Magic Plank

Brian Jacobs, Kenneth Santiago Jr, Stephen C Fraser II

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE, UNIVERSITY OF CENTRAL FLORIDA, ORLANDO, FLORIDA, 32816-2450

***Abstract* — The Segway Personal Transportation vehicle, a self balancing, single user, transportation vehicle, was originally conceptually conceived by Dean Kamen. Here, Dean Kamen addressed the classic control problem, known as the inverted pendulum problem; he approached this problem with redundant array of tilt and gyroscopic sensors that measured angles of incline and acceleration/speed. In addition to proprietary software developed, a self balancing personal transportation vehicle.**

***Index Terms* — Self Balancing Platform, Personal Transportation Vehicle, Embedded Software Design, Bluetooth Communications, Wireless Steering, Sabertooth, Atmel.**

I. INTRODUCTION

The Segway PT, in itself a marvel of modern engineering that has seen many applications in the civilian and military fields. Segways allow users to agile use and navigate routes not normally taken. Segways are very popular personal transportation vehicle used in many different tours, in many different countries around the world. In addition, the Segway is used by private security to patrol large areas quickly while maintain a low profile, which in turn keeps the security personnel safe from potential threats. Furthermore, Segway has coupled with the military and various universities around the country to be developed into a military platform to be used in the field.in length to the Summary, many contributors opt to prepare their Summary in the format required for the Digest. This template contains the instructions for the proper preparation of such a document.

Approaching the inverted pendulum problem using an Atmel ATMega328p as our main microcontroller allows for an extremely customizable, user – friendly interface that allows easy access to various data streams for each aspect of the platform's balancer. A skilled engineer will be able view how the platform reacts to its surrounding environment; therefore, understanding how to adjust the software of the balancer to become more compatible with the platforms current environment. Moreover, the Saberthooth 2x12 motor controller allows an engineer to adjust the control input for the platform. Currently the platform utilizes the Atmel ATMega328p to process data from a Bluetooth module and an Inertial Measurement Unit (IMU) and outputs processed commands to the motor controller.

The primary motivation for this senior design project comes from the intended nature of engineering and implementing a unique, challenging, yet feasible and rewarding senior design project. Moreover, this project is the first of its kind here at UCF, and hopefully paves the way for future projects addressing the inverted pendulum problem.

II. THE MAGIC PLANK

The Magic Plank will share the same principles with that of the Segway in the sense that the Magic Plank will be addressing the inverted pendulum problem, be a platform that will be balanced of two wheels, and will have a custom tailored algorithm that will serve as the balancer and therefore maintain the equilibrium of the platform.

The Magic Plank addresses balancing in a different manner than the current implementation of that of the Segway. The Magic Plank's base platform will be more elongated length wise to resemble more of a skateboard. This forces the wheels to be placed in the center of the platform. Furthermore, the Magic Plank will be driven by chain motors, which are less responsive than the servo motors used on a traditional Segway PT. This slight downfall will hinder the effectiveness of the Magic Plank's ability to maintain equilibrium, therefore providing a greater challenge in the development of the balancing algorithm.

In addition to a different physical layout, the Magic Plank takes an unconventional approach to steering with the use of Bluetooth wireless communication. By taking advantage of the Nintendo Wii Wireless remote's easy and intuitive interface, the Magic plank can be steered remotely. In addition to steering, the wireless interface provides for an easy to use "dead man" switch that will terminate operation of the Magic Plank on demand. Since this is a software operation, the platform can resume functioning at any time with the use of a "revive" command, which is a unique and useful application that is otherwise not easily achievable through conventional hardware means.

The challenge of designing the Magic Plank comes from limited budget, less responsive motors, less data acquisition redundancy, and an added element of wireless steering that is not present in conventional platforms such as the Segway PT.

III. ATMEL ATMEGA328P MICROCONTROLLER

The Atmel Atmega328p microcontroller is the brain behind the operation. The Atmega328p is a highly customizable, user friendly and low power consumption device. The main advantage of the Atmega328p is the Arduino software platform, which allows easy development to tailor the microcontroller to the specific needs of the user. This microcontroller was chosen as a result of identifying the needs of the platform and finding the cheapest controller to suit the needs of the platform. The first piece of hardware, the Bluetooth RN-42 SMD, operates via UART interface, so hardware supported UART is preferred. The second piece of hardware, the Sabertooth motor controller, requires only a single serial transmission, which any microcontroller would be able to do. The third piece of hardware, the IMU3000 digital gyroscope, operates using I^2C Interface, so a microcontroller that has hardware support for this interface is required. The Atmega328p features hardware support for I^2C interface and UART interfaces along with a handful of other general purpose digital pins, which perfectly fits the needs of the Magic Plank. Utilizing these hardware supported interfaces provides a minimal waste of processing power and power consumption. In addition to its hardware needs, the Magic Plank must be able to execute code fast enough to respond rapidly to incoming data. The Atmega328p is capable of running at variable frequency at 1MIPS per MHz. The Magic Plank clocks the 328p at 16MHz and is capable of running the main code loop in around 12ms, which is more than enough for the Magic Plank's 50ms loop time. This leaves roughly 40ms as a timeout interval for receiving information from the Bluetooth module. The Atmega 328p is fast enough to suit the needs of the system, making wireless communication the bottleneck of the system.

IV. BLUETOOTH MODULE

The Magic Plank utilizes the Human Interface Device protocol through the SMD Module RN-42. This module has a small profile as well as a small energy foot print. In addition the SMD Module RN-42 can deliver a three megabit data stream up to twenty meters or sixty feet away. This allows a user to use a wireless steering device off the platform and operate the platform a respectable range. Therefore, a user can assess and analyze the operation of the platform.

The SMD Module RN-42 Human Interface Device does not require any software configuration to pair up with any other Human interface Device protocol device. Any device with the Human Interface Device protocol with automatically detects the SMD Module RN-42 when the module is receiving power.

The Atmega328p read the data stream from the SMD Module RN-42 in the form of hardware UART. Hardware UART allows high resolution baud rate generation; coupled with the data overrun detection, framing error detection, and the noise filtering from the Atmega328p microcontroller can establish a near errorless communication. With Human Interface Device protocol coupled with the advantages of using hardware UART, using the SMD Module RN-42 creates a safe form of communication that has little influence from outside the system.

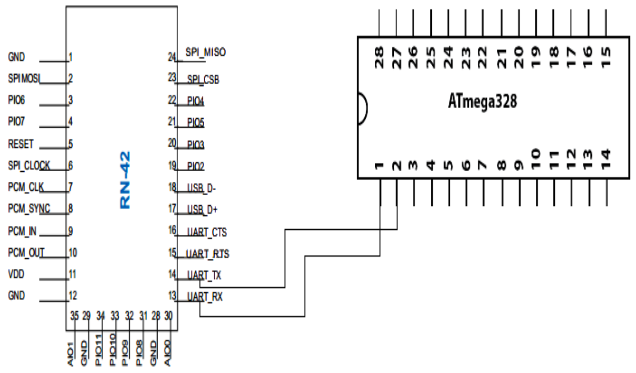

Figure 1: Block diagram illustrating the pin connections between the Atmel Atmega328p and the SMD Module RN-42.

Bluetooth RN - 42 Pin Connection Summary	
Pin	Connection
13	To IO0 Connected to ATMEGA 328P Pin 1
14	To IO1 Connected to ATMEGA 328P Pin 2

Table 1: Table addressing the pin outs from the SMD Module RN-42 to the Atmel Atmega328p.

A. Wiimote Wireless Steering

Wireless steering is achieved through Human Interface Device protocol for Bluetooth communications. This allows a user to pair up a Wiimote wireless controller, to be used as the steering device to control the platform. The Wiimote wireless controller connects to the SMD Module RN-42 using the Broadcom 2042 controller chip. This controller chip utilizes the Human Interface Device protocol. Furthermore, the Wiimote wireless controller utilizes a three-axis linear accelerometer located on the top circuit of the Wiimote. The three-axis linear accelerometer measures accelerations over a range of at least +/- three Gs with ten percent sensitivity.

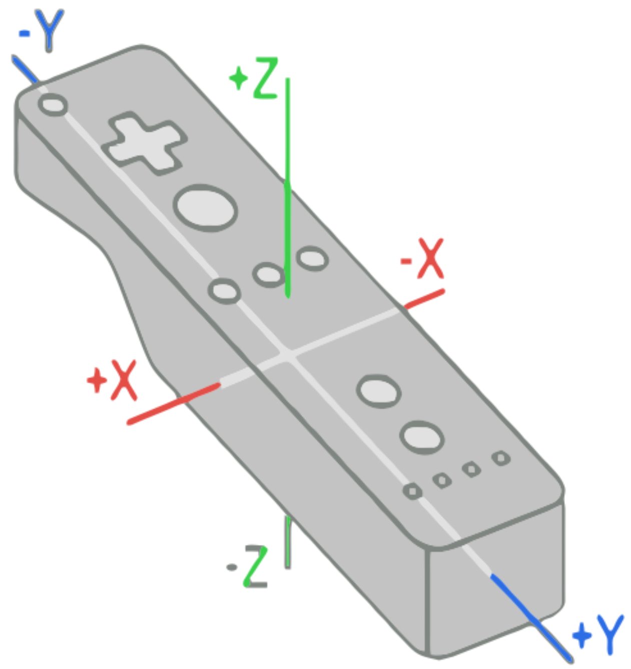

Figure 2: Diagram illustration how the Nintendo Wiimote wireless controller's accelerometer measures data .

The Wiimote wireless controller has many different numbers of data reporting modes. Each of these modes combine certain core data features with data from external peripherals and sends the data to the host device through one of the report IDs determined by the current mode in use. A hexadecimal data packet is sent out every time a value of one of the reported features has changed.

The reporting mode used for the Magic Plank implementation sends a hexadecimal Human Interface Device packet consisting of both button data and accelerometer data. The data sent in the Human Interface Device packet will be in the following formant:

(a1) 31 BB BB XX YY ZZ

(a1) indicates that the example packet is input from the Wiimote wireless controller. 0x31 shows that the data report mode is reporting both button and acceleration data. BB BB are a combination of hexadecimal values that describe which button is currently being pressed on the Wiimote wireless controller. Hexadecimal values XX, YY, and ZZ are unsigned bytes that represent the acceleration in each of the three axis.

Button	**Number (dec)**	**Value (hex)**
Two	1	0x0001
One	2	0x0002
B	3	0x0004
A	4	0x0008
Minus	5	0x0010
? motion ?	6	0x0020
? motion ?	7	0x0040
Home	8	0x0080
Left	9	0x0100
Right	10	0x0200
Down	11	0x0400
Up	12	0x0800
Plus	13	0x1000
? motion ?	14	0x2000
? motion ?	15	0x4000
? Reading Mii ?	16	0x8000

Table 2: A table listing all possible hexadecimal vales that could be in a Human Interface Device packet.

V. IMU 3000 FUSION BOARD

The IMU 3000 Fusion Board is a combination of the IMU 3000 and the ADXL345. This fusion board is a six axis inertial measurement/moment unit that uses the single I^2C bus on the Atmel Atmega328p microcontroller. IMU 3000 is a three axis gyroscope with programmable ranges of +/- 250 to +/- 2000 deg/sec. Having a very wide range of programmable sensitivity allows a user to adjust the IMU 3000 to adjust the platform to be used for any purpose; furthermore, the customizable range maybe altered to be used on a variety of different drivable surfaces.

The ADXL345 is a three axis accelerometer with programmable ranges of +/- 2 to +/- 16 gees. The primary purpose of the IMU 300 Fusion board is to be a redundant sensor to the IMU 3000. Just like the IMU 3000 the programmable sensitivity of the gee range allow the ADXL345 to complement the IMU 3000's programmed sensitivity.

A secondary I2C is located on the IMU 3000 Fusion Board. The secondary I2C acts as a line of communication between the IMU 3000 and the ADXL345. The IMU 3000 acts as the master to ADXL345; as a result two distinct advantages of the setup of master and slave allows less processing power of Atmel Atmega328p. This allows more important data to be processed without having other subroutines fight for data computations using microcontroller resources.

The IMU 3000 is connected to the Atmel Atmega328p microcontroller through pins twenty-seven which connects to SDA(data line) on the IMU 3000 Fusion Board, and to pin twenty-eight to SCL(clock line).

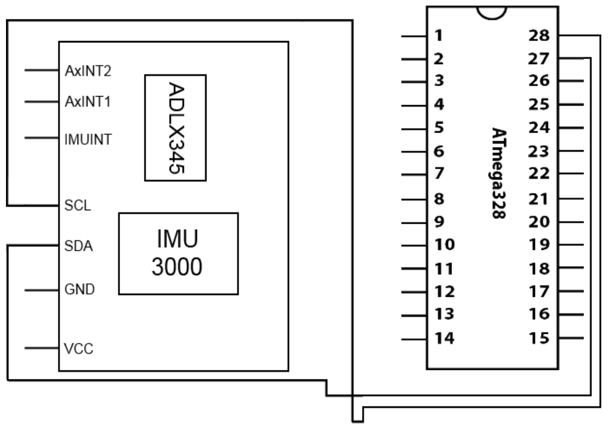

Figure 3: Block diagram illustrating the pin connections between the Atmel Atmega328p and the IMU Fusion Board.

Inertial Measurement/Moment Fusion Board Pin Connection Summary	
Pin	Connection
SDA	Pin twenty-seven on the Atmel Atmega328p
SCL	Pin twenty-eight on the Atmel Atmega328p

Table 3: A table listing the pin out connections between the IMU 3000 Fusion Board and Atmel Atmega328p.

VI. MOTORS AND MOTOR CONTROL

A. SABERTOOTH 2X12 MOTORCONTROLLER

The Sabertooth 2x12 Motor Controller, designed and manufactured by Dimension Engineering, is a very robust motor controller cable of handling heavy payloads. Sabertooth 2x12 is capable of supplying power to two identical DC motors to be driven at the same time at twelve Amps nominal and eighteen Amps peak. This motor controller allows hearty protection for the platforms more sensitive electronic components. Protection offered by the Sabertooth 2x12 was not planned as forethought for the platform, but an actual built in design in the Sabertooth 2x12 Motor Controller.

Leads from the power supply plug directly into the Sabertooth 2x12 Motor Controller. With the motors and the power supply directly connected to the motor controller, this allows the signals from the Atmel Atmega328p microcontroller to be read by the motor contoller, the motor controller will then draw the necessary power from power supply and the route them to motors. Moreover, the Sabertooth 2x12 Regenerative Motor Controller recycles all excess power, now in stepped down voltage, into the circuit and therefore limits the amount of power drawn from the power supply extending the run time of the power supply.

The Sabertooth 2x12 Motor Controller takes input signals in the form of Packetized Serial Protocol from the Atmel Atmega328p. Packetized Serial Protocol is a point to point protocol, and provides a very reliable form of communication between the Atmel Atmega 328p microcontroller and the Sabertooth 2x12 Motor Controller. This is due to the ability to flash a boot loader onto the Atmega328p. With the boot loader flashed onto the microcontroller a predefined library for the Packetized Serial protocol can be loaded on to the Atmel Atmega328p. Then the dip switches on the Sabertooth 2x12 Motor controller a configured with switches “one” and “two” in the down position. Switches “four” through “six” determine the address of the motor controller; the address can range from one hundred twenty-eight to one hundred thirty-five. The address can be configured so that multiple motor controllers can share the same serial transmitter. However, in the case of the Magic Plank we will only be using one.

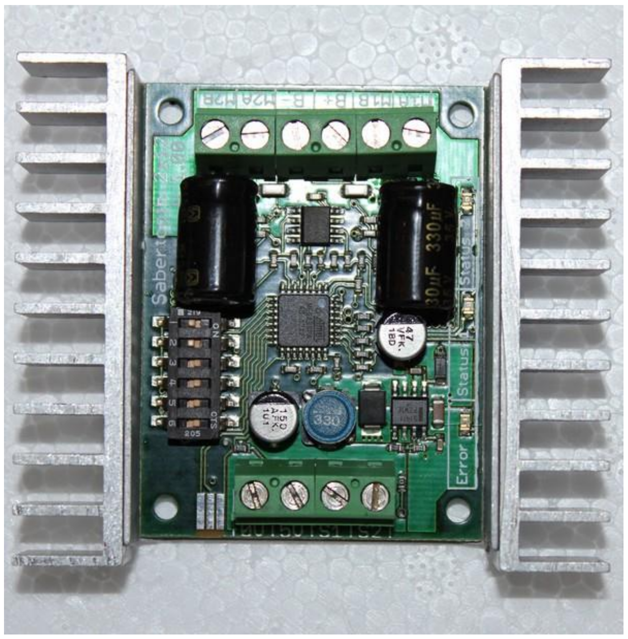

Figure 4: Picture of the Sabertooth 2x12 Motor Controller with the dip switches configured for Packetized Serial Protocol.

The Sabertooth 2x12 Motor Controller will be connected via pins “S1” and S2” to the Atmel Atmega328p pins nineteen and eighteen. The pins are the RX and TX and establish the line of communication between the two. Recycled power in the form or five volts is transferred from pin “5V” to the pin labeled ‘VCC” on the Atmega328p. Battery leads are connected positive into “B+” and negative into “B-“. The motors are connected “M1A”, “M1B”, “M2A”, and “M2B” respectively.

B. 250W Brushed DC Motors

The Magic Plank utilizes 2 250 watt brushed DC motors to balance. These powerful motors are rated to draw a current of 13A. Although the Sabertooth motor controller is rated for 12A nominally, it can support up to 18A peak current. With the batteries supplying 24V, 250 watt motors should be drawing around 10.4A nominally, which is well within the supported range of the motor controller. Even with motors rated for such high current draw, the motors are never used to their full capacity, which means that, under most circumstances, the motors will not be requiring a high current draw. Even in the event of a motor stall where the motors reverse direction and draw a large spike in the current, the motor controller offers overdraw protection, preventing the controller from burning out or becoming damaged.

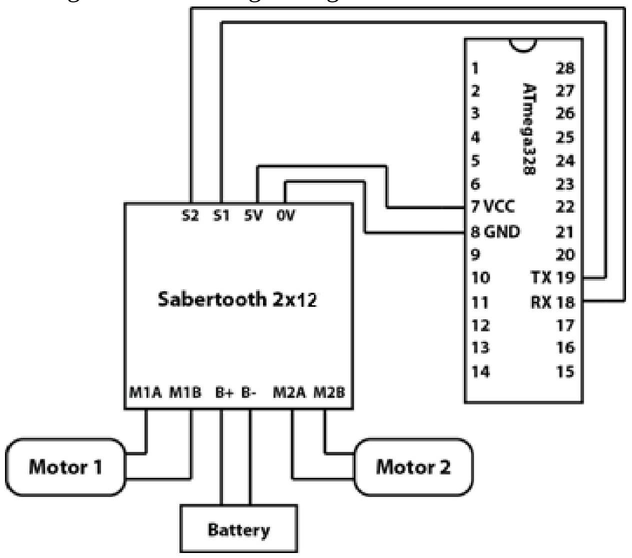

Figure 5: Figure illustration the pin connections between the Saberthooth 2x12 Motor Controller, Atmel Atmega328p Microcontroller, power supply, and motors.

Sabertooth 2x12 Motor Controller Pin Connections	
Pin on the Sabertooth 2x12	Connection
S2	Pin 19 on Atmega328p
S1	Pin 18 on Atmega328p
5V	VCC on Atmega328p
0V	Ground on Atmega328p
B+	Positive lead on Power supply
B-	Negative lead on Power Supply
M1A	Lead from Motor 1
M1B	Lead from Motor 1
M2A	Lead from Motor 2
M2B	Lead from Motor 2

Table 4: A table listing the pin connections between the Saberthooth 2x12 Motor Controller, Atmel Atmega328p Microcontroller, power supply, and motors.

VI. SOFTWARE OVERVIEW

The Magic Plank is has an intricate network of thought out classes. Each class provides a compartmentalized function that by itself useless, but combined with the other classes, provides all the necessary data that is needed to ensure that the balancer algorithm runs as smoothly as possible. The software design behind the magic plank consists of five distinct classes, each serving a compartmentalized function, but is important to overall effectiveness of the balancer algorithm.

The Sabertooth class handles the interaction of the motors and how the Packetize Serial Protocol sends and receives data from the Atmel Atmega328p Microcontroller. Furthermore, the Sabertooth class is responsible for handling the voltage required by the motors, as well as recycling unused power. Moreover, the sabertooth class handles the software revive and kill switches that either provide power to the motors or cut all power to them.

The Bluetooth class handles how the SMD Module RN-42 receives and sends data from Wiimote wireless controller and the Atmel Atmega328p. In addition, Bluetooth class, assists in processing the raw values that is transmitted from the Wiimote wireless controller and for the initial “pairing” of the Wiimote Wireless controller and the SMD Module RN-42.

The IMU class provides an important and complex job that is extremely essential to the efficiency of the balancer protocol. The IMU class takes full advantage of the “master-slave” relationship of the IMU 3000 Fusion board. Using the secondary I^2C, IMU 3000 is able to read the data from the ADXL345 registers; this data is passed through IMU 3000 as positional data measured in float point. The IMU class also calibrates the IMU 3000 Fusion Board for each use. The calibration is achieved by “zeroing out” the IMU 3000 Fusion Board.

This means that each time Magic Plank is powered on, the platform must be held in a level position to get a base reference read of being level. If Magic Plank is held in a position that is not horizontally level, then IMU 3000 Fusion board will that position as reference and strive to balance the Magic Plank to the reference position. The raw data that is measured by the IMU 3000 and the ADXL345 is converted into angular rate or degrees per second for the IMU 3000 and gees for the ADXL345. The data from the IMU class is then passed on to the Filter class, here the vibrational noise from the accelerometer and the drift from the gyroscope's drift. This ensures pure unadulterated data, which allows easier processing in the balancer algorithm. The data that is outputted by the IMU class in the form of current angle or projected angle. This is combined data of the IMU 3000 and the ADXL345. The filter class manages the IMU class.

The fifth and final class is the balancer itself. The balancer manages the Sabertooth, Bluetooth, and Filter classes. The Primary balancing algorithm is a Proportional Integral Derivative control loop. The data from the filter is passed into the balancer. The balancer using the Proportional Integral Derivative control loops creates data that is sent to the motor controller. The data is then interpreted by the motor controller to provide the proper power to the motors to keep the platform balanced. Moreover, in the same data packet, the balancer is also transmitting steering comrades to the motor controller. Again, the motor controller will interpret the data to determine which motor must spin faster than the other to either turn left or right.

VII. SOFTWARE DESIGN

The software design for the Magic Plank is really the main part of the project. It is implemented using the Arduino programming language, which is syntax compatible with C++. Taking advantage of this object oriented code base, the code is designed as a collection of modules which each perform their functions independently. The hierarchy takes a "bottom up" approach by first addressing bare hardware interfacing and handling the hardware functions to make it easier to handle from the higher level modules. The Sabertooth, IMU, and Bluetooth modules handle hardware interfacing while the Filter class manages the IMU and handles angle processing for the Balancer class, which manages all the classes in the hierarchy and incorporates them into a main control loop.

A. Balancer

The balancer class is the main "master" class of the system. Balancer, as seen in the code hierarchy, instantiates and manages the Bluetooth, Sabertooth, and Filter classes. The balancer contains the Proportional Integral Derivative (PID) control algorithm that calculates a speed for the motors. The main control loop operates by first updating the Bluetooth class, which will either result in a new steering value from the Bluetooth module or result in a failure. In the event of a failure, the loop will send a signal to terminate movement of the motors, which creates the effect of a software "dead man" switch. More on this functionality is described in detail in the section regarding the Bluetooth class. In the event of a success, the loop then continues to a loop timing control, which regulates the loop time of the algorithm. Currently, the algorithm executes a loop time of 50 ms, or 200 loops per second. After the loop regulation time, the PID control function is executed.

The PID control function operates by first updating the Filter class, which implicitly updates the IMU class, therefore sampling new data from the current data from the IMU3000. Once this update is complete, a new angle is returned for use in the PID calculation.

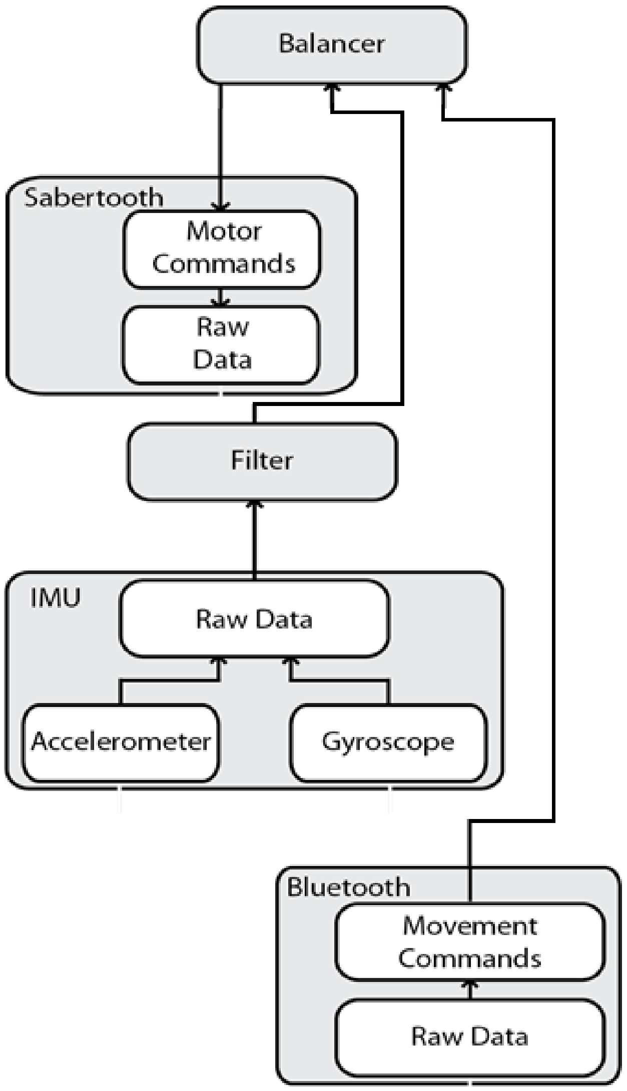

Figure 6: Block diagram that displays the relationship between each class.

The first thing to calculate is P, or the proportional gain of the system. P is the simplest part of the algorithm, simply corresponding to a motor offset directly proportional to the current angle of the system. P is calculated thusly:

$$P = \frac{9}{2}\theta + \frac{1}{2}\delta$$

Where θ is the current angle of the system and δ is the current rate of change of the angle as read from the gyroscope.

Next is I, or the Integral calculation. I can be considered a lag time that catches up to the Proportional response over time. The primary purpose of I is to provide a smoothing effect that prevents the platform from overcompensating and reduces the effects of oscillation on the system. I is calculated as follows:

$$I = \frac{I + P_{Prev} * I_{response}}{I_{response}} * 0.999$$

Where $I_{response}$ is a constant that affects the speed at which I will respond over time and P_{Prev} is the Proportional response of the previous loop.

The final piece of the puzzle is D, which is the Derivative. D can be considered a response to spikes in the system, where D increases with a rapid change over time and approaches zero when there is little change over time. This results in the ability of the system to more readily compensate for a rapid change so that it can compensate accordingly. D is calculated as follows:

$$D = \frac{[D + (P - P_{Prev}) * D_{response}]}{2}$$

Where P_{Prev} is the Proportional response of the previous loop, P is the Proportional response of the current loop, and $D_{response}$ is a constant that determines how rapidly D will change over time.

Finally, these are all summed together to produce the raw speed σ.

$$\sigma = [P * K_P + I * K_I + D * K_D] * \omega$$

Where ω is the gain, a multiplier used to increase the raw magnitude sent to the motor controller, and constants K_p, K_i, and K_d are constants used to determine the influence of P, I, and D.

Next, the raw speed is factored in with the steering command, which is a number that is added to the first motor and subtracted from the speed of the second motor in order to provide equal and opposite offsets to the motor, allowing for the platform to turn with a zero turn radius and thus preserving equilibrium.

B. Bluetooth

The Bluetooth class is the driver that handles all input from the Bluetooth RN-42 Bluetooth module. The first function is the update function, which reads in and parses a new two byte signed integer value from the Bluetooth module. This value is then cross-referenced against a pre-defined "kill" command. If the "kill" command is received, a killSwitch flag is marked. If the connection times out and does not receive any data, the update function returns false. If the killSwitch flag is marked or the update returns false, the Balancer takes appropriate action and terminates movement of the motors. Upon termination of the motors, the Balance r then calls the next function, which is the DeadLoop. In the loop, the Bluetooth module continues to be sampled until it receives a pre-defined "revive" command. Upon receiving this command, the loop exits and the Balancer is free to resume the balancing loop. In addition to handling the "kill switch", the secondary function of the Bluetooth class is to interpret incoming steering commands, which, if valid, are passed up to the Balancer class for integration into the control loop.

C. Filter

The Filter class handles processing the raw data passed up from the IMU class and passes the data through what is known as a Complementary Filter. The Accelerometer's angle can be calculated by taking the current gravitational force in Gs upon all three axes. First, the force vector R is calculated using the following formula:

$$R = \sqrt{{F_X}^2 * {F_Y}^2 * {F_Z}^2}$$

The relevant axis of tilt for the Magic Plank is the X axis. The angle upon the X axis can be calculated as follows:

$$\theta_{Accel} = \cos^{-1}\left(\frac{F_X}{R}\right) - \frac{\pi}{2}$$

The Gyroscope's angle calculation can be calculated by taking the current rate of change in degrees per second and factoring it in with the time elapsed since the previous sample as follows:

$$\theta_{Gyro} = \theta_{Gyro} + \frac{\delta_X * \delta_T}{1000}$$

Next, since the gyroscope suffers from gyroscopic drift over time and the accelerometer suffers from vibrational noise, the angles are passed through the filter. The complementary filter can be expressed as follows:

$$A = \frac{\tau}{\tau + \delta_T}$$

$$\theta = A * \left(\theta + \theta_{Gyro} * \delta_T\right) + (1 - A) * \theta_{Accel}$$

Where τ is adjusted to increase or decrease the ratio of the accelerometer to the gyroscope. With this filter, the accelerometer's vibrational noise can be suppressed while the gyroscope's drift can be corrected, returning a more accurate angle for the Balancer class.

D. IMU

The IMU class is the hardware driver for the IMU3000. The IMU works by utilizing the hardware supported I^{2}C interface and the Arduino I^{2}C software library. Initializing the IMU causes it to enter into the calibration function, which calibrates the IMU, assuming that the current orientation is level. This initialization is actually critical to the startup of the Magic Plank and determines the level at which the platform will attempt to balance. Updating the IMU causes the class to sample from the IMU3000's registers, sampling the most recent raw data from the gyroscope and its auxiliary accelerometer. It then subtracts the calibration offsets determined upon initialization and also converts the raw data into usable data: degrees per second for the gyroscope and force of gravity in Gs. These values are then passed up for the Filter class to use.

E. Sabertooth

The IMU class is the hardware driver for the IMU3000. The IMU works by utilizing the hardware supported I^{2}C interface and the Arduino I^{2}C software library. Initializing the IMU causes it to enter into the calibration function, which calibrates the IMU, assuming that the current orientation is level. This initialization is actually critical to the startup of the Magic Plank and determines the level at which the platform will attempt to balance. Updating the IMU causes the class to sample from the IMU3000's registers, sampling the most recent raw data from the gyroscope and its auxiliary accelerometer. It then subtracts the calibration offsets determined upon initialization and also converts the raw data into usable data: degrees per second for the gyroscope and force of gravity in Gs. These values are then passed up for the Filter class to use.

VII. CONCLUSION

The Magic Plank allows exploring a problem that has not been attempted yet at UCF. Exploring the inverted pendulum problem sheds light in an area that has not actively researched at UCF. The Magic Plank has and continues to provide much knowledge in embedded software programming and understand how Proportional Integral Derivative control loop handles the delicate balance of the inverted pendulum problem with the added challenge of wireless steering control.

BIOGRAPHY

Brian Jacobs is a student at the University of Central Florida pursuing a Bachelor of Science in Computer Engineering. Brian is currently employed as a software engineer at the DiSTI Corporation providing Human Machine Interface development and maintenance training services to a variety of clients ranging from private companies to the US military. His preference is to work on low level software design, and, after graduating, aspires to work in the field of embedded systems engineering.

Kenneth Santiago Jr, 24, is currently pursuing a Bachelor of Science in Computer Engineering degree at the University of Central Florida. After graduating in Summer of 2012, Kenneth plans to pursue a career in embedded systems design. Kenneth is currently employed at HostDime International as a System Technician. Here Kenneth provides a wide variety of support ranging from software to hardware. Through Kenneth's employment at HostDime International, Kenneth has gained valuable experience with a growing provider of web hosting and server solutions.

Stephen Colby Fraser II, 25, is currently pursuing a Bachelor of Science in Computer Engineering degree at the University of Central Florida. After graduating in the Summer of 2012, Stephen plans to pursue a career in the Defense Industry, in the fields of embedded systems design, database development and security, or network security engineer. Stephen is currently employed at HostDime International; Stephen has gained valuable experience with a continuously growing provider of web hosting and server solutions. Stephen is a self motivated learner who regularly maintains interest and works on web design, network security, database development and security, as well as embedded systems design projects. Stephen enjoys and looks forward to each new challenged that each project presents.