**PERSISTENCE OF VISION ON A BICYCLE**

**Matthew Egan**

**Sean Schindzielorz**

**Gian Thomas**

**Nancy Zanaty**

**SPRING 2014**

**Sponsored by** 



**Table of Contents**

[1 Executive Summary 1](#_Toc386391979)

[2 About the Sponsor 2](#_Toc386391980)

[3 Project Description 3](#_Toc386391981)

[3.1 Objectives 3](#_Toc386391982)

[3.1.1 Mobility 3](#_Toc386391983)

[3.1.2 Endurance 3](#_Toc386391984)

[3.1.3 Range 3](#_Toc386391985)

[3.1.4 Compatibility 4](#_Toc386391986)

[3.2 Project Requirements and Specifications 4](#_Toc386391987)

[3.2.1 Hardware Requirements 4](#_Toc386391988)

[3.2.2 Micro-controller 5](#_Toc386391989)

[3.2.3 LED array 5](#_Toc386391990)

[3.2.4 LED controller 6](#_Toc386391991)

[3.2.5 RPM sensor 6](#_Toc386391992)

[3.2.6 Display 7](#_Toc386391993)

[3.2.7 Wireless 7](#_Toc386391994)

[3.2.8 Power 7](#_Toc386391995)

[3.2.8.1 Battery 8](#_Toc386391996)

[3.2.8.2 AC adapter 8](#_Toc386391997)

[3.2.8.3 Generator 8](#_Toc386391998)

[3.2.9 Printed Circuit Board (PCB) 9](#_Toc386391999)

[3.2.10 Extended features 9](#_Toc386392000)

[3.2.11 Housing and mounting 9](#_Toc386392001)

[3.2.12 Software Requirements 10](#_Toc386392002)

[3.2.12.1 Performance 10](#_Toc386392003)

[3.2.12.2 Accuracy 10](#_Toc386392004)

[3.2.12.3 Stability 11](#_Toc386392005)

[3.2.12.4 Ease-of-Use 11](#_Toc386392006)

[3.3 Equipment 11](#_Toc386392007)

[3.4 Specifications 13](#_Toc386392008)

[3.5 Project constraints 14](#_Toc386392009)

[4 Existing Projects 15](#_Toc386392010)

[4.1 Cornell University Student Final Project Persistence of Vision Display 15](#_Toc386392011)

[4.2 University of Central Florida Senior Design Persistence of Vision Display 16](#_Toc386392012)

[4.3 Texas Instruments Intern Design Project “Spoke Ink” Persistence of Vision Bicycle Wheel 17](#_Toc386392013)

[5 Hardware Research 19](#_Toc386392014)

[5.1 Microcontrollers 19](#_Toc386392015)

[5.1.1 MSP430g2553IN20 19](#_Toc386392016)

[5.1.1.1 Features 20](#_Toc386392017)

[5.1.1.2 Calculation 20](#_Toc386392018)

[5.1.2 PIC32MX340F512H 21](#_Toc386392019)

[5.1.3 Attiny2313 21](#_Toc386392020)

[5.1.4 ATMega8515 21](#_Toc386392021)

[5.2 LED Controllers 24](#_Toc386392022)

[5.2.1 TLC5940NT 25](#_Toc386392023)

[5.2.1.1 Features 25](#_Toc386392024)

[5.2.2 LT3746 26](#_Toc386392025)

[5.2.3 TLC 5947 26](#_Toc386392026)

[5.2.4 HT16D724 26](#_Toc386392027)

[5.3 Generators 27](#_Toc386392028)

[5.3.1 bike2power Bottle Sidewall Bicycle Dynamo Power Generator 28](#_Toc386392029)

[5.3.2 Human Creations Bicycle Dynamo USB Charger 28](#_Toc386392030)

[5.4 AC Converters 29](#_Toc386392031)

[5.4.1 AC-to-DC Converter with Transformer 29](#_Toc386392032)

[5.4.2 AC-to-DC Converter (No Transformer) 30](#_Toc386392033)

[5.4.3 AC- to- DC Converter with op-Amps 30](#_Toc386392034)

[5.4.4 AC-to-DC Converter Circuit Components 31](#_Toc386392035)

[5.4.4.1 Diodes 31](#_Toc386392036)

[5.4.4.2 Resistors 32](#_Toc386392037)

[5.5 Battery 32](#_Toc386392038)

[5.5.1 Nickel Cadmium (Ni-Cd) Battery 32](#_Toc386392039)

[5.5.2 Nickel Metal Hydride Battery 33](#_Toc386392040)

[5.5.3 Lithium Ion Battery 33](#_Toc386392041)

[5.5.4 Types of Li-ion Batteries 33](#_Toc386392042)

[5.6 LED Types 39](#_Toc386392043)

[5.6.1 Miniature LEDs 39](#_Toc386392044)

[5.6.2 High Power LEDs 39](#_Toc386392045)

[5.6.3 Application Specific Variations LEDs 39](#_Toc386392046)

[5.6.3.1 Flashing LEDs 39](#_Toc386392047)

[5.6.3.2 Bi-Color LEDs 40](#_Toc386392048)

[5.6.3.3 Tri-Color LEDs 40](#_Toc386392049)

[5.6.3.4 RGB LEDs 40](#_Toc386392050)

[5.7 Wireless Communication 40](#_Toc386392051)

[5.7.1 Wi-Fi 40](#_Toc386392052)

[5.7.2 GSM 42](#_Toc386392053)

[5.7.3 Bluetooth 43](#_Toc386392054)

[5.7.4 Possible Bluetooth Modules 43](#_Toc386392055)

[5.7.4.1 RN-42 43](#_Toc386392056)

[5.7.4.2 Wireless Bluetooth V2.0 RS232 TTL Transceiver Module 44](#_Toc386392057)

[5.7.4.3 HC-06 serial 45](#_Toc386392058)

[5.8 Displays 45](#_Toc386392059)

[5.8.1 Cathode Ray Tube 46](#_Toc386392060)

[5.8.2 Plasma 46](#_Toc386392061)

[5.8.3 Liquid Crystal 46](#_Toc386392062)

[5.8.4 7-Segment LED 46](#_Toc386392063)

[5.9 RPM Sensors 47](#_Toc386392064)

[6 Hardware Summary 51](#_Toc386392065)

[6.1 Selected Microcontroller 51](#_Toc386392066)

[6.2 Selected Battery 51](#_Toc386392067)

[6.3 Selected Generator 52](#_Toc386392068)

[6.4 Selected LEDs 52](#_Toc386392069)

[6.5 Selected LED Driver 53](#_Toc386392070)

[6.6 Selected RPM sensor 54](#_Toc386392071)

[6.7 Selected AC converter 55](#_Toc386392072)

[7 Software Research 57](#_Toc386392073)

[7.1 Battery Regulator Module 57](#_Toc386392074)

[7.2 User Interface Modules 57](#_Toc386392075)

[7.2.1 Phone 58](#_Toc386392076)

[7.2.1.1 Android 58](#_Toc386392077)

[7.2.1.2 Apple 59](#_Toc386392078)

[7.2.2 Computer 60](#_Toc386392079)

[7.2.2.1 Java 60](#_Toc386392080)

[7.2.2.2 C# 61](#_Toc386392081)

[7.2.2.3 C++ 61](#_Toc386392082)

[7.2.2.4 PHP 61](#_Toc386392083)

[7.3 Wireless Modules 64](#_Toc386392084)

[7.3.1 GSM Send 64](#_Toc386392085)

[7.3.1.1 Java Communication API 64](#_Toc386392086)

[7.3.2 Bluetooth Send 65](#_Toc386392087)

[7.3.3 Communication 66](#_Toc386392088)

[7.3.4 Receive 66](#_Toc386392089)

[7.4 Image Conversion Algorithms 67](#_Toc386392090)

[7.4.1 Scaling Algorithms 67](#_Toc386392091)

[7.4.1.1 Nearest Neighbor Image Scaling 67](#_Toc386392092)

[7.4.1.2 Bilinear Image Scaling 68](#_Toc386392093)

[7.4.1.3 Trilinear Image Scaling 69](#_Toc386392094)

[7.4.2 Radial Image Conversion 70](#_Toc386392095)

[7.5 7-Segment Displays 72](#_Toc386392096)

[7.5.1 Common Anode 72](#_Toc386392097)

[7.5.2 Common Cathode 73](#_Toc386392098)

[7.5.3 Power Consumption 73](#_Toc386392099)

[7.6 LED Controller Modules 73](#_Toc386392100)

[7.6.1.1 TLC5940 73](#_Toc386392101)

[8 Hardware Design (Prototyping) 77](#_Toc386392102)

[8.1 PCB Layout 78](#_Toc386392103)

[8.2 Microcontroller 80](#_Toc386392104)

[8.3 LED Controllers 82](#_Toc386392105)

[8.4 RPM Sensor 83](#_Toc386392106)

[8.5 Power 83](#_Toc386392107)

[8.5.1 Voltage regulator 84](#_Toc386392108)

[8.5.2 AC Converter 85](#_Toc386392109)

[8.5.3 Voltage and Current Restraints 86](#_Toc386392110)

[8.5.4 Battery Regulation 87](#_Toc386392111)

[8.5.5 Battery Switching 88](#_Toc386392112)

[8.5.6 Seven-Segment Battery Readings 89](#_Toc386392113)

[8.6 Cellular Device Charging Station 89](#_Toc386392114)

[8.7 Mechanical Design 90](#_Toc386392115)

[9 Software Design 96](#_Toc386392116)

[9.1 Dataflow Diagrams 96](#_Toc386392117)

[9.2 Driver Programs 97](#_Toc386392118)

[9.2.1 Hall Effect RPM Sensor 97](#_Toc386392119)

[9.2.2 Bluetooth Wireless 98](#_Toc386392120)

[9.2.3 7 Segment 98](#_Toc386392121)

[9.3 LED ARRAY Prototyping 100](#_Toc386392122)

[9.4 GUI Prototyping 102](#_Toc386392123)

[10 Hardware Testing 106](#_Toc386392124)

[10.1 Mechanical Testing 106](#_Toc386392125)

[10.2 Microcontroller Signals 106](#_Toc386392126)

[10.3 LED Driver Testing 106](#_Toc386392127)

[10.4 LED Testing 107](#_Toc386392128)

[10.5 Power Testing 107](#_Toc386392129)

[10.5.1 Generator Testing 107](#_Toc386392130)

[10.5.2 Battery Endurance Testing 108](#_Toc386392131)

[10.5.3 Battery Charger Testing 108](#_Toc386392132)

[11 Software Testing 110](#_Toc386392133)

[11.1 GUI Testing 110](#_Toc386392134)

[11.2 General Software Testing 110](#_Toc386392135)

[11.3 Handle Bar Display Testing 111](#_Toc386392136)

[11.4 Wireless Testing 111](#_Toc386392137)

[11.5 Text Processing Testing 112](#_Toc386392138)

[12 Software Design Summary 113](#_Toc386392139)

[13 Operational Instructions 115](#_Toc386392140)

[14 Administrative Content 117](#_Toc386392141)

[14.1 Research and Design Methods 117](#_Toc386392142)

[14.2 Budget 118](#_Toc386392143)

[14.3 Project Scheduling 120](#_Toc386392144)

[14.4 Suppliers 121](#_Toc386392145)

[14.5 Team and Project Organization 122](#_Toc386392146)

[15 Works Cited 124](#_Toc386392147)

[16 Permissions 128](#_Toc386392148)

**Table of Figures**

[Figure 1:The bicycle that will be used in this project 12](#_Toc373613433)

[Figure 2: Rear wheel of the bicycle 12](#_Toc373613434)

[Figure 3: Bicycle Handle Bars 13](#_Toc373613436)

[Figure 3: UCF POV Senior Design Final Result 17](#_Toc373613437)

[Figure 4: MSP PIC CMP Diagram. 23](#_Toc373613438)

[Figure 5: PIN Attachment Diagram. 24](#_Toc373613439)

[Figure 6: AC to DC Converter with Transformer Circuit Schematic 29](#_Toc373613441)

[Figure 7: AC to DC Converter (No Transformer) Circuit Schematic 30](#_Toc373613442)

[Figure 8: AC to DC Converter with Op Amps Circuit Schematic 31](#_Toc373613443)

[Figure 9: A web diagram of the strengths of a LCO battery . 34](#_Toc373613444)

[Figure 10: A web diagram of the strengths of a LMO battery 34](#_Toc373613445)

[Figure 11: A web diagram of the strengths of a LFP battery 35](#_Toc373613446)

[Figure 12: A web diagram of the strengths of a NMC battery 35](#_Toc373613447)

[Figure 13: A web diagram of the strengths of a NCA battery 36](#_Toc373613448)

[Figure 14: A web diagram of the strengths of a LTO battery 36](#_Toc373613449)

[Figure 15: The general setup of a Hall effect Senor 49](#_Toc373613450)

[Figure 16: A general layout of Photo-Variable circuits. 49](#_Toc373613451)

[Figure 17: A simple example of a laser RPM sensor. 50](#_Toc373613452)

[Figure 18: Pin Information for OVS-33 Series SMD Super Bright LED 53](#_Toc373613453)

[Figure 19: Calculated Image Ratios 68](#_Toc373613454)

[Figure 20: Nearest Neighbor Scaling. 68](#_Toc373613455)

[Figure 21: The process of Trilinear Scaling. 70](#_Toc373613456)

[Figure 22: Radial Image Conversion 71](#_Toc373613457)

[Figure 23: Alternating Column Technique 75](#_Toc373613458)

[Figure 24: High-level hardware block diagram 77](#_Toc373613459)

[Figure 25: PCB Layout 1. 79](#_Toc373613460)

[Figure 26: PCB Layout 2 80](#_Toc373613461)

[Figure 27: Schematic for the MSP430 Launchpad (handmade) 81](#_Toc373613462)

[Figure 28: Schematic for the TLC5940 LED controller 82](#_Toc373613463)

[Figure 29: A prototype of the circuit for the Hall RPM sensor. 83](#_Toc373613464)

[Figure 30: Showing the general power block diagram for the entire project. 84](#_Toc373613465)

[Figure 31: The prototype circuit for the voltage regulators used in the project. 85](#_Toc373613466)

[Figure 32: The prototype circuit for the AC to DC rectifier. 86](#_Toc373613467)

[Figure 33: Battery Step Down Schematic 87](#_Toc373613468)

[Figure 34: Switching circuit 1 88](#_Toc373613469)

[Figure 35: Switching circuit 2 89](#_Toc373613470)

[Figure 36: Brush connected diagram #1 92](#_Toc373613471)

[Figure 37: Brush Connector Design #2 93](#_Toc373613472)

[Figure 38: Finalized Brush Connector 94](#_Toc373613473)

[Figure 39: Handlebard Mounting 95](#_Toc373613474)

[Figure 40: Hall Effect RPM Sensor Mounting 95](#_Toc373613475)

[Figure 41: Battery Positioning Diagram 95](#_Toc373613476)

[Figure 42: High-Level Software Architecture Diagram 96](#_Toc373613477)

[Figure 43: Text Conversion Diagram 97](#_Toc373613478)

[Figure 44: Bluetooth Subsystem Block Diagram ..…………………………………98](#_Toc373613479)

[Figure 45: 7-Segment Display Schematic 99](#_Toc373613480)

[Figure 46: The Letter ‘A’ and The Hex values 103](#_Toc373613481)1

[Figure 47: Adjusting Message Position 104](#_Toc373613482)1

[Figure 48: Increasing Spacing Message Position.](#_Toc373613483) 101

[Figure 49: The Letter ‘M’, Increase Width](#_Toc373613484) 101

[Figure 50: GUI Actions diagram](#_Toc373613485) 103

[Figure 51: Final GUI Design 1](#_Toc373613486)04

[Figure 52: GUI Data Transfer Characterisitcs 1](#_Toc373613487)05

[Figure 53: GUI Protoype 1](#_Toc373613488)12

[Figure 54:Summary GUI 1](#_Toc373613488)13

[Figure 53: Project Calendar/Timeline 1](#_Toc373613488)20

**Table of Tables**

[Table 1:Microcontoller comparison 22](#_Toc373619504)

[Table 2: Table LCCompare 1.0 27](#_Toc373619505)

[Table 3: Speed to Charging Electric Current 28](#_Toc373619506)

[Table 4: Showing a comparison of a few possible main batteries for the project. 37](#_Toc373619507)

[Table 5: A comparison of a few possible secondary batteries for the project. 38](#_Toc373619508)

[Table 6: Left: Wi-Fi Infrastructure Mode. Right: Wi-Fi Ad-Hoc mode 41](#_Toc373619509)

[Table 7: GSM data transmission 42](#_Toc373619510)

[Table 8: Comparison of several different RPM sensing methods. 48](#_Toc373619511)

[Table 9: Comparison between iPhone and Android 59](#_Toc373619512)

[Table 10: Comparing programming languages 62](#_Toc373619513)

[Table 11: Comparing phone and computer as user interface 63](#_Toc373619514)

[Table 12: Estimated Needed Voltage/Current Inputs 87](#_Toc373619515)

[Table 13: Budget 118](#_Toc373619516)

[Table 14:Project Milestones 121](#_Toc373619517)

# Executive Summary

The ability of the human eye to retain an image for fractions of a second after the image is presented to it is known as Persistence of Vision. While the exact causes of it are still disputed among researchers, this apparent phenomenon has motivated and inspired the engineering of many creative technologies. Particularly, many devices utilize rapidly moving lights to create the illusion of solid lines and shapes that can be used to form a clear image in the eyes of the viewer.

In this project, the creation and manipulation of light-based images is accomplished by spinning LEDs on the wheels of a bicycle. A user interface will allow a person to upload an image of their choosing, to be displayed on the wheel.

The bicycle wheel will be equipped with a wireless receiver, as the bicycle will receive the image via wireless communication. When the image is received, a microcontroller on the bicycle will direct the activation of the appropriate LEDs with the colors that the image contains. The human eye will recreate the image to the viewer, by observing the LEDs moving at a fast rate when the bicycle wheels are spinning.

The intended use of this bicycle is for advertising. As advertisers are always seeking new ways to promote their products and services, there are few ideas more creative than to have an LED display on a bicycle wheel spinning out their logo or company slogan. Having the bicycle receive images wirelessly contributes to the practicality of this idea, as an advertiser would have the freedom to upload and change images as they pleased without any manual manipulation of the bicycle itself.

This Persistence of Vision Bicycle display is designed and developed by electrical and computer engineering students at the University of Central Florida, as a Senior Design project. Each member of the team has contributed to the research and design of this bicycle, and the final project will be built by all team members.

This project is sponsored in part by Boeing. Any extra costs beyond Boeing’s contribution are self-funded.

# About the Sponsor

As the world’s largest Aerospace company and manufacturer of commercial jetliners, Boeing is a leader in engineering and innovation. Along with their development of commercial and military aircraft, Boeing’s products and services include satellites, weapons, electronic and defense systems, launch systems, communication systems as well as many others. Boeing employs over 170,000 people in the United States, and has customers in 150 countries. (Boeing)

Boeing has been a long-time supporter of engineering education and provides a great deal of support to students at the University of Central Florida. Each year, Boeing provides over $30,000 worth of scholarships to engineering students at UCF through the Boeing Scholarship Fund. Boeing also encourages the creative thinking and development of engineering students by providing sponsorship for several Senior Design projects.

Our team had the opportunity to apply for funding from Boeing through our university by offering them a proposal with our project objectives and budget.

Boeing’s funding will cover the majority of the expenses of this project, and any additional costs will be self-funded. Their contribution is indispensable, and we are extremely grateful for their generosity.

# Project Description

In this section we provide an overview of what the project intends to be.

## Objectives

The primary use case for this project is for advertising purposes. The Persistence of Vision bike can be ridden in public areas while displaying specific logos and information about local businesses. Due to its bright and visible nature, the display naturally draws the eye, and has excellent viewing angles. The device’s objectives are primarily related to optimizing it for mobility, endurance, range, compatibility, and ease of use. A potential advertiser could sell time slots on the bike, and wirelessly upload images to it. The final product implementation should appear seamless, and users should not need to understand how any functions work. The main objectives are described below.

### Mobility

The primary aspect of this device is its mobility. The rider must be able to approach speeds normally obtainable on an average bicycle without feeling excessively encumbered by the addition of the display unit. Therefore, the entire assembly, including batteries and the generator, must be very light. Further the POV display units must be relatively self-contained to prevent loose cables from interfering with the bicycle’s moving parts. The electronics on the bike also need to be able to withstand the conditions of this environment. They should be relatively resistant to the natural vibrations caused by the road, and damage from splashing water. Lastly, the drag created by spinning the generator on the bike’s peddles must not be extreme enough to make the bike difficult to peddle.

### Endurance

It is extremely important that the bike has enough endurance to last through a reasonably long shift. It was agreed upon by the group that the display must be able to light up continuously under a combination of battery and rider-generated power for a bare minimum of 4 hours. This allows enough for a rider to have a ride the bike for reasonably long shift and ensures that a large amount of people see the display for advertising purposes. For this reason, power consumption was an important factor in choosing all of the parts picked for use on the bike.

### Range

This section does not relate to the vehicle’s range. It is related to the transmission range of the wireless subsystem. The final products should to allow the end user to upload images to the bicycle remotely using a computer or smartphone. The range should be great enough to allow this to be done from inside an office, to the bicycle outside. This is so the bike does not need to be wheeled inside every time the image needs to be changed. This saves the advertiser time, and will help to maximize displays exposure to the public.

### Compatibility

It is important that the device and the corresponding image processing software are compatible with a variety of devices. The image software should be able to downscale images from a variety of formats for display. In order to ensure compatibility it was decided that the device should use a Bluetooth Module for receiving data. This allows the display to communicate with virtually any device using a standardized interface. The more computationally intense part, image ad text processing, will be done using client side software. A special packet containing all of the information about what to display on the bike will be sent over the Bluetooth connection to the microcontroller. Dividing the computational labor his way allows a less powerful microcontroller to be used onboard the device.

## Project Requirements and Specifications

The purpose of this section is to present the overall functions and features of the project. This section will also describe how each component should be able to perform in order to exemplify the features of the project. The top components and systems best matching the required design needs and project needs will be compared and vetted to single out the best match. This evaluation and selection will be performed in later sections; this section is mainly to outline the specifications and requirements of the design.

### Hardware Requirements

The hardware component is the body of the POV display. This section includes all components which require power. In order to have a successful project, it is imperative that the hardware is performing appropriately and at a high performance rate. In order to achieve this, we must first establish what is expected and needed from each hardware component.

### Micro-controller

The project and design in general is comprised of several different components, and so needs a ‘brain’ to control and seamlessly integrate all the parts of the project. The ‘Brain’ of the project would need to be a microcontroller. The microcontroller would need to be able to handle all the processing and overall functioning of the POV design.

Among the list of needs of the microcontroller, the most important ones is the need to control the output timing and data to the LEDS, to give the proper image. In order for the POV display to work at all, the image that is transmitted to the microcontroller must be delivered to the LED controller at the same rate at which the bicycle wheel turns. As such, the microcontroller also needs to be able calculate the speed of the bicycle’s wheel (rotational speed) and accordingly adjusted the timing of the LEDs. The microcontroller should also display back to the rider the speed and distance traveled by the user in order to maintain user awareness of the performance of the bicycle and hardware. In the same respect, the microcontroller would need to calculate voltage levels of the battery and display it back to the user so that they could be aware of the batteries level and if it needs to charge.

The microcontroller’s specification should allow it to have enough storage space to hold multiple images and animations as well as have enough I/O pins to be able to support all the devices which we would need to connect to it. Overall this component is one of the most important pieces, if not the most important piece of the project; it is the central hub for all data and would determine if the project would fail or succeed.

It is by these specifications that we will choose among several microcontrollers. It is important to ensure that whichever microcontroller we choose is able to sustain the data and power to execute these tasks effectively and up to standard.

### LED array

The main idea behind the ‘Persistence of Vision’ (POV) concept incorporates that whatever image is displayed needs to be visible to the observer. With this in mind, the best display would incorporate bright LEDs with emitted light from the visible wave length of the electromagnetic spectrum. The LEDs would also need to display multiple colors or have multiple LEDs within the array.

The viewing angle of the LEDS must also as wide as possible to display the images from the widest angles from the display. The array should be oriented on the bicycle to provide best display as the wheels spin and also the array should be laid out in a pattern which most efficiently can be coded to display the desired message.

Additionally, the cost per LED must be low to allow for the purchase of as many LEDs to effectively complete the display. The LEDs need to be small enough to fit several LEDs in a desired space.

Fortunately, LEDs are typically of low cost and come in several different varieties, which provides us with flexibility in ensuring that the LEDs we choose for our final design are the best for our needs. The LED types in consideration are discussed later in this document.

### LED controller

At the core of the project, the LEDs are the essential components that are needed and used to produce images and animations. However, in reality we will not be simply using LEDs, but more specifically, be controlling the LEDs. To control the display we will be turning specific LEDs off and on at the right intervals and speeds, and we will be adjusting the brightness and duty cycles of different LEDs to produce different colors. We will be doing this all at an extremely fast rate. To accomplish this level of control we will need to employ the use of LED controllers. The LED controllers would need to effectively handle the data flow and power flow to all the individual LEDs. To this, they would need to have a lot of output pins to handle the large number of LEDS which would be used. Additionally, the controllers must be small enough to fit on the available space and also draw as little power to them but still be able to maintain the flow of power to all the LEDs.

### RPM sensor

The sensing of the rotational speed of the wheels, and by effect the speed and distance traveled by the bicycle, is required to not only display back to the rider, but also in the calculations of the timing, duty cycle and overall functioning of the LED POV display

The speed of rotation of the wheel allows for the precise timing of the flashing of the LEDs, giving the correctly displayed image. The precise measure of the timing of the wheels rotation is undoubtedly one of the most important pieces in the design and needs to be captured using a method with high accuracy and precision. . This is why an RPM sensor is a needed; in order to give us this speed information.

Additionally, while precision and accuracy is needed in the design of this component the need for small, lightweight and low cost cannot be neglected. We cannot have an RPM sensor that weighs down the bicycle or that takes too much space, since space and weight should be reserved for larger components like the PCB.

Thus, the requirements of the RPM sensor including something that is accurate and precise, but is also lightweight and inexpensive. We will explore a few options throughout the document, but these are the preliminary requirements.

### Display

The bicycle handlebars will contain a 7-segment display. The display simply needs to display to the rider three different measurements. The first measurement would be the speed of the bicycle. The second is the distance travelled. The third is the battery level of the one board batteries. The need for a simple, low power, low weight, and large enough display (so the rider can see while riding) is ideal for this job. The data for the speed and distance travelled would come from the RPM sensors readings, and the battery level would come from the measured current coming from the battery. These measurements would be passed from their respective inputs through the micro- processor then from the micro-processor to the display.

### Wireless

The entire project is intended to be mounted on a bicycle which for the most part will operate as a closed system. In that, once a specific amount of display images and animations have been uploaded to the micro controller there is no real way of changing them without stopping and programming new images to the bicycle. This restricts the project functionality, causing the purpose of design to be less than we intended for the project (a real-time advertisement billboard).

To change this, the incorporation of a wireless means of uploading new images to the bicycle is needed. To effectively complete this task the wireless transmission method would need to be simple, fast, have a suitable bandwidth and used as little power as possible. Additionally, the wireless hardware components must be small enough to fit onto the PCB easily and occupy as little space as possible.

In the later sections of this document we will provide an overview of the options available to us for wireless transmission. These components will be analyzed in the context of these requirements

### Power

All of the bicycle hardware components require power, and it is therefore necessary to establish the needed power elements and their requirements.

#### Battery

The project, being a closed and mobile system, requires a power source to operate all the components and features of the design. The solution to this power requirement is the installation of a battery. For this project, we will actually require two batteries; one will act as the main power source and the other as a reserve, as well as a receiver for pedal-generated power (this generated power will be covered shortly). The main battery should have enough power to supply power for at least a few hours, and the secondary battery for at least an hour. The batteries should be able to supply a suitable rated voltage, current and power to all components.

It is by these standards that we will choose batteries, and it also by these standard that we will test our batteries to ensure that they operate to the full needed performance.

#### AC adapter

When the user is not riding the bicycle and the batteries are low, the bicycle must have a way of recharging the batteries for the next time the rider wishes to use the bicycle and all its components. The solution to this is that the bicycle should have built in power adapter to draw power from an AC outlet. The adapter should convert the AC to DC and recharge the battery. The adapter needs to be small and light weight, taking up as little space but proving safe converted DC power to the battery to recharge them.

Additionally, the AC adapter must be able to charge the bicycle components at a reasonable rate. We expect that it should take no more than a few hours to charge the battery fully. The design and testing of the AC-to-DC converter is discussed later in this document.

#### Generator

While riding the bicycle for a long period of time, the use of all the features and components would eventually deplete the battery. To counteract this problem, the bicycle will be fitted with a secondary battery, which would only be activated when the primary battery is no longer able to power effectively. This secondary battery is unique in that it is charged by means of a pedal-powered DC generator.

The generator needs to be small enough to be easily mounted to the bicycle, but be able to produce enough power to regenerate the battery. The generator should be hook up not to the main battery but to the secondary battery as a sort of buffer to the main battery. The generator does not need to provide enough power to fully operate all of the components in real time, but it should provide enough power for the secondary battery to be able to sustain all of the hardware components until the primary battery is able to be recharged.

### Printed Circuit Board (PCB)

This component will be very important; the project’s success depends on the design and functioning of the printed circuit board. This component would house most of the components circuitry or at least act as the common point of data transmission and reception between all components. The PCB must be as small as possible in order to be effectively mounted to the bicycle wheel (while supporting all the designed components and connections). The PCB must have low internal resistance to reduce any power losses while transferring the full current and voltages to their respective components. The size of the PCB must allow it to fit into any space or locate it where ever it is needed.

### Extended features

In addition to the primary POV display, there are a few features we plan on including in the final design of the bicycle. One of those features which we decided to implement was an onboard phone charger. The charger would draw power from the battery or it would be hooked up directly to the generator. The battery should be able to provide enough power to charge the common 5V batteries found in many modern phones. The charger would need a suitable DC voltage and current from the battery, so a voltage regulator circuit would be appropriate to integrate this feature. This voltage regulator is described in later sections of the document.

### Housing and mounting

Effective housing and mounting of all components onto the bicycle is an important requirement because it ensures that the project is effectively operational. All the mounting and housings would need to be designed to specific needs. The first of these needs would be that all the parts needed to be small enough to fit on the bicycle without hindering the rider. No parts should be sticking out or adversely affecting the ride of the bike.

Second, the parts need to be balanced on the bicycle; no components should be heavy and protruding to the sides, offsetting the natural balance of the bicycle, or causing the bicycle to be unstable as the rider is operating it.

The moving parts of the bicycle can also be a hazard to the sensitive and delicate electronics components. Therefore, stable and reliable mounting is required. It is important for us to ensure that none of the hardware elements of the POV display are shaking or moving as the bicycle is operating.

Additionally, the running of wires needs to be flushed along the bicycle to prevent tangles with the moving parts as well as the rider. It is imperative to ensure that while the rider is operating the bicycle, that none of the wires become entangled in their feet or in any of the hardware components of the display. This will not only damage the display, but will also be a safety hazard to the rider.

One issue that will require special attention is the lack of space on the wheel well. This is particularly troublesome since many of the components of the POV display are quite large and require a great amount of wiring. It will therefore be important to design the bicycle and the wiring such that any wiring does not become entangled around itself when the bicycle wheel is spinning. It is also important to design the bicycle so that the rotating components on the bicycle are able to be attached to the stationary components without trouble.

Finally the bicycle is going to be exposed to all kinds of environmental factors. Aside from the changes in Florida weather such as heat, humidity, cold, and rain, we must account for debris like rocks and pebbles which may hit the components as the rider is traveling across different terrain and daily environments. So proper housing and PCB design is needed in ensuring that all components are durable enough to stand up to the weather and terrain or at least be water and impact proof.

### Software Requirements

This section will describe the requirements of the behind-the-scenes software work that makes the POV display possible. Having software that works to its highest performance is crucial, and will ultimately make or break the project. The proper processing of images, the transmission of those images, and the displaying of the images on the proper LEDs depends on the software component. The software requirements are described below.

#### Performance

For this project, top performance is key to providing the user a simple and pleasant experience using the display. The text display software must quickly convert text for display prior to transmission. The entire conversion should take no more than a couple of minutes using a basic Windows computer that would be found in an office setting. Once the text is converted, there should only be a minimal delay between when an the message is transmitted from the software, and when it appears on the display. Under normal conditions, this delay should be no more than thirty seconds. Once the input is displayed, the LEDs should be timed in such a way that the produced image remains relatively stable. Lastly, the handle-bar display should respond to user input (pressing the switch) by instantly changing modes. This level of performance is the key to providing the user a seamless experience while using the system.

#### Accuracy

The next important software requirement for this project is accuracy during every single stage during the process of using the display.

During the first stage of text conversion the software must produce an image that is at the very least recognizable compared to the original. Some form of pixel averaging must be used during this stage to ensure accurate color values. Next, the Bluetooth system must send the finalized image to the display without any loss in accuracy. That is to say all color values received by the microcontroller must exactly match those sent by the client software. The image should then be accurately rendered on the POV display, with color values closely approximating those in the received image arrays.

It is of equal importance that all types of information displayed by the handle bar box are numerically accurate to within a few percent of true. A Hall Effect sensor on the frame of the bike will be used to calculate distance. The front wheel is chosen because it is generally known to slip less than the back wheel, which makes it more accurate. If any one of these features is inaccurate, the system’s usefulness dramatically diminishes.

#### Stability

At the final development stage, it is vitally important that all software is free from any catastrophic bugs. All subsystem programs should run without any crashes, halts, or infinite loops. This is of particular importance at the microcontroller level, since troubleshooting the display without access to a computer is nearly impossible. All software will be thoroughly tested to help prevent any kind of malfunction because all bugs that cause the system to crash or stall in anyway will negatively impact the user’s experience.

#### Ease-of-Use

Lastly, it is important that an average nontechnical user can display on the device quickly and easily. The step-by-step process of converting a message for use on the bike’s display must be as intuitive as possible. For simplicity, the desktop software interface will be very stripped down. A text box will allow the user to type an image. Three sliders will allow the user to adjust the message’s color and brightness. The final button, “upload”, will wirelessly transmit the newly converted message to the display’s microcontroller, where it should appear. This process should make sense to the user with very little extra training required.

## Equipment

In order to save funds, the bike being used for this project is an old leftover bike from a group member’s garage. The bike is a men’s Huffy with rims that are 24 inches in diameter. It is a twenty-one speed bike with front and rear hand brakes. Images showing the eventual location of all new hardware will be shown. A profile image of the bike is shown in Figure 1.



Figure 1: The bicycle that will be used in this project

The entire POV display will be mounted in the rear wheel, consisting of three PCBs. Each PCB will contain three LED controllers and a bank of sixteen LEDs. This location was chosen because it allows for hand peddling in the lab during testing. It also provides more attachment points for various wires.

On the left rear fork the generator will be mounted to the frame. This location gets it out of the way of most of the bike’s moving parts. Additionally a separate Hall Effect sensor, mounted to the frame near the rear wheel will feed information about the wheels rotation to the handle bar box. The rear wheel is shown in figure 2.



Figure 2: Rear wheel of the bicycle

Lastly, the 7-segment display, secondary micro-controller, and phone charger will be mounted on the handle bars. This location allows the user to easily view the display and have access to the phone charger. The handle bars are shown below in Figure 3:



Figure 3: Bicycle Handle Bars

## Specifications

This section describes the final specifications of the POV display bicycle. This gives a general view into what comprises the POV display and what is on the bicycle overall. The specifications below were based on estimations, research and assumptions of the components that we needed to complete the project.

* 48 RGB LEDs
* 6 watt generator
* 2 X 7.4V Lithium ion batteries
* 4 Hall effect sensors
* Total weight of components: 8-10 lbs.
* Charges using 120V AC
* USB input via computer
* 128 kilobits of storage space for images and animations.
* Bluetooth wireless adaptor
* 4 voltage regulators(3V, 5V)
* 20-30 resistors of varying resistances
* 20-30 capacitors of varying capacitance
* 20-30 diodes (regular, zener, Schottky)

## Project constraints

As with all projects and all designs, the development of the POV bicycle display has limitations and constraints. This section deals with the limitation and constraints that the project is faced with on a design level.

The nature of the project revolves around everything operating in a closed system. This is not a stationary display; the bicycle will be in motion and used as a regular bicycle. This requires that the workspace or environment where we design our project around be versatile enough to handle everything the bicycle can, while also not hindering the movement of the bicycle itself. This is one constraint the project is faced with.

Of all the constraints the biggest one is that of bicycle. More specifically that the workspace of the project is relatively small. There is limited space as to where and how we can attach things to the bicycle. Parts and components need to be certain dimensions to not only fit on the bicycle but also to avoid interfering with the moving parts of the bicycle and the rider; it also necessary to maintain the balance of the bicycle. The project aims to minimize the presence of the project components to the rider and continuously present a bike ride not different in any way from that of a normal bicycle ride.

Bicycles today travel over many different types of terrain, and deal with a lot influence from the elements. This affects the types of material we need to complete the POV project. The materials, components and overall design need to be able to withstand shaking, heat, water and many more environmental influences.

Another big constraint deals with the fact that the operating environment of the components is one that has many moving parts. The POV display can be restricted to just parts which don’t move, or parts which move constantly but this restricts space available to place features and components. And so we are presented with the problem of components located in moving parts interfering with those located on or in relatively stationary parts. This constraint is a big one and so we need to develop with ways of attaching and interfacing these components. And generally we need a way of effectively, easily and yet securely attaching all of these parts to the bicycle.

# Existing Projects

Persistence of Vision has endless creative applications; as such, it has inspired the design and creation of many projects among engineering students and engineers alike. While the final design of our POV display will be entirely original, the concept of a POV display and even a POV display on a bicycle is not unique. It is therefore important for us to give credit to the sources which have inspired the creation of this project.

The following section outlines some existing projects related to our Persistence of Vision bicycle, in concept and application. These projects are used as a foundation for understanding the challenges of designing and building our POV bicycle, and for providing meaningful references for exploring and comparing parts and approaches.

It is important to note that while these projects are referenced for inspiration for our final design, no component of these projects is borrowed directly for our use. Additionally, all borrowed ideas are given due credit as they are cited throughout this document.

## Cornell University Student Final Project Persistence of Vision Display

Electrical Engineering students at Cornell University created a Persistence of Vision Display as their Final Project, which has some overlapping characteristics with what our project design intends to be. The Cornell team designed a display that allowed users to upload an image, which would be wirelessly transmitted to a receiver on the display. The display was a cylindrical model which spun a column of LED’s around a central motor shaft (A Bodine Electric Company AC motor was used), and would light up the appropriate LEDs in order to produce a two-dimensional image. (Cornell)

This project consisted of two components: the spinning LED display, and a stationary system which acted as a user interface. The user interface allowed the user to upload their image, and select functions for the spinning display such as clearing the display or performing certain animations. (Cornell) Our user interface will not be attached to the display as in this project.

The wireless component of this project is a particularly significant resource for us, as our Persistence of Vision Display relies on the wireless transmission of images, and this team may provide us with insight on how to best execute or utilize wireless components. Although the Cornell team used Wi-Fi, while we are using Bluetooth, their testing methods may be of use to us.

The Cornell project, similar to ours, has a hardware component, and a software component. For the main design, 14 RGB LEDs were mounted to a Plexiglas support arm. The chosen microcontroller for this project was an ATmega644 on the display, as well as an identical microcontroller on the stationary board5 MAX 6966 controller chips acted as slaves to the master, the microcontroller. The microcontroller communicates to the slave chips via an SPI bus, and sends commands to write to registers on the chips. The chips receive 4-bit values that correspond to a pulse-width modulation duty cycle which outputs on a port connected to a target LED. The onboard microcontroller calculated rotations per minute at each rotation of the arm in order to adjust the LED display depending on the speed of the arm. The device was powered by a 9V VCC on the microcontroller. While we do not plan on using the same microcontroller, we will still look to the Cornell’s team for ideas in execution of microcontroller tasks. Particularly, the fact that they have used an RPM sensor to determine the speed of the rotating display is of relevance to our project. One of the greatest challenges of this project will be ensuring that the microcontroller instructs the LED controllers to act appropriately in relation to the speed of the bicycle wheel. Therefore, having a reference for how another team accomplished this task is extremely useful to us.

Furthermore, the software component of the Cornell project had three main tasks: 1).To calculate the RPM at every revolution 2). To write the data to the appropriate slave chip via the SPI 3).To run the different animations of the POV display. Again, this is similar to the operation of our software component. Our software tasks will include calculating the RPM of the bicycle wheel, and using this information to communicate to the LED controllers which LEDs should be activated and when.

A notable fact about the Cornell team’s Persistence of Vision project is that all hardware and software components were completely original and built from scratch. They did not use any patented or copyrighted material in their design. No part of our project will contain any materials developed by the Cornell team, however, we plan to use their project for inspiration and as a reference in developing our Persistence of Vision Bicycle. (Cornell)

## University of Central Florida Senior Design Persistence of Vision Display

A group at the University of Central Florida built a Persistence of Vision display as their Senior Design project in fall 2012. The cylindrical display was built as a portable chassis with two motor-driven LED arrays mounted to it. The UCF group’s project was similar to the Cornell group’s project in that it had a user interface which allowed a user to upload an image that would be wirelessly transmitted and displayed through the LEDs using persistence of vision.

The UCF team used two microcontrollers: one was stationary, and used for image processing purposes, while the other rotated with the POV display and controlled the LEDs. The chosen microcontroller for this project was the PIC32 by Microchip. The UCF group’s persistence of vision display was successful in meeting its design objectives, and is highly referenced in the execution of our design. Although we do not plan on using the same microcontrollers, nor do we plan on building our display in a similar setup as theirs, this team’s approach is still of use to us for inspiration and for reference in things such as image processing algorithms.

Additionally, since this team is a former team from our university, we have access to information on the materials they used, and even have access to viewing their display in person for reference. Their final design output is shown in Figure 5, (Al, 2012)



Figure 3: UCF POV Senior Design Final Result. Printed with Permission from UCF Senior Design Group 8 (UCF)

## Texas Instruments Intern Design Project “Spoke Ink” Persistence of Vision Bicycle Wheel

In 2012, Texas Instruments held a design competition in which they challenged their interns to create a project based on one of TI’s microcontrollers. A team called “Spoke Ink” built a persistence of vision bicycle wheel, similar to our project. Unlike our project, however, their bicycle was stationary and upside down, and the wheel was spun by hand, rather than by being ridden as an ordinary bicycle. As their project was built over the course of a summer internship, it was also not as intricate as our design.

The team’s design used the MSP430 Launchpad microcontroller by TI, and featured: an array of 96 RGB LEDs, three wireless, batter-powered PCBs, and three Hall Effect magnetic sensors to keep track of the wheel’s speed and to adjust the display accordingly. Spoke Ink’s project is of significance mainly for its use of a bicycle wheel similar to our project, as well as its use of the MSP430 microcontroller, which was strongly considered for our project.

Furthermore, Spoke Ink used Hall Effect sensors to calculate the bicycle RPM and report that information back to the microcontroller. This is very similar to our design, as we are also using Hall Effect sensors in connection with the MSP430 Launchpad.

This team mounted their MSP 430, as well as the LED arrays directly onto the PCB. Since this is likely what we will be doing, we will reference their images for ideas on how to arrange our PCB layout. (Texas Instruments)

Spoke Ink released much of the data for their design, including a mini user manual on how to recreate their project. While this manual is not very specific, it gives us an idea on how they developed their persistence of vision display, and is open for reference when we are building our display.

# Hardware Research

The following section describes the research behind one of the major component of the Persistence of Vision Bicycle: the hardware component. Throughout this section, each piece of hardware utilized in the design of our POV display is described with its purpose in relation to the design. Additionally, for each piece of hardware we provide a comparative analysis on the different parts that were considered to meet the requirements of the final product. Finally, we provide justification for the selected parts.

## Microcontrollers

The following will discuss how the microcontroller section will be organized. First, it will be discussed as to why our project merits the use of a microcontroller. Then, a paragraph will discuss why a particular microcontroller should even be considered. Then, the microcontroller’s specific feature set will be discussed and how some features could affect the results of the project. After all the microcontrollers have been assessed, a section will be dedicated to comparing each of the microcontrollers to one another. Refer to Table 1 for all the specs of each microcontroller. This table is what will ultimately decide which microcontroller we select.

Imagine having 15 LEDs, one by one, right next to each other. Let’s assume the goal is to turn on LED 0, then turn on LED 1 and turn LED 0 back off, and so on. So each LED would be on one at a time, in a snake-like manner. To do this without a microcontroller would require a lot of digital logic that is complex and not easy to change. Our project is doing far more than flashing LEDs one at a time. Therefore, a microcontroller is absolutely needed.

### MSP430g2553IN20

The MSP430 is made by Texas Instruments, which is a reputable company. It was important for us to choose parts from reputable brands, since it somehow ensures a sense of reliability. The particular model that is being considered is the MSP430g2553IN20. This microcontroller is discussed first because it is the one that is taught in the EEL4742 Embedded Systems Class at University of Central Florida. Thus, Group 29 has the most experience with it.

Like many microcontrollers, their feature sets are very broad, and not all of the features provided will be relevant to our project. Therefore, the feature set will be reduced somewhat, but not completely. Since this is research, the use of some features may not become apparent until the full design of the project is complete, or unfortunately even after prototyping has begun. Features will be filtered in combination with educated guesses and based on the works of similar projects.

#### Features

The MSP430g2553IN2 has a maximum clock speed of 16MHz. The frequency can be altered by adjusting the proper registers. This is not the ideal speed for a POV display, but is sufficient for the number of LEDS we intend to use. The number of I/O pins that it has is 16. This may seem like a low number of I/O pins compared to many other microcontrollers out there, however, it is surely sufficient. This microcontroller is fairly inexpensive too; after visiting various vendors it seems to cost less than $3.00. An advantage to the MSP430 is that it has a wide and low voltage supply range. It can support a voltage between 1.8V and 3.6V. This is good because simple batteries could power it without any trouble. The amount of memory that the chip has is pretty important for our project for both still images and animations. Below is the calculation for how much memory we actually need. (Texas Instruments)

#### Calculation

In this section we will try to calculate how much memory the microcontroller would need to perform an animation for 10 seconds.

Let us assume we need 25 frames per second in order to get a clear image. 25 frames multiplied by 10 seconds equals 250 total frames. That is the equivalent of 250 single images in memory. Now we will calculate how much memory a single image takes up.

First of all, if we are using 16 LEDs, that may end up being equivalent to 32 by 32 pixels which equals 1024 pixels. Each pixel needs to store an RGB value. Each of the colors are 12 bit values. 12 bits x 3 is 36 bits for a single pixel. From this point there are a couple of ways of determining how many bytes are required for storing one pixel.

There are three possible answers 4.5 bytes, 5, bytes, or 6 bytes. To use 4.5 bytes per pixel, this would be rather annoying in code; a single byte would contain data on adjacent RGB LEDs. A similar problem occurs when trying to pack the bits into 5 bytes. Each of the RGB colors is 12 bits, so rounding 36 bits to 40 bits, would mean adjacent RGB LEDs no longer share data. However the individual colors themselves share data among bytes. The easiest approach to handle in code would be to have each color stored in its own secluded set of bytes. This would mean 2 bytes per color, which is 6 bytes per pixel. This is the value that will be used in the rest of the calculation. 1024 pixels x 48 bits is 49152 bits for a single image. That’s 6144 bytes, which is 6 kilobytes just for a still image. For a full 10 seconds of animation 6 KB x 250 equals 1500 KB.

Essentially what it comes down to is that our microcontroller needs more than 6 KB of memory, which this particular MSP430 does have. The .5KB of ram is on the low side of memory, but the 16KB of flash is plenty to store an image. From the previous calculation to store all 10 seconds of a clip, at least 1500 KB is required. Storing all of the information all at once is probably unnecessary; in software the data for the animation could be handled in a streaming video type manner. This topic will be covered in more detail in the software section of this document. It is clear through that we are not going to use a microcontroller that has 1500 KB of memory.

### PIC32MX340F512H

The PIC32 is made by Microchip. This chip is relevant to discuss for a couple reasons. At the University of Central Florida, another senior design group used this microcontroller in their Persistence of Vision display and they were very successful.

We first look to an overview of this microcontroller’s features. Firstly, its maximum frequency is 80 MHz which is far more than necessary to make a POV display. The number of I/O pins this controller has is far more than our group would ever need. In total there are 51. With 51 I/O pins, each RGB led could be hooked up individually with leftover I/O pins for sensors. It is clearly overkill. The voltage supply range is 2.3V to 3.6V, which is similar to the MSP430, but not quite as wide, although this microcontroller could just as easily be power by batteries. The total amount ram that it has is 32KB which is sufficient for displaying a 6KB image. There is also 256KB of flash that can be utilized. This microcontroller is double the cost of the MSP430 at under $6.00. More than likely this microcontroller will not be selected, although it would be a very good backup plan to keep this one in mind. (Microchip)

### Attiny2313

ATTiny2313 is made by Atmel. This microcontroller is being considered is because of the following. First, it is the microcontroller of choice in other POV projects; specifically some online build-your-own kits use it. After briefly going over the specs, it has been determined that it a good candidate for microcontroller selection.

This microcontroller’s maximum frequency is 20MHz which is sufficient. It has 18 general purpose I/O pins. It has 128 B of ram, which is quite small, and 2KB of flash which is also quite small. Its voltage supply range is 1.8V to 5.5V which can be powered by a battery. Overall this microcontroller is good but probably not the best candidate for our project. If it had at least 6KB of memory, then it may actually be worth using, but it does not. (Atmel)

### ATMega8515

The ATMega8515 is also made by Atmel. We are revisiting an Atmel product again because it was a disappointment that the first Atmel product we found wasn't good enough. This microcontroller seems to fit the specs that we are looking for a bit better.

The microcontroller has a required supply voltage range of 2.7V to 5.5V. Its maximum frequency is identical to the ATtiny2313 at 16 MHz. It has far more general purpose I/O pins than the ATtiny; a total of 35 compared to 18, which is more than enough. It has slightly more ram, at .5KB. It has 8KB of flash, which is cutting it too close to our calculated values to be comfortable with. Its total cost is less than $7.00, which is expensive compared to the others. So basically its cost per getting the job done ratio is far too high. (Atmel)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Microcontroller | Maximum Frequency | General Purpose I/O | Ram | Flash | Cost |
| MSP430g2553IN20 | 16 MHz | 16 | 0.5 KB | 16KB | < $3.00 |
| PIC32MX340F512H | 80 MHz | 51 | 32 KB | 256 KB | < $6.00 |
| ATtiny2313 | 20 MHz | 18 | 128 B | 2KB | < $3.00 |
| ATMega8515 | 16 MHz | 35 | 0.5KB | 8KB | < $7.00 |
| Minimum Required: | 16 MHz | 8 | .5 KB | 6 KB | < $50.00 |

Table 1: THIS table compares all the microcontrollers that have been discussed in this section.

From Table 1, we will determine which microcontroller will be selected for use in our project.Right off the bat it is quite clear the ATtiny2313 microcontroller will not work, it does not have enough memory. It could be made to work if external memory is purchased, but there are other candidates that don't require that, so that will not happen. The ATMega8515 is very similar to the ATTiny, however it happens to be too expensive. The PIC32MX340F512H on the other hand, is definitely qualified for the job. It's probably even too qualified. We're not trying to control anything other than LEDs, not any complicated robot. Not only that, the price is kind of on the high side. The MSP430g2553IN2 seems to be the closest to what we are looking for, there is uncertainty in how it will perform when it comes to animations, but it seems to have all of the proper stats without going too far over. Not only that, but it is the cheapest one. The decision will be finalized in the summary.

When it comes time to prototype there are some tricks that will make it a lot easier. Let's compare the MSP430 to the PIC32. Let us start by looking at Figure 6, it shows the actual processors on Launchpads.

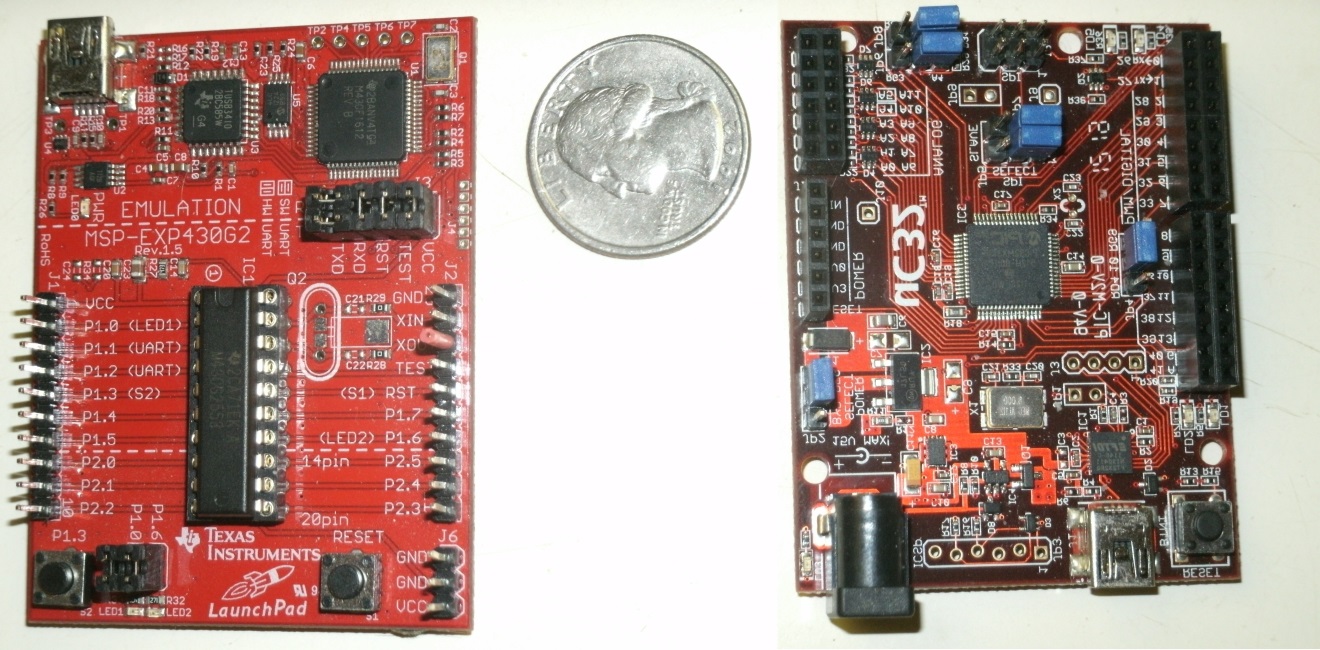


Figure 4:This diagram shows the MSP430 Launchpad and a PIC32 Launchpad. Photos taken and edited by Matthew Egan.

The point to these diagrams is to show that one of these Launchpads is better than the other, and it has nothing to do with the actual processor. If we use these Launchpads in our prototype we would have a much harder time with the MSP430 and here is why:

Let us assume we need to hook up a wire from our MSP430 Launchpad into a breadboard. First, what we would do is plug the first end of the wire into the breadboard. The other end of the wire would have nothing to plug into; that is a problem. The PIC32MX340F512H Launchpad however happens to have nifty breadboards already built into itself. This makes it extremely easy to prototype. The MSP430g2553IN2 Launchpad is not hopeless though. There are attachments that can be purchased and they plug right onto the pins of the MSP430 Launchpad. Refer to Figure 7 for an illustration. This is a cheap and effective way to make it work. A part such as this could be made by hand, or purchased for a few dollars.

What needs to be realized is that this part will be attached to the microcontroller which will be spinning pretty rapidly on the bike. We need to prevent this attachment from flying off if it happens to be on loosely. The best solution to this problem is still unknown, but at the moment one possible solution is to glue it on. The drawback of gluing it on is that if it was loose in the first place, that means the contact between the pin attachment and the pins themselves may not be reliable. If their connection was to break and it is already glued on, we may end up destroying the pins trying to take off the attachment. Another way to handle this is to still glue it, but melt solder in the holes, but it has risks of its own. If this is a legitimate problem, it will make testing extremely difficult, but then again, we have thought of it as a potential problem already, so maybe it will not be as painful.

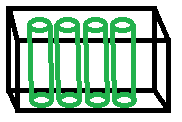


Figure 5: Shows how the MSP430 launch can be modified to make prototyping easier.

## LED Controllers

The LED driver—or controller—is an essential piece of the POV display. This is the component which takes in information from the microcontroller related to the image, and then translates this image to the LED array. This component is responsible for determining which LEDs are activated, and the final view of the display. Options for an LED driver are presented below.

This section will begin by discussing why we even need an LED controller in our POV project. Then, similarly to the microcontroller section, a particular LED controller will be justified as to why it should even be considered. The specific feature set of each LED controller will be discussed and may be briefly discussed. Lastly, all LED controllers in question will be compared and contrasted with one another, in Table 2 these features are what we will ultimately base our decision off of.

It has already been established why a microcontroller is needed. So now the use of LED controllers also needs to be justified. Let us assume we do not have any LED controllers, and that our microcontroller has eight general purpose I/O pins available. Unfortunately, with this situation, no more than 8 single colored LEDs may be hooked up at one time while only using a single microcontroller. Its true POV could be done without any LED controllers at all, but it is impractical. LED controllers do more than extend the number of outputs, but also allows for controlling the brightness of an LED in a simple manner. LED controllers are not a requirement to do POV, but they are a requirement to do POV in a smarter way.

### TLC5940NT

The TLC5940NTLED controller is made by Texas Instruments. Not only are they reputable with their microcontrollers, but also with their LED controllers. Similarly, this particular product has been used in money making devices. It has also been used in other POV projects and LED applications. If by coincidence we happen to choose the MSP430 microcontroller, the fact they are made by the same company may imply compatibility. However, this has not been confirmed. This also happens to be a part that one of Group 29’s members happens to have experience with. (Texas Instruments)

#### Features

Not all features will be listed, only those deemed relevant to the project. One of the most obvious features of an LED controller would be how many LEDs it can support. For this particular controller, it can handle 16 LEDs, which would equate to about 5 and 1/3 RGB LEDS. The 1/3 is legitimate because an RGB led could be hooked across multiple controllers; this implies that these LED controllers can be linked to one another.

The next most obvious feature of an LED controller would be the brightness level that it supports for the LEDs. This particular LED controller has dedicated 12 bits for handling the brightness, which gives 4096 possible brightness levels for a single LED. In an RGB LED, in order to create different colors the brightness of the 3 LEDS would be altered using this grayscale value. For example, if we wanted the output of an LED to be purple, we would have the green grayscale value far lower than the blue and red. This is controlled by software; all the details about the software will be covered in the software design section.

Another feature that is less obvious is if the LED controller is drive capable. What this means is that the LED controller can power the LEDs that it controls. This is a very convenient feature. Another less obvious feature that it has is dot correction. To understand how dot correction works, it must first be understood that LEDs are not all identical and they are not all perfect; no two LEDs are the same. For example, if we tell two LEDs to be at maximum brightness, in software, we may think, now they're the same. But they're actually not, what dot correction does is that it stores an offset of brightness for each LED to actually make the LEDs the same brightness. The last feature to discuss is the voltage required to power it, the voltage range is from 3V to 5.5V. As far as the cost goes it's fairly cheap. It costs less than $6.00. (Texas Instruments)

From this set of features, it is quite clear that this LED controller will be a strong candidate for selection. It has a lot of LED outputs, it has a couple convenient features for both powering and brightness control and it is very low cost.

### LT3746

The LT3746 LED controller is made by Linead Technology. It has a total of 32 LED outputs. That means it can control 10 2/3 RGB LEDs. The advantage of that is, it would only take two of these controllers for all of our LEDs. It costs less than $8.00, two of them would cost less than $16.00. As opposed to the TLC5940NT, which would require 3 at $6.00, making it $18.00 in total. Overall, it would be cheaper. The number of grayscale bits is identical to the TLC5940; it has 12 bits to control the brightness of the LEDs. As to whether this controller is drive capable is unknown. After reading the list of features and the data sheet, it does not mention it at all. It does have dot correction though, which may come in handy. Because of the uncertainty associated with some of the features, this is more than likely not the LED controller that we will pick. (Linead Technology)

### TLC 5947

The TLC5947 from Texas Instruments is a viable option for an LED driver chip. Its large number of input pins greatly helps with the powering of large numbers of LEDS, which makes it a perfect fit for the project. This chip comes with a constant current sink, as well as a 4096 PWM steps. The TLC5947 also has a special thermal shutdown (TSD) feature that basically turns off all the output drivers if the over-temperature condition is reached. The chip then restarts whenever the normal temperature is achieved again. This feature makes the chip very safe and reliable, and ensures that we do not lose a valuable component to overheating. A table comparing important specifications and data on this chip is located at the end of this section. (Texas Instruments)

### HT16D724

The HT16D724 is a 16 channel LED driver from the HOLTEK Company and is considered to be a reasonable choice for the operating and functioning of the LEDs for the project. The HT16D724 is specially designed for LED applications and boasts high accuracy constant current driver ability. The chip is meant to be connected with other chips of its type in a cascade manner and data be fed serially through all of them. As was previously said the chip has a 16 channel array of constant current LED drivers. Table 2 is a data sheet comparing all the LED drivers and their specifications. (HOLTEK)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Chip | TLC5940NT | LT3746 | TLC 5947 | HT16D724 |
| Manufacturer | Texas Instruments | Linear Technologies | Texas Instruments | HOLTEK |
| Price | <$6.00 | <$8.00 | $1.95/1000 Unit | <$3.00/ 10 units |
| No. of Channels | 16 | 32 | 24 | 16 |
| Input voltage | 3-5.5V | 6-55V | 3-5.5V | 3.3-5V |
| Data Input | Serial | Serial | Serial | Serial |
| Data Transfer rate | 30MHz | 30MHz | 30MHz | 25MHz |
| Ability to Cascade | Yes | Yes | Yes | Yes |
| Constant Current Sink | 0-120mA | 30mA | 30mA | 3-45mA |
| PWM Grayscale control | Yes  12-bit | Yes  12-bit | Yes  12-bit | No |
| DOT correction | Yes  6-bit | Yes  6-bit | No | No |
| Special features | Error Notification | Full Diagnostic and Protection | Thermal Shutdown | No |

Table 2: Shows the comparison of several LED controllers

## Generators

A main objective of the design of our Persistence of Vision Bicycle is to have a supplemental, backup power source for all power-consuming devices on the bike, and to have this supplemental power supplied by a pedal-activated generator.

The following diagram provides a visualization of this process and the generator’s purpose. The following sections provide an overview of the most viable options for a generator in the final design.

### bike2power Bottle Sidewall Bicycle Dynamo Power Generator

The Bottle Sidewall Bicycle Dynamo Power Generator by bike2power is an attractive option for a pedal-powered generator. This generator attaches to the front wheel, and is easily removable and adjustable which provides flexibility in experimenting with the mechanical design of the bicycle display. It generates up to 6V DC current, which we estimate should be sufficient to power a supplemental battery (more meticulous estimations would be made if it were the primary or single power source). Additionally, this generator weighs a mere 4 oz and is 3.5”x”2.5”x1”, which is relatively small and will not weigh down the bike. This generator is only $23, which is well within our budget. (bike2power)

### Human Creations Bicycle Dynamo USB Charger

The Human Creations Bicycle Dynamo USB Charger sold by Amazon is another option for a pedal-powered generator. While this generator is more expensive than the bike2power generator at $56.40, it offers a wider range of features. Mainly, this generator is designed for the purpose of charging a cell phone or any other USB-connected device. In our case, instead of charging a cell phone, this connection would be used to charge a battery pack on the handlebar. This generator provides different outputs of current based on the speed of the bicycle. The speed to current relationship is shown in table 3.

|  |  |
| --- | --- |
| Speed (mph) | Charging electric current (mA) |
| 5.3 | 100 |
| 5.9 | 150 |
| 6.5 | 200 |
| 8.4 | 250 |
| 9.3>=12.4 | 300>=500 |

Table 3: Speed to Charging Electric Current for the Human Creations Generator

Having this figures is convenient for us as it allows us to approximate how much power the generator can supply to the battery at any given speed, and can help us in making power considerations overall.

## AC Converters

To keep the batteries at full power after they have been running for some time, a power source is needed to recharge the battery. While the generator is one source, it is not very effective to recharge the batteries because it would require the rider to ride extra distances without using any of the features or functions of the bicycle. This defeats the purpose of the project, since the concept of the POV design plays on the ability for the POV display to be on and advertising all the time. Given this, the other viable source is recharging the bicycle by means of a power outlet. This presents a problem, however, since the power coming from a typical outlet is 120V AC, 60 Hz. This output needs to be converted to a lower DC voltage in order to charge the batteries. To do this, a rectifier circuit is needed. There are a few designs for the AC to DC rectifier circuit; several circuits are considered in the following sections.

### AC-to-DC Converter with Transformer

The first option for a circuit configuration is shown in Figure 9. This circuit involves an input AC power source connected to a transformer which steps down the current and voltage from 120V AC to about 20Vac. After passing through the transformer, the AC stepped down voltage is passed through a full wave rectifier circuit and effectively converted to DC voltage

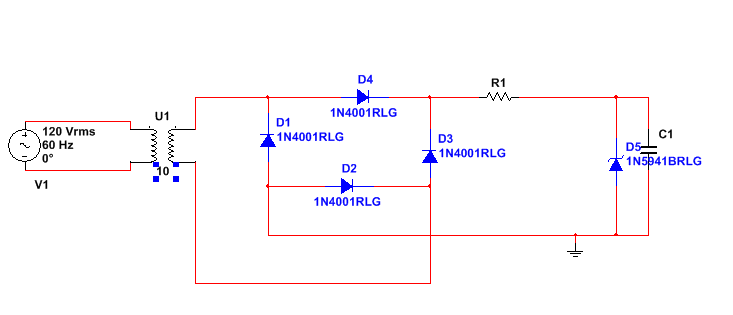


Figure 6: AC to DC Converter with Transformer Circuit Schematic

While we expected this converter with a transformer to be effective, what we have found is that it is quite difficult to maintain a steady output of close to 9V, which is our goal DC voltage output. Additionally, the needed transformer is extra space that would not be needed otherwise if another circuit was used. Thus, this converter, while possibly effective, is not the ideal option for our AC-to-DC convertor for this project.

### AC-to-DC Converter (No Transformer)

The second method involves immediately passing the 120V AC source through a full wave rectifier circuit, which then converts it to DC voltage. The diodes in this case would have to be much better quality and higher voltage rating in order to withstand the high AC voltages and current. With this approach there are two ways the rectifier circuit can be built. The first way is with the use of just diodes in the rectifier configuration as shown in Figure 10.

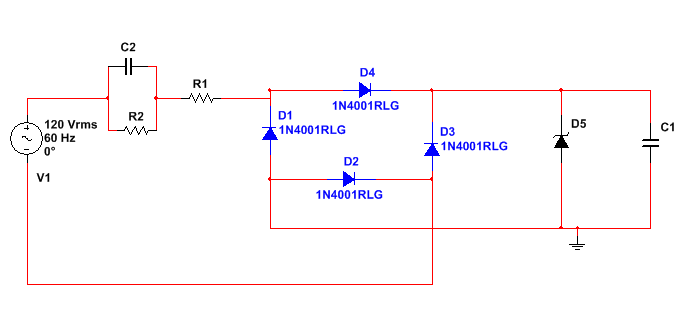


Figure 7: AC to DC Converter (No Transformer) Circuit Schematic

Another method is a rectifier circuit which utilizes op-amps. That circuit is discussed in the next section.

### AC- to- DC Converter with op-Amps

The third method is with the use of operational amplifiers to rectify the voltage. While this method is much more effective in rectifying the voltage, it is more expensive than just using diodes since the operational amplifiers contribute added cost. Additionally a DC voltage is required to bias the op-amps transistors. This voltage would need to come from another source which is not plausible. This schematic is of the dual op-amp convert circuit is shown on the next page.

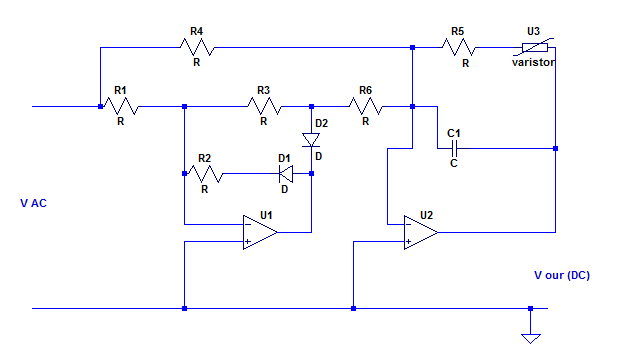


Figure 8: AC to DC Converter with Op Amps Circuit Schematic

### AC-to-DC Converter Circuit Components

The following sections describe the research behind the different circuit components needed to build the AC-to-DC converter. While the exact voltage requirements for our bicycle are relatively unknown, we estimate, based on research of other previous projects, that components with a maximum operating voltage of at least 150V and diodes with a power rating of at least 1500W should be sufficient for our power needs.

#### Diodes

As specified earlier, having a diode power rating of at least 1500W will allow up to 10 amps to flow through the diodes. While we cannot guarantee that this will be sufficient power for our bicycle until we test each component, we feel that this is an adequate preliminary estimate.

For diodes, we could use the Multicomp MUR 460 at a power rating of 1500W with a maximum voltage of 600V and a forward current of 4A. These diodes are relatively expensive however, at $0.68 apiece. (Multicomp)

Another option involves buying assorted value diodes in bulk, and experimenting with several different values. For example, amazon sells kits of 100-piece rectifier diodes of assorted values. One kit, sold for $12.97, includes 1N4148, In4007,1N5819 , 1N539, FR107 , FR207, 1N5408 , 1N5822 diodes. At $.1297 apiece, it will be easier for us to experiment with these types of diodes in order to find the best value through testing. (Amazon)

#### Resistors

Since resistors are extremely inexpensive, it will be simple to experiment with several different types of resistor values, both in simulation, and on a breadboard implementation of a circuit. The most important factor for consideration with resistors is proper heat dissipation, to ensure that the electrical components of the bicycle do not overheat. This will require a resistor power rating of at least 5W. A resistor in consideration is the RS00510K00FE12, which is a 10kohm resistor, rated at 5W.

## Battery

The entire project operates in a closed system and so a power source is needed to run all of the components on board the bicycle. The ideal power source would be a battery which can provide the required voltage and current to power all the components for a significant amount of time. The battery must additionally have a long shelf life, lasting many charge and depletion cycles. Most importantly however, the battery must be durable and safe.

This section is focused on the research done on different types and models of batteries. We will be looking at the different types of batteries which have been used in portable electronics over the years, and their effectiveness. We will focus down into the type which we will use and explore the different models of that type.

Over the years there have been 3 main types of batteries used in portable electronics, and more specifically, laptops. Theses batteries have helped power many components in laptops for a significant period of time and they have lasted a while without being replaced or breaking. These characteristics are what made them ideal for our application. (Lechnyr)

### Nickel Cadmium (Ni-Cd) Battery

The first type of battery is the Nickel Cadmium (Ni-Cd) battery. This battery was very popular at one point but has now become outdated (but is still being sold and marketed). These batteries, while effective, have many problems. The biggest problem faced by this battery is that it suffers from ‘memory effect’. Memory effect means that if the battery, while in use, is only partially discharged before recharging, will ‘forget’ that it can further discharge. This would cause it to only charge back up the level of its original charge, and continue there from the next time the battery is used. This can be solved by periodically fully discharging the battery and charging it fully again. This memory effect greatly reduces the life span of the battery, however. Apart from suffering from memory effect, the Ni-Cd battery has some additional problems in that they are heavy and additionally contained cadmium, which is a toxic material.

### Nickel Metal Hydride Battery

The second type of battery is the Nickel Metal Hydride (Ni-MH) battery which is similar to the Ni-Cd battery, but with the cadmium was replaced. This made the battery safer, with the hazardous material gone. Additionally the Ni-MH battery has a few more advantages over the Ni-Cd battery in that it is less affected by memory effect and so it doesn’t need as much maintenance and supervision. The other advantage of the Ni-MH battery is that they have a higher energy density than the NiCd batteries, and so there can be an increase in operation time of about double that of the NiCd batteries without the additional weight. While the NiMH battery has these advantages over the NiCd battery, there are still some disadvantages to using them. The first disadvantage being that the NiMH batteries have problems at very high and low room temperatures. And secondly, while they are made from less hazardous materials, they cannot be fully recycled and so disposal of the batteries is an issue.

### Lithium Ion Battery

The third and final battery type and the most practical for use in this project is the Lithium Ion (Li-ion) battery. These batteries have become the new standard for portable power. The Li-ion battery could produce the same energy as the NiMH battery, but weighs significantly less than them; about 20-35% less. Additionally, they are composed of non-hazardous material and do not suffer from the memory effect. However, the Li ion batteries did once suffer from a major and hazardous disadvantage. Early Li-ion batteries could sometimes become over charged and in some cases explode, destroying components and causing harm to the people using them. Luckily, in most current Li-ion batteries is a built-in internal circuit which monitors and regulates the charging of the batteries so that the IC can prevent overcharging of the battery.

### Types of Li-ion Batteries

While we are most certain that a Li-ion battery will be used for this project, there a few different types that could be considered for this project. The below section provides the research and evaluation of a few Li-ion battery types and then a model for each type. (Buchmann, Types of Lithium-ion, 2011)

There are 6 different types of Li-Ion batteries, each of which have a special composition which gives it special properties. The first one is the Lithium cobalt Oxide (LiCoO2) also known as LCO or Li-cobalt. This composition gives the battery a high capacity and is often used in cell phones, laptops and cameras. The LCO battery is ideal for high specific energy uses but is moderate when it comes to performance, specific power, safety and life span. The strengths and weaknesses are expressed as a web diagram in figure 12.

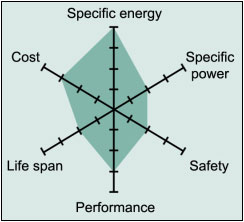


Figure 9: A web diagram of the strengths of a LCO battery printed with permission from Battery University.

The next Li-Ion composition is Lithium Manganese oxide (LiMn2O4) or LMO, Li-manganese, or spinel. Typical Li-manganese batteries are used for applications like power tools and e-bikes. They boast a high specific power and a long life span and are also among the safest Li-ion batteries on the market. However their overall performance is moderate and their cost can be a little high.

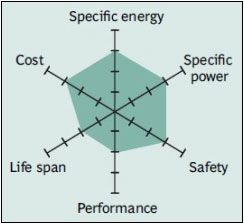


Figure 10: A web diagram of the strengths of a LMO battery printed with permission from Battery University.

Lithium Iron Phosphate (LiFePO4) is the next battery in consideration; its abbreviation is LFP and it is also known as the Li-phosphate battery. These have the same application uses as the Li-manganese and are found in many medical devices, and are popular among hobbyist. Typical Li-phosphate batteries have high safety ratings, a long life span and perform quite well. However they have a moderate specific energy and give out a lower voltage than most other Li-ion batteries (around 3.2-3.5 volts. Additionally, their cost is slightly higher than some other Li-ion batteries.

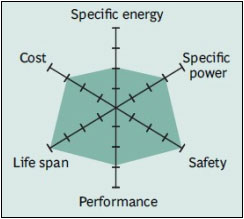


Figure 11: A web diagram of the strengths of a LFP battery printed with permission from Battery University.

Another Li-ion composition is the Lithium Nickel Manganese Cobalt Oxide battery (LiNiMnCoO2) or NMC. These batteries display very high specific energies and are the batteries of choice for many power tools and a wide range of electric vehicle applications. This battery’s performance is overall very good and is very popular, having the lowest safe-heating rate.

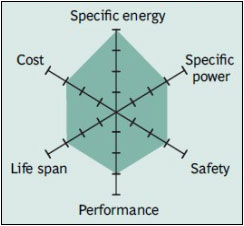


Figure 12: A web diagram of the strengths of a NMC battery printed with permission from Battery University.

The next composition is Lithium Nickel Cobalt Aluminum Oxide (LiNiCoAlO2) or NCA. NCA batteries have high specific energy and specific power; additionally they have a long life span and perform well. NCA batteries are favorite candidates for electric vehicle (EV) powertrains. Unfortunately NCA batteries have a high cost and have some of the lowest safety ratings; making them not very popular or not widely used in the consumer market.

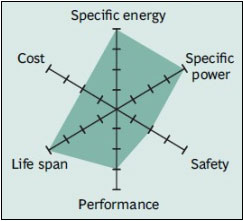


Figure 13: A web diagram of the strengths of a NCA battery, printed with permission from Battery University.

Finally the last composition is Lithium Titanate (Li4Ti5O12) or LTO and Li-titanate batteries. These batteries have excellent safety ratings, very high low temperature performance and a long life span. Their draw backs are that they have low specific energy and power; their typical nominal cell voltage is around 2.4V. Additionally, these batteries can be a bit expensive. However, current efforts are being made to raise the specific energy and lower the cost of the LTO batteries to make them perform much better.

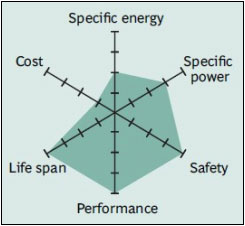


Figure 14: A web diagram of the strengths of a LTO battery, printed with permission from Battery University

The large amount of research done on the types and composition of the Li-ion batteries has led to the narrowing down of three possible batteries, whose specifications match our needs quite well. The three candidates are listed in Table 4 form comparing their different specifications and ratings.

|  |  |  |  |
| --- | --- | --- | --- |
| Battery Type | LiNiMnCo / NMC | LiMnNi / NMC | LiFePO4 / LFP |
| Image |  |  |  |
| Voltage  (Working) | 7.2V | 7.4V | 6.4V |
| Capacity | 7.2Ah (51.84Wh) | 8.0Ah (59.2Wh) | 5.0Ah (30Wh) |
| Cycle  Life | ≈750 cycles | ≈1000 cycles | ≈ +1000 cycles |
| Price | $54.45 | $54.95 | $45.00 |
| Dimensions (LxWxH) | 133mm(5.25") x 64mm (2.5") x 29 mm (1.13") | 114mm(4.5") x 30mm (1.2") x 75 mm (3.0") | 70mm ( 2.76") X 47mm (1.85") x 101mm (3.98") |
| Weight | 430  grams  (0.95 lbs.) | 390  grams  (0.86 lbs.) | 370grams  (0.82 lbs.) |
| Protection | YES  (8A PCB) | YES  (9A PCB) | YES  (10A PCB) |
| Charging rate | 4.0 A (max) | 6.0A (Max) | 2.5A (Max) |
| Charge time | @1.2 Amps ≈7Hrs | @1.2 Amps ≈10hrs | @1.2Amps ≈5.9Hrs |

Table 4: Showing a comparison of a few possible main batteries for the project.

Now that we have assessed the possible batteries that would be used for the Main battery, we now need to evaluate a few more batteries for use as the secondary battery. As stated in the project requirements, because we are using a generator to subsidize some of the power needs we will need to put that power into a secondary battery. The reason for the generator being attached to the secondary battery is that we do not want to be using and charging the main battery at the same time, for safety reasons. We also wanted a reserve power for our system to run our components for a short time or at least essential components. The list of possible candidates for the second battery is found in Table 5.

|  |  |  |  |
| --- | --- | --- | --- |
| Battery Type | LiCO2 / LCO | LiMnNi / NMC | LiFePO4 / LFP |
| Image |  |  |  |
| Voltage  (Working) | 7.4V | 7.4V | 6.4V |
| Capacity | 4.4Ah (32.6Wh) | 4Ah (29.6Wh) | 2.4Ah(15.36Wh) |
| Cycle  Life | \_ | ≈600 cycles | ≈+1000 cycles |
| Price | $19.95 | $27.95 | $29.95 |
| Dimensions  (LxWxH) | 0.735’’ x 2.86’’ x 2.86’’ | 59mm(2.3") x 30mm (1.2") x 75 mm (3.0") | 75mm(3.0") x 72mm (2.8") x  21mm(0.8") |
| Weight | 198.5 grams  (0.45 lbs.) | 195 grams  (0.43 lbs.) | 145 grams  (0.32 lbs.) |
| Protection | YES  (3A PCB + polyswitch) | Yes  (3A PCB) | Yes  (5A PCB + polyswitch) |
| Charging rate | 3.0  (max) | 6.0A  (max) | 1.2A  (max) |
| Charge time | @1.2A  ≈5.5hrs | @1.2A  ≈ 5hrs | @1.2A  ≈3.6hrs |

Table 5: A comparison of a few possible secondary batteries for the project.

## LED Types

LEDs are a critical aspect of this project. The type of LED used will affect the final result of how the bicycle looks and operates. Given this, it is critical that all possible LED options are carefully explored for consideration. The following section gives a brief description of several types of LEDs, and how are compatible they are with the needs of our project.

LEDs are generally of three main types: miniature, high-power, and application-specific. We will consider each below.

### Miniature LEDs

Miniature LEDs are commonly used in everyday items such as cell phones and televisions. They are generally single-die lights and range in size from about 2mm to 8mm. Most miniature LEDs are sold in “ready to fit” mode, which means they can be directly mounted to circuit board. The current flow of miniature LED lights is around 1mA to over 20mA. Miniature LEDs come in several shapes such as round, flat, triangular or square top, or variations of those shapes.

Miniature LEDs can be further classified into three types: Low current, Standard, and Ultra-high-output. Standard is the most likely to be used.

### High Power LEDs

High power LEDs, known as HPLED, can carry large amounts of current. Most HPLED can carry current from hundreds of miliamps, and even up to over one ampere. Some manufacturers such as Seoul Semiconductor produce these LEDs to operate on AC power, which eliminates the need for an AC-to-DC converter. For the purpose of this project however, the capacity of HPLEDs will most likely be unnecessary.

### Application Specific Variations LEDs

There are several types of application-specific LEDs; they are described below.

#### Flashing LEDs

Flashing LEDs typically display one color continuously. Some models flicker between several different colors. For the persistence of vision affect, flashing LEDs may not be necessary, as the image is created in the human eye through the means of spinning the bicycle wheels rapidly, not through the LEDs themselves flashing rapidly

#### Bi-Color LEDs

Bi-colored LEDs emit two different colors in a single LED. There are two types of bi-colored LEDs. The first includes two matrices connected to two wires serially, and the current in each wire produces a different color. The second type includes two separate wires for both emitters so that each color can be controlled independent of the other. This type of LED would be ideal if we were creating a display that produced two main colors. Since that is not a primary objective, it is not likely that bi-colored LEDs will be used in this display.

#### Tri-Color LEDs

Tri-color LEDs are similar to bi-color LEDs, but emit three colors instead of two. In this case, there are three emitters in a single case. Each emitter operates independently of the others, and each is connected to its own lead.

#### RGB LEDs

RGB LEDs are tri-colored LEDs that emit red, green, and blue lights.

## Wireless Communication

For our Persistence of Vision bicycle, one of our objectives is to have the image to be displayed uploaded by the user in the user interface, and then wirelessly transmitted to the bicycle. This gives the bicycle a creative edge since the user will be able to customize the types of images displayed, and also strengthens the possibility of the bicycle being used by advertisers.

Meeting this objective requires a cost-effective, yet reliable means of wireless transmission from the user interface to the bicycle. When it comes to wireless communications, there are several options that may be utilized. The main options for consideration are Wi-Fi, GSM, and Bluetooth. The following section provides a comparative analysis of the benefits and drawbacks to each option, and a justification for our ultimate decision to use Bluetooth.

### Wi-Fi

Wi-Fi is a method of wireless connectivity that operates on the Institute of Electrical and Electronics Engineers' (IEEE) 802.11 standard radio technologies. General use Wi-Fi networks operate in 2.4 GHz radio bands. (Wi-Fi Alliance)

For the purpose of this project, Wi-Fi was considered for use in two different modes: Infrastructure, and Ad-Hoc. Infrastructure mode would require a transmitter and receiver known as the Access Point and Client. Both of these components would be relatively inexpensive. Wi-Fi also includes a variety of encryption methods including WEP and WPA. It has an effective range of about 100 meters.

Ad-hoc mode eliminates the need for an Access Point. In this mode, two Clients communicate with each other. For this, two cell phones with Wi-Fi chips inside would be the two Clients, and would transmit and receive data from each other. Alternatively, an FPGA and a microcontroller could act as the two clients. The Table 6 provides an overview of how images would be transmitted in each Wi-Fi configuration.

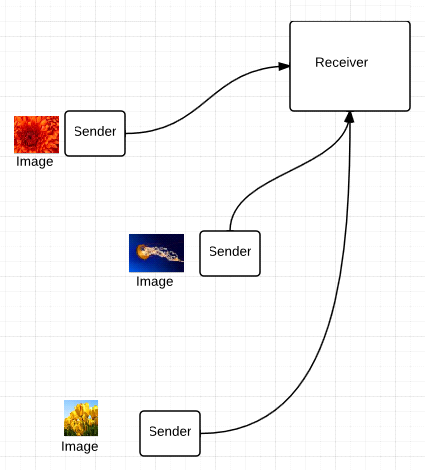
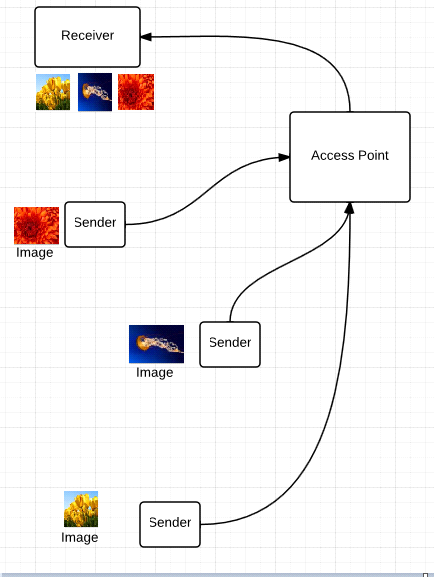


Table 6: Left: Wi-Fi Infrastructure Mode. Right: Wi-Fi Ad-Hoc mode

One of the major drawbacks to utilizing Wi-fi is its high power consumption. Since there are many components on the bicycle such as the LEDs that require electricity to operate, it is important to minimize the amount of power that each individual component utilizes whenever possible. When compared to other options such as Bluetooth, Wi-Fi power consumption is simply too high to justify choosing it.

Additionally, if infrastructure mode was used, it would require attaching an access point to the bicycle. This would be extremely impractical since there is no place on the bicycle able to house such equipment. Ad-Hoc mode could be sued, but if an FPGA or two cell phones were used, this would be much more expensive than the alternative options.

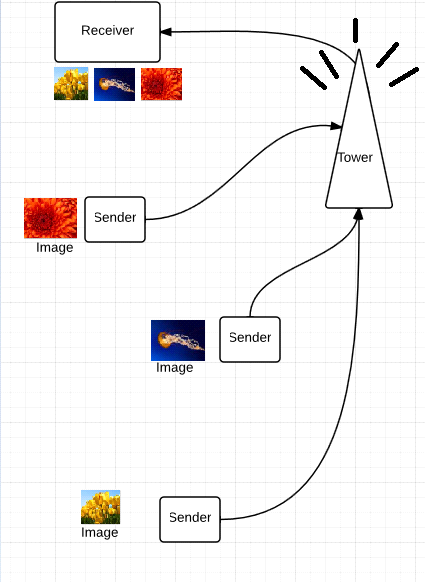


Table 7: GSM data transmission

### GSM

GSM, short for Global System for Mobile Communications is a wireless data technology originally designed for use in by cell phones. In the United States GSM primarily operates on the 850 MHz 19,000 Mhz. GSM transmits data over a providers cellular network.

GSM was considered for this project primarily because it has unlimited range, as long as the user is in a coverage zone. Input data can be transmitted to the display easily and universally by texting/emailing a predetermined phone number. Practically emailing and texting are the same thing because most phone providers provide SMS gateways that allow emails to be received as text messages. GSM is extremely low power since it was designed for mobile devices, and is typically limited to 2 watts.

GSM does have its own set of unique disadvantages. GSM modules are typically more expensive than Wi-Fi or Bluetooth modules. The maximum bandwidth is typically also much lower, and software implementation is typically more centuplicated. It would allow anyone to remotely upload images to the bike using virtually any device from virtually any distance. Also, in the United States many wireless carriers are wary of activating bear GSM modules, since they can easily be used for malicious purposes.

### Bluetooth

Bluetooth is a method of wireless connectivity that operates on the Institute of Electrical and Electronics Engineers’ (IEEE) 802.15 standard radio technologies. The primary difference between Wi-Fi and Bluetooth is that Wi-Fi tends to be access point based. Bluetooth was intended for portable equipment and its applications. Bluetooth is designed for interfacing symmetrically between small low power devices. Up to 7 devices can be connected to each other on the “piconet”.

Bluetooth was considered because it requires very low power and is easy to implement on the chosen microcontroller. Bluetooth also allows for easy paring with a variety of common devices including computers and cell phones. Software implementation would also be fairly straight-forward since most microcontrollers support a Bluetooth stack However, it does have several downsides. It has an extremely short range of only about 35 meters. It also has somewhat lower bandwidth than Wi-Fi. Bluetooth is also relatively unsecure because it limited to key-matching for security.

However, in the end Bluetooth is probably the best wireless protocol for use in this project. Bluetooth is by car the easiest wireless technology to implement. It doesn’t suffer from the need for a dedicated wireless provider like GSM, and it doesn’t need a common access point like Wi-Fi. It’s very low power nature makes it particularly fitting for such a mobile device.

### Possible Bluetooth Modules

This section describes the Bluetooth modules we have considered for use in this project.

#### RN-42

The RN-42 module is a very cheap and simple way to implement Bluetooth. The greatest advantage of the RN-42 module is that it includes an integrated antennae and contains all of the components necessary for the Bluetooth protocol. Many modules require a special external microcontroller that includes a Bluetooth stack to operate.

Specifications (Roving Networks):

* Fully qualified Bluetooth 2.1/2.0/1.2/1.1 module
* Bluetooth v2.0+EDR support
* Available with on board chip antenna (RN-42) and without antenna (RN-42-N) Postage stamp sized form factor, 13.4mm x 25.8 mm x 2mm (RN-42) and 13.4mm x 20 mm x 2 mm (RN-42-N)
* Low power (26uA sleep, 3mA connected, 30mA transmit)
* UART (SPP or HCI) and USB (HCI only) data connection interfaces.
* Sustained SPP data rates - 240Kbps (slave),
* 300Kbps (master)
* HCI data rates - 1.5Mbps sustained, 3.0Mbps
* burst in HCI mode
* Embedded Bluetooth stack profiles included (requires no host stack): GAP, SDP, RFCOMM and L2CAP protocols, with SPP

and DUN profile support.

* Bluetooth SIG certified
* Castellated SMT pads for easy and reliable PCB mounting

The module does have several disadvantages. The device is in a surface mount package, but it is possible to solder wires onto each pad of the RN-42. Additionally, the module is rather large, and will take up a large amount of space on the PCB. It is also rather expensive, with a listed cost of around 35 dollars.

#### Wireless Bluetooth V2.0 RS232 TTL Transceiver Module

This is a generic Bluetooth module used for connecting to a microcontroller’s UART pins wirelessly. It also includes everything necessary. Like the RN-42, the module is a self-contained solution that includes everything needed for the Bluetooth protocol. However, this module is also much smaller and cheaper than the RN-42, with a listed cost of only 7 dollars.

Specifications ("Wireless Bluetooth RS232 TTL Transceiver Module."):

* Radio Chip: CSR BC417
* Bluetooth V2.0
* Memory: External 8Mbit Flash
* Output Power: -4 to +6dbm Class 2
* Range: Up to around 30 feet (10 m)
* Sensitivity: -80dbm Typical
* Bit Rate: EDR, up to 3Mbps
* Interface: UART
* Antenna: Built-in
* Voltage: 3.3V DC

The default configuration is:

* Baud Rate: 9600 bps
* Data : 8 bits
* Stop Bits: 1 bit
* Parity : None
* Handshake: None
* Passkey: 1234

Bluetooth Module Dimensions:

* Width: 27mm
* Height: 13mm
* Pin Spacing 1.5mm

#### HC-06 serial

This Bluetooth module is designed for simple serial over Bluetooth communication to a microcontroller. It includes everything required for a Bluetooth connection. This includes the antenna, and the Bluetooth module itself, and a small separate microcontroller with a preconfigured Bluetooth stack. It comes on an easy to install 4 pin board. It has a development community around it that includes some documentation using it with the MSP430. For these reasons it is the best fit for this project. Its basic specifications are listed below.

* Specifications HC-06:
* Bluetooth protocal : Bluetooth Specification v2.0+EDR
* Frequency : 2.4GHz ISM band
* Modulation : GFSK(Gaussian Frequency Shift Keying)
* Emission power : <=4dBm, Class 2
* Sensitivity : <=-84dBm at 0.1% BER
* Speed : Asynchronous: 2.1Mbps(Max) / 160 kbps, Synchronous: 1Mbps/1Mbps
* Security : Authentication and encryption
* Profiles : Bluetooth serial port
* CSR chip : Bluetooth v2.0
* Wave band : 2.4GHz-2.8GHz, SM Band
* Protocol : Bluetooth V2.0
* Power Class : (+6dbm)
* Reception sensitivity: -85dBm
* Voltage : 3.3 (2.7V-4.2V)
* Current : Paring - 35mA, Connected - 8mA
* Temperature : -40~ +105 Degrees Celsius
* User defined Baud rate : 4800, 9600, 19200, 38400, 57600, 115200, 230400,460800,921600 ,1382400.

## Displays

A secondary display will be mounted on to the bicycle’s handle-bars. This display will show a variety of useful information to the rider. This includes current speed, distance ridden, and estimated remaining battery life. The display needs to be small and light weight so it does not interfere with the normal riding or steering of the bike. This display must also be visible under direct sunlight, since this is a common use scenario for the bike. Most importantly, it must consume as little power as possible, in order to help prolong the endurance of the bike’s batteries.

### Cathode Ray Tube

Cathode Ray Tubes, or CRTs, use cathode rays in a vacuum tube to create an image on a fluorescent screen (Brain , “How Television Works."). The greatest asset of this technology is that it provides very wide viewing angles and produces very accurate colors. However, they are a poor choice for this project for several reasons. The vacuum tubes tend to be heavy, bulky, and fragile. They are also very power hungry. Further they are relatively dim, and highly reflective of sunlight. These attributes make them a very poor display for use on an outdoor mobile device.

### Plasma

Plasma displays run an electric current through a gas chamber, usually containing Neon or Xenon. The ionized gas contains both positive ions and electrons, whose collisions release photons to produce an image (Harris, “How Plasma Displays Work.”) They are considerably thinner and brighter than CRTs and have much better contrast ratios. They also very bright, with good viewing angles. However, the gas chambers tend to be large and fragile, and they also require some sort of cooing system to avoid burning out. They also tend to be somewhat large and power hungry. In short, they are just too cumbersome for this application.

### Liquid Crystal

Liquid Crystal Displays pass a small electric current through a nematic phase liquid crystal. The crystal molecules exist in a naturally twisted state. When an electric current passes through the crystal, these molecules untwist. This changes the appearance of the crystal. By controlling the current precisely, a highly detailed image can be produced. LCD has a lot of advantages over CRT and plasma displays. They are thinner, lighter, and consume much less power than either. However, they do have their own set of unique problems. They are very heat sensitive and pressure sensitive, which makes them vulnerable to heat related malfunction outside in Florida. They are also extremely fragile, and also tend to rather reflective, making them very hard to see in direct sunlight. These problems make LCDs poorly suited for use in an outdoor mobile display.

### 7-Segment LED

7-Segment LED displays are probably the oldest and most simple type of electronic display. They simply use an array of LED segments that independently light up to display numbers. Since all the handlebar display needs to read out are numbers, this limitation doesn’t matter. They are very inexpensive, extremely light weight, and very easy to see in direct sunlight. They come in self-contained modules, which make them relatively durable. Most importantly, they require every little power to drive, and are extremely easy to interface with a microcontroller. They are by far the best suited display type for use with this project. Thus, a 7-segment display will be used to display information to the rider from the handle bars of the bike.

## RPM Sensors

In the project there is an important need for finding out the speed of travel of the bicycle. With the use of speed we can calculate the timing of the LEDs to produce the overall display, as well as display back to the rider the speed of travel and the distance traveled. Since the speed or RPM of the wheels of the bicycle is so important, we need an accurate yet small, and simple RPM sensor to measure and record the speed of the bicycle.

There are many ways to get accurate RPM readings from the bicycle and below we will explore a few of them giving their advantages and disadvantages when compared to each other in a table found on the next page. (Meyer) (Paisley)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sensor | Description | Sensory type | Size | Power Used | Accuracy | Cost |
| LED | Uses visible light sensors to detect LED light emissions | Visible Light | Small | Low | Moderate | Low |
| Infra-Red (IR)LED | Uses infra-red detecting sensors to pick up the IR light | Infra-red Light | Small | Low | Moderate | Low |
| Hall | Uses change in magnetic field to pick up flux dense areas | Magnetic | Small | Low | Moderate | Low |
| Laser | Uses intense light to gauge speed | Light | Moderate | High | High | High |
| Relay Switch | Uses a mechanical switch to detect movement | Mechanical | Moderate | Low | Moderate to low | Low |

Table 8: Comparison of several different RPM sensing methods that may be used in the project.

From the Table 8 we can see a simple spread and comparison of all the RPM sensors that are relatively applicable. Each of them has their strengths and weaknesses. For example, if we look to the laser sensor, while it will give us a high accuracy its cost is too high for reasonable use.

In the case of the mechanical relay switch, as the wheel turns, the use of a simple gear can trigger a completion of a circuit and the speed can be determined by the counts of the circuit completion. This is very simple to use and is quite inexpensive, but it is subject to mechanical failure and questionable accuracy.

The LEDs sensors circuits, both the IR and the simple LED, are subject to interference from environmental factors. This could reduce their accuracy but they are simple to make, cheap, quite, small, and low power. The Hall probe sensor is also quite effective, in that it is easy to make, is low cost, quite small, and doesn’t need much power to operate. However its accuracy is not as high as if a laser was being used.

Figure 38. shows what some of the possible RPM sensors would look like and how they may possibly work.

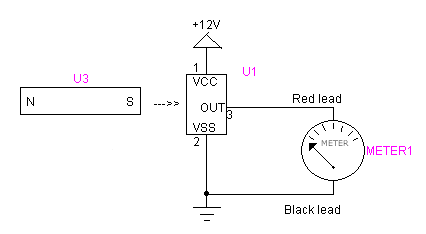


Figure 15 :The general setup of a Hall effect Senor (Printed Pending Permission from Rob Paisley 2012)

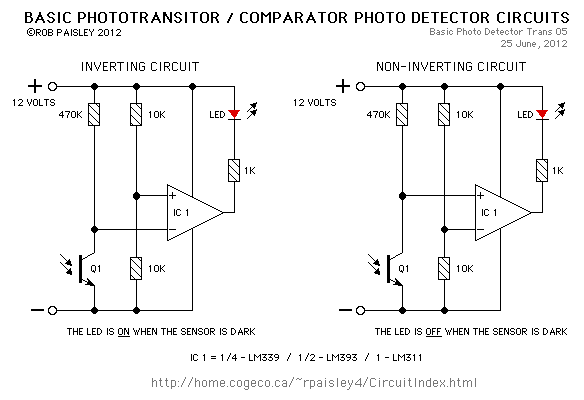


Figure 16: A general layout of Photo-Variable circuits. (Printed Pending Permission from Rob Paisley 2012)

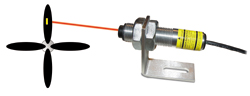


Figure 17: A simple example of a laser RPM sensor. (Printed Pending Permission from Rob Paisley 2012)

# Hardware Summary

Now that we have provided the research behind each component of the hardware on the bicycle, we will continue by giving a summary of the selected parts.

## Selected Microcontroller

Our group has selected the MSP430g2553IN2 to be used in our POV project. Our reasons for selecting this one in particular is for both technical and personal reasons. The order in which we mention the reasons is not necessarily the order of significance. First, the MSP430g2553IN2 is the microcontroller that is taught and used at University of Central Florida. All members of our group have experience hooking up and programming with it. Aside from that, it also happens to be powerful enough for controlling something such as a POV display. As opposed to some of the others, which were overkill. There has been at least one other POV project in existence to use this microcontroller and our section that discusses previous projects mentions it. Also, not only is it a good processor, but it's made by TI. This is relevant since our group has decided to support TI products, as opposed to other chip companies. Our reason for this is because we acknowledge TI as a respectable company worthy of our support.

In short, The MSP430g2553IN2 is powerful enough to get the job done, our group has experience with it, and our group likes TI.

## Selected Battery

For the POV bicycle we have chosen to use a Lithium Nickel Manganese Cobalt Oxide 7.4 volt battery as our primary battery. This battery is reasonably priced at $54.95, and is slightly less than 1 pound at 390 grams. We estimate that 7.4 V will be more than enough to meet all of our power requirements on the bicycle. As most of our hardware components require less than 7.4V to operate effectively, we will likely need to “step-down” this voltage in order to use this battery. However, we feel that it is better to have a battery that may be too powerful, and step it down, than have a battery that is too weak to operate.

As stated previously, we will have a secondary battery being charged by the generator while the primary battery is operating. Our secondary battery is meant to merely supplement the primary battery for a short time until it is able to be recharged through the AC power source. Thus, for the secondary battery we have chosen to use a smaller less expensive battery: another Lithium Nickel Manganese Cobalt Oxide 7.4 volt battery operating at 4Amp-Hours. This battery is $27.95. Along with the primary battery, it slightly exceeds our battery budget. However, it only exceeds the budget by a few dollars, so we are willing to be flexible in this area, and be more stringent in other parts that we need to purchase.

## Selected Generator

We have chosen Human Creations Bicycle Dynamo USB Charger for our pedal-powered generator. This generator is quite expensive at $56.40, however it does provide us with more features that are convenient to us. Specifically, it contains a USB charging setup, which would be simple for us to manipulate into a battery charger. Also, this generator clips to the forks of the front bicycle wheel, which means it will not interfere with the wiring of any other equipment in our display. Additionally, it comes with a piece that clips onto the handlebars of the bicycle which is designed to hold a phone or GPS unit. This piece can be used to house our seven segment display. Since this generator comes with a preset speed-to-current-output

Overall, this generator is the best fit for our needs and appears to be worth its cost.

## Selected LEDs

In the research of existing projects, rarely did we find a persistence of vision project that did not utilize RGB LEDs. RGB would give us a wider range of color options by using the RGB additive color model. Thus, we will most likely be using miniature RGB LEDs. Specifically, we are using common anode LEDs. This is because our LED drives cannot support common cathode since they do not have the necessary current output.

From the comparisons of the different types of LEDs and their special features we have decided to go with common anode RGB LEDs with frosted bulbs. The presence of the three colors in the RGB LEDs would allow us to produce a much larger range of colors from a single LED. This large variety of colors would allow for a much better display effect and quality from our POV design.

The LEDs would have a frosted bulb because this allows for a much better color dispersion of the light in the individual LEDs, making their mixed colors stand out more. This cuts down on how bright the LEDs can get because of the loss of lumens through the opaque frost. However the frosting also has a great advantage, in that it allows for a much wider viewing angle of the LED. This wide viewing angle would make the POV image visible to more people.

Each of the RGB LEDs has three diodes in them, each of which should require about 20mA operating each of them and so each of the RGB LEDs should draw at most 60mA when operating. This small current is required to operate the LEDs is very helpful because we are using so many LEDs the current of each one adds up. So the smaller the current draw the better it is for the design and the longer we can operate the POV display without worrying about the battery discharging.

Additionally, since the LEDs will be spinning to produce an image, it is important to ensure that the images are as clear as possible. To avoid streaks or gaps in the images, we would like our LEDs to be as close together as possible. What we have learned from the previous UCF persistence of vision senior design group is that while the T-1 ¾ package LEDs are quite popular, they are not ideal for minimizing space between LEDs, since they have a width of 5.9mm. (Al, 2012)

Instead, by using the previous UCF group’s project and experimentation as a reference, it is better to use surface mount LEDs, which are smaller and easier to integrate with minimal space in between.

The previous UCF group chose to us MulticompS OVS-33 Series SMD Super Bright LEDs. We have chosen to follow their lead and use the same LEDs for our project. The pin information for the OVS-33 Series SMD Super Bright LED is shown in Figure 21. (Multicomp)

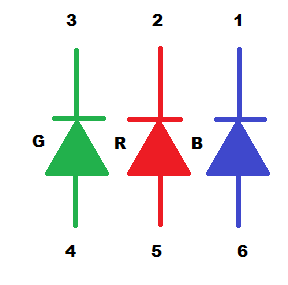


Figure 18: Pin Information for OVS-33 Series SMD Super Bright LED

## Selected LED Driver

The LED drivers are one of the most essential and important parts to the project. Therefore the selecting of the LED driver is an important decision and requires much thought and weighing the pros and cons of each of the possible devices.

After reviewing all the LED driver options the driver that was decided upon is the TLC5940NT IC manufactured by Texas Instruments. This driver is very popular when it comes to projects and application which involve the powering and control of many LEDs. The TLC5940 is an easy IC to setup and use, allowing for easy learning and operation.

The chip has a wide range of feature sets which stand out in our minds and convinced us that this was the right choice for our project. One of these important features is that the chip has the ability to be daisy chained, which allows us to connect and control multiple 5940 devices together. The connecting of multiple TLC5940 ICs allows us to able to drive a large number of LEDs all at once, making it perfect for our project. Designed with 16 channels per chip, the 5940 is one of the few chips on the market which can power a large number of LEDs at once.

Another feature of the driver is that it has a 12 bit (4096 step) pulse width modulator build into it; allowing us to be able to adjust each individual LED’s light output, giving us a wider range of colors and effects. This is very useful because it allows us to produce a higher quality and a more dynamic POV effect and design.

The TLC5940 is also outfitted with a dot correction feature with up to 6-bits of correction. Dot correction is basically a feature which allows us to correct the lighting in the individual LED, so that if we wanted a mixture of light to produce a color, one LEDs color won’t drown out all of the other. For example, if we wanted a purple light we would use the blue and red LEDs in the RGB LED. However, the blue light might be very bright and drown out the red light not giving us our purple light. The dot correction feature would make it possible to correct this, by lowering the intensity of the blue light to allow it to mix smoothly with the red giving purple light.

Overall the TLC5940 is a great driver to use for our project; It would provide us with the necessary constant current to effectively drive large amounts of LEDs, it gives us low power consumption while providing a lot of useful features. The chips are able to cascade easily taking up only 5 pins total on our microcontroller and they are relatively inexpensive in large quantities. On a side note because this device is so popular there are many helpful examples of the chips in use which we can familiar with, allowing us to effectively use the TLC5940 to its full potential.

## Selected RPM sensor

Out of all the methods of RPM sensing, the use of the Hall Effect sensor was found to be the best choice. The Hall Effect circuit is very simple and easy to implement, test, and is considered relatively accurate. Each of the circuits are simple to make because they are mainly composed of the Hall sensor and a few resistors and capacitors. This simple circuit is has the additional benefit of being very cheap.

When comparing the other RPM sensor designs there were several faults with each of them which convinced us that the Hall Effect sensor was the best candidate for the job. In the case of the laser RPM sensor, the design was just not viable because it was not only expensive to acquire a reliable laser, but also required a strong power source. We felt that this additional current draw from the battery for operating the laser continuously would be too much since we need as much power to operate all the LEDs and the main components of the POV system.

The laser sensor maybe highly accurate but it comes with its own applicable complications, in that its setup is a relatively larger circuit when compared to the other rpm sensors.

The mechanical rpm sensor, like the laser sensor is larger when compared to other methods and can possibly get in the way of the user riding the bicycle. The mechanical sensor is faced with another problem; as the name says it is a mechanical device sensor which has moving parts and these devices will eventually go bad and would then need maintenance and servicing. Additionally the mechanical rpm sensor, depending on how it is set up, can have varying levels of accuracy when compared to other methods.

The LED and IR (infra-red) sensors were the second best choice. They are easy to set up, they are inexpensive, they don't use much power, and they can be very small. However there were few problems with these methods. The first problem was that they are prone to interference from the outside light. The circuits work based on a photo varying resistor, where the intensity of light exposed on to the resistor would change the resistance of photo varying resistor, trigger a signal to the micro controller. The problem occurs when you get "noise" from the infrared and visible light wave of the sun which may accidently trigger the circuit giving inaccurate readings. We could add a Schmitt trigger and thresh hold values to the circuit but this only would complicate things more and raise the cost of the implementation of the sensor. While one could argue that, the circuit with LEDs or IR LEDs would be easy to implement because we already are using LED drivers to power so many LEDs (and consequently could just use one of the ports to make the circuit), this is in reality only hinders us because using up channels on the drivers would cause us to have less LEDs or more driver chips. Therefore this is not the best solution because we would need all the channels on the LED drivers to run the LEDs essential for the POV design.

Therefore as said before the Hall effect RPM sensor is the best model for the job, and we will be using multiple sensors to get a much more accurate reading of the speed of the bicycle.

## Selected AC converter

Selecting the AC converter is very important as well. The right converter would allow the battery to charge efficiently in a reasonable amount of time and also charge the batteries safely.

When talking about AC we are dealing with very dangerous voltage and current levels; levels that have caused fatalities to those who have improperly handled AC voltages. The converter chosen is the full wave rectifier without the use of a transformer. We went with this circuit design because it is reliable and easy to implement. Additionally, it relatively small compared to the circuits with a transformer in it. The transformer circuit would normally be relatively large and heavy due to the many windings which make up the transformer and so it’s not best suited of the job of rectifying the voltage.

The second choice was the ideal full wave rectifier. This is a rectifier circuit with the use of operational amplifiers, but this is problematic because op amps require the use of biasing DC voltage to operate the op-amps. The problem with this case is for example, say both the main and secondary battery are depleted and we wanted to charge them. The op-amps would not be able to work, rectifying the AC voltage because they wouldn’t have the necessary DC biasing voltage and so this design is not considered a reliable and effective solution to our problem. That’s why we are going with the full wave rectifier without the transformer. Many everyday appliances use this method and it’s simple to create a design which will match our charging needs.

# Software Research

This section describes the research behind the development of the software component of the bicycle.

## Battery Regulator Module

The battery regulator module will be a function written in C for our MSP430g2553IN2. The goal of this module is to come up with some method of safely regulating how charged the battery is. To do this, the AD converter pin on the microcontroller should be used to measure the voltage output of the battery. From this value, we can determine how charged the battery is, and send an on/off signal to the charger.

The module itself will more than likely be just a few lines of code. Should this module fail, there is a hardware safety net to prevent our batteries from exploding. Standard coding practices should be used to ensure that reliable code is being made. This area of the code should be well tested in order to prevent errors, such as turning the charger on when the battery is full or turning the charger off when the battery is empty.

## User Interface Modules

The following section will discuss how the user interface section will be organized. First, an introduction paragraph will justify as to why a user interface is even necessary in this particular project. It will then go on to mention some possible user interface technologies that our group could take advantage of. Each of those technologies will be discussed in detail and at the end of the section all of the possibilities will be compared and contrasted. Refer to Table 11 for the comparison table. This table will merely list out the advantages of the selected phone over the selected computer GUI. From this table we will determine whether to make our GUI using a phone or the computer.

Justifying the use of a user interface is quite easy to grasp. Imagine that you are a marketing company representative, and you have asked a senior design group from University of Central Florida to craft up some really cool POV bicycle. At the end of the semester they give you a really cool, fancy, best ever POV bicycle, but you have no way of telling it what to display! The entire project no matter how fancy it is, is worthless. Therefore, it is necessary to have a user interface, and there are many possibilities.

Since this happens to be the software research section of the document, it will be assumed that all hardware technologies are a viable option for user interface interaction. Some of these technologies include phones and computers. Phones could be broken down even further into Apple and Android. Using the computer would open up the possibility for countless more user interface options. For each possibility deemed worthy, we are going to discuss how each module would need to be handled given a particular piece hardware.

### Phone

This section will discuss the two of the major phone manufacturers. One phone being from Apple and the other from Microsoft, an Android phone. Both phones are worthy of our project, and there are a limitless number of resources out there for programming with them. The main reason as to why we are not considering the phones of any other companies is because of a few factors. One being, that the resources available that assist in programming these devices are typically for Apple and Android, and are very difficult to find for off brand manufactures. Another reason as to why we are only considering these two is because Apple and Microsoft are known for making reliable phones that do not break (as often). It would be a waste of time to write a program for a phone if it is not reliable. Not only that, an even stronger reason is because they are the most popular, if we were to market this product, it would be unreasonable to expect the customer to have (or purchase) a phone that they are not likely to have.

Topics such as the language required, the IDE required, and external libraries required will be discussed for both possibilities of phones. Refer to Table 9 for the comparison table. The comparison table merely lists advantages and disadvantages for both Apple and Android. From this table we will determine which one of the phones we are more willing to program with.

#### Android

In order to craft an Android application, the ADT (Android Development Tools) bundle must be downloaded and installed. The ADT bundle is a package that has all required configurations done for us. It's is convenient for getting started in Android development, and fairly quickly too. Alternatively, the entire development space could still be manually configured, if that is what we wanted (or needed). (Android)

The IDE that comes with the ADT bundle is a modified version of eclipse. It is essentially the same eclipse, just with some extra tools built in for Android development. Naturally, eclipse is most known for java, and java is the language required for Android development. Our group happens to know java, which is a major plus.

There is some configuration that actually does need to be done however. Once eclipse is up and running, we would then want to create a new project. While making a new project, eclipse will prompt us for obvious data such as the project name and location of the project. However, it requires another piece of information. The information it requires is which Android phone to emulate. Not all Android phones have the same functionality or even the same screen size. To decide which phone to select for emulation, we would need to consider the phones that we, as a group, have access to; those phones are the phones that we would select for this option.

All required libraries are handled for us in the ADT bundle. Therefore there is likely to be little reason to download any others to create the core of an Android application. Just to clarify the idea, the phone that was selected when the project was made, may determine which libraries are automatically included in the project. Implying that each of the Android phone models may require a different library.

#### Apple

In order to craft an IPhone application, Apple's Xcode 5 SDK must be downloaded. In order to even download this SDK, one must first create an account with an Apple ID and then must login at the Apple website. Right off the bat this leaves a bad impression on developing for Apple. Once this is done, a download link will then be provided to download Xcode. Believe it or not, there are actually some advantages to developing with Apple. To develop an IPhone application there are a variety of languages that can be utilized. Some of those languages include Objective-C, Cocoa, C++, C, Javascript, Lua, or any other interpreted language with an interpreter. (Apple)

Apple's Xcode is fairly easy to use. Someone could make a basic application without writing any lines of code just by manipulating the tools Xcode provides, much like visual basic for windows. However, to do anything actually useful, Objective-C is the language that is primarily used. The fact that other languages can be used is a double edged sword. Yes, there are a lot of languages, but finding well-made tutorials may be very difficult.

|  |  |  |
| --- | --- | --- |
| Phone | Advantages | Disadvantages |
| Android | * Nice tutorials * Easy to setup | * Java is only option |
| IPhone | * Tutorials exist * Variety of languages | * Annoying to setup * Must create an Apple account |

Table 9: Comparison between iPhone and Android

Now that we have Apple and Android compared it is quite obvious that Android takes the cake. The reasoning is as follows. Android was researched first. It was quite easy to find all the information needed to get something up and running pretty quickly. The tutorials for Android seemed nice and easy to follow. After that, we researched Apple. It could have been coincidence, but at first we could not find out what Apple's SDK was called. Perhaps, it could have been due to bad search criteria. After we finally found out it was Xcode 5, we wanted to download it. This is where we started to get annoyed with Apple. In order to download it, we had to create an Apple ID, which is pretty annoying, and if something as simple as downloading it is annoying, who knows what else is? So after that, we decided to search for some tutorials, all of which were mediocre. The difficult part about the IPhone tutorials is that it is necessary to find a decent tutorial in the programming language we intend to code in, which was not easy. Thus concludes the reasons as to why we chose Android over IPhone. These are of course just the opinions of our group, after researching both, Apple and Android. Really though, it would only truly become obvious as to which one is better after actually programming with both of them.

### Computer

The following will discuss how the computer section will be organized. The first topic that will be discussed is why our group should even consider the option of using a computer UI. The second topic that will be discussed are programming languages that may make the programming easy. In addition, we will discuss why they should be considered. And finally, the languages in question will be compared to one another, exposing the advantages and disadvantages to each of them. Refer to Table 10 for the comparison table. That table is how we will decide which programming language is really right for our group.

Electing to use a computer based UI opens the window to unlimited possibilities. Just in case it is not obvious as to why, here is why. First, any programming language is a viable option. This alone is a strong advantage. Second, the only hardware it requires is a computer, as opposed to a smart phone, which, believe it or not, not everyone has. There actually is a downside to using computers. Although computes may be thought of as a single entity, the operating system that is installed may determine what we can and cannot do with our computer GUI application. For example, if we were to write our GUI in C, it would need to be compiled for whatever Linux distribution our client is using or for windows if that is what our client is using. This idea of platform independence is quite attractive, so interpreted languages are probably going to rank higher than compiled languages. This is a very similar to the problem to the phone section, in just the idea that there was Android vs. Apple. If we do not pick an interpreted language, then we will assume our client is using a Windows computer, since this is what most of the world uses.

#### Java

To justify why java should be even considered as a viable option for writing our GUI is because of the following. First, this is a language taught at University of Central Florida, so we have experience in it. Second, java is a versatile language that can be used in a wide variety of applications including GUI applications, or even web based applications via Applet. Applets are not the only way to create a web based java program, but for the sake of simplicity, we will not be using or discussing those methods. Also, Java is an interpreted language, so it's platform independent.

Some IDEs that may be used with Java are Eclipse and Netbeans. It is mainly personal preference as to which should be used. The members of our group have mainly used Eclipse in the past, so more than likely that is what would be used.

#### C#

C# is very much like java in the fact it's Object Oriented, and the way in which GUI's are built are similar. Also, it is just as versatile as java is. However, we have relevantly little experience programming GUIs in C#. So, in essence, we do not know exactly what kind of GUI we can truly build unless we were try it.

The required IDE is Visual Studio, made by Microsoft. Should we pick this language, there are many resources out there for it, making it easy to get started. Also, as students of University of Central Florida, we get this software for free. One disadvantage is that Visual Studio seems to take up a lot of system resources while running it, which adds to lag, and makes working in that environment painful sometimes.

#### C++

C++ is like both C# and Java, a GUI application could easily be made using this language. Like C# this language will be need to be learned. Some advantages are that some C syntax can be used, and we know C.

There is no required IDE for C++. Visual Studio could be used; DevC++ could be used, or any other. This is an advantage over C# and java. There are plenty of resources out there for learning GUI programming in C++.More than likely we will not be using C++. We believe that C# is a much better candidate; this is based on personal experience.

#### PHP

PHP is probably the most unique language listed here. PHP has many strong points and an equal number of weak points. First of all, the only external entity required to run a PHP script, is a PHP compatible browser which almost all modern web browsers have. This means that our application could run on Linux, Mac, and Windows, without any trouble at all. On top of all that, our group has experience with PHP. From that experience, we know that PHP is very easy to write in, and it is also very easy to make the application look nice with HTML or CSS. However, PHP does have its downsides. PHP cannot run by itself, even if we have a web browser. PHP requires a server that can interpret the PHP code into something that the browser can display. There are two options for setting up a server. Host it ourselves, or have a company host it. Both have their advantages and disadvantages. If we host it ourselves, we would be the one who need to configure the server to run on the internet. Any other configurations that would need to be set, would need to be done so by us. However, the advantage is that we are not sharing our data with any strangers, and could end up being more secure (not that our project requires security). The advantage of having someone else host it is that it is easier to get started, so there would be no need to modify our router and no need to get permission from our ISP. Should we decide to host our own web server there are applications out there that assist in setting up the server. WAMPP and XAMPP are bundles that include several servers, including databases. These packages are fairly easy to configure too, after some experience.

Another advantage of PHP is that there is no IDE associated with PHP. There are text editors that do syntax highlighting, such as notepad++. One of the strongest advantages of using PHP is that it can potentially run on both computers and phones. The phone must have a PHP compatible browser, and so do the computers. The most difficult part of using PHP is setting up all the configurations.

|  |  |  |
| --- | --- | --- |
| Language | Advantages | Disadvantages |
| Java | * Already Known | * Requires Java to be installed |
| C# | * Nice tutorials * Good to learn * IDE is available for free, normally paid for. | * One IDE available * Need to learn |
| C++ | * Nice tutorials * Good to learn | * Need to learn |
| PHP | * Already Known * Can be used for Phone and Computer | * Requires Server * Can be Annoying to set up |

Table 10: Comparing programming languages

The computer clearly has a wide variety of options when it comes to programming languages. Some of the most appealing choices are PHP and Java. PHP is appealing because web based interfaces are very pleasing to use in many cases; in general they just look good. With a PHP application, the only software the user needs is a web browser, which all computers have by default. As opposed to java, java must be in installed on the user’s computer in order to run any java application. This is usually not an issue, since java is fairly widespread and most computers already have java installed. C# and C++ are very good to consider, however, the fact that our group isn't experienced in either one of them is a big disadvantage. The focus of the project is on POV, and not creating a GUI with languages we are not familiar with. The main disadvantage of PHP is that it requires a server in order to run. There are plenty of servers out there that are very easy to set up, but they tend to be unreliable and are very strict with configurations. Overall, if our group were to choose to use a computer based GUI, the best choice would be java. There are java libraries out there that provide support for a variety of wireless transmission protocols, which is super convenient.

|  |  |  |
| --- | --- | --- |
| Device | Advantages | Disadvantages |
| Phone | * Mobile * Voice Recognition | * Will need to be learned * Small GUI * Texting |
| Computer | * Easy * Mobile if it's a laptop. * Keyboard | * Easy * Large |

Table 11: Comparing phone and computer as user interface

So now that we have all the options out, it is time to decide on whether it is more advantageous to program for a phone or program for the computer. How the UI is made is a non-critical part of this project, although a GUI is definitely necessary. One clear advantage for the computer is that it is the most basic kind of programming. Everyone who has written a program probably learned on a computer. Also, none of our members have experience programming for phones, this could add a significant delay in progress. There are tutorials out there to make it easy, but there will most likely be unanticipated issues that could make a simple program into a complicated one. One clear disadvantage of the phones is the size of the GUI. This probably is not a unique opinion, but it can be a pain dealing with a small interface. Yet another disadvantage of the phones is that in order to write a message out. More than likely it would need to be written just like a text, which varies from phone to phone. Texting in general is much more difficult than typing on a keyboard. Alternatively, an advantage could be, if the phone supports it, that the text could be written out with speech recognition. The computer has very few disadvantages. One disadvantage is that writing a GUI program on the computer is something all computer engineers have done before, and there would be relevantly little challenge to it, although this is also an advantage. Also, laptops are much bulkier than a phone is, but the larger GUI outweighs that disadvantage.

After considering all of these facts, our group has decided to use Java on the computer to construct our GUI application. It is just the most reasonable choice considering the requirements of our project.

## Wireless Modules

This section is going to assume that we are using a particular wireless communication hardware, and it will describe how each of them would be programmed. The general idea behind the module is that, in the GUI when the upload button is pressed, there is some sort of button listener waiting to send the data to the microcontroller. In a sense there are two modules, a send module and a receive module. The send module is on the GUI side, and the receive module is on the microcontroller side.

Before the send and receive functions are discussed, it needs to be understood that there are two possible data types that are being sent; images and animations. Animations pose a problem. If the entire animation cannot be stored in the microcontroller’s internal memory, it may need to be streamed, and this has problems of its own.

One possible solution is to simply add a hardware memory module to our design that can handle the animation size we want. This is not the ideal solution for a couple obvious reasons. First, adding hardware means it will cost more money. Second, let us say we add the hardware memory module, and it turns out we want to have a longer animation. We would then have to upgrade to an even larger memory chip. More than likely this is not what our group will be doing.

To stream our animation from our computer to our microcontroller we need to determine the necessary bit rate to transfer our data. 6KB \* 1024 \* 8 is 49152 bits for just a single frame. 49152 \* 25 is 1228800 bits per second, which is more than 1 Mbps or 150 kbps. From this calculation, to stream video would require a pretty fast wireless connection between our computer and microcontroller. This speed as of now is questionable.

### GSM Send

The data that the GSM Modem is sending is assumed to have already been converted, and is in the form that the microcontroller will expect. Details about the algorithms are in the algorithm section. This section will discuss various ways of sending data to the GSM modem. Sending data is not so simple as to just send it off and hope it arrives. It must be done so in an organized manner, and there are libraries out there that do this for us. The library may provide a send function and may provide a receive function. There are many details associated with sending and receiving data, and insuring all of the bits send are received, those kind details will not be discussed; rather, the libraries that do this for us are the topic of discussion.

#### Java Communication API

The first option we will look at is using the java communication API from oracle. Now that it is downloaded we must then install this API. Let us assume that we are working in eclipse and are writing our code in java. First, the .jar file that came with the API should be added to the build path of the eclipse project. The API has now been installed. The way in which the code is communicating with the GSM modem is through the communication (COM) ports on the computer.

Rather than straight up using this API from scratch, we have found open source java code that set up a connection for GSM communication. The author states that the code may be used, so long as the header comments remain unchanged. There are 5 java files in total that we intend to borrow, called SerialConnection.java, SerialConnectionExeception.java, SerialParameters.java, Sender.java, and SMSClient.java. None of these java files contain a main method, so we have to create it ourselves. Within our main class, we would need to create an instance of SMSClient and provide it the appropriate arguments. This should be the only class that we actually have to interact with, assuming we do not need to modify their code. SMSClient provides all the methods to connect to our GSM Modem via the COM port. (Alexander).

There is of course no guarantee that these functions will work for our particular GSM modem. There are about a dozen reasons why this may or may not work so the more options that are written in this section. The easier it will be later.

If after a couple days go by without any successful transmissions, and if we have tried tweaking the code, and if we have tried writing it from scratch with this API, we will proceed to move on to another potential solution.

To make this section as beneficial as possible, we are going to list GSM API's to try out after we acquire a GSM modem. Without an actual GSM modem, it is very difficult to accurately determine what actually works and what does not. As known already, there is the java communication API from Oracleand SMSLib. After researching GSM API's there actually does not seem to be too many other options available. It seems that the mass majority of the people on the internet are using SMSLib to handle their GSM modem communications.

These issues are important to solve as early as possible, and should we actually choose GSM, we have found a couple resources on the internet that have a lot of information. They are online java programming forums that allow us to search for people's problems and their solutions. It turns out there are quite a few posts related to GSM, SMS, and communication ports. During this research, any sources that seem good are being saved. They are being saved so we still have access to the solutions even if their websites were to shut down.

### Bluetooth Send

This section will discuss how we can write a software module to send data via Bluetooth. To start with, after just a little research, we found an API that works. It is called javax.bluetooth. It is not so simple as to just download it; it seems to be pretty hard to find. After searching around the internet, there are many ways of getting this API, and it generally means downloading some other library that happens to contain it.

Another API that we were successful with, was a library called Bluecove. There is example code out there for this particular library and we were able to confirm that the source code was compatible with the library. However, until we actually have Bluetooth device, it is extremely difficult to determine what will work and what will not work. To communicate to a Bluetooth chip, instead of using the COM ports, the API uses something called a Bluetooth stack. The API handles all details of this in the background, however, there is always an uncertainty that the Bluetooth device we end up getting would be compatible with the library found here. Assuming we pick Bluetooth to be our transmission protocol that is.

Just as was done in the GSM section, we will merely list java Bluetooth APIs to try out, after we have the actual Bluetooth device. One as we know is bluecove, another is named Harald which is a joke by the way for Harald Bluetooth, look him up. AveLink, JavaBluetooth, avetanaBluetooth, JBlueZ, and Impronto DK. Each of these has their own set of tutorials and are set up in about the same way. (Fitton)

### Communication

So now that we finally have communication, we can finally start sending data. The module is fairly simple, ignoring any of the communication setup. This module does not need to worry about converting any data. The data is assumed to be good to go, ready to be sent by the time this module is executed. The receive section will discuss certain issues such as how to detect the end of the message. Other than that detail, the method will basically be a loop that sends the data one data unit at a time until it's all been sent. It really should not be too complicated. The most difficult part will be establishing communication.

### Receive

Assuming the send module sent the data properly, the way in which the received data is handled is always the same, regardless of the technology. As the data is received, it should be stored in the flash of the microcontroller. After all the data is received, the microcontroller should then run its update LEDs function. This portion of the code will more than likely be written in C.

There are a couple ways that we can notify the microcontroller that all the data has been sent. One way is if the very first piece of information that is received is the total number of expected bytes to be sent. The receive module would just wait until all bytes have arrived via a loop. Once all the data arrives it would just continue with its routine. Programming it this way open opportunities for infinite loops. Imagine the case when the send module has finished sending, but the receive module is still waiting. Careful testing will be needed to prevent these kinds of mistakes if this method is chosen.

Yet another way to tackle this problem is to have a terminating character determine when all the data has been received. This way is pretty nifty, but it does have its downsides. One downside to this is that every single piece of data needs to be checked if it's the terminating string. If the terminating string is a byte long, and the image is 6kB, a total of 6000 comparisons would need to be made in order to find out if the job is done. There is also the possibility of the receiver confusing some other byte as the terminating string, when it's not really the case. This would be difficult to test for, considering that images being uploaded are in a sense random.

## Image Conversion Algorithms

This section outlines the various algorithms that will be used by the client software to prepare images and text for use with the bike. In the case of image files, they will first be converted into the portable pixmap format, or PPM. This is a widely used format in the field of computer vision because it is very easy to work with. All images are stored as three two-dimensional arrays. Each array contains red, blue, and green color values respectively. This allows the conversion software to easily and efficiently read and modify image files. For this purpose a freeware image conversion utility will be used. The programming required to convert most image formats into PPM is readily available and beyond the scope of this project. All other image manipulation software will be written specifically for use with this project. The algorithms needed for this are detailed in Figure 22.

### Scaling Algorithms

In order for an image to be shown on the bicycle’s POV display, images must be scaled down to an appropriate size. The display will use comparatively few LEDs to display the image, so it must be first scaled to a smaller resolution. The lower resolution also allows for a cheaper and less power hungry microcontroller to be used to implement the display. In this section several common image scaling algorithms will be examined for potential use in this project (Tech Algorithm, 2007).

#### Nearest Neighbor Image Scaling

The Nearest Neighbor algorithm is the least complicated and quickest scaling method. The main premise of this algorithm is to first build a reference image from the original image, and use this to construct a new scaled image. This new image may be larger or smaller than the original depending on the scaling ratio used. The scaling ratios for the new image are calculated using the following formula:

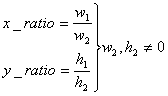


Figure 19: Calculated Image Ratios. Used with Permission from Stereforth John at tech-algorithm.com

If the algorithm is scaled up, this is accomplished by creating a new 2D array to represent the image. This array has dimensions equal to the scaling ratio multiplied by the original images dimensions. Then the new array is populated with the original image’s pixels by using a double loop to go over all of the pixels fill pixels according to the x-ratio and y-ratio of the new image. The image is completed by matching all pixels empty spaces, representing regions that require new pixels, with their nearest neighbor. This is shown in Figure 23.

If the algorithm is scaling down, instead of a new array populated with empty spaces, pixels are just removed according to the same pattern. This does result in a permanent loss of information.

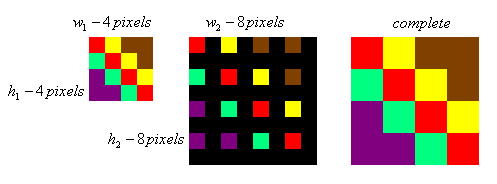


Figure 20: Nearest Neighbor Scaling. Used With Permission from Stereforth John at tech-algorithm.com

The primary advantage of Nearest Neighbor is speed. The software implementation of the algorithm is very simple, but this comes at the cost of frequent in accuracy. The other algorithms typically produce much less jagged looking images.

#### Bilinear Image Scaling

Bilinear image scaling is very similar to nearest neighbor scaling. The only real difference is that interpolation is used to create new pixels, instead of merely copying them from neighbors. This generally produces much smoother images than nearest neighbor (Algorithm, "Bilinear Image Scaling", 2009).

Linear interpolation attempts to estimate the color of a new pixel by doing a weighted average of the two nearest original pixels based on the distances between the original pixels and the new one. For example, suppose a new pixel, *Y*, needs to be created between two pixels *A* and *B*. Let *L* be the distance in pixels between *A* and *B*, and let *l* be the distance in pixels between *A* and *Y.* The value placed in Y is defined by the following equation:

As the name implies, Bilinear interpolation creates pixels by doing interpolation on new pixels in both the x and y directions. When the image needs to be scaled up, a new image is template is constructed in the same way as in Nearest Neighbor. The empty pixels are now filled in using interpolation on the four nearest original pixels. However, when an image is scaled down interpolation is done in reverse. Typically when a pixel is removed this is done by combining it with nearby pixels, averaging their values. This results in more accurate color values than nearest neighbor because nearest neighbor merely throws away pixels while down scaling.

Bilinear interpolation provides a good balance between speed and quality. It produces a reasonably smooth image with relatively low complexity. It is a little slower than nearest neighbor, but the quality gained from pixel interpolation is generally worth it.

#### Trilinear Image Scaling

Trilinear is a slight modification of bilinear image scaling. It further sacrifices speed for increased accuracy. Unlike the previous algorithms, trilinear interpolation requires two reference images. One reference image should be exactly half the size of the other. The first reference image is usually the image being scaled, and the second reference images is generated via bilinear downsize (Algorithm, Trilinear Interpolation Image Scaling, 2009).

The smaller reference image is then scaled up using bilinear scaling to the intended size. The original reference image is usually bilinear scaled to the intended size. These two reference images are then linearly interpolated to create a new image. This image is said to have undergone “trilinear interpolation” because it was bilinearly interpolated and then linearly interpolated.

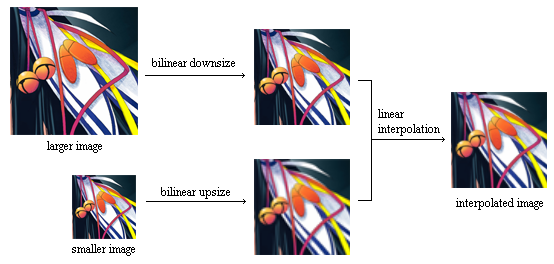


Figure 21: The process of Trilinear Scaling. Used With Permission from Stereforth John at tech-algorithm.com

As shown in Figure 24, Images created in this way tend to be extremely smooth. However, this algorithm is also usually very slow because it requires that bilinear interpolation is run on the image at least twice, followed by running linear interpolation. Also, images produced this way tend be less sharp, almost slightly blurry.

Since fast conversion is an important goal for this project, and the image is being downsized by an extremely large factor, the use Trilinear Interpolation would be overkill. Bilinear Interpolation will produce images that are nearly as accurate as Trilinear, but it will do so much faster. The downscale image is of such low quality anyway that any difference would be extremely minimal. For this reason, this project will use bilinear scaling to downsize images.

### Radial Image Conversion

In Order to display an image on the POV,

After our image has been reduced in size, then some method of reorganizing the data is needed. Refer to the Radial Image Conversion Diagram 1to understand the following explanation. Notice the red and black checkered image in the diagram. This is the sample image that will be converted.

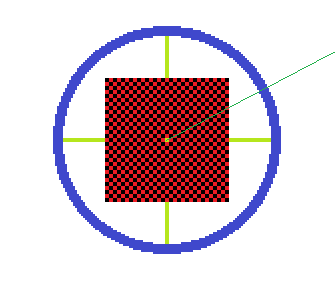


Figure 22: Radial Image Conversion

First, it should be noted that the image starts as a standard rectangle centered at the origin on a Cartesian coordinate plane. The image should be viewed as a 2D array of numbers that represent color values. What the algorithm needs to do is essentially rearrange the image data into a different order. The order in which the data needs to be arranged in, are represented by the green line in the image. All the pixels that the green line passes need to be added into a new array. Each value in the array represents the color value of each pixel the line passes through. This array will only contain sixteen pixels, and each element will represent the color value sent to an RGB LED on the bike’s display.

The reason for organizing the data this way is so that all the data is split up into chunks and ready for the microcontroller to use. Once this is done, the microcontroller just has to load the LED array with the data already organized, rapidly to display the image.

First of all, it needs to be decided how the lines are going to be represented mathematically. Around sixty lines of pixels will be sampled from the down scaled image. This will be done by checking if a given pixel lies within a certain tolerance value of the line, since most of the time the line will not cleanly pass though exactly sixteen pixels. I

In the case of more than sixteen pixels falling within the tolerance value around the line, pixel averaging will be used to combine neighboring pixels, like in the previously discussed bilinear interpolation. This process should help to preserve accurate color values without the missing pixels.

In the case of less than sixteen pixels falling within the tolerance value around the line, pixel averaging will be used to determine what color value should be used to fill in any open spots left in the LED array, like in the previously discussed bilinear interpolation. This process should help to preserve accurate color values without the missing pixels.

At the end of the conversion process, a set of arrays that represent the color values of each LED at any given angle will be ready to be sent to the display. Using the Hall Effect sensor to estimate the current speed of rotation of the bike, these arrays will be used to light up the LEDs at each corresponding angle. The after image left on the viewer’s eye will fill in any gap left by the algorithm resulting in the appearance of a complete image.

## 7-Segment Displays

It was decided that the project would use a 7-segment display to show relevant information to the user from the handle bars of the bike. This display will show information regarding speed, distance traveled, and remaining battery life. As previously discussed, a 7-segment display fits this role very well because it is easily viewed in direct sunlight, has wide viewing angles, requires very little power, and is very light weight. The following section discusses the two types of 7-segment modules, and which was ultimately picked for use with this project.

There are two different common types of 7-segment LED display: Common Anode and Common Cathode. Both typically require a resistor in series, to limit the amount of current going to the LED segments. For a 5V power source with a forward bias current of 20mA and a forward bias voltage of 2V, the resistor value is defined as follows:

### Common Anode

In a Common Anode display pins 3 and 8 are typically shorted with the Anode of the LED segments. This common pin is then also connected to positive voltage. In order to light up any particular LED segment, proper Cathode pin must be wired to ground. This completes the circuit and LED segment will be put in forward bias mode, which causes the LED segment to light up. In this configuration the current limiting resistor should be wired to the Cathode to prevent the LED from burning out.

### Common Cathode

In a Common Cathode display, the pins 3 and 8 are shorted with the Cathode of all 8 LED segments. This common connection is then wired to Ground. In order to light a particular segment, positive voltage must be given to the corresponding LED Anode pin. The completed circuit will put the LED into forward bias mode, which causes the LED segment to light up. In this configuration the current limiting resistor should be wired to the Anode to prevent the LED from burning out.

### Power Consumption

Usually, Common Anode 7-segment displays requires significantly less power from the microcontroller to drive. With the Common Cathode configuration, the LED driver must directly output the required current. However with the Common Anode LEDs the controller only needs to sink the required current. Since power use is a major concern in this project, common anode 7-segment LEDs will be used to display information on the handlebar box.

## LED Controller Modules

This section will take into consideration all suggested LED controllers in the hardware research section of this document. Each subsection will discuss how, in code each LED controllers would be programmed.

#### TLC5940

When programming with a TLC5940, there are at least 5 output pins that need to be known.

* SIN
* SCLK
* GSCLK
* XLAT
* BLANK.

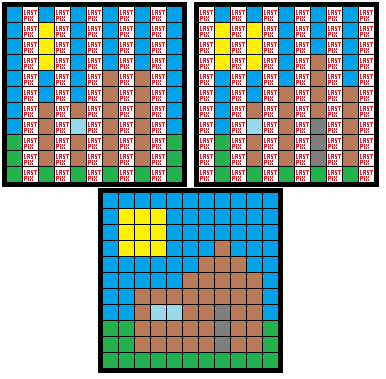
To start with, I will give a brief overview of how the LED controller works. There are 16 LED outputs and each of these receives a 12 bit number. This number represents the brightness that the LED (color) should reach.

At this point we know we need a 12 bit number for each of the outputs. What this tells us in code is that we're going to have to do a total of 12 \* 16 shifts, which is a total of 192 shifts to fill the entire LED controller, it will be more if we chain them together. This number, 192, also represents how many times we will need to pulse the SCLK. The SCLK shifts the data into the LED controller. So now the question is, what data is SCLK shifting in? The data that it is shifting in, is the value of the SIN pin at the moment when the SCLK is pulsed. If SIN is high, a high bit is shifted in and vice versa. Still, there is more that needs to be understood about this; we have said we need to do 192 SCLK pulses, but it's every 12 bits that are important. It is important to keep in mind that if the brightness we want happens to be 1011 1000 0001, we will have to change the value of the SIN pin after each and every SCLK pulse.

Now that we have our LED controller loaded with all the values we wish to show, we must take another step before the effects are reflected on the LEDs themselves. There is another register called XLAT. When XLAT is pulsed, it triggers a latch signal that will cause the LEDs to reflect the input data. It is more complicated than it just reflecting the 12 bit numbers onto the LEDs. There is a register called the GSCLK. It is what really makes the brightness of the LEDs. GSCLK is short for the grayscale clock, which counts up until it reaches that 12 bit number. The 12 bit number is essentially the amount of time the LED should stay on, and the GSCLK controls it. It must be pulsed in the actual code. Let us say we have reached the maximum GSCLK value, it must then be reset back to 0. To do this, we must adjust the BLANK register. If BLANK is **high**, then all LED outputs will be reset also this is where the GSCLK gets reset. When BLANK is **low**, it allows the LED outputs to controlled by the gray scale pulse width modulation.

As far as the software module goes, the code will affect these five pins in such a way to produce a POV image. For animations, a separate module will be needed to manipulate the LED controller accordingly. The animation function could really be the same function as the picture function, just called several times.

Animation in general can be handled in the following way. If the microcontroller we choose happens to be too slow for complex animations, there are coding tricks that may be done to compensate. It is easiest to understand the technique if you picture a rectangle, and imagine its 100 pixels by 100 pixels. To cut the required time to animate this block, imagine lighting up one solid column of pixels. Skip a row and light up another column until the end is met. Changes would be made to the lit up columns in one frame. In the next frame, changes would be made to the dark columns. There is a limit to how much this technique can be taken advantage of. For example, could three columns of alternating work? Four? Instead of doing complicated math to figure it out, it is best to write this animation function in such a way that is general, where the number of alternating columns could be changed with the flip of an integer. Some of our members have experience with this technique and has used it in animating in Java, applying it to POV should be just the same. Refer toFigure 26 for an illustration of the technique.

Figure 23: This Diagram illustrates how the alternating column technique would work given the still image of a house in the grass, with the sun shining.

The unaltered columns will remain as they are, or could be turned off completely depending on fine detail of programming with this particular led controller. We only consider that because the led controller is essentially a large shift register and is more than likely to end up being blank rows.

NOTE:

More contemplation has been done on this technique and there are some problems that may need to be addressed. Firstly, it is known by this point the LED controller is essentially a large shift register. So in order to get data to the LEDs, all pixels must be traversed and stored with some value to move on to the next. In order for this technique to work, we would need direct access to each LED, which is possible, but is not in the current design plans. The reason why we need direct access to each LED is because to change the color the LED in the lowest right corner, every single LED must be set with a value before arriving at that corner. In short, it is because the data is shifted inward. This technique is still valid as long as the possibility of us wiring our PCB this way exists. In other words, this technique will not work under our hardware design, so far.

The way the module handles still images is far simpler. Before we can display an image, we first need that image stored in memory. This module does not need to worry about the data very much, by the time the data reaches this module, the data is in the form it needs to be and is merely shifted into the LED controller.

There are some issues that need to be planned for carefully in this module. For example, imagine a circle; it has 360 degrees. At which set of degree marks will the LED array be updated? Clearly, the more the LEDs are updated, the more detailed the image will be. However there are factors that limit how many times this can occur. For simplicity the module should be programmed in such a way that is general, where the number of updates is a matter of changing an integer value.

Of course if we change the number of updates, the data would need to be adjusted as well. To clarify, imagine that the image we want to display is a circle directly in the middle of the POV display. This is the simplest case; the number of updates needed would be **zero**, minus the initial uploading of the image. This should prove that the required number of updates changes from picture to picture.

We have already shown that updating too much can never be a problem when it comes to picture quality. So now, just to mention the fact, if we update too few times, we may lose parts of the picture.

So by now, it should be clear that building this module in a general way will save us a lot of trouble later. As far as how to actually do this, can be done in the following way. First of all, the module is made of up a couple nested loops and the goal is to load the LED array. So we are basically saying we need to load the LED array X number of times. X would change depending on the type of data. For example, we could have three different update LED modules, one for images, one for animations, and one for text.

Another issue this module needs to worry about is the timing. Let us say we want to load the LED array at every 5 degrees, which is a total of 72 updates. There is one piece of external data that this module needs, and it is the time it takes for the tire to do one full rotation. Let us assume it takes 3 seconds to do one full rotation. 3 seconds / 72 is 41.6666 milliseconds between each update. To handle this issue of timing, we would need a wait function that takes the number of milliseconds as an argument. This function does not just have to waste time. It is uncertain at this point in time, but it is possible we could have this function also load the LED array with the next line of data, and only after the 41.6666 milliseconds have passed would the data be XLAT’ed in. Another issue to discuss is that a bike is a moving object, and to complete one full rotation is not a constant number. The problem proposed is if the biker is going too fast, then it is possible the system may not be able to keep up. This calculation can be redone with different time values. For example, if it takes .2 seconds for the tire to do one full rotation the time between each update would be 2.77 milliseconds.

# Hardware Design (Prototyping)

The hardware design of this POV bicycle will determine whether or not the display works effectively. Thus, it is extremely important that all aspects of the design are carefully considered. The following sections discuss the preliminary hardware design. To begin, we provide the highest level block diagram that presents the relationship between all hardware and power-consuming devices on the bicycle.

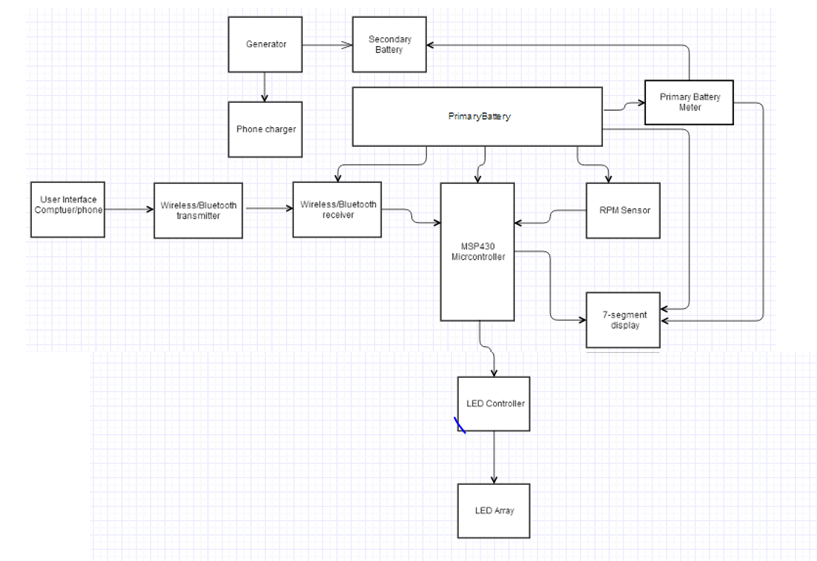


Figure 24: High-level hardware block diagram

As shown, the primary battery powers all of the devices that comprise the POV display including the MSP430 microcontroller, the RPM sensors, the seven-segment display, and the wireless modules.

The microcontroller is the “brain” of the project, and has inputs and outputs from most of the components including: the wireless transceiver, the LED controllers, the seven-segment display, and the RPM sensors.

The user interface is where the user uploads an image they would like to be transmitted to the bicycle. The image is passed to the wireless transmitter, which delivers the image to the wireless receiver, where it is fed into the microcontroller.

The microcontroller processes the image and feeds the information to the LED controllers. The LED controllers are connected to the LED array, and through proper data transmission, the appropriate LEDs for the image are created.

The generator is present to charge the secondary battery, and also acts as a phone charger for the bicycle user. The secondary battery is used as a backup when the primary battery is low, and is therefore represented as an output to the primary battery meter (although absent from the high level diagram is the switching circuit which activates the secondary battery).

An RPM sensor on the bicycle measures the speed of the wheel, and feeds this information to the microcontroller. The microcontroller delivers this information to the seven-segment display, which displays the information on the speedometer. Additionally, the speed of the bike as measured by the RPM sensor is used by the microcontroller to determine the alignment of the image, and how it should be displayed through the LED array.

As this is a high-level diagram, it merely shows the general data and power flow of the hardware. Many of these components are mounted to the PCB. The PCB is analyzed in greater detail in the following sections.

## PCB Layout

The following section describes the layout of the printed circuit board. The PCB will be fixed to the axle of the bicycle wheel, and will house all of the controls of the display, include: microcontroller, LED controllers, the LED array, the RPM sensor, and the wireless GSM module.

When thinking up and designing the project and all its pieces the PCB was one of the most debated and complicated parts that we had a hard time finalizing. Figure 28 shows a general layout of how the PCBs would be connected. It was decided that there were going to be two central PCBs one located in the wheel and the other located on the handle bars. The main PCB in the wheel was also going to be broken down into three separate pieces. This was done because of the difference in location of the 3 LED arrays. The first PCB part in the wheel of the bike would house one LED array of 16 RGB LEDs, 3 TLC5940 drivers, the blue tooth module, the RPM sensor circuitry, a voltage regulator for the Bluetooth module and finally the MSP430 microcontroller. This first part was what made up the main PCB, but there would also be two additional PCB which would connect from the main PCB and be place at different phases on the wheel. These two sub-PCBs would each house an array of 16 RGB LEDs and three TLC5940 drivers. The power flowing into these PCBs would come from the main batteries regulated voltage and would be rated at 5V.

While the exact layout of the PCB will be ongoing design process, the following schematic provides a general plan for the PCB setup:

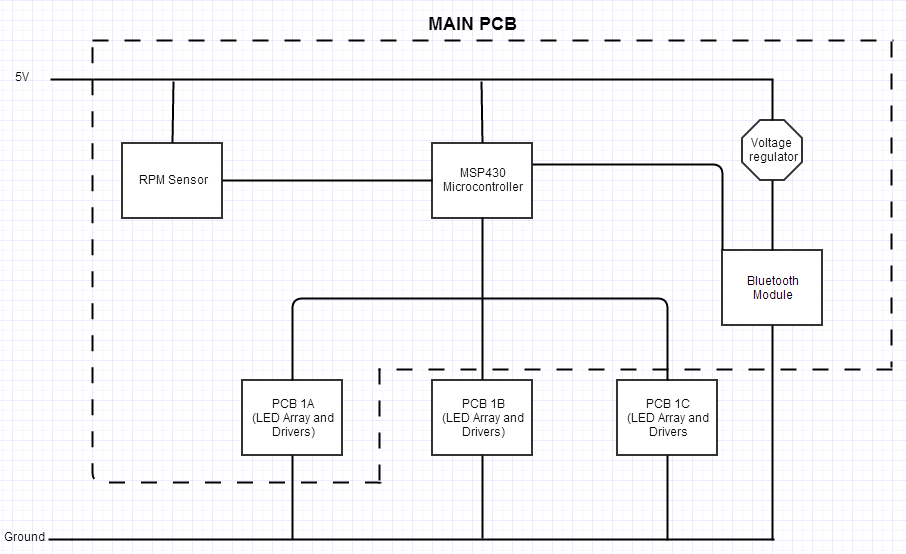


Figure 25: showing a general block diagram of the PCB layout found in the Wheel of the bicycle.

The general layout of the secondary PCB board found on the handle bars of the bicycle. As is stated in the mechanical design section, we are using two micro-controllers and RPM sensors because not only do we not have enough pins on the selected microcontroller but we also have mechanical issues transferring multiple data or power buses of two or more channels from the stationary bike frame to the rotating wheel well. Therefore to provide the both the user and POV display with the current speed of the bicycle two separate RPM sensors and microcontrollers were designed and placed in their respective spots. The secondary PCB found on the handle bars would only contain the second micro-controller, the connections for the 7-sengment display and the RPM sensor. The main battery also powers this PCB.

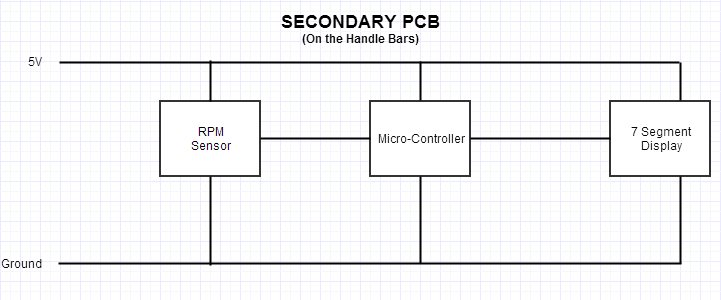


Figure 26: Showing a general block diagram of the PCB layout found on the handle bars of the bicycle

## Microcontroller

The chosen microcontroller for the POV display is the MSP430 Launchpad by Texas Instruments. Our schematic representing the input and output pins of the microcontroller are shown in Figure 30. The Launchpad is also equipped with a port to intake external power. The transmission of external power to the microcontroller is also represented in the schematic.

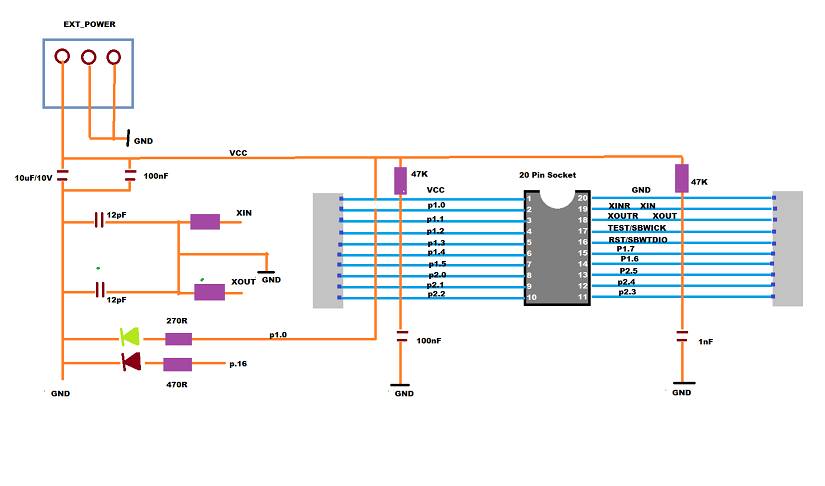


Figure 27: Schematic for the MSP430 Launchpad (handmade)

The microcontroller found on the main PCB will be responsible for operating and controlling the entire POV display. This microcontroller will be connected to the RPM sensor for the back wheel, as well as the LED drivers and the Bluetooth module. All the LED drivers need a total of 5 pins from the Microcontroller, plus the VCC and ground connections. The RPM sensor needs one pin to the microcontroller to be able to send the rotation speed data. Finally the Bluetooth module would require 4 pins for proper communication with the microcontroller and POV display.

The microcontroller in the secondary PCB is going to be connected to the 7-segment display, the second RPM sensor, and the battery monitor. The 7- segment display will require a lot of pins from the microcontroller, about 10 to be specific. The RPM sensor will require one input pin to the microcontroller. Finally the battery level monitor would require two pins so that it could communicate to the user the battery level.

## LED Controllers

The chosen LED controller for this project is the TLC594 by Texas Instruments, and its schematic is shown in Figure 31.

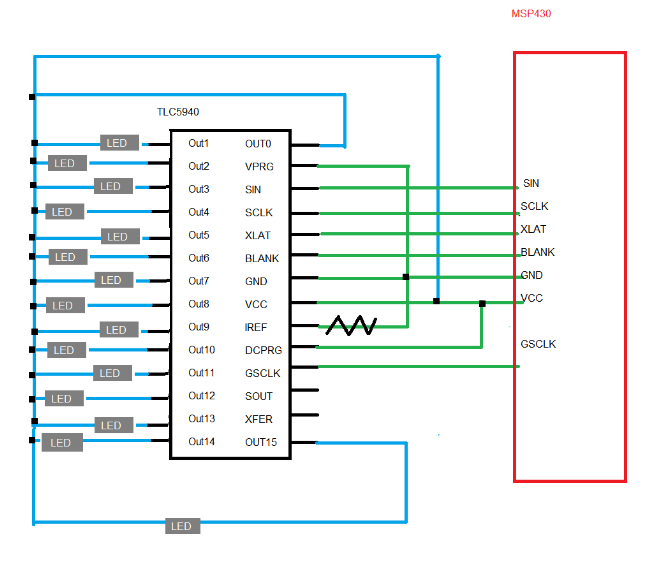


Figure 28: Schematic for the TLC5940 LED controller

When connecting the LED drivers in cascade/ daisy chaining them there is only a total of 5 pins which need to be connected to the microcontroller. Each of the LED drivers can share (connect in parallel) the pins for the GSCLK, BLANK, XLAT, and SCLK. However the important thing for cascading the drivers is that while the SIN of the first chip is connected to the microcontroller. The SOUT pins of that driver is then connected to the SINs of the next chip, whose SOUT is then connected to the next SIN and so on. This is how the cascading of the drivers is done. As always the VCC and ground connections are hooked up in parallel to all the drivers.

## RPM Sensor

This is the RPM sensor which will be used in the project. This RPM sensor setup contains multiple hall sensors hooked up in series (Meyer). The Hall sensors main circuit which is housed on the main PCB will be located in the back wheel well. The rotating wheel and stationary bicycle would make it hard for the data to come from the stationary part to the rotating wheel well. Therefore our setup is where we will have the strong magnets placed on the stationary bicycle frame and the sensors in the wheel well.

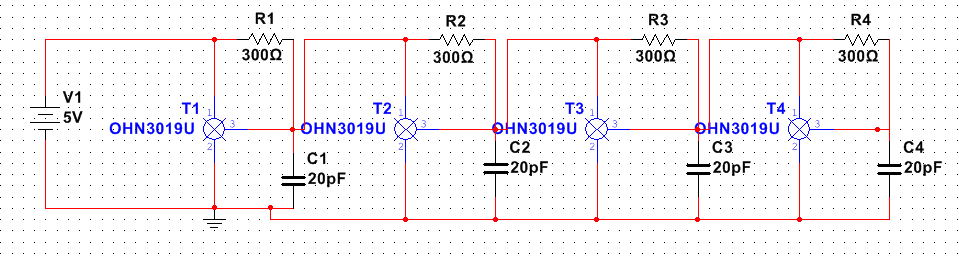


Figure 29: A prototype of the circuit for the RPM sensor using hall effect sensors.

The sensor operates in such a way that the circuit is ON, always sending a high signal ‘1’ to the micro controller. As each sensor pass the magnet on the frame it will cause the circuit to drop low and send a low ‘0’ signal to the microcontroller, telling the microcontroller that a sensor has just cut through the flux field of the magnet. We can measure the timing of each ‘0’ input and calculate the speed of the wheel from that.

In terms of displaying the speed and distance travelled to the user via the 7-segment display, we would use the same circuit idea found above but this time measure the front wheel speed. However this sensor and flux trigger/magnet set up is reversed, in that we will only use one sensor segment attached to the frame of the bicycle and put several of the magnets on the wheel. This is so that it is the moving magnets which trigger the circuit and not the moving sensors.

## Power

This section will describe the requirements and considerations for powering the Persistence of Vision Bicycle. As there are many components of the final design which will require electricity to operate, this is a critical part of the project design. Powering the POV display is perhaps one of the greatest challenges of this project, as it requires not only determining accurate voltage and current input for each element of the bicycle, but also ensuring that the power sources are safe and reliable.

One of the major challenges is determining how several different parts, each with their own power requirements, can function effectively from a single battery, and how long the display can operate before the battery needs to be recharged. Overcoming these challenges will require thorough examination and testing of all components and their power needs. The section below offers a preliminary overview of the POV display power element. Figure 33 provides a high-level power block diagram.

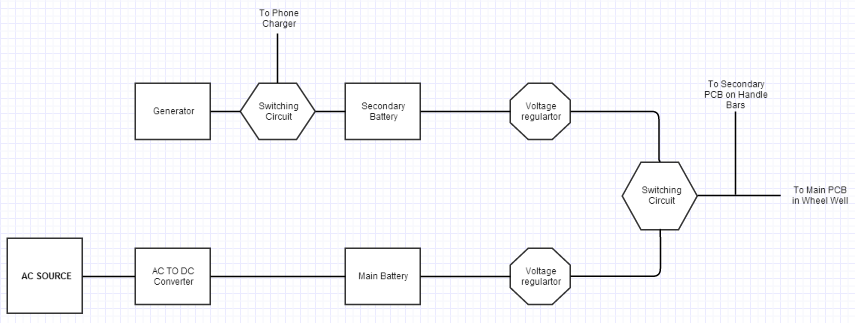


Figure 30: Showing the general power block diagram for the entire project.

### Voltage regulator

This is the voltage regulator prototype that would be used to step down the main batteries voltage and feed 5V into the main and secondary PCBs. The circuit is designed to give the appropriate voltage to the inputs of all the components no matter the increase in load and additional loading effect. The voltage regulator used in the circuit has a low drop out voltage. In that as the battery’s power gets depleted close to the voltage regulators constant output about 5V then the dropout voltage determines when the output of the regular would start falling below 5V.For example say our dropout voltage is 0.5V the output of the regulator would start dropping below 5V when the battery is drained down to 5.5V. The regulator is also designed to be able to transfer as much of the current needed by all the LEDs and the components in the PCB. These devices and components allow have current draw which when added up turns into a high current load and so the regulator is designed to not restrict the flow of current.

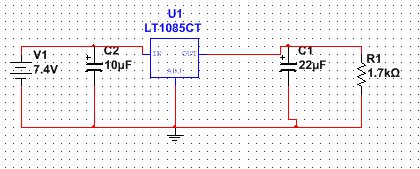


Figure 31: The prototype circuit for the voltage regulators used in the project.

A secondary voltage regulator or just a voltage divider would be needed to operate the Bluetooth module, since its operating voltage is lower than the 5V with which most of the other components would be using. This voltage regulator would be very similar to the one powering the main and secondary PCBs. The only difference would be this voltage regulator would have a lower rated output voltage, something around 3.5-3 volts. The drop out voltage of this regulator would still need to be low since we don’t want cascading voltage drops across the components resulting in the turning off of the blue tooth module. But further testing would need to be done to correctly balance all the voltage and current needs of the entire project.

### AC Converter

The circuit is very important because it is responsible for charging the battery via converting the AC voltage and current to DC voltage and current. This can be very dangerous because of the high current and high voltages of the AC outlet, and even more dangerous because of the risk of overcharging the battery which can cause it to explode and/or cause fires from integral leaks.

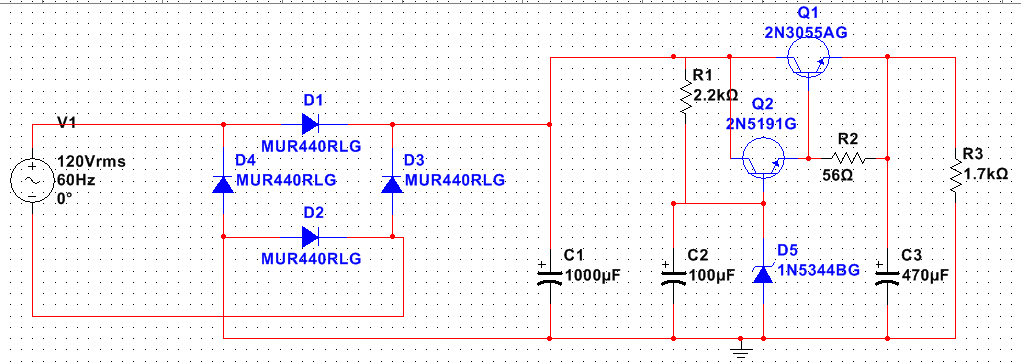


Figure 32: The prototype circuit for the AC to DC rectifier.

The battery luckily has built in voltage and current shut off, so if the input voltage and current gets too high the battery wont charge, or if the battery is full it cuts the charging off the battery to prevent an over charged situation. Regardless of these built in limiters, when we design the AC converter we are doing it so that that fail safe built into the battery is never triggered or never put into a situation where it would need to be triggered.

Safely handling the AC voltage is a big issue because we decided not to use a step down transformer. This means that all the AC voltage and load is going into our circuit. The prototype circuit is based on a voltage regulator setup, in that if the output voltage rises, the circuit would automatically adjust it back down to the voltage of the Zener diode. Alternatively if the voltage across the output were to drop with load then the circuit would automatically raise the voltage back up to the zener diode voltage.

This circuit is designed to hook up straight to the main battery for charging. The circuit has a higher than normal current output. This is great for charging the battery at a fast rate while at the same time not going over the maximum current charging level as specified by the battery. This circuit is however just a prototype and it would require some actual testing under real would application and real load scenarios. This circuit is designed to output about 7.7 V and about 2.5 A which is well within the safe range for charging the battery.

### Voltage and Current Restraints

While the exact power requirements of each component of the bicycle are still unknown, and will require thorough testing to determine accurate values, we are able to estimate ballpark ranges for component power needs. To begin, we estimate the values of the needed voltage and current, as applicable, to each hardware component. For some components, the input value is the important factor, the input current is more relevant for others. These estimations are examined in Table 12.

|  |  |  |
| --- | --- | --- |
| Component | Estimated Voltage/Current Input Needed | |
| MSP 430 Microcontroller | | 3-5V |
| Total LEDs | | 2.7A |
| 7-Segment Display | | 5V |
| LED Drivers | | 5V |
| RPM Sensor | | 5V |
| Blue Tooth | | 3-3.3V |
| Phone Charger | | 5V |

Table 12: Estimated Needed Voltage/Current Input for Each Hardware Component

### Battery Regulation

The main power source for the POV display and its components will be the 7.4V DC battery. As our POV bike will only use one battery at a time (with a supplemental battery being charged by the generator), there are some necessary battery manipulations that must take place.

The first element of battery regulation is adjusting the voltage levels being fed into different parts. Most of the hardware parts have a lower estimated voltage input than the 7.4 V that the battery supplies. This requires ‘stepping down’ the voltage as it moves from the battery to the input of each component. This can be done a few ways, but the simplest way is to use a basic voltage divider circuit. An example of a circuit to step-down the voltage to 5V is presented in the Multisim schematic Figure 36.

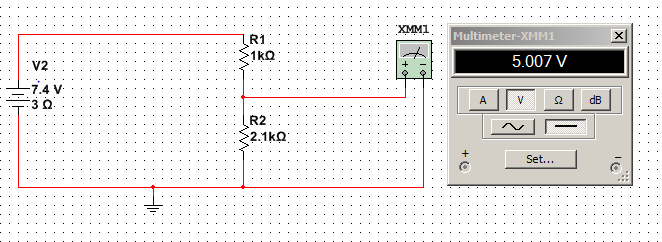


Figure 33: Battery Step Down Schematic

### Battery Switching

The second element of battery regulation will be switching from the primary battery to the secondary battery as the main power source when the primary battery runs low. This is a challenging task to ensure that transition between the two batteries does not cause a disruption in the power flow, or cause damage to any of the hardware components. One way to make the switch between the primary battery and secondary battery is to use a simple diode switching circuit.

The way that this circuit works is by the use of a switching diode. While the primary battery is on, the diode prevents current from flowing from the secondary battery. As shown in Figure 37, the Multisim simulation below, the voltage of the parallel terminals comes solely from the primary battery.

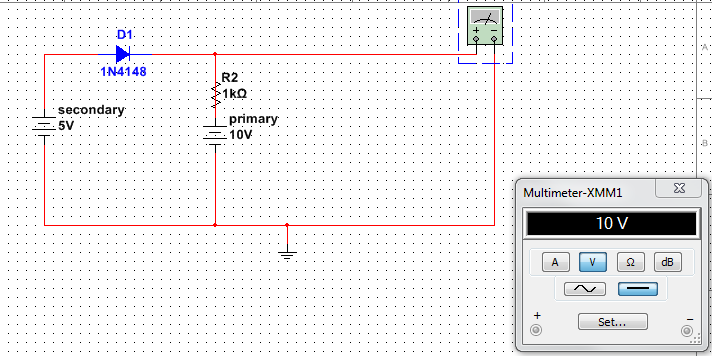


Figure 34: When the voltage potential at the cathode of the diode is positive, no current from the secondary battery flows to the multimetere

When the primary battery goes to zero, this provides a zero voltage potential at the cathode, and the secondary battery provides a positive potential at the anode. This forward biases the diode, as allows it to act as a short circuit through which current can flow easily. This essentially “activates” the secondary battery. As shown in Figure 38, once the primary battery goes to zero, this “switching” via the diode occurs, and the voltage represented at the parallel terminals is that of the secondary battery.

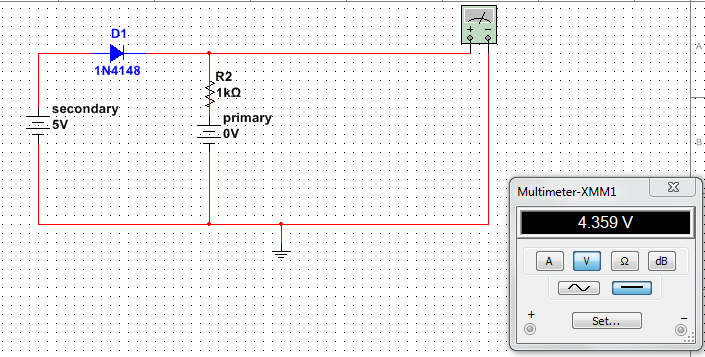


Figure 35: When the primary battery goes to zero, the secondary battery is able to send current to the multimeter This switching circuit is the basic model that will be used to automatically switch between the battery sources when the primary battery goes low.

### Seven-Segment Battery Readings

One main purpose of the 7-segment display is to output the levels of the primary battery, and to display when the battery levels are running low. To do this, there must be a way to measure and report the levels of battery. This can be done by estimating the time that the battery has been used, and comparing this value to the approximate life cycle of a fully charged battery. However, a more accurate representation can be made by actually measuring the battery levels. One way this can be done is by place a resistor at the output of the battery, and measuring the voltage across the resistor.

This voltage reading would be communicated back to the 7-segment display, and displayed for the user to see.

## Cellular Device Charging Station

As was mentioned in the extended features of the requirements, the implementation of an onboard charging system for cellular devices was carried out. The original idea for the cell phones charger was that it was going to be powered by the generator, syphoning some of the power from the generator that was mainly attached to the secondary battery. However after more experimenting and reviewing the output of the generator, we deemed this original idea not plausible, or effective. This was due to the low current being supplied by the generator. Another way to put it was that the secondary battery requires about 5.5 hours to fully charge, with an input current of about 1.2A. Unfortunately, the generator can only output about 0.5A (maximum) if the rider were biking at around 12-15 mph (a regular biking speed).

Additionally, if the phone battery was hooked up in parallel with the secondary battery, it would have to share the low generated current, further decreasing the current sent to the secondary battery. This would cause an increase in both the charging times of both the cellular device’s battery and the secondary battery.

To solve this problem, we came up with the idea of having a circuit which would switch its output to either the secondary battery or to the cell phone charger. The circuit would be designed to prioritize the cellular device charging, so that if a cell phone was attached to the charger all the current generated would flow to the cell phone. However if there was no cell phone plugged in, then all the current generated would flow into the secondary battery. This was a simple idea to fix our problem of sharing the current but unfortunately it does not fix the underlying problem that the selected generator can only provide a relatively small amount of current to effectively charge the devices quickly. But to acquire a generator which can supply the appropriate current levels would just be too expensive for our budget or it would be just too big to operate effectively without disturbing the rider.

The generator selected was indeed the right choice as was stated in previous section, because it does come with a built in phone charging station. This allows us the luxury of not having to design the voltage regular for the charger. The generator additionally comes with its own rectifying circuitry so we can just simply send the generated power to a USB port where we can attach any device to it easily. The generator is rated at 5V which is well within many cellular phones and devices charging range of operation.

## Mechanical Design

During design the layout of the bicycle went through several revisions before reaching the current arrangement. Originally the bike was only going to make use of a single micro controller, but it was determined that this would not work. Unless a very large power hungry microcontroller was used there would not be enough input pins to incorporate all of the desired features. Additionally this would have required many wires to be run into the rotating back wheel without tangling, which posed a pretty daunting mechanical challenge.

The first possible solution that was discussed involved separating the bike into two separate subsystems. The entire POV display, with its battery was going to be mounted within the rear wheel, eliminating the cable issue. The generator would have been connected to modular battery charger, so that when the display’s batteries died, they could simply be swapped out by the user. Additionally current battery level was going to be estimated by the handle bar box display using only software. Eventually, it was decided that this design cut too many corners, and too many features that needed to be included.

To overcome the lack of input/output pins on our chosen microcontroller, two separate microcontrollers will be used. One will be located on a PCB inside of the rear rim. This one will contain the Bluetooth subsystem and drive the LED arrays for the POV display. The other will be located inside a box on the handle bars, and it will drive the 7-Segment display. Additionally two separate Hall Effect drivers will be used. This further reduces the amount of wires being run into the rear rim from four to just to, power and ground. One sensor measures the speed of the rear wheel, and is used to time the LEDs in the POV display. The other is used by the handle bar box to estimate speed and distance. This solves the need for more pins. Another solution was still needed to prevent cables from tangling.

Three PCBs will be mounted to the rear wheel of the bicycle using plastic clips to isolate them electrically. A ground and a power wire from the battery will be run along the bike frame to the main PCB containing the microcontroller. To prevent the wires from getting tangled, a special connector needed to be developed. This connector needed to allow rotation inside the wheel, while keeping the lines outside of the wheel fixed. Two different avenues for accomplishing this were explored.

The original idea involved mounting a series of copper rings around the wheel hub to form a disc. There would be plastic between the rings to keep them electrically insulated. Brushes would ride along the grooves formed by the rings to conduct electricity while the wheel was rotating. This disc of rings would be allowed to rotate freely and the brushes would be fixed. A rough diagram illustrating the idea behind how this connector would have worked is shown in Figure 39.

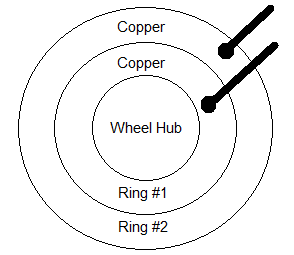


Figure 36: Brush connected diagram #1

However, it was ultimately decided that this connector would be a poor choice for use on a bike. The rim of the bike flexes as the rider goes over bumps and imperfections in the road’s surface. It was though that this would probably cause the brushes to temporarily move and disconnect the power to the microcontroller. Also since the side of the connector is completely exposed, riding through a puddle could create a path from the power wire to ground, causing a short circuit. A design without exposed brushes that could potentially move around was needed and eventually found.

A new connector was developed, this time with the rings in parallel. The input wires are ran through the side of the connector and connected to copper rings on top of the connector. Brushes are then looped around the copper rings to transmit power the microcontroller inside of the rotating rim. The looped brushes will prevent connection loss when the wheel flexes from bumps. The copper rings are also significantly less exposed to the elements than the previous connector. This new connector meets all requirements for use on the bike display. It should allow power to be sent to the microcontroller from the battery reliably. A detailed schematic of this connector is shown in Figure 40.

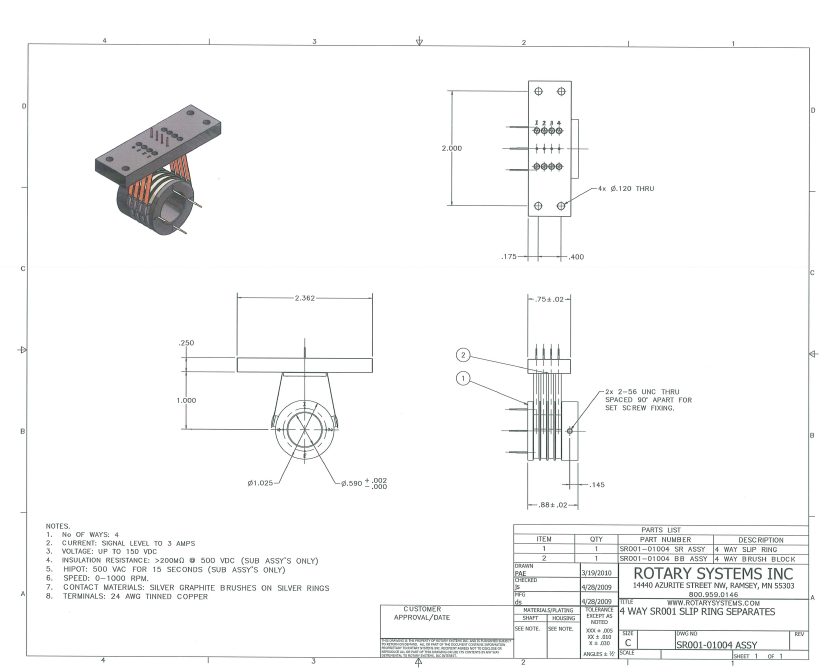


Figure 37: Brush Connector Design #2 (printed wIth permission from rotary systems inc,)

Unfortunately this idea also failed to give constant power connections. A third connector was the developed. This connector was had two stainless steel rings in parallel, one for power and one for ground. A brush connector from a car alternator was mounted to the frame. Springs in the brush holder applied constant pressure on the carbon brushes maintaining contact to the steel rings. This ultimately was also unsuccessful. With our materials and budget were unable to machine a ring with precise enough rings. Additionally the bike’s wheel itself is slightly bent. This caused oscillations that would occasionally cause disconnections.

A fourth idea, used one copper ring instead of two stainless steel ones. This allowed the two brushes to be wired in series. The brush connector only transmitted power. The PCB’s themselves were grounded to the bikes axel. This meant that the current bike grease needed to be replaced with a conductive graphite based lubricant. This new connector has much lower resistance. It also has much more surface area and two redundant brushes which ensures connectivity. This final brush connector design was successful. It is shown as figure 41.

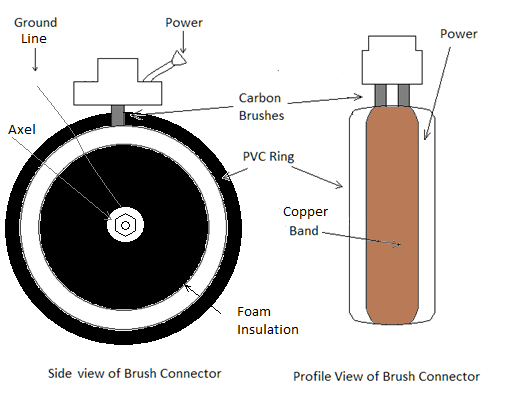


Figure 38: Finalized Brush Connector Schematic

The following section details how each component of the POV system will be physically connected to the bicycle. The security of these attachment points is very important to the overall function of the bike. Bikes undergo a large amount of vibration while being ridden, so the system’s components must be firmly attached.

The generator will be mounted on the right front fork of the bicycle. A special mounting bracket was included for the generated that fits well at this location. Putting the generator on the front wheel also reduces drag on the bike’s peddles, and reduces the amount of components be crammed into the rim of the back wheel.

The Handle Bar Box will contain the second msp430 which will be wired to a second Hall Effect sensor on the front. The 7-segment display will sit directly on top of the microcontroller, and the whole thing will be house in plastic box for protection. The box will be mounted on to the handle bars using hose clamps as shown in Figure 42.

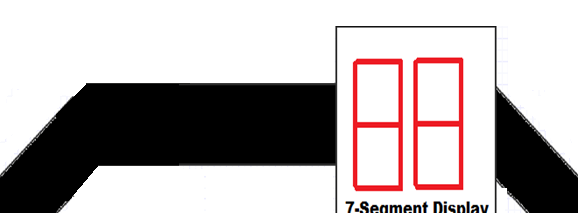


Figure 39: Handlebard Mounting

One Hall Effect sensor will be mounted on each bike wheel. The sensor used to measure speed for the POV display will be directly mounted to bicycle rim, and the magnet that triggers the senor will be mounted on the rear fork. This arrangement reduces the amount of wires that have to run into the rear hub. For the front wheel sensor, which outputs to the handle bar box, the sensor will be mounted on the fork, and the magnet will be mounted in the wheel. A diagram showing the positions of the three sensors is shown in Figure 43:

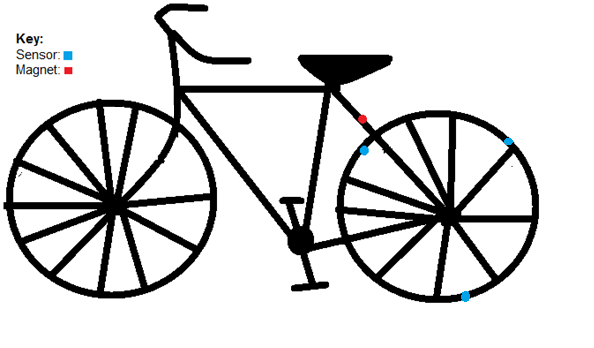


Figure 40: Hall Effect RPM Sensor Mounting

The battery will be mounted across the bicycle’s cross bar using hose clamps keep them in place. Since the batteries are the heaviest components being added to the bike, it was decided that they should be centrally located to avoid throwing off the bike’s balance. The central location also makes it easy to run the necessary wires along the frame from the generator to the battery and from the battery to the POV display’s microcontroller. A diagram of this is also shown in Figure 44:

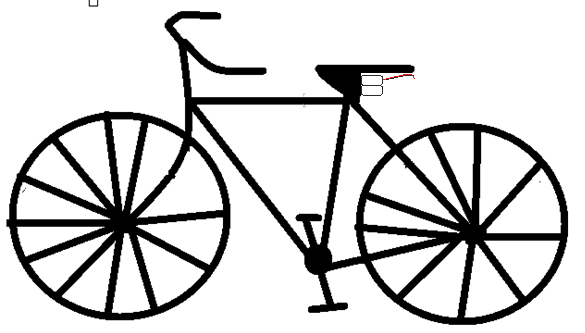


Figure 41: Battery Positioning Diagram

# Software Design

## Dataflow Diagrams

This section will break down all of the software sections from the highest to the lowest levels. All parameters will also be indicated. Figure 45 is the high-level software architecture block diagram. This shows the interactions between all software modules

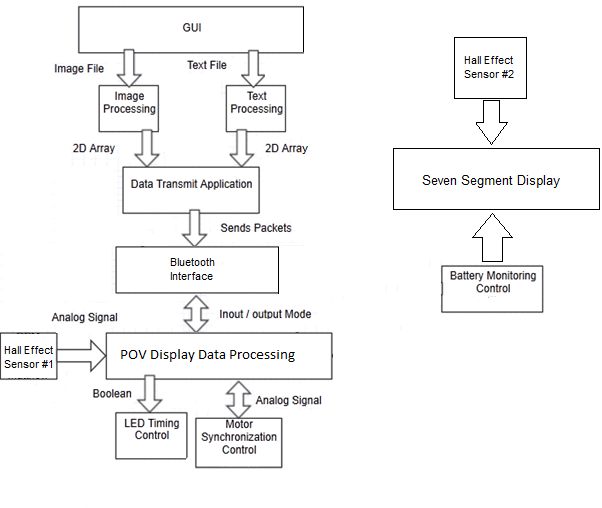


Figure 42: High-Level Software Architecture Diagram

The following block diagrams illustrate the sequence of steps in converting an image to be used on the display, and converting a text message. All images will first be converted to the PPM format. This is a very easy to work with file type, and should make the implementation easier. Please note, that to use make a text message canned images are mixed together to produce text. This just allows the program to convert faster.

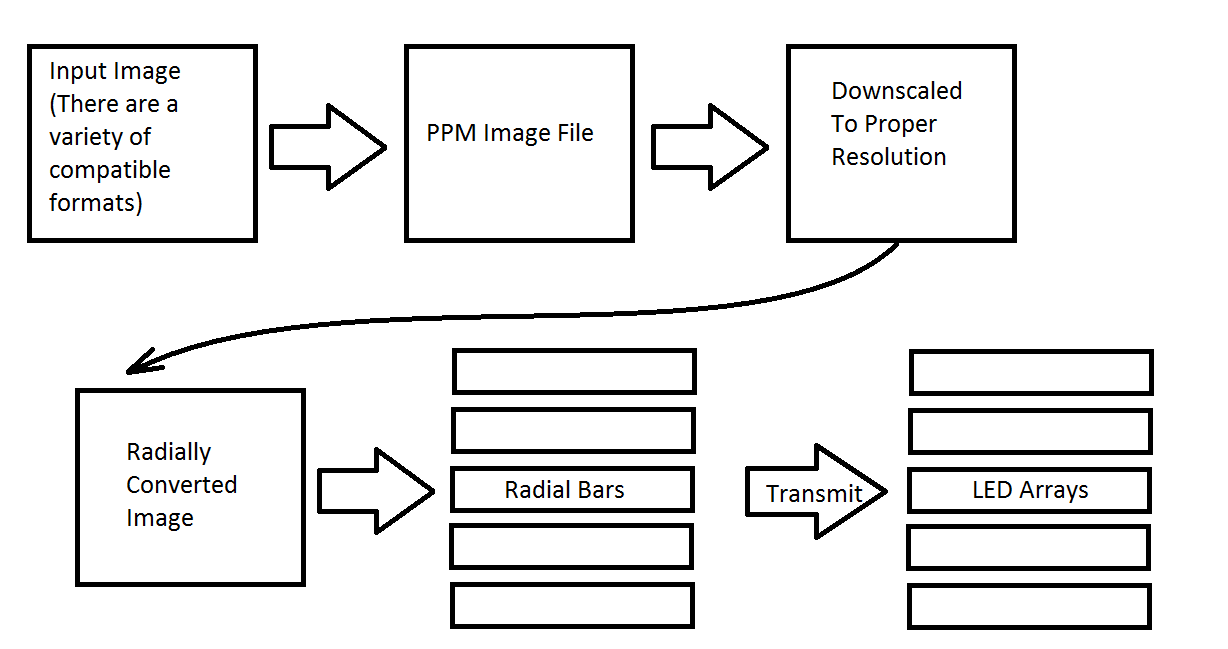


Image Conversion Diagram

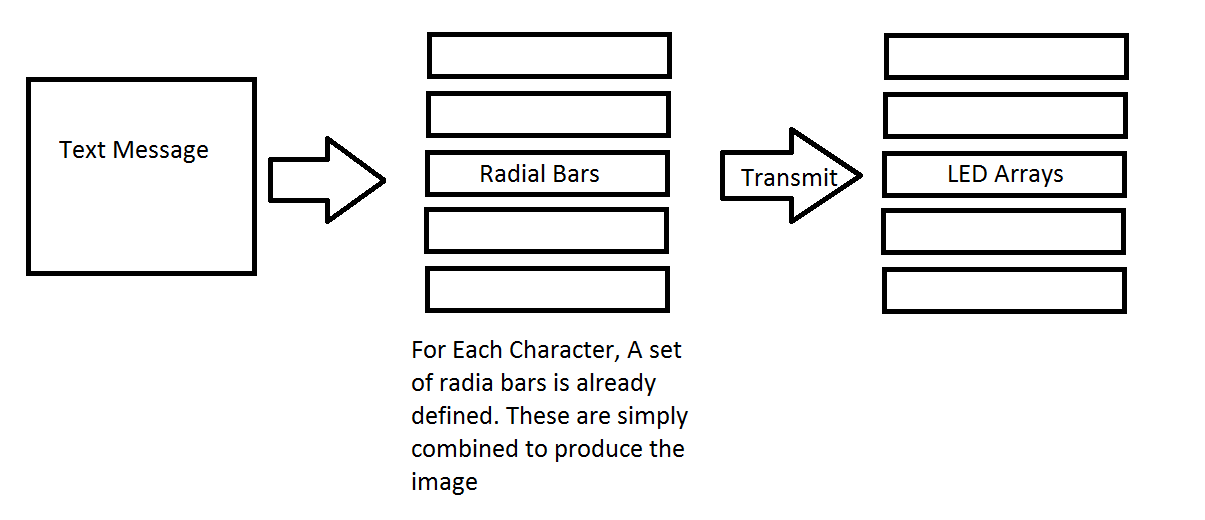


Figure 43: Text Conversion Diagram

## Driver Programs

The following section describes the driver programs of this project.

### Hall Effect RPM Sensor

The rpm sensor will be connected to a single input pin on the msp430. A continuously running “while” loop will be listening to the pin. If the voltage on the pin goes active high, a software interrupt will be triggered. This interrupt will increment an integer variable, count, which represents the number of times the wheel has gone around, over a set time interval. This time interval will be determined experiment. Using a running average of” count” will allow the program to calculate a relatively continuous RPM value, over a given time interval. The RPM value will be defined by the following equation:

RPM= Count/timeInterval \* 60

An RPM will be re-calculated every few. This value will be used in the speedometer and odometer functions of the 7-segment handle bar display.

### Bluetooth Wireless

The Bluetooth module will be driven using the MSP430s built in UART interface (MrChips)The MSP430 will see the data as nothing but a stream of characters. A parsing algorithm will translate these characters into the arrays that light up the LEDs and display the image. A block diagram illustrating this procedure is shown in Figure 47:

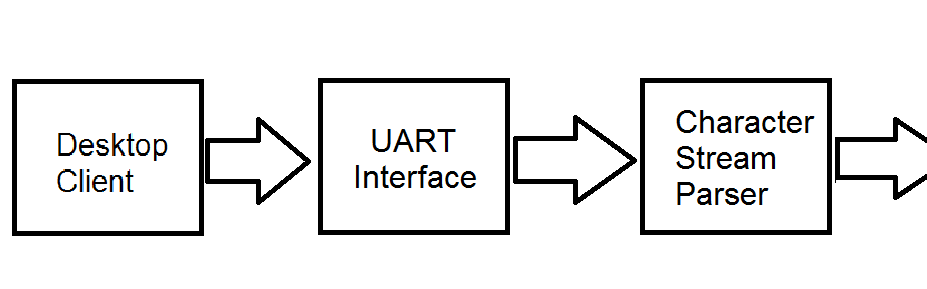


Figure 44: Bluetooth Subsystem Block Diagram

### 7 Segment

Typically, multiple 7 Segment displays are wired in parallel and time division multiplexed. Only one display is ever lit at a time, but the processor cycles through them so fast that they all appear lit. There is a limit to the amount of LEDs that can be driven this way. As displays are added, the amount of time each is lit becomes less. This can make them appear dim. However, since only 3 digits are required this is not a problem.

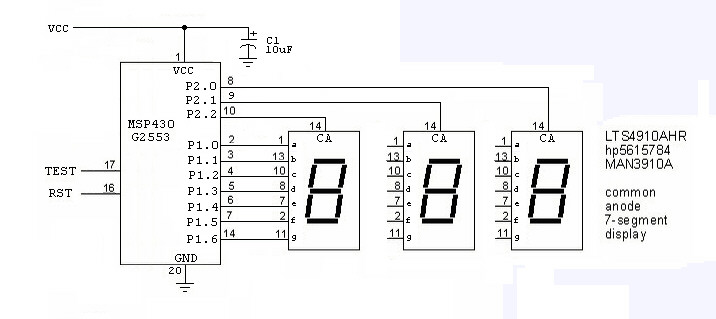


Figure 45: 7-Segment Display Schematic. Used with permission from:

<http://forum.allaboutcircuits.com/blog.php?b=559>

For the display, three Microtivity IS122 seven segment modules will be used. These modules will be multiplexed to form a three digit decimal display (Vividvilla, 2013).To conserve power, only one LED segment will ever be lit at a time. The lit up segment will cycle through so fast that it will trick the eye into thinking that the whole display is it up at the same time. This timing will be accomplished using a series of “if statements” within in a “for loop”. Within each statement will be the proper sequence that must be stepped through to represent a certain number. These strung together statements will light up the whole display.

The LED segments are green which will allow for increased visibility in direct sunlight. It has a forward biasing voltage of 2.2V. The seven segment display will be directly connected to the microcontroller. With input from the Hall Effect sensor that gives RPM, the 7 segment will be capable of displaying the following three modes of operation:

* The first mode will display the bicycles speed. This will be calculated using the following equation in miles per hour:
* The second mode will record the total distance traveled according to the following equation:
* The third mode will display the current battery level of the bike. To calculate this, an analog to digital converter will measure the voltage across a small resistive load connected to the battery. This will be divided by to give a percentage. This percentage will be displayed. Dataflow Diagrams

## LED ARRAY Prototyping

Before we could actually start programming our LEDs there was some prerequisite knowledge. We needed to understanding the problem. If we have a spinning disk with a single strip of LEDs, the LEDs will need to turn on and off rapidly. So then we needed to understand in what way do the LEDs turn on and off, clearly it’s not random. The simplest form of LED array manipulation is to turn on all the LEDs on the strip.

To do this, we assigned each LED controller pin to a corresponding microcontroller pin. There are 5 pins, not including ground and power. SIN, GSCLK, SCLK, BLANK, and XLAT. The important part here is that SIN sets the data, and the SCLK shifts the data. For simplicity, let’s say we have 8 LEDs, and the data required to turn on an LED is binary. So, in figure 46, which represents an ‘A’, to load the leftmost bar we would perform the following actions, SIN(LOW), PULSE(SCLK), SIN(HIGH), PULSE(SCLK) x 6, SIN(LOW), and PULSE(SCLK). This will load the very first column that represents the letter ‘A’. In reality each of these LEDs requires 36 data bits each that determine its brightness and color.

For each ASCII character we have a constant char double array that merely stores each letters position data. It says which of the 8 LEDs are on and off. There is a separate array that stores the actual “data” for the colors and brightness. So in our update LED function, it first checks if the LED should even be on, if it shouldn’t it breaks out and tries the next LED. If the LED should be on, it will parse the color data one bit a time and set the SIN accordingly. Every LED will always be the same color and same brightness with the current design

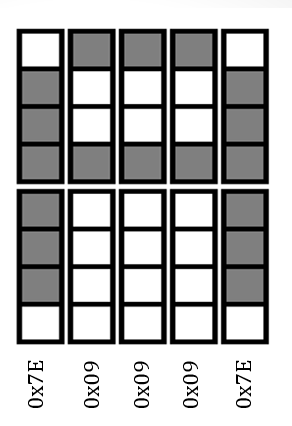


Figure 46:an image of the letter ‘A’ and the hex values that IT representS.

So now that we are displaying the letter ‘A’, we can manipulate it even further. Let’s say our message is “Hello.” There are three effects we can do. We can increase the width of each individual character, figure 49, we can increase or decrease the space between each character, figure 48, and we can position the entire message anywhere on a 360 degree circle, figure 47.

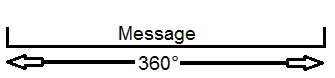


Figure 47:Asjusting message position

C:\Users\Mammoth\Desktop\timing2.bmp

Figure 48:increasing spacing between letters

C:\Users\Mammoth\Desktop\timing3.bmp

Figure 49:The letter M, increase width.

Of the three forms of timing each one can be faked without actually taking any timing measurements. First to change the message position, whitespace can be added to the front of the message and that will have the same affect. For letter spacing, in the double array that stores all of the characters, increase the number of columns by X, and set its value to 0x00. To increase the space between columns, merely add the whitespace between each data value that represents the actual character.

The proper way of having these same effects is to calculate the number of clock cycles that happen within a single revolution of the tire, and delay in the code accordingly (which is easier by the way). It could be taken a step further and we could find the number of clock cycles per degree. With this, we could have a delay function that takes in the number of degrees as an argument. This part of the project was probably the most difficult.

## GUI Prototyping

This will discuss the route we took to prototype our GUI. We’ll illustrate some of the GUI’s features via UML diagrams. The first diagram we drew out was a use case diagram, so we could see all possible user actions and compare it to our requirements. Since we’ve predefined all user actions already, we must then match all of our GUI components to fit this. Refer to Figure 49.

For the use case diagram, the user can perform four distinct actions. They may choose a color for the LEDs, type text to display, clear the display, and finally upload their message. The use case diagram is simple, and this is exactly what we wanted for the user, simplicity. In our original prototypes, they may have been a bit complicated, and a bit cluttered.

Diagram 50 shows what our GUI actually looked like. The GUI has three basic functions, color, text, and uploading. Using a slide bar is the first/second best way for a user to change the color. The alternative would be using a color wheel. The most immediate advantage to that is that the user can observe the color being mixed, this is an extremely strong advantage, but much harder to do in code. The downside to this is that brightness would need to be controlled separately from color selection, which adds more components and complicates the GUI further. Slide bars are very easy to incorporate, but it’s up to the user to understand the color wheel and mix their colors correctly.

There is actually a hidden feature by using slide bars, and it was a pleasant surprise. We actually have total brightness control. To have a dim red, only slide the red bar a little bit, and zero out green and blue. The three boxes to the right of red, green, and blue are actually functional as well. They require an integer between 0 and 4095. Should the slide bar change, this box will be updated with its current value. Should the integer box change, the slide bar will automatically be readjusted.

To give the LED array an actual message, the Bike Message text field should contain only ASCII characters and be no longer than 100 characters. For a reasonable message, it should be no longer than 25 characters. Unicode could be implemented, and if it were, Chinese and Japanese could be displayed as well.

The upload button has quite a bit of work behind it. From the users perspective they would never know. C#’s .NET library handled all of our Bluetooth transferring one byte at a time at 9600 baud. It will update both the text and color of the LED Array that this GUI controls. The clear button is a special case of the upload button. The clear button will in fact change the color of your display, it just merely sets the message to “”, a blank message, this cleared the display. The rainbow text button in the image was a feature we wanted to implement to have each character of the display alternate colors in a loop of ROYGBIV.

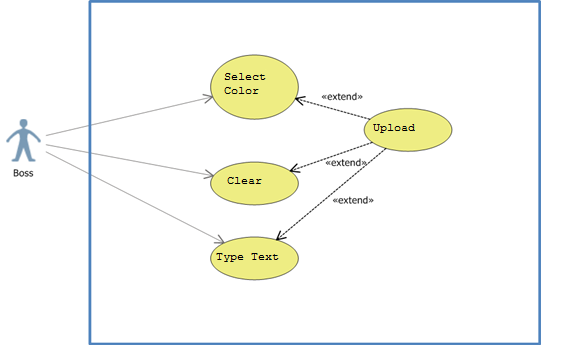


Figure 46: Diagram is used to show all possible actions a user may take in the GUI application. Made in Visual Studio 2012.

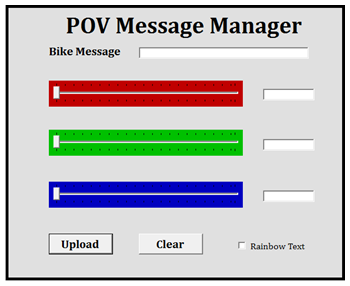


Figure 47:This diagram shows a prototype of what the GUI will look like after the GUI has been created. The graphics of the GUI are all made from scratch by us, group 29. We have no intention of using graphics from any other people.

To actually build this GUI, we used C#. To build a GUI in C# is very simple, it’s completely visual and components are added by dragging and dropping them into a visual workspace. Each component can be customized further by clicking on it, and adjusting its properties in the properties panel. Just about everything about that particular component can be changed within that panel; font, color, size, specific attributes, name, or default values.

Double clicking on a component will bring up the method that controls it in the code. From the method that shows up, we can finally have the button or slide bar “do something.” To the right of the slide bars there are text input boxes. Their function is to display the current value of the slide bar; the text boxes themselves may be edited to change the value of the slide bar. If the value typed in is larger than the maximum value of the slide bar, it just sets the value as the maximum. The integer value of each of these colors in stored in a global variable within the GUI class. These color integers need to undergo some conversion now. On the microcontroller end, it expects two bytes per color. To convert an integer into two bytes is fairly simple. We limit the maximum size of the integer to be 4095, which occupies 12 bits. To get the first byte, just assign the byte value to the (integer & 0x0FF). To get the second byte, the integer should just be right shifted by 8.

At this point we have 100% of our color data, we now need our message. To do this we need to create a new method by double clicking on the “Bike Message” text box. Within this code, it’s straightforward to extract the string from the text box. The actual message should be stored in a global variable. We need to modify this string just slightly before we can actually send it off. For the microcontroller to know when it has a complete message, we used a termination character. In our case we chose the character ‘~’ to end our message, since invisible characters seem not to work (such as ‘\n’, definitely strange).

The Upload button is the brain of this GUI and gives this GUI purpose. The Upload button merely creates an instance of our “Serial Port Communication” class and calls the appropriate methods to send the data. We actually use two different send data methods, one that sends bytes and one that sends characters. We had to split this up mainly because of how C# operates. Brace yourselves, what we’re about to say is absurd. In C# if you declare a character, you may not store a hex value into it, such as c = 0x0F. Likewise, if we declare a byte, we may not store a character in it, such as, b = ‘A’. They are both 8 bits, yet they make this distinction. Thus, we have two send functions (within “Serial Port Communication”). This class is written using the .NET library that accompanies C#. This class is open source and is fairly simple in retrospect. The code that makes it up is fairly short too. The Clear button is 99% identical, except the message is blank. Refer to figure 51 to see the order in which the data needs to be sent.

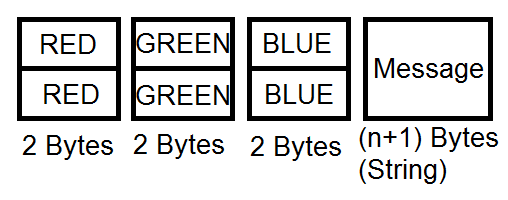


Figure 51:This shows the order in which the Gui sends the data, and the order the microcontroller expects.

# Hardware Testing

The following sections describe the testing and validation of the prototype of the main components of the POV bicycle display.

## Mechanical Testing

After all of the electronics are mounted on the bicycle, it must be tested for mechanical soundness. The bicycle will be ridden for five miles to ensure that all of the parts are solidly mounted, and are unaffected by the stress and vibrations caused by riding. If any stress points are found, corrections will be made. Additionally the brush connector used to transmit power from the batteries to the microcontroller must be thoroughly tested to makes sure that it maintains a constant electrical connection, even after extensive use.

## Microcontroller Signals

When testing the micro-controller signals we did simple I/O testing. We made sure that when we turned on a pin that there was sufficient current and voltage to that particular pin. This testing was closely related to and done conjointly with the software testing of the micro-controller. We tested the noise to signal ratios of all the pins which would be in use. We tested the loading effect of cascading several of the LED drivers. We tested that the microcontroller itself had the correct input voltage levels at different loads. We added capacitors to sources and special I/O pins to reduce bounce and surges. The micro-controller testing was a simple but important job.

## LED Driver Testing

The second stage is going to involve testing if each of the LED drivers will be working properly while all of their channels are occupied with LEDs. We will be monitoring whether or not all the ICs channels will have the correct current required for the driving of the LEDs. The third test will involve using multiple drivers daisy chained together with each of their channels connected to LEDs. We then will continue the driver testing with the altering of the PWN steps of individual LEDs, adjusting their brightness up and down the 12 bits steps of the PWN. Next we will examine the dot correction feature of the TLC5940 making sure that it is functioning properly. While testing the dot correction and the PWN feature we will record the best combination of the both for the best output of certain colors. Additionally as the tests are done we will be monitoring the input current and voltage levels of the ICs to see if the devices will be functioning as they are expected to. The final test for the drivers will be when we connect them in cascade and test the serial data streaming from the microcontroller to all of the drivers

## LED Testing

The testing of the LEDs is going to be simple. We are going to basically construct a test circuit with which we can place each of the LEDs into, making sure that they turn on and that they would be able to handle high frequency color shifting and changing. We are going to test different combinations of current input to the individual diodes in the RGB LEDs to test the transitions from one color to the next. We are then going to give different duty cycles to the individual diodes to see how it will affect the brightness and color composition. We will experiment and test the dot correction feature, streamlining the levels for the best color effect. We will be monitoring and testing the maximum and minimum current levels for LED operation. Finally we will approach the LEDs from different angles to deduce the range of the view angles for the best placement of the array on the bicycle.

## Power Testing

Testing the reliability of the power sources of the hardware components of the POV bicycle is one of the most important aspects of this project. The following sections highlight what kinds of testing we will perform on different power aspects of the display.

### Generator Testing

There are a few tests that will be performed in order to validate the reliability of the generator, and to ensure that it is operating properly.

First, in order to ensure that the expected current outputs of the generator match the expected outputs for each bicycle speed, we will run the bicycle at different speeds (as determined by the RPM sensor). Using a multimeter, we will measure the current being output by the generator at each speed, and compare it to the expected value. If the current output is drastically different than what was expected, then we know that the generator may be defective. If the values are similar but not exact, and these differences are generally uniform across different speeds, we will understand what adjustments—if any—need to be made to compensate for these differences.

Second, we will compare the current at different radial speeds (revolutions per second). Understanding how the current output changes as the bicycle speed increases or decreases will allow us to approximate the ideal speed of the bicycle for solid performance and reliability of the display.

Finally, we must ensure that the generator will be able to withstand variations in environment. While the POV bicycle will not be able to operate in rainy weather, it should be able to handle Florida humidity and occasional contact with small amounts of water. This is so that we can predict what would happen to the POV display in case the bicycle rides through a puddle, for example, and water gets splashed onto the devices. Thus, the generator will be secured to be mostly waterproof, and tested for its security against splashes of water contact.

### Battery Endurance Testing

Battery endurance testing is to examine how long the battery powering the POV display will last under certain loads. To do this, we will first simulate the battery and several loads in NI Multisim. That will give us our expectations for how much power should be delivered to each component after battery adjustments are made.

Knowing this information, we can measure and test how long the battery lasts while it’s operating at several different loads. We will measure the battery life when all hardware components of the POV bicycle are in full operation. These tests will be performed for both the primary battery and the secondary battery.

We would like to have the primary battery last for at least two hours of consistent bicycle riding before running low and being replaced by the secondary battery. We would like the secondary battery to be able to power the bicycle for at least one hour of consistent riding.

The main problem with testing the battery endurance is that we will not be able to test all of the power-consuming components for battery endurance unless we ride the bicycle will all components running for several hours continuously. This is impractical because multiple tests will require multiple hours of consistent running. Instead, what we can do is make estimates by simply measuring the battery endurance while all power consuming parts are turned on. This would exclude devices that operate while the wheels are spinning such as the RPM sensor. However, doing this and measuring the time it takes for the battery to deplete, as well as the rate of depletion, will give us a fair estimate for how long our battery should last.

### Battery Charger Testing

The primary battery of the POV bicycle will be charged by an AC power source charger. This component will also require testing. First, we will test our AC-to-DC converter design in Multisim to ensure that we have designed the circuit properly, and that it is giving us the DC value that we expect to see. Knowing what values we should expect, we can move on to physical testing of the charging system.

.Since the batteries we have chosen have controls inside of them to protect the battery from overcharge, we must test to make sure that the battery stops charging once it is full. We can do this by attaching a resistive load to the battery, and measuring the voltage across the load as the battery is charging.

Additionally, we will measure how long it takes for the battery to charge. The primary battery will be connected to the AC power source while a multimeter measures the voltage across the connected load to determine how long it takes for the battery to reach its maximum voltage.

Since the secondary battery is charged by the generator, we must also test this battery charging independently. Similar to the primary battery, we will connect a load to the secondary battery and allow the generator to charge it. Since the generator operates by pedal power, it will be nearly impossible to manually pedal the bicycle wheel consistently for the entire duration of the battery charging. Thus, we will measure the battery charging level by a consistent bicycle speed over a short period of time. Then, we will use the measured rate of charging over the short time period to estimate how long it would take the entire battery to charge, and how much pedal power it would require.

# Software Testing

This section describes the testing of the software component.

## GUI Testing

Testing the GUI is for the most part straight forward, except for one part in particular. The GUI is made up of two parts that need to be tested, graphical and functional. Graphical testing is by far the simplest. We have 4 components in our GUI, a textbox, slide bars, and upload and clear buttons. Of these parts, only the buttons and slide bars have moving parts.

Regardless of function, when a button is pressed, it’s animation that represents a press should be smooth. So the test is to press the button in different ways, quickly, long press, while moving the mouse, with the GUI frame in a different position, while the button is halfway cut off the screen, with different text and font, or for any other graphical change to the button. The test is mainly just trying it and seeing what it looks like.

The slide bars should increase smoothly, with one integer of accuracy. The tick marks surrounding the slide bar should look cosmetically appealing. One slide bar should not affect any other regardless of GUI position, mouse velocity, slide bar position, etc. Just like the button, the test will pass if it “looks” appealing.

Sending data was the most important attribute to test with this GUI. To test this, we actually first did a couple of sanity checks. We downloaded a program called Putty for serial communication and sent the bits directly to the microcontroller and ensured the effect we wanted occurred. This completely eliminates Bluetooth from the testing. Once this was working, we plugged in our Bluetooth and did the exact same test with putty with the serial port, except this time we connected to the port Bluetooth was on. Only after this was working did we actually test our GUI’s upload button. The Upload button did not work on its first try; some of the main problems we encountered were the data being in the wrong format, or type.

## General Software Testing

General software testing refers to standard practices we took while programming our display and GUI. To begin this type of testing, we first need a base point. A base point is a piece of code to return to whenever there is severe trouble encountered. The very first base point in any project is the main loop. Have the microcontroller run an empty program and confirm that it builds, compiles, and uploads to the microcontroller.

This base point is dynamic, so of course we won’t always revert back to running an empty program. But we do need to keep a history of base points and store them in our storage database. Ideally they will be stored on a reliable server, but we always kept local copies as well. One question may be, when do we switch out base points? And this is a very good question. There are three answers, whenever we “feel” like it, daily, or whenever a new feature is added. We never changed our base point on a daily basis because we would end up with a lot of code to name and sort through. Changing it on a feature and feel base worked out extremely well.

Some testing issues that came hand in hand with this testing method are natural for any project. This issue is the naming convention. For example, let’s say we name our first base point “start” and we add a feature to light up a single LED. This new base point might be named “Single LED”, and this is fine. However in our work environment, Windows, it will automatically sort our files by name, which would reverse the order of these two items. Ideally we would want them in chronological order. Yes, we could sort by date modified, but this has problems of its own. If we store all our files on a database as well, and update the database with both of these files at the same time, they date modified will no longer be in chronological order. So the potential exists to lose track of which base points came earlier, and later. One solution to this is to add a year / month / day to the name of the base point, but this is not ideal.

## Handle Bar Display Testing

To test the handlebar display, we would cycle the numbers 0 – 9 on the each of the 7 segment displays. This will confirm that they are wired up correctly. Once that has been confirmed, we will test the 7 segments in pairs of two, to create a two digit number. Should the test pass up to this point, we will sample some random maximum digit numbers. If the test passes up, then we will consider it working.

## Wireless Testing

The wireless communication subsystem is extremely important. It is the bridge between the image conversion client software and the microcontroller powered display. First it will be thoroughly tested by sending basic text messages. These messages will be sent to the display using a variety of Bluetooth devices to ensure compatibility. No messages should ever be dropped, and the message received must match the message sent. These messages will also be sent from a variety of ranges, to ensure that long distance transfer is possible.

Once these basic text messages are working correctly, the specialized packets that contain the arrays corresponding to the converted radial images must undergo the same rigorous testing process. No corrupted, altered, or dropped packets are allowed. Once this level of consistency is achieved, the wireless system will be considered complete.

## Text Processing Testing

All features related to displaying text will be tested. Since each character will be hard coded, all supported characters will be sent to the bike for display using the GUI. Once all characters are confirmed to work individually, longer messages of up to the maximum of 12 be tested: “UCF KNIGHTS!”, as shown in Figure 54:

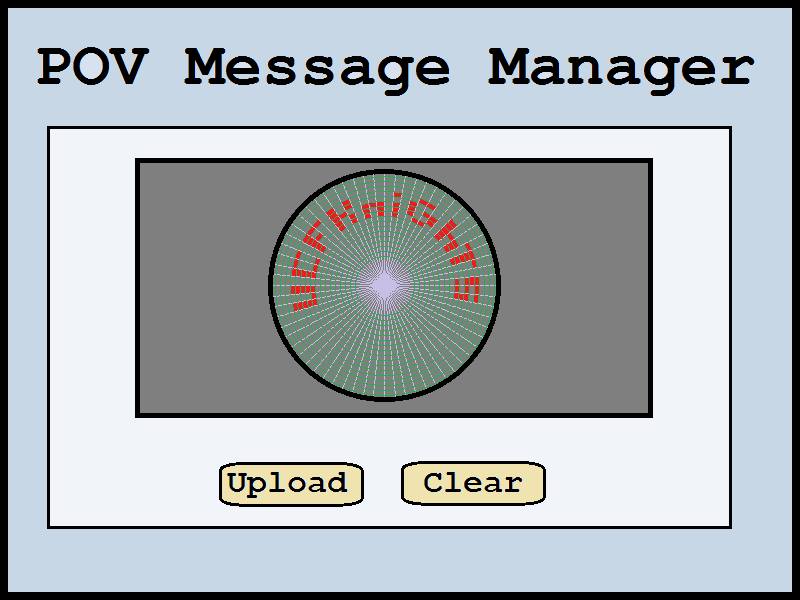
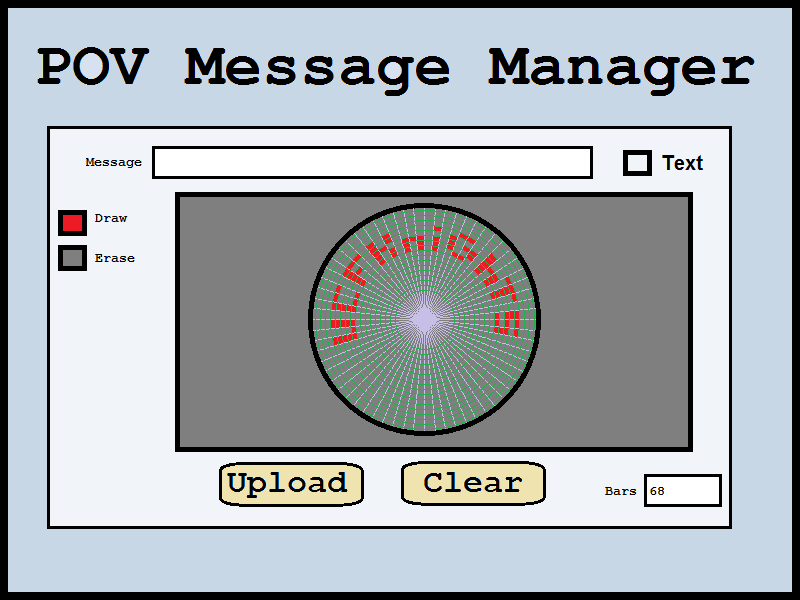


Figure 48: *Expected Output of Text Display*

In addition all supported font colors of the messaging program will need to be tested. It is important that the message on the bike must be legible and appear to be stable while being ridden. Once the text processing has been thoroughly tested across all the listed scenarios, this subsystem will be considered complete.

# Software Design Summary

The software portion of this project has been divided into three distinct parts, which will all be summarized below.

 The first part is a client based program that allows a user to prepare a text message or image file for use with the bike’s POV display. This program itself is further divided into several subparts: a GUI, image conversion algorithms, text conversion algorithm, and wireless transmission. The only part of the client software that the user directly sees is the GUI, and it is fairly unique. The first GUI is a very simplified implementation. Essentially, the user can do two things; browse for an image or type in text. The second GUI has an animated wheel that the user can to draw on. By design, the user may add in additional bars around the wheel. These bars represent when the LEDs are supposed to update. The user is given the capability to adjust the number of these bars, which updates the animated wheel. This suggests that eventually the user may have a hard time drawing on the wheel if the gaps become too narrow. A new feature was added to allow the user to type in the text that they want. A preview of this text will be shown on the animated wheel. Refer to Figure 55 for an illustration of the GUIs.

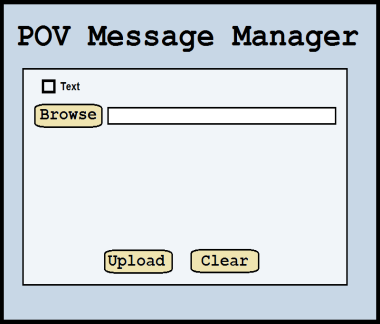


Figure 49: These are the same images from the Design portion of this document. The Image on the left is the simplified GUI. The Image on the right is the creative GUI.

The image processing system converts most common image types into a special format to use with the bike in a series of steps. The first step converts that image into a PPM. This format allows for quick and easy parsing of the image for the rest of the conversion process. It also means that the remaining code only needs to be optimized for one image format.

Next, the PPM is downscaled to a small resolution in order to fit within the display. This process is accomplished using Bilinear Interpolation. This algorithm was chosen because it represents a very good compromise between speed and quality. When pixels are removed to down scale the image, a weighted average of the missing pixel and its neighbor’s takes replaces it. This prevents total data loss when a pixel is dropped, since its color values still have an effect on the remaining pixels.

After the image is downsized, it must be converted into a radial representation that better matches the layout of the LEDs used in the display. This is done by “drawing” lines through the center of the image of various slopes. Any pixel that the line crosses through, or is close enough to is added to a new array representing the new. Pixel averaging is then used to increase or reduce the amount of pixels in these new arrays to exactly sixteen. These arrays contain the color values needed when LED array is at certain angles with the display. They are what will be sent wirelessly to the display’s microcontroller

The text conversion processing is far simpler than the client software. In this case, the user would input a short message and select a font color. All supported characters will have corresponding hard-coded arrays of the same form used in the image display algorithm above. These arrays will be used to directly construct an image by arranging the arrays in the same order as the characters in the message string. This process will be considerably faster than converting an image, but the data sent to the display will be in the exact same format.

The second major software subsystem is software used to by the POV display’s microcontroller to light up the LEDs in sync. After the input is converted by the user into the format described above, it will be sent over a Bluetooth wireless connection to the microcontroller. The data will be passed to the controllers UART pins, and into a buffer. This action is very similar to what happens during a wireless telnet session. The arrays are sent in-order, one at a time. Once they have all arrived and been stored in memory the microcontroller will being displaying them. Each LED array corresponds to the color values used when the wheel is at a certain angle. This angle will be estimated using data from the first Hall Effect sensor to calculate the wheels rotation in revolutions per minute. This value will be the key to timing the color changes on of the LED arrays. Essentially, all the microcontroller will be doing at this point is running a loop that fills the LED array with color values from memory in a timed pattern. This will continue until the microcontroller is interrupted by a new input message or turned off.

The third and final piece of software is used by the microcontroller inside of the handle-bar box. There are three devices will be inputting to this microcontroller. The first is an Analog to Digital converter that is measuring the battery voltage across a small resistor. This will allow the software to estimate the remaining battery by dividing the load voltage by the batteries expected output voltage to produce a percentage. The other device is a second Hall Effect sensor that measures the revolutions per minute on the front wheel. This minimizes the amount of wires needed to be put inside of the rear wheel while providing more accurate information to the handle bar box. The box will use that value to calculate speed by multiplying the current RPMs with the wheel’s circumference. Additionally the box will calculate distance by keeping track of the total number of times the bike has rotated and multiplying this times the wheel’s circumference. The third and final device on the handle bar box is a simple toggle switch. When the switch is pressed the device will alternate between the three calculate values on the 7-segment display. The 7-segment display itself is lit up only one segment at a time to save power and minimize the number of required output pins.

In summary, the software will consist of three parts. The first part is a client program that allows the user to prepare input for use with the display. The client will then wirelessly transmit the converted data to the microcontroller. The second piece of software governs all of the functions of the first microcontroller. This includes receiving data over Bluetooth, reading data from the first Hall Effect Sensor, and timing and lighting the LEDs correctly. The third piece of software governs the second microcontroller located inside the handle bar box. These functions include a toggle switch, a second Hall Effect sensor for measuring speed and distance, an analog to digital converter for measuring the current state of the battery, and a 7-segment display for outputting all of this information. Together this software should form a seamless system that transforms the bike into an effective mobile billboard.

# Operational Instructions

To power on the bike, the user needs to connect the battery to the connector located under the seat. Then set the switch located under the seat to the on position to activate the bike. If the red light located on the Bluetooth module is blinking, the bike is ready to receive messages.

To operate our Persistence of Vision device, the user must have a computer with any version of windows installed; Mac or Linux will not work. They must then download our GUI application that accompanies our Persistence of Vision kit. The GUI assists in controlling the LED array on the bicycle. Once the application is opened, the control screen should show up. This screen is the only screen the user should see for the duration of the session. From the control screen the user has a couple of choices.

The first thing the user does is enter a message into the textbox. Next they can set the messages color by adjusting the three slider bars. The bars represent standard red, blue, and green color values. If the user wants to enter a more precise color, they can type an exact value in the text box adjacent to each slider. The user can then click the upload button, and after a short delay, the specified message will appear on the bike. Should any error occur during the uploading, the application will let the user know about the problem. In this situation, the user must restart both the LED array and the GUI. The user can be sure the device was reset properly when they see the default message on the bike. Lastly the user can press the “clear” button to erase their message. The user should then switch the bike off to conclude the session.

To recharge the battery, remove it from the slot under the bike seat. Charge it by plugging it into a standard wall outlet with the included AC adapter. It takes approximately 10 hours to reach full charge.

# Administrative Content

As with all projects, the design and development of this Persistence of Vision bicycle requires a large amount of organization and planning. The following sections discuss the logistics of this project, including an overview of the team financial and decision making processes.

## Research and Design Methods

The way in which our group decided to do our research is in the following way. First, on our own, we would do some general research on our topics, via the internet. Then at some later time we would all meet up and discuss our research. During these meetings we expose advantages and disadvantages of the topics we've researched, in hopes of reaching a concrete design. If it turns out that the research we have come up with is not viable for the design but still reasonable, we would still add it to our paper. In the paper, it would discuss why it appeared to be a valid answer to the problem, and why it doesn't work.

The easiest way to start working on a concrete design is to have a list of parts that we intend to use. By doing this, we can easily show how everything is connected and spot possible defects. One possible defect could be that the microcontroller doesn't have enough I/O pins. So either a new microcontroller would need to be selected, or a new piece of hardware would need to be added to extend the number of I/O pins available. Even adding in another of the same microcontroller.

As far as the software design goes, coordination especially critical. Even if all the hardware was set up perfectly, the project would fail with faulty software. Some of the most important software elements is having each module agree on a certain data structure when transferring data. For example, if we were to represent an RGB color with a total of 36 bits, would it take up 4.5 bytes, 5 bytes or would it take up 6 bytes? All these troubles will be void with proper communication at group meetings.

The technique that is about to be mentioned we would recommend to any group in senior design. Whenever we find a really good source, one that we do not want to lose, and it happens to be on the internet, we save the .html file to our computer that way we don't have to hope their website stays up. Typically, people in general assume that a website would never shut down, but as some of us have found out, they can. The website in our case eventually came back up, but we can never be too sure.

## Budget

The development of our persistence of vision bicycle is expected to cost over $600 and no more than $1000. Table 13 gives a breakdown of our projected costs, and the following section provides a brief overview of each component.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Quantity | Cost (each $) | Cost (total $) |
| LEDs | x100 | 00.15 | 15 |
| LED Controllers | x5 | 05.00 | 25 |
| Power Supply | x1 | 70.00 | 70 |
| Bike | x1 | 00.00 | 0 |
| Micro-controller | X2 | 50.00 | 100 |
| PCB | x1 | 200.00 | 200 |
| Bike RPM Sensor | X2 | 40.00 | 80 |
| Bluetooth Module | x1 | 30.00 | 30 |
| Battery | X2 | 40.00 | 80 |
| Generator | X1 | 60.00 | 60 |
| 7-segment modules | X1 | 5.99 | 5.99 |
| Mounting hardware |  | 20.00 | 20.00 |
| Wiring |  | 10.00 | 10.00 |
|  |  | Total: | 675.99 |

Table 13: Budget

 **LEDs:** This is the core of our project. The LEDs are the light source which will create the images on the bicycle through means of persistence of vision. We have estimated that approximately 100 LEDs will be used. However, since LEDs are relatively inexpensive, we are able to have some flexibility in purchasing more LEDs as needed without greatly exceeding the budget.

 **Battery:** Two batteries will be needed for this project. The first will be used to power the LED setup and its associated microcontroller. The second battery will be charged by means of a generator while simultaneously powering the seven-segment display and its microcontroller. When the primary battery runs low, the batteries will be swapped.

 **LED controllers:** An LED controller will abstract some information about the LED as to make programming easier.

 **Power supply:** A necessary component to charge the display components while the bicycle is not being used. It is possible the power supply could be designed from scratch, but it is listed here in the parts in order to get an accurate upper end of our budget.

 **Bike:** The bike is project specific, persistence of vision can be done in many ways, and we intend to use a bicycle to show its effects. We intend to use both wheels to enhance the effects of POV. This bicycle comes to no cost to us as it will be donated by a group member.

 **PCB**: The PCBs are where all of our electronics will be, and will provide the appropriate connections to all those electronics. As PCBs are relatively expensive, we will try to limit the design of our bicycle to only use one. However, if it is determined that a second or third PCB would improve the final design of the bicycle, we will consider expanding the budget as needed.

 **GSM Module:** A GSM module will be used on the bike to allow remote data uploads to the display.

 **Generator:** Will be used to charge the second battery while the primary battery is in use in order to have a backup power source.

 **Seven-segment modules:** A seven-segment display on the handlebars of the bicycle will display battery life, as well as bicycle speed

 **Mounting hardware:** This refers to miscellaneous hardware needed to mount components to the bicycles, such as nails, screws, and other materials as necessary

 **Wiring:** Wire will be needed to make electrical connections. We have provided an upper-limit cost estimate of how much wire will be needed.

## Project Scheduling

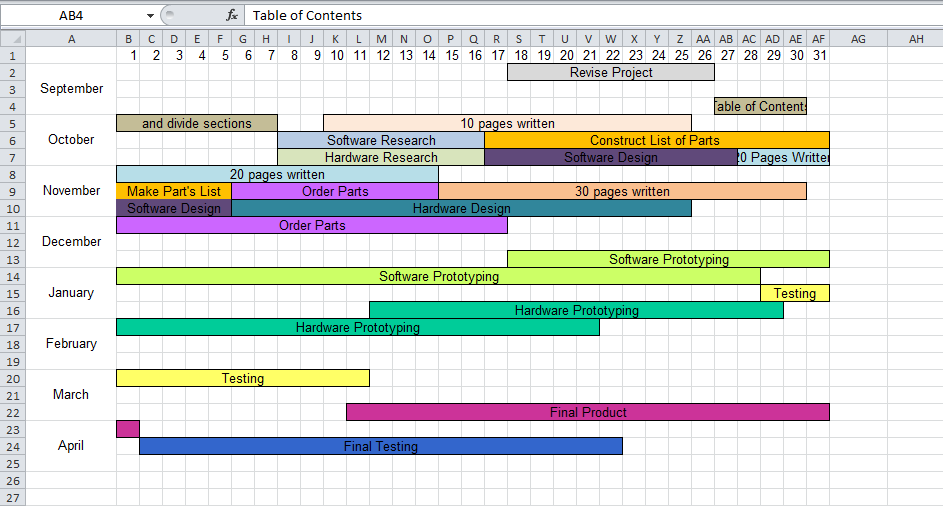
In order to ensure that all project objectives are met, and that all components of the project are designed to their maximum performance, maintaining a project schedule is essential. At the beginning of this project, the team created a list of project timeline with designated milestones. These milestones are the basis around which the team operates on. At all times during the project we have consistently worked to stay on or ahead of schedule. The project timeline with milestones is shown in Figure 56.

Figure 50: Project Calendar/Timeline

|  |  |  |  |
| --- | --- | --- | --- |
| ACTIVITY | START | END | NOTES |
| Have 10 pages written (each) | 10/18/2013 | 10/25/2013 | 14 days |
| Have 20 pages written (each) | 10/28/2013 | 11/14/2013 | 14 days |
| Have 30 pages written (each) | 11/15/2013 | 12/2/2013 | 12 days |
| Meet with Dr. Richie | 9/17/2013 | 9/17/2013 | 1 day |
| Revise Project | 9/18/2013 | 9/26/2013 | 7 days |
| Write Table of Contents. Divide Sections | 9/27/2013 | 10/7/2013 | 7 days |
| Hardware Research | 10/8/2013 | 10/16/2013 | 7 days |
| Software Research | 10/8/2013 | 10/16/2013 | 7 days |
| Construct List of Parts | 10/17/2013 | 11/5/2013 | 7 days |
| Hardware Design | 10/8/2013 | 10/16/2013 | 14 days |
| Software Design | 10/8/2013 | 10/16/2013 | 14 days |
| Prototyping | 1/13/2014 | 2/21/2014 | 14 days |
| Programming | 12/18/2013 | 01/28/2014 | 30 days |
| Testing | 01/29/2014 | 03/11/2014 | 30 days |
| Final Product | 03/12/2014 | 04/01/2014 | 30 days |
| Final Testing | 04/02/2014 | 04/22/2014 | 30 days |
|  | 04/12/2014 | 04/14/2014 | 15 days |

Table 14:Project Milestones

## Suppliers

For materials in this project, no single supplier was used. Rather, all components are chosen from whichever supplier has the lowest price, fastest shipping rate, best return policies, and highest credibility. Considering this, there are a few notable suppliers from which many of our materials will be purchased from.

Amazon (amazon.com) is where many of our project parts will be purchased. We have found that Amazon has the lowest prices on materials such as diodes, resistors, and capacitors, as well as larger pieces such as the generator and 7-segment display components. As we are students, Amazon provides us free 2-day shipping on most items through Amazon Prime, as well as free return shipping and guaranteed refunds. By using Amazon, we are essentially ensuring that we do not waste money on parts that do not get used, as any part we are unhappy with is able to be returned or exchanged for another part with ease.

Home Depot is a supplier that is local to us, that will be used mainly for assembly parts such as screws and bolts and wires.

Battery Space (AA Portable Battery Corp) is where we will be purchasing both the primary and the secondary battery for the POV display. We chose Battery Space because they have the exact battery that we were interested in, and all of their batteries in made and delivered in the US. This eliminates any concern about purchasing products from overseas or waiting for international shipping.

Texas Instruments is the producer of our chosen microcontrollers and LED controllers. Therefore, we will purchase those components as needed through the Texas Instruments website.

DigiKey (digikey.com) is also an important resource. Since this website specializes in electronic components, it will be used to purchase any specific components that cannot be found on a general website like Amazon. If we need any capacitors, resistors, or LEDs which cannot be found on Amazon at a lower cost, they will most likely be purchased from Digikey.

SparkFun (sparkfun.com) is another electronics store. From SparkFun we will likely purchase components that will allow us to convert the pins on the microcontroller so that it can be breadboarded. We will also purchase components for the 7-segment display and Bluetooth modules from SparkFun.

For all documentation purposes, we will be using OfficeMax‘s print and copy services center in order to print and bound our reports.

## Team and Project Organization

This section describes the management of the team which makes this project possible. Since this Persistence of Vision Bicycle was created as a Senior Design project for computer engineering students and electrical engineering students, our team is comprised of equal numbers of members in each major; specifically, we have two computer engineers, and two electrical engineers.

The Persistence of Vision bicycle conveniently has two major components that cater to the skills of the two types of engineers in our team. It contains a hardware component, which encompasses all power considerations, wireless transceivers, microcontrollers, and LEDs. It also contains a software component, which includes all image processing and power-monitoring algorithms, wireless communications, and microcontroller programming. All members of the group contribute to each part of the design and building of all parts of the Persistence of Vision bicycle, however, naturally, our electrical engineering members have taken leadership over the hardware components, while our computer engineering members have lead the development of the software components.

A few components of our project have required equal contribution from each team member. First, the mechanical design of the bicycle has presented a great challenge as no single team member can be named the expert in mechanics. We have collectively all had to brainstorm, plan, and design the way in which our bicycle can be mechanically built, with all components functioning and secure, in the most convenient and practical manner.

Additionally, all team members are included in financial decisions. No parts are purchased until all team members are consulted and are presented with all possible part options. This ensures that our team stays within its budget, and that each purchase is made carefully and responsibly.

# Works Cited

AA Portable Battery Corp. (n.d.). *Battery Packs.* Retrieved from Battery Space: http://www.batteryspace.com/

Al, B. e. (2012). *3D Persistence of Vision Display.*

Alexander, W. (n.d.). *GSM API.* Retrieved from http://www.java-samples.com/java/free\_send\_SMS\_using\_Java\_Sample\_program.htm

Algorithm, T. (2009, June). *"Bilinear Image Scaling".* Retrieved from http://www.Tech-algorithm.com

Algorithm, T. (2009, July). *Trilinear Interpolation Image Scaling.* Retrieved from tech-algorithm.com

Amazon. (n.d.). *Human Creations Bicycle Dynamo USB Charging Kit with Universal Phone/GPS/iPhone/MP3 Holder*. Retrieved from http://www.amazon.com/Human-Creations-Bicycle-Charger-Universal/dp/B008KET1V2

Android. (n.d.). *Android SDK.* Retrieved from http://developer.Android.com/sdk/index.html

Apple. (n.d.). *Apple SDK.* Retrieved from https://developer.apple.com/xcode/

Atmel. (n.d.). *ATMega8515 Datasheet.* Retrieved from http://www.atmel.com/Images/doc2512.pdf

Atmel. (n.d.). *ATTiny2313 Datasheet.* Retrieved from http://www.atmel.com/Images/doc2543.pdf

bike2power. (n.d.). *Bottle Sidewall Bicycle Dynamo Power Generator* . Retrieved from http://www.bike2power.com/Bottle-Sidewall-Bicycle-Dynamo-Power-Generator

Boeing. (n.d.). *The Boeing Company*. Retrieved from http://www.boeing.com/boeing/companyoffices/aboutus/index.page?

Brain, M. (2013, November). 'How Television Works".

Buchmann, I. (2011, March 15). *Types of Lithium-ion*. Retrieved November 4, 2013

Buchmann, I. (2011, March 15). *Types of Litihium-Ion .*

Burlison, S. E. (2012). *EECS UCF Senior Design.* Retrieved from UCF EECS: http://www.eecs.ucf.edu/seniordesign/su2012fa2012/g08/documents/senior-design-i-documentation.pdf

Cornell. (n.d.). *Persistent of Vision Display* . Retrieved from http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2012/jm787\_dab355/jm787\_dab355/

Corp., A. P. (2000, January 1). *AA Portable Power Corp.* Retrieved November 5, 2013

Daycounter Inc. (2012, August). *Transformerless AC to DC Power Supply Circuits.* Retrieved from http://www.daycounter.com/Circuits/Transformerless-Power-Supplies/Transformerless-Power-Supplies.phtml

exploreB2B. (n.d.). *Usage and Types of LED Lights.* Retrieved from https://exploreb2b.com/articles/usage-and-types-of-led-lights

Farnell. (n.d.). *VISHAY DALE RS00510K00FE12.* Retrieved from http://uk.farnell.com/vishay-dale/rs00510k00fe12/resistor-wirewound-10kohm-6-5w/dp/1203687

Fitton, D. (n.d.). *Bluetooth API.* Retrieved from http://www.caside.lancs.ac.uk/java\_bt.php

Future Electronics. (2010). *Roving Networks: Wireless for Less.* Retrieved from http://www.futureelectronics.com/en/manufacturers/roving-networks/Pages/index.aspx

Harris, Tom. "How Plasma Displays Work." *HowStuffWorks*. Discovery Channel, n.d. Web. 29 Nov. 2013.

HOLTEK. (2012, September). *16-Channel Constant Current LED Driver.*

HOLTEK. (2012, September 19). HT16D724, 16-Channel Constant Current LED Driver. Unknown. Retrieved November 5, 2013

Inc., R. S. (2013, january 1). *Slip-ring Series 001.* Retrieved October 24, 2013, from http://www.directindustry.com/.

Ink, T. I. (2012). *Spoke Ink- Persistence of Vision for Your Bike*. Retrieved from http://e2e.ti.com/group/microcontrollerprojects/m/msp430microcontrollerprojects/664376.aspx

Instruments, T. (2008, September 15). TLC5947. Unknown.

Instruments, T. (n.d.). *TLC4940 Datasheet.* Retrieved from http://www.ti.com/lit/ds/symlink/tlc5940.pdf

Lechnyr, D. (2003, July 30). *Types of Batteries*. Retrieved October 25, 2013, from http://www.tldp.org/HOWTO/Battery-Powered/battery.html

Lechnyr, D. (2013, October). *Types of Batteries.* Retrieved from http://www.tldp.org/HOWTO/Battery-Powered/battery.html

Linead Technology. (n.d.). *LT3746 Datasheet.* Retrieved from http://cds.linear.com/docs/en/datasheet/3746fa.pdf

Linear. (n.d.). *LT4746 Datasheet.* Retrieved from http://cds.linear.com/docs/en/datasheet/3746fa.pdf

Loflin, L. (2011, March). *Using Ratiometric Hall Effect Sensors.* Retrieved from http://www.bristolwatch.com/hall\_effect/ratiometric\_hall\_effect.htm

Meyer, A. (2011, july 5). *A Strange Attraction. Various Hall Effect Sensors*. Retrieved October 20, 2013, from http://bildr.org/?s=hall+effect

Meyer, A. (2013, October). *A Strange Attaction: Various Hall Effect Sensors.*

Microchip. (n.d.). *PIC32MX340F512H Datasheet.* Retrieved from http://ww1.microchip.com/downloads/en/DeviceDoc/61143H.pdf

Monarch Instrument. (2013, January ). Retrieved from http://www.monarchinstrument.com/senchart.htm

MrChips. (n.d.). *All About Circuits.* Retrieved from http://forum.allaboutcircuits.com/blog.php?b=559

Multicomp. (n.d.). *SMD Super Bright LED.* Retrieved from http://www.farnell.com/datasheets/385441.pdf

Paisely, R. (2010, March). *Basic Visible and Infrared Light Detectors.* Retrieved from http://home.cogeco.ca/~rpaisley4/PhotoDetectors.html

Paisley, R. (2010, March 7). *Basic Visible and Infrared Light Detectors*. Retrieved October 26, 2013, from http://home.cogeco.ca/~rpaisley4/PhotoDetectors.html

RICK. (2011, Octorber 11). *Regulated DC Power Supply*. Retrieved November 4, 2013, from http://www.circuitstoday.com/regulated-dc-power-supply#comment-65722

Rotary Systems Inc. (2013, January ). *Slip-ring Series 001.* Retrieved from http://www.directindustry.com/

Roving Networks. (n.d.). *RN-42 Datasheet.* Retrieved from SparkFun Electronics: https://www.sparkfun.com/datasheets/Wireless/Bluetooth/rn-42-ds.pdf

SourceCorge. (2013, November). *PPM Format Specification.*

Tech Algorithm. (2007, October). *Nearest Neighbor Image Scaling.* Retrieved from Tech Algorithm.

Texas Instruments. (n.d.). *"Adding Bluetooth to Your MSP430 Project".* Retrieved from http://www.msp430launchpad.com/2011/08/adding-bluetooth-to-your-msp430-project.html

Texas Instruments. (2008, September). *TLC5947.* Retrieved from http://www.ti.com/general/docs/lit/getliterature.tsp?literatureNumber=sbvs114a

Texas Instruments. (n.d.). *CC2560-PAN1315.* Retrieved from http://www.ti.com/product/cc2560-pan1315

Texas Instruments. (n.d.). *MSP430g2553 Datasheet.* Retrieved from http://www.ti.com/lit/ds/symlink/msp430g2553.pdf

Texas Instruments. (n.d.). *MSP430g2553 Datasheet.* Retrieved from http://www.ti.com/lit/ds/symlink/msp430g2553.pdf

Texas Instruments. (n.d.). *TLC4940 Datasheet.* Retrieved from http://www.ti.com/lit/ds/symlink/tlc5940.pdf

Today, C. (n.d.). *Regulated DC Power Supply.* Retrieved from http://www.circuitstoday.com/regulated-dc-power-supply#comment-65722

Unknown. (2012, August 17). *Transformerless AC to DC power Supply circuits*. Retrieved November 2, 2013

Vividvilla. (2013, October). *Seven Segment Display: Everything You Need to Know About.* Retrieved from Electronics Blog.

Wi-Fi Alliance. (n.d.). *Discover and Learn.* Retrieved from http://www.wi-fi.org/discover-and-learn

# Permissions



