A Feedback System for Baseball Swings

by

Gerzain Mata

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2016

Author .....
Department of Electrical Engineering and Computer Science
May 9, 2016

Certified by .....
Jacob K. White
Professor
UAP Supervisor

A Feedback System for Baseball Swings

by

Gerzain Mata

Submitted to the Department of Electrical Engineering and Computer Science
on May 9, 2016, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Electrical Engineering and Computer Science

Abstract

In this project, I designed and implemented a feedback system which samples roll, pitch, and yaw coordinates, saves them in nonvolatile memory, and provides the saved coordinates as desired values when performing feedback during replication attempts; this involves the sampling of an integrated accelerometer, gyroscope, and magnetometer unit, processing the raw measurements through an implementation of Madgwick's gradient descent filter, and storing floating point numbers in a ferroelectric RAM for later retrieval. Feedback is given to the user through four vibration motors placed 90 degrees from each other along the main axis of the bat. The construction of the swing feedback system involved the physical modification of a plastic bat, the addition of reinforcing members from laser-cut acrylic, the addition of LEDs and pushbuttons for user interaction, and the addition of four vibration motors. The code incorporates open-source code available through GitHub and is based on the Arduino programming environment for accelerated embedded systems development.

UAP Supervisor: Jacob K. White
Title: Professor

Acknowledgments

Several individuals were instrumental in the realization of this project. The workspace and equipment for this project were provided by the Cypress Engineering Design Studio along with the 6.302 lab workspace. Many thanks to the project supervisor, Professor Jacob K. White. Sincere thanks to Joe Steinmeyer and Nick Arango for their recommendations in how to appropriately provide feedback to the user. Many thanks to Gavin Darcey and the staff at the EDS for their assistance in using the laser cutter and their patience with the constant noise produced. A special thank you to two students from the Electrical Engineering department: Hugo Malpica for your constant company and support in the development of this project. Grant Gunnison for your words of encouragement and feedback in the development of this project.

Contents

List of Figures

Chapter 1

Introduction

Chapter two describes the design process, which includes the development of the enclosure and its modifications.

Chapter three describes the user input/output and a high-level overview system diagram.

Chapter four describes the integration of the system, including the 9-DOF (Degrees Of Freedom) IMU, the FRAM module, and the motor controller board.

Chapter five describes the software and the feedback mechanism performed. Along with this a state machine representation of the device is presented and a high-level overview of major software routines are described.

Chapter six provides performance and accuracy observations along with recommendations for future work and improvement.

1.1 Background

There are various techniques employed in the art of feedback. There is the use of lead and lag compensators, Proportional, Integral, and Derivative (PID) controllers, and even the use of State Space modeling to achieve stability within a given model. Each type of feedback carries its advantages and tradeoffs, which determines what type of

feedback a designer uses when desiring stability.

Analog circuits enjoy the advantage of continuous feedback, whereby the current input of a system is always present and the desired feedback signal is immediately provided. On the other hand thanks to the proliferation of cheap microcontrollers it is now relatively inexpensive to develop code that samples an analog or digital signal, performs the appropriate operations, and provides the resulting feedback at a later (albeit short) time. This is also contingent on the clock speed of a given microcontroller.

1.2 Motivations for Swing Feedback

The rise of MEMS technology has enabled the development of cheap IMUs (Inertial Measurement Units) to be included in almost every single piece of mobile computing available today. There has been a recent interest in "Augmented Reality" and "Virtual Reality" within the context of gaming and social interaction. Unfortunately very little emphasis has been placed in the realm of feedback, whereby the user experiences a reaction with respect to a given input.

Every single mobile device available today includes a small vibration motor, which generally provides the user with vibration whenever a message, a notification, or a call has been received. Recently Apple Inc. has introduced "Force Touch" whereby a user believes they have clicked on a surface but in reality the user has experienced vibrations from a linear actuator located under the trackpad surface.

It seems a natural evolution of the concept of User Interface design that users would like to experience feedback in the realm of sports or other physical motion. Small children or mobility-limited individuals undergoing physical therapy demand care and attention when relearning motor skills. Sometimes movements can become staggered or irregular after undergoing surgical procedures or after being struck by IEDs (Improvised Explosive Devices).

It would be advantageous if "proper" movements could be recorded that could serve as a reference in order for a user to learn those same reference movements. It would be even better if realtime feedback could be provided where current movements could be compared to reference movements and an indication of corrective movement could be provided.

This is the idea behind the device implemented: record reference coordinates taken from an IMU, save them in nonvolatile memory for later retrieval, and give a user feedback when they try to implement the same movements through the motion of vibration motors.

Chapter 2

Design Process

The original idea was inspired from the swinging motion required to swing a sledgehammer to drive a wooden stake. The motion requires precision at the moment just before the sledgehammer strikes the stake since any significant deviation from the head of the stake means the sledgehammer will miss the stake and the work required to bring the hammer up and swing the weight downwards will have been lost.

2.1 Enclosure

Within the context of a physical implement it became apparent that space would be required to store all the electronics and accessories. Anything composed ferromagnetic metal would have imposed additional challenges since the magnetometer works on the assumption that it is distant from any strong magnetic fields or anything capable of being induced by magnetic fields. Wood itself is rigid and has qualities advantageous to being subjected to significant forces. Unfortunately working with wood also requires specialized tools and significant care that every cut and bore be planned.

Plastic is cheap, fairly flexible, and can be quite rigid if properly supported. It also allows for large enclosures to be made with fairly thin walls. It is also immune

to EM (electromagnetic) fields and is an excellent insulator.

The enclosure chosen for the electronics is a "MLB Jumbo Plastic Bat" developed by Franklin Sports. It measures approximately 25 inches with an internal radius of $3\frac{1}{2}$ inches for a length of 11 inches with a decreasing radius down to $1\frac{3}{16}$ inches. The bat can be seen in Figure A-1

The plastic bat was cut in half lengthwise along the mold joint. This allowed all of the electronics, wiring, pushbuttons, and LED to be placed inside the bat at the loss of vital structural integrity. The development of structural supports were needed to be able to join the plastic bat together and provide support when the bat was used.

2.2 Reinforcements

Since the structural integrity of the bat was reduced the laser cutter at the EDS located in the fifth floor of building 38 was used to make internal supports for the plastic baseball bat. Figure A-2 shows the general dimensions of the internal supports. The baseball bat had slots cut into it so that the protrusions of the supports could serve as slats on which the bat could support itself. Within each structural support four holes were cut so that wires could be passed from one section of the bat to the other. The supports were then adhered to one half of the bat with epoxy. The other half was left detached so that the bat could be disassembled. The main electronic components were able to be placed securely inside one of the partitions the supports formed. Ample space was provided for the purpose of inserting masses to change the center of mass. The electronics and motor assembly proved to be quite heavy so two steel bolts were placed inside the bat handle to balance the bat.

2.3 Motor Assembly

Four vibration motors needed to be placed 90° from each other in order to provide yaw and pitch feedback. Seeing as how rotation along the principal axis of a baseball bat is largely irrelevant it later became clear that roll coordinates were largely unnecessary. The motor's vibrations could interfere with the measurements of the IMU, which meant that a damping adhesive would be advantageous to reduce noise in the coordinate measurements. Four holes were cut in the bat: two holes perpendicular to the cut plane of the bat and half holes on each side of the cut bat. All of the holes were placed four inches from the top of the bat. The motors were secured with hot glue to plastic IC packaging tubes. The empty channel inside the packaging tube allowed the vibration motor wiring to be passed through the tube and into the baseball bat.

Chapter 3

Hardware

The swing feedback system involves significant use of electronic components. The objective was to use only as many wires as necessary due to the constrained environment inside the baseball bat after the microcontroller, the peripherals, and the motor supports were included. It became an even more pressing issue when the addition of supports limited the bending of wires inside each partition.

3.1 System Overview

The bat device is composed of the following components:

1. An ATmega328p 16MHz microcontroller in an Arduino development board
2. A Cypress CY15FRAMKIT-001 Serial F-RAM Development Kit
3. An Adafruit V2 Motor Shield for Arduino
4. An Adafruit 9DOF IMU
5. Four vibration motors
6. A rocker switch

7. A pushbutton

8. One RGB LED

A system diagram can be seen in Figure A-3

3.1.1 ATmega328p Microcontrolller

The microcontroller uses an 8-bit RISC architecture and runs at 16MHz with 32Kbytes of program memory and 2Kbytes of RAM. It has three onboard timers: one is used by the Arduino library for the millis() and micros() timekeeping functions. The second timer is used for button interrupts, and the last one is used for other optional timing events. The Arduino programming environment exposes two external interrupt pins. The microcontroller also includes serial, SPI, and I2C modules which allow serialized communications with external peripherals. There are also various PWM (Pulse Width Modulation) pins which allow for time-averaged variable voltage. This specific microcontroller has also enjoyed widespread adoption amongst the "maker" crowd, which has allowed the development of various third party "libraries" (in essence header and include files) that facilitate the rapid prototyping of embedded systems.

3.1.2 Cypress FRAM Module

Ferroelectric RAM is a relatively recent technology that is useful for embedded applications. Cypress Semiconductor claims their FRAM modules can widthstand up to 100 trillion write cycles. This specific memory is also nonvolatile and have fast read and write times. The Cypress module includes 32Kbytes of memory in a I2C module and another 32Kbytes of memory in a SPI module.

3.1.3 IMU

The Adafruit 9DOF IMU module is composed of the L3GD20H 3-axis gyroscope and the LSM303 magnetometer/accelerometer module. Each device is addressed through

its own I2C address at a clock rate of 100KHz. The module outputs three floating point integers for each axis of the accelerometer, gyroscope, and magnetometer. Each sensor has configurable gains and can be programmed to trigger interrupts given a predetermined condition. For this project the raw sampling mode was employed.

3.1.4 Motor Shield and Vibration Motors

The vibration motors are driven at 5 volts through a motor controller board. The controller board is essentially a I2C-addressed shift register connected to two dual H-bridges that are capable of driving up to four vibration motors bidirectionally.

3.2 User Input/Output

User input is provided through a rocker switch and a push button. The rocker switch serves to indicate to the microcontroller when it should start saving measurements into FRAM (this will be explained in further detail in chapter 4). The pushbutton is pressed whenever the user desired to reproduce the reference coordinate measurements (the functionality of which is explained in chapter 4). The RGB LED serves to let the user know in what state the device is currently in.

3.3 Considerations

The devices were chosen for their ease of integration with the Arduino board and due in part to the total number of ports used by the I2C devices. I2C uses a master-slave protocol whereby the master addresses an individual device using a 7-bit address and sends data bits afterwards. The slave responds with an ACK bit acknowledging receipt of a message. The physical bus itself uses two wires: SCL (Serial Clock) and SDA (Serial Data). The number of physical pins used by a device become a point of consideration when trying to integrate multiple components. The RGB LED uses

three pins (specifically pins 8,9, and 10) on the Arduino board, which also happen to be PWM pins. This allows for future iterations to be able to light up to potentially 16 million colors (8 bits of color for each R,G, and B LED) on a single LED. The switches were connected to pins 2 and 3 which allowed the pins to be configured as external interrupts (the functionality which will be explained in chapter 4).

Chapter 4

System Integration

4.1 Power

The bat device and its peripherals are powered by four AA batteries housed in a battery pack. A toggle switch located opposite the rocker and pushbutton switches turns on the apparatus. The terminals are connected to the motor driver board, which then provides power to the ATmega328p.

4.2 IMU Measurements

The Adafruit 9DOF module is read continuously. As previously mentioned in subsection 3.1.3 the IMU outputs three floating point integers for the x,y, and z axis of the accelerometer, magnetometer, and gyroscope. Each floating point integer uses 32 bits of memory, which means that it takes 12 floats to store the x,y, and z values of each sensor. 48 bytes in total are needed for each sample period of the sensors. It becomes apparent that memory is a limiting factor when storing floating point integers sampled at high rates. If a user desired to store the 48 bytes sensor data of a 30 second sampling period at a sampling rate of 60 Hertz then that would require approximately 86Kbytes of data, more than the total ATmega memory space. It is also desirable to

compress the sampled data into a smaller number of bits without losing the capability of extracting absolute coordinate values with respect to a reference frame.

4.3 FRAM Storage/Retrieval

The Cypress FRAM board has 32Kbytes of memory. Data transfers from the ATmega328p microcontroller to the I2C module occur in bytes. Up to 32 bytes of memory can be transferred in a single transaction. Because the Cypress FRAM stores individual bytes in its memory and floating point integers are 32 bits wide care must be taken to ensure that the floating point integers can be separated into four bytes and recovered as 32 bits from a memory read. Solution is the C union datatype, whereby consecutive memory addresses are mapped as the maxima of two different data types. In this case both an array of integers and a single floating point integer can be mapped to the same memory address.

4.4 Motor Board

The final version of the motor board used in the device provides the advantage of effectively using only two pins for control of four bidirectional motors since it's a I2C device. The supporting software header files developed by Adafruit provide an easy mechanism whereby speed and direction can be controlled easily.

4.5 Microcontroller

The microcontroller is the center of every input and output to the bat. Since the board runs at 16 Megahertz it is important to take into consideration the time spent calculating yaw, pitch, and roll values and ensuring that the sampling rate of the IMU allows for effective feedback. The microcontroller's pin interrupts are triggered when the user presses either the rocker switch or the pushbutton switch. Triggering on an

interrupt allows the microcontroller to immediately "interrupt" currently-executing code and execute the Interrupt Service Routine. This same mechanism allows the microcontroller to calculate accurate time wth regards to the number of micro and milliseconds passed since the device last turned on.

Chapter 5

Software

The compiled code uses 26,726 bytes out of a total 32,256 bytes (82.9% of program storage space) and 1,192 bytes out of a total 2,048 bytes (58.2% of dynamic memory). The relatively high program space use is impressive considering that various functions are used to talk to three different I2C devices and each device stores its own variables within the microcontroller memory space.

5.1 State Machine Description

At startup the microcontroller configures all of the I2C using their respective header and implementation files.

Pins 2 and 3 of the ATmega328p are configured as interrupt pins. The pushbutton switch triggers an interrupt when the logic level changes to a low logic level. The rocker switch triggers an interrupt on a rising edge.

Pins 8,9, and 10 are configured as output pins. The RGB LED has a common anode, so a high logic level turns off the LED and a low logic level drives current through the LED.

Initially the state of the device is configured to START. The RGB LED is illuminated at a constant blue color in this state. When the user holds the rocker switch

the interrupt to which it's configured to triggers a change of state to RECORD. The RGB LED is illuminated at a constant red color in this state. As long as the user holds the rocker switch the microcontroller will continue to store coordinate samples to FRAM.

Once the user releases the rocker switch it triggers another interrupt and the microcontroller changes the state to START again. The last memory address written to is stored in the writeFramAddress variable.

When the user momentarily presses the pushbutton switch the microcontroller sets readFramAddress to address 0x0000. The microcontroller then transitions to the DELAY state.

The DELAY state was implemented to allow the user time to prepare his/herself to the initial position of the reference swing. The RGB LED is illuminated at a constant yellow color in this tate. After two seconds the microcontroller transitions to the PLAYBACK state.

In this state the microcontroller reads the FRAM memory locations starting with address 0x0000. The current read FRAM address is stored in readFramAddress. At every sample the roll, pitch, and yaw measurements are added to three parallel PID controllers with individually configurable gains in code. The output from the PID controllers are calculated and are applied to the pitch and yaw directions. The vibration of the motors is proportional to the error of the current coordinate versus the reference coordinate. The motor that vibrates is the opposite direction of the corrected direction of travel.

5.2 Measurement Filtering

Madgwick's paper [2] describes an implementation of a gradient descent algorithm useful for calculation of roll, pitch, and yaw coordinates requiring 277 scalar arithmetic operations. Assuming a 16 MHz clock rate and an average of one instruction

completed for every two clock cycles then there is a theoretical upper limit of 28.8 kHz sampling rate. This is without taking into consideration the 100 kHz clock rate of the serial I2C and the time it takes to poll the IMU and FRAM. Madgwick's algorithm takes in raw accelerometer readings and performs an efficient gradient descent algorithm where a measurement of the Earth's magnetic field within the sensor frame will allow an orientation of the sensor frame relative to the earth frame to be calculated.

This algorithm is updated at each sample and the calculated roll, pitch, and yaw are provided as output. The output of that same is then used as input to three parallel PID controllers.

5.3 Feedback Implementation

The continuous time PID controller can be described by the following equation:

$$output(t) = K_p * error(t) + K_i * \int_0^t error(\tau)\, d\tau + K_d * \frac{\mathrm{d}}{\mathrm{d}t} error(t)$$

where error(t) is defined as:

$$error(t) = desired(t) - measured(t)$$

Whereas the discrete time PID controller can be described by the following:

$$output[n] = K_p * error[n] + K_i * \sum_{x=0}^{n} error(x) + K_d * \frac{error[n]}{\Delta t}$$

where error[n] is defined as:

$$error[n] = desired[n] - measured[n]$$

In these equations there are a couple of changes that can be tweaked depending on

the application. Beauregard's onine article [1] states that if we assume that our error term can become too large at any one time then we can also prevent the output from going outside of the range of PWM values. In the following code one can appreciate the modifications necessary to prevent overshooting of the PID controller. At the same time the dt term may be irregular, so it is calculated depending on the change in time since the PID controller was last run. This is included elsewhere in the source code.

```
void updateState(float newDesired, float current) {
  desired = newDesired;
  prev_error = error;
  current_meas = current;
  error = desired - current_meas; // current error
  error_integral += error * dt * ki;
  if (error_integral > MAX_PWM) error_integral = MAX_PWM;
  else if (error_integral < MIN_PWM) error_integral = MIN_PWM;
  error_deriv = (error - prev_error) / dt;
  output = kp * error + error_integral - kd * error_deriv;
}
```

Chapter 6

Performance and Future Work

6.1 Speed and Accuracy

The filter achieves stability within two-three seconds from startup. This is somewhat limiting in its responsiveness and can probably be corrected through Madgwick's filter tuning. The PID controller achieves good performance at small angle deviations. This can also be corrected through PID tuning. The principal limiting facts include the relatively low clock rate of the ATmega32p and the IMU's 100 kHz sampling rate. The peripherals themselves are capable of being clocked at higher rates (up to 400 kHz according to their respective datasheets) so in effect the microcontroller has become the bottleneck.

6.2 Improvements

Madgwick's filter provides fast reaction to large movements, but struggles to converge to its final value, with an average time constant of 1.2 seconds. While disappointing the filter response can probably be improved through filter tuning.

6.3 Future Work

This project provided a good foundation from which further work in the realm of user feedback can be started. With the eventual proliferation of the Internet-Of-Things further applications of feedback will be found and eventually better-performing sensors will be developed that will allow control designers to achieve even better performance in areas such as autonomous flight and responsive critical systems.

Appendix A

Figures

Figure A-1: The bat used as an enclosure

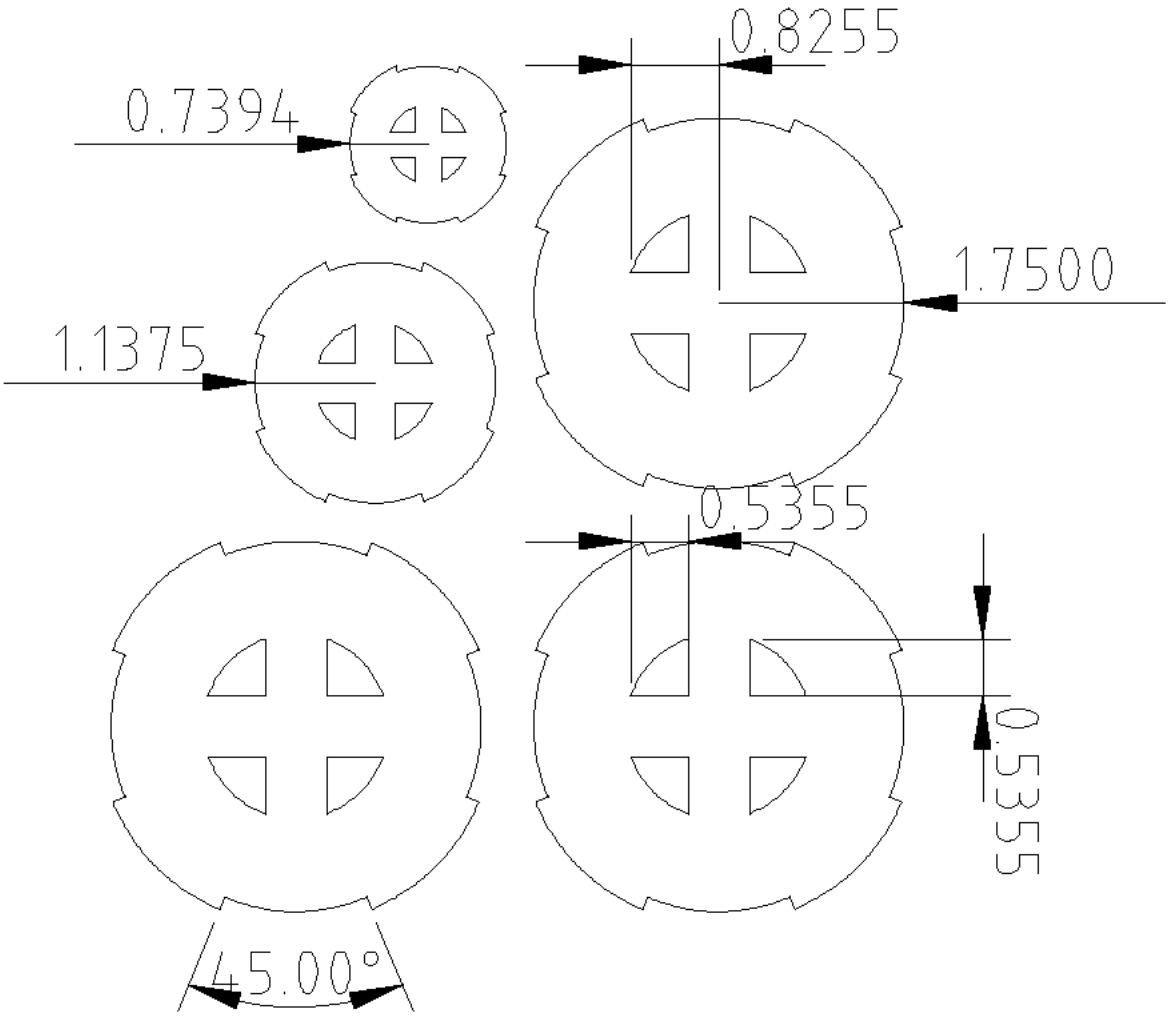

Figure A-2: CAD design and dimensions for bat internal supports

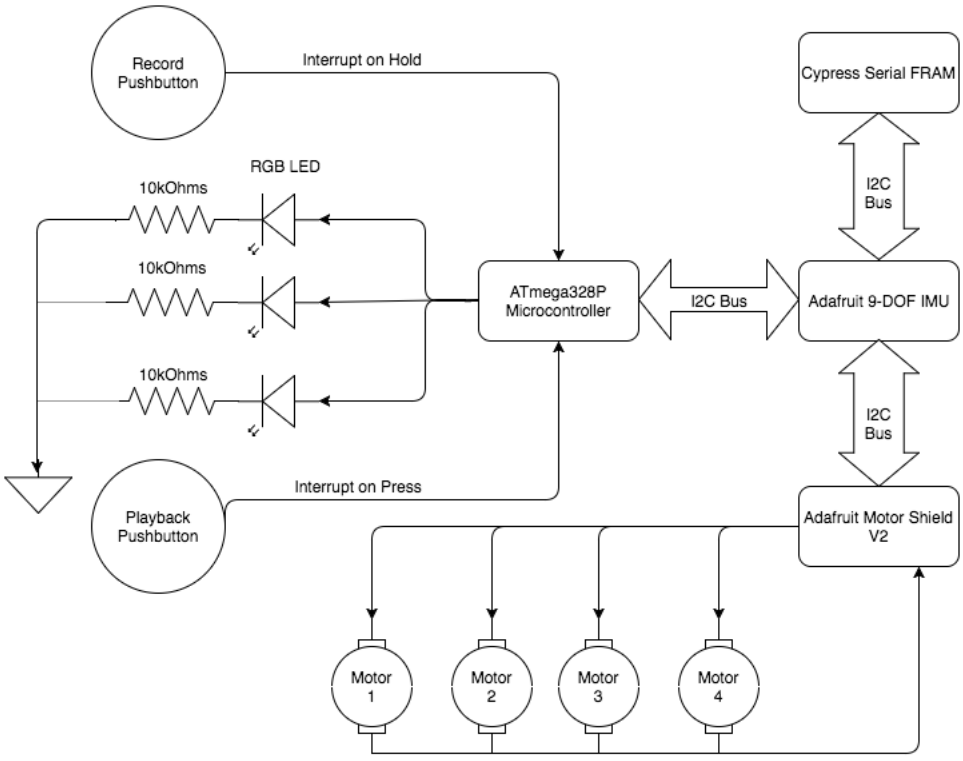

Figure A-3: Hardware Overview

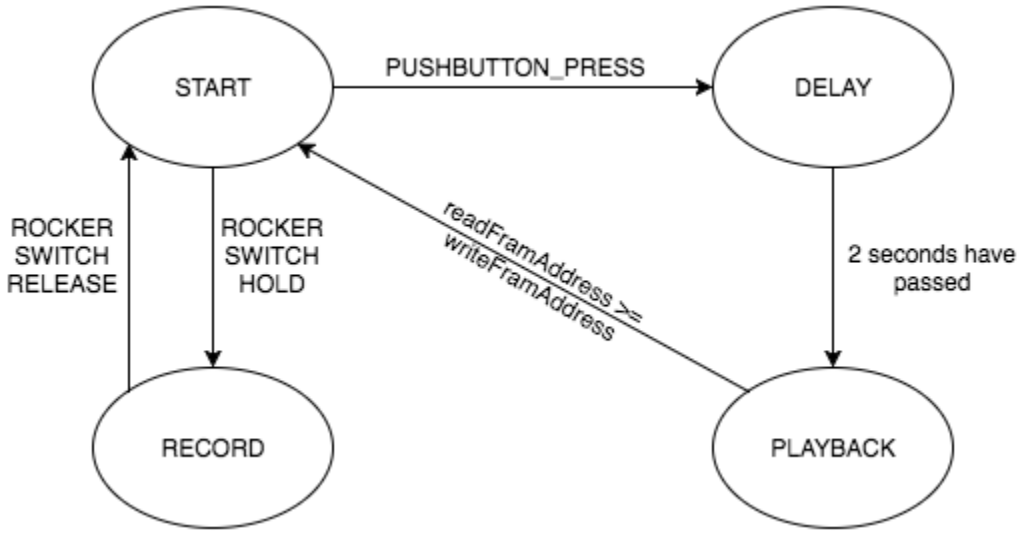

Figure A-4: State Machine representation of software

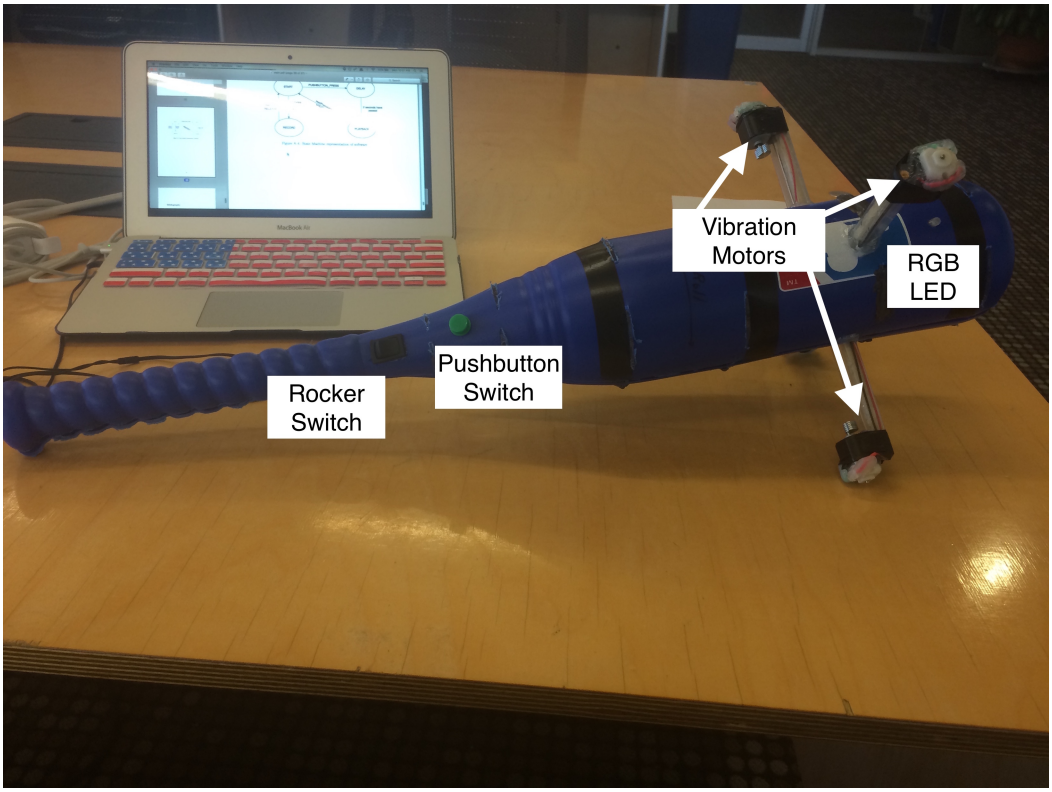

Figure A-5: Outside view of apparatus

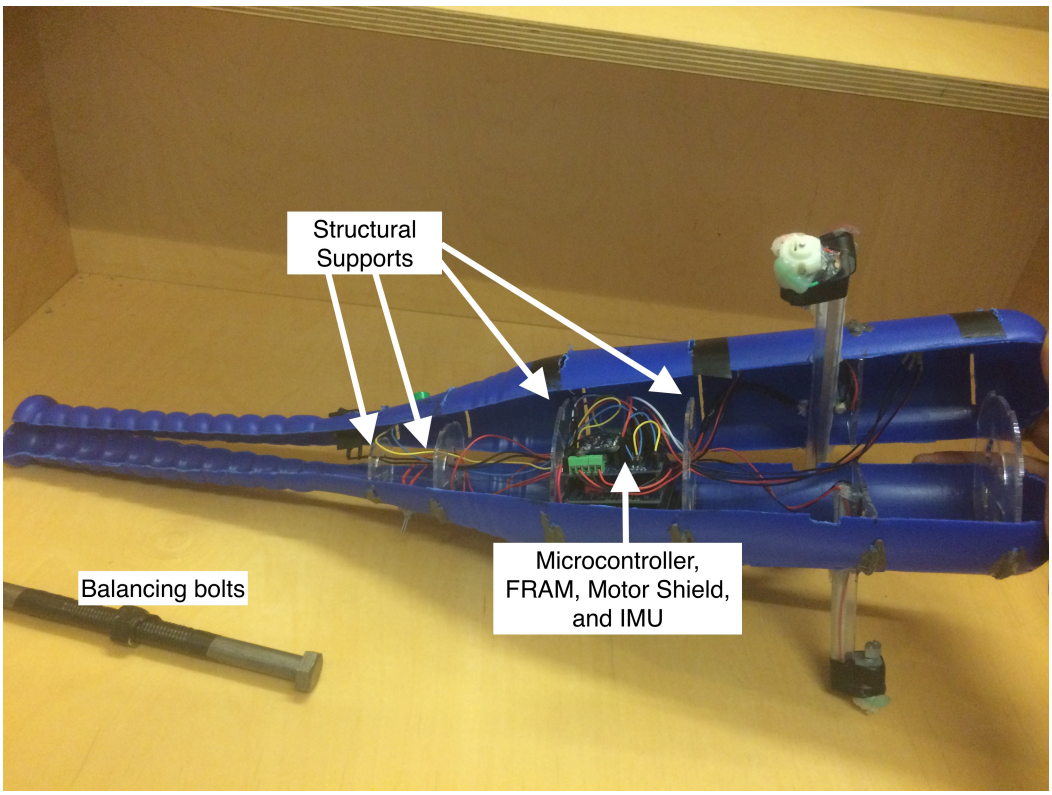

Figure A-6: Inside of apparatus and hardware

Bibliography

[1] Brett Beauregard. Improving the beginner's pid - introduction, apr 2011.

[2] Sebastian O. H. Madgwick, Andrew J. L. Harrison, and Ravi Vaidyanathan. Estimation of IMU and MARG orientation using a gradient descent algorithm. In *Proceedings of the IEEE International Conference on Rehabilitation Robotics (ICORR)*, pages 1–7. IEEE, June 2011.