot.

We will also present National Instruments with complete documentation over our design process, including derivation of our mathematical model of the system, as well as development of our control algorithm. Both of these documents should greatly reduce the amount of time that it would take for another team to get up to speed should they wish to continue the project.

In addition, we have created a model of the robot within the Robotics Environment Simulator. This model, able to be controlled by our control algorithm, shows the effectiveness of creating an initial, virtual model within LabVIEW before moving on to a physical prototype. This knowledge can now be applied to other NI projects.

Although we came short of our original goals, what we did accomplish was of very high quality. We came up with several innovative solutions, including the wireless router connected to the cRIO, which we think that National Instruments will be very pleased with, and hope that the project can continue on with another team to finish using the foundation that we have established.

7. References

[1] Kappeler, Fabian. *Unicycle Robot.* June 28, 2007.

[2] A. Kadis, D. Caldecott, A. Edwards, M. Jerbic, R. Madigan, M. Haynes, B. Cazoolato, Z. Prime, *Modeling, Simulation, and Control of an Electric Unicycle*, University of Adelaide, Australia.

[3] D'Souza-Mathew, Neil, *Balancing of a Robotic Unicycle*, 2008

[4] S. Majima, T. Kasai, *A Controller for Changing the Yaw Direction of an Undergraduate Unicycle Robot,* University of Tsukaba, Japan.

[5] R. Throne, *Discrete-Time Control Systems*, Rose-Hulman Institute of Technology, Chapter 6, 7, and 11.

[6] *Generating a Control Algorithm in LabVIEW using State Variable Description,* Team G Research Memo, Rose-Hulman Institute of Technology

[7] *Inertial Measurement Unit (IMU) Sensor,* Team G Research Memo, Rose-Hulman Institute of Technology

[8] "NovAtel Integrated GPS/IMU", AutonomouStuff, October 4th,2012. [Online]. Available: http://www.autonomoustuff.com/novatel-integrated-gpsimu.html

[9] Rescue Robotics, *A UAV team competing in the Australian Outback Rescue challenge* Available: http://rescuerobotics.com.au/