Design Specifications for Sole Power-Up Project

Daniel Peerson, Francois-Olivier Morin, Maziar Sharifikhah, Mohammad Kabir, Yekaterina Lim

University of British Columbia

EECE 409/429/419/439/400/469

April 06, 2017

University of British Columbia (UBC)

Design Specifications for Sole Power-Up Project

Pedal-Powered Mobile Device-Charging Station

Prepared by: Francois-Olivier Morin, Maziar Sharifikhah, Yekaterina Lim, Mohammad Shadman Kabir and Daniel P. Peerson

ELEC 491 Capstone Design Project

April 6, 2017

Executive Summary

The following document describes the electrical design for a pedal powered phone charging station. The overall structure of the system is explained along with the motivation for selecting each component. The specific components are discussed, including their specifications and implementation into the overall design. Also discussed are the project's challenges, safety measures, and design updates. While a functional electrical design is described in detail, the mechanical portion of the station is not discussed.

Table of Contents

List of Figures

1. System Architecture

1.1. Previous Capstone Architecture

The previous capstone group left a system that included a working mechanical design as well as a number of high quality electrical components. However, it needed to be plugged into the wall as the passive power consumption was only sustainable for hours without a direct connection to an outlet which is requirement FR3 for the project. Power from pedalling was being wasted on converting voltage from 12V DC to 120V AC then again back down to 5V DC to charge the phone battery causing unacceptable losses. The mechanical system, while functional, had pedals which exhibited rapid changes in acceleration that were not conducive to a pleasant user experience.

1.2. Current System Architecture

The system is currently designed to allow up to two users to charge their mobile devices using energy they generate by pedalling on stationary bikes (FR1). It converts mechanical energy from the rotation of pedals to electrical energy, regulating the voltage and current, to charge a battery which in turn charges the user's mobile device. The high level system design to achieve this goal is shown in Figure 1 below. It should be noted that our design covers only the electrical portion of this system. The mechanical portion referred to as the "Bike Mechanical System" block (encompassing the bikes and differential gearbox) in Figure 1 is the responsibility of this project's AMS representative. A rendering of the station can be found in Appendix 5 Figure 12. A list of parts selected for our design is included in Appendix 3. This section will describe the block diagram in more detail.

To start the mobile device charging process, a user inputs mechanical energy by rotating the pedals of a stationary bike. These pedals rotate a shaft which is connected to an alternator. The alternator converts this rotational energy to electrical energy, outputting DC current using a rectifier. The generated 12V electricity is step down to 5V to latch two relays (K1A and K2A). The system thus connected to the battery through NO contact K1B and K2B. The energy is sent to a Morningstar Prostar-30 charge controller, which provides the system with power and safely stores any excess energy in a battery (requirement FR3).

User detection by pedalling is required to start the station, the system detects that a user is pedalling using a magnetic sensor. Sensors are attached to the crankset of both bikes, allowing the system to determine how many users are pedalling (requirement FR1). The sensor detects the motion of the pedals and sends a signal to the system's microcontroller. As the microcontroller receives that signal, it latches a relay switch allowing current to pass onto the charging circuitry. This process regulates the power before providing it to the user's mobile device. If the user stops pedalling for 10 seconds, the microcontroller opens relay K3A and K4A, which shuts down the system through NO contact K3B and K4B .

An LCD screen is used to show the user how much power they are generating as they pedal, which can be used to educate users about sustainable energy (requirement NFR1). This power value is recorded by the microcontroller using data from a power measurement circuit, which measures the voltage and current at the output of the alternator.

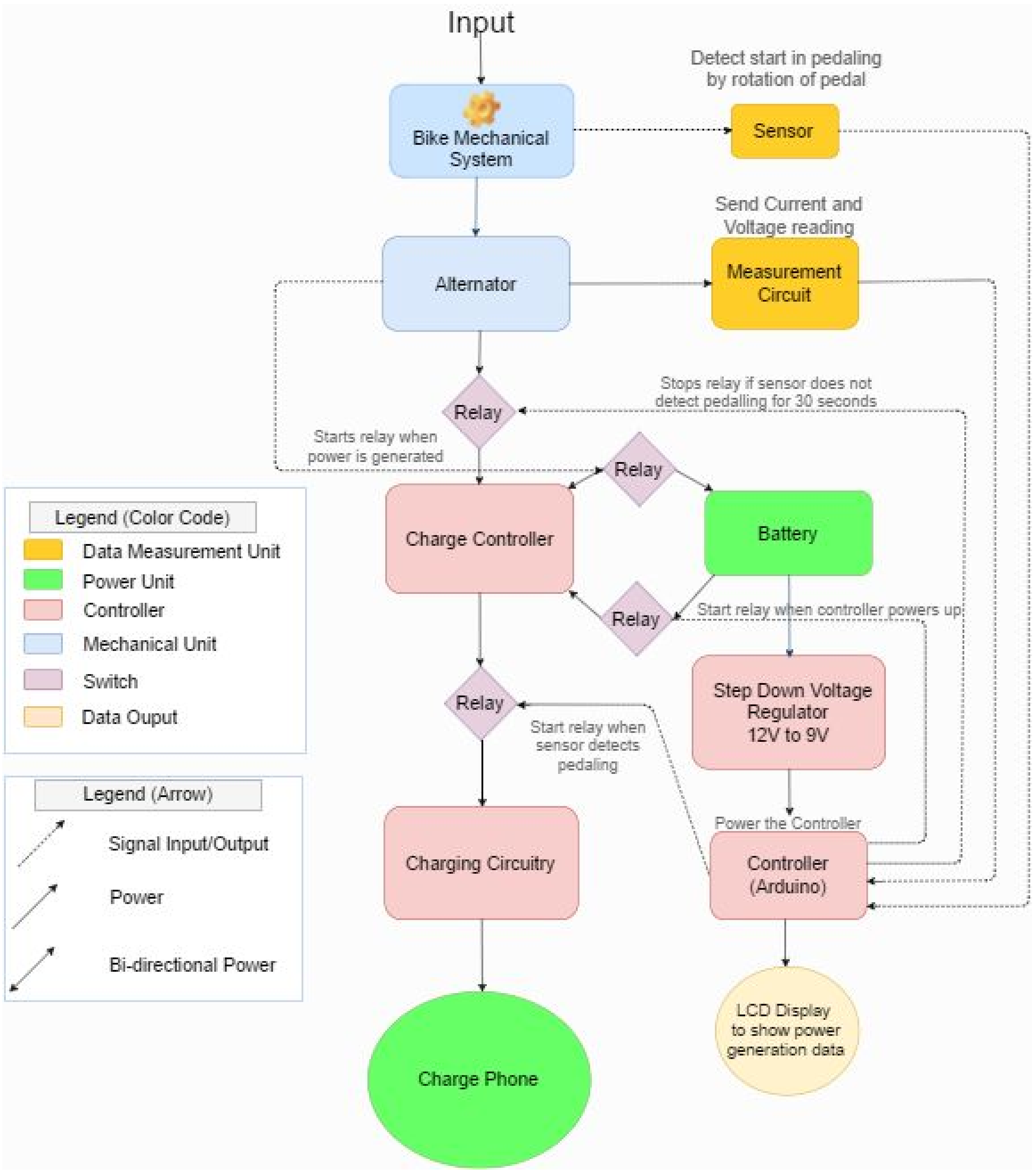

Input
Detect start in pedaling by rotation of pedal
Bike Mechanical System
Sensor
Send Current and Voltage reading
Alternator
Measurement Circuit
Stops relay if sensor does not detect pedalling for 30 seconds
Relay
Starts relay when power is generated
Relay
Legend (Color Code)
Data Measurement Unit
Power Unit
Controller
Mechanical Unit
Switch
Data Ouput
Charge Controller
Battery
Relay
Start relay when controller powers up
Relay
Start relay when sensor detects pedaling
Step Down Voltage Regulator 12V to 9V
Legend (Arrow)
Signal Input/Output
Power
Bi-directional Power
Power the Controller
Charging Circuitry
Controller (Arduino)
LCD Display to show power generation data
Charge Phone

Figure 1: System Block Diagram

2. System Components

2.1. Power Generation

2.1.1. Alternator

An alternator is required to convert mechanical energy, provided by user when they pedal, into usable electrical energy. We are using a WindBlue DC-520 permanent magnet alternator to generate electrical energy to charge a 12V battery. It has a number of features that fit well into our design:

- Reach 12V at a low RPM (rotations per minute) of 240 of the alternator - Since a user will provide the rotational force the alternator needs to work at low RPM[1]. 240 RPM in the alternator translates to 45 RPM for the user.
- Built-in rectifier - This converts alternating current to direct current which means less external circuitry is required
- Completely brushless design - It reduces friction and minimizes the need for maintenance (requirement NFR3)

It has been tested to generate more than 17.6 Watts with a single user and 44.7 Watts with 2 users pedalling (FR1,C1) (see Validation Document, Appendix 1: Alternator Power Generation). The measurement for the power output is taken at a low speed of 260 RPM of the alternator which translates to about 50 RPM for the user.

2.2. Energy Management

2.2.1. Charge Controller

A charge controller is required for our design to regulate the power generated by pedalling on the bike. The user will pedal at different speeds which will generate fluctuating level of power. The charge controller regulates the voltage input to a constant voltage to charge a battery. The input power is used to power the load and any power remaining charges the battery. The controller also prevents overcharging, and allows safe delivery of power to a load. A simplified charge controller is shown in Figure 2.

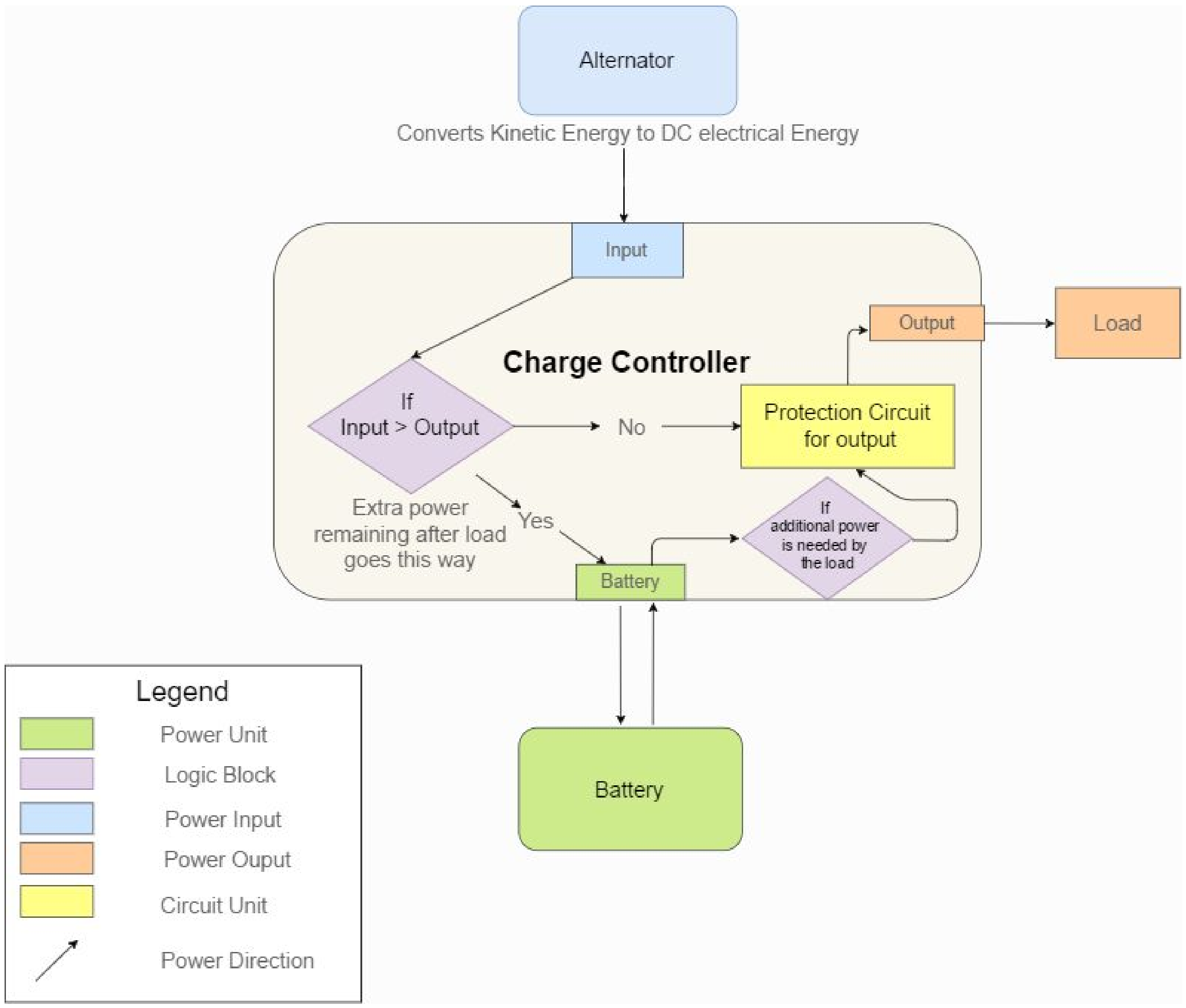

Alternator
Converts Kinetic Energy to DC electrical Energy
Input
Charge Controller
Output
Load
If Input > Output
No
Protection Circuit for output
Extra power remaining after load goes this way
Yes
If additional power is needed by the load
Battery
Battery
Legend
Power Unit
Logic Block
Power Input
Power Ouput
Circuit Unit
Power Direction

Figure 2: Charge Controller Block Diagram

The charge controller we have selected is the Morningstar Prostar-30 (PS-30) which is validated (see Validation Document, Appendix 4) and works very well for our design:

- High Current Input/Output - it has a max charging/load current of 30A which prevents any damage done to the battery by surge currents[1]

- Protection Circuit - Short circuit, overload, and reverse polarity protection
- High Voltage/Current Disconnect - High voltage and high current disconnect for surges and part failures
- Low self current consumption of 20mA - It is important as we need very low passive power because our power input is only from a user pedalling

2.2.2. Battery

To store energy generated by the alternator (meeting requirement FR3) and to power a load using a charge controller, we need a battery that is reliable and can be easily replaced. This battery was also available at no cost from the AMS as it was used by a previous capstone project. A lead acid battery made available to us was selected for this application over other battery chemistries because of availability, reliability and cost.

The particular model is Rolls Surrette S12-128AGM, a 12V 128Ah AGM (Absorbent Glass Mat) battery (see Validation Document, Appendix 4). The characteristics of this battery is well suited for our system for the following reasons:

- 12V Nominal Voltage - Very easy to convert to 5V to power our circuitry.
- 115 AH Capacity (@ 100 hr rate) - A large capacity allows us to store the excess generated energy, and ensure that a user's mobile device can be charged immediately even if the station has not been used for an extended period of time (FR1)[2]
- Designed for small charges and discharges - This battery is designed for a car which charges and discharges the battery a little bit at a time. We will be doing the same thing and staying within the standard operating use of the battery.[2]

2.3. Power Measurement

2.3.1. Measurement Circuitry Overview

In order to educate users about how much power they are generating by pedalling (requirement NFR1), the system requires appropriate circuitry to measure the power coming from the alternator. This is done using the circuitry described by the schematic below in Figure 3. It measures current and voltage independently, then sends those values to the microcontroller to calculate the final value. A sample calculation can be found in Appendix 1.

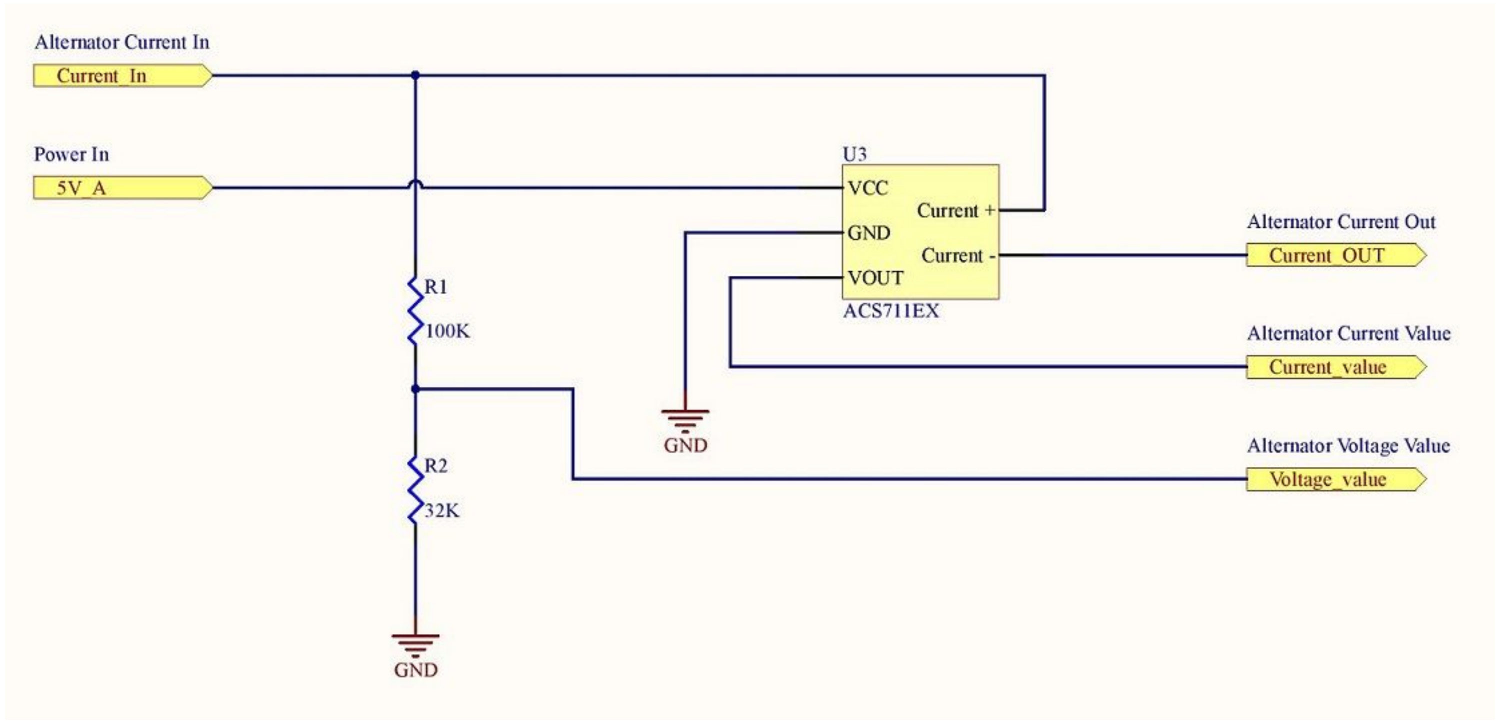


Figure 3: Power Measurement Circuitry Schematic

To measure the current, a ACS711EX current sensor board is used. The following features makes it ideal for our system:

- Input current range of -15.5 to 15.5 A - Ensures that the full range of currents being output from the alternator can be captured
- Current sensing resolution of 0.090 V/A - Gives high precision, making the power calculation more accurate[4]
- 4 mA supply current at 3.3V to 5V - Minimizes the passive power draw, helping to meet requirement FR3
- Read voltages and current from range 0V to 20.6V and -18A to 18A respectively

To measure the voltage, a simple voltage divider is required to make the alternator's 12V output low enough to be read by the microcontroller (0-5V). Once the lowered value is read by the microcontroller, we can easily convert the actual value. High resistor values (100kΩ and 32kΩ) are used in order to ensure that the current sunk by the microcontroller does not exceed its rated limit (~40mA). The circuit has been tested to work to get data for the Result section of Appendix 1 of the Validation Document.

2.4. Microcontroller

A microcontroller is an integral part of our project as it acts as the main controller of the system. Having a controller allows us to gather data and display them to the user, as well as control parts of the circuit without having to make changes in the circuit.

We have chosen an Arduino Uno for our project. The Arduino Uno is easy to set up, has a large library of code for different applications, a wide array of products tailored for its use, and numerous tutorials on how to use it. The current use of the Arduino is listed below:

- The Arduino controls the mobile device charging circuit by using a relay. Providing an output voltage at the analog pin of the Arduino connected to the relay will close the switch to the mobile device charging circuit.
- The Arduino is connected to a pair of magnetic sensors which gives an analog signal input as soon at it detects pedal motion. When the bike station is not in use for more than 10 seconds the Arduino powers off the system entirely by opening

relay K3A and K4A. More precisely, two NO (normally open) contacts (K3B,K4B) will disconnect the system from the battery. This way we can ensure the system is powered off after the user has stopped using the station.

- The current and voltage measurement circuit is connected to the analog input of the Arduino. The Arduino uses an ADC (analog to digital converter) to read the voltage and current value at its input and uses arithmetic operation to convert the values into useful numbers. Gathering real time power generation data helps the system determine if the user is generating enough power to charge a mobile device (C1). The Arduino will not close the relay if the power generation is not sufficient.
- Finally, the measurement circuit data is displayed to the user using an LCD display (NFR1). The data on screen can be used to educated the user about the effort required to generate power to charge a mobile device. This gives an insight on the importance of energy conservation and the development/usage of sustainable energy.

All the connections described above are provide in the schematics in Appendix 4. Validation of the microcontroller is provided in Appendix 4, Validation Document. Algorithm for the code is represented by the flowchart in Appendix 8.

2.5. Sensors

2.5.1. RPM / Pedal Detection

To detect if/how fast a user is pedalling, we are using a simple magnetic switch. The switch itself is fixed to the bike shaft, and a magnet is attached to the shaft of the pedals. The circuit is closed each time that the switch and magnet pass by each other, allowing the Arduino to detect that a user is pedalling and to count the number of rotations the user is pedalling. This information can be later displayed to the user as a rolling average to show a relationship between rotations and power.

Each bike crankset will have its own magnetic switch, allowing the microcontroller to detect how many users (1 or 2) are pedalling. This fulfills requirement FR1.

Using a magnetic switch is a suitable solution because of its simple implementation and reliability.

3. System Design Implementation

3.1. Automatic System Startup and Shutdown

When the user starts pedalling, the alternator generates a voltage that latches relay K1 and K2 through a voltage regulator (5V). Stepping down the voltage allows the relay to latch a very low power generation at about 30 RPM (see Validation Document Section 2). The NO contact of K1 and K2 connects the battery to the charge controller. This allows all components of the system to be powered up. The microcontroller latches relay K3 and K4. We can see in Figure 9 that the relays are connected in parallel with K1 and K2 respectively. This logic circuit allows the microcontroller to keep the system up and running during dips in user pedalling and even short periods of downtime between users. When the system is idle for more than 10 seconds the microcontroller opens the relay and the system turns off until another user starts pedalling.

We can see the state machine visualisation of the system in Figure 4. The system starts in the ‘off’ state, when the user starts pedalling the system goes to the ‘setup stage’. At this stage the controller is connected to the battery and this starts up the rest of the system. It goes to ‘on’ state where the phone starts charging. If the user stops pedalling momentarily, station is kept ‘on’ for 10 seconds. From here we can either stay in the ‘on’ state if the user resumes pedalling or we can go to the ‘off’ state if the user stops pedalling for more than 10 seconds. More detailed flowchart can be found in Appendix 8. Validation for the mentioned components are in Validation Document, Appendix 4.

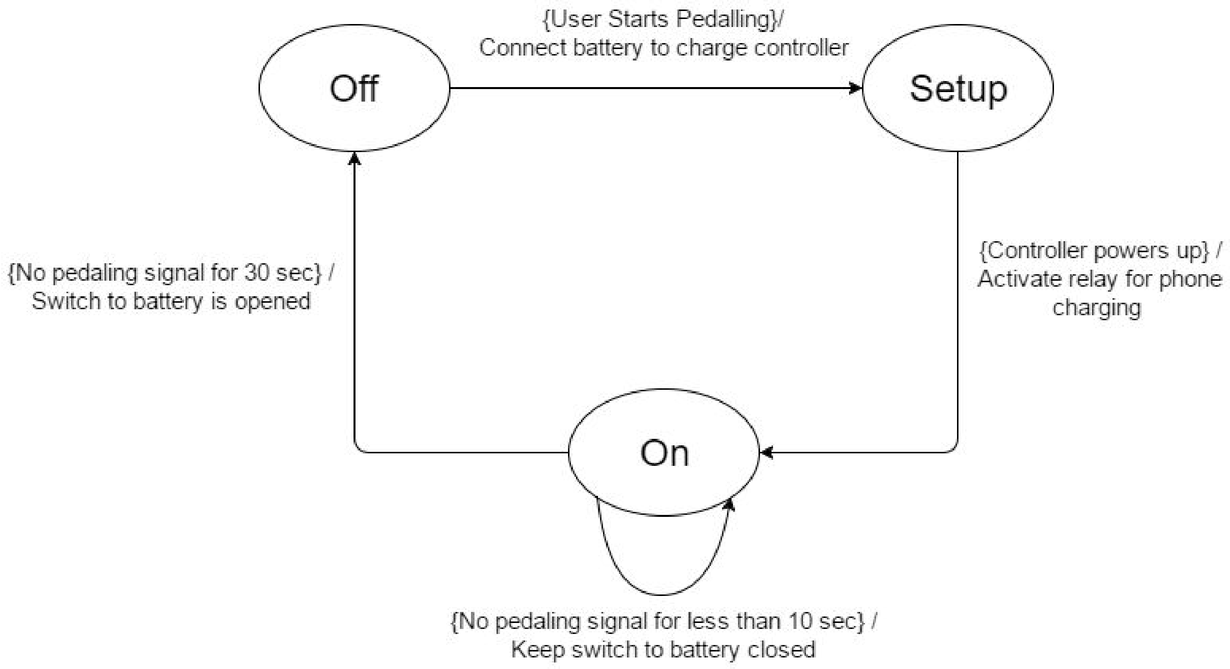

{User Starts Pedalling}/
Connect battery to charge controller
Off
Setup
{No pedaling signal for 30 sec} /
Switch to battery is opened
{Controller powers up} /
Activate relay for phone
charging
On
{No pedaling signal for less than 10 sec} /
Keep switch to battery closed

Figure 4: State machine diagram

3.2. Mobile Device Charging Circuitry

A block diagram of the charging circuit is shown in Figure 5 (Appendix 2). The charge circuit takes 12V from the charge controller and steps it down to 5V using a buck controller, a switching voltage regulator which steps down voltage from a higher to lower value. The selected buck controller, as shown in Figure 16 (Appendix 5), uses a XL4051E1[5][6] chip to control the voltage and current at a specific level to charge a mobile device battery. A cell mobile device commonly charges at 5V at 800mA. Figure 6 (Appendix 2) shows the circuit for the buck controller used. The resistors R1 and R2 are variable resistors which can be adjusted to fix the voltage and current we require. The board has two separate potentiometer to adjust the required voltage and current. The controller has around 95% efficiency[7] when charging at 5V 1.5A as seen in Figure 8 (Appendix 2). The output of the controller is then fed into a universal charging cable, allowing both Android or iOS devices to be charged (FR2). Validation for the components in the charging circuitry is included in Validation Document, Appendix 4.

3.3. Safety

Safety of our system is of utmost importance. The wire used for the input end of the system, (from the alternator to the charge controller) is 14 Gauge. The maximum power generated by the user is 58.23 Watt (see Validation Document Appendix 1). At 12V this gives a maximum current of 4.85A. 14 Gauge wire can handle a maximum of 24 A[9] which is above the maximum input. The other reason why 14 gauge wire is used is because the UL standard stipulate that the minimum gauge to be used for power application is 14 AWG. The charge controller takes in all the power for the system and is rated at 30A which is multiple times higher than the maximum current input. The wire for the rest of the system is 22 Gauge, which sufficient for the current in the system as the current is limited by fuses.

The alternator has a built-in rectifier which converts AC output to DC current. We are using a commercial step down voltage regulator to charge a mobile device instead of an inverter to convert the DC current to AC. Voltages more than 50V are dangerous[13] and therefore stepping up voltage to 110V would have posed a safety issue. No component in our system requires AC voltage, so we are able to avoid this. Furthermore, voltage in the entire system is below 20V which is below the safe voltage value of 50V (constraint C4). There is a fuse after the alternator to protect the system from any surge voltage or currents. A fuse is placed where the user's mobile device is connected and another to the voltage input of the microcontroller.

Updating the components without proper knowledge or authorization can cause the system to fail which may injure users. In order to prevent any user contact with components and prevent them from being stolen (requirement NFR2), components such as the battery, charge controller, and Arduino will be in a locked box below the station's table. This is represented by the blue box in Figure 10 (Appendix 4).

3.4. Edutainment System

In order to increase engagement and educate users about power consumption/generation, the system is equipped with an entertainment/education (edutainment) system (goal G1). This system comprises a 7” touchscreen (the 4D Systems GEN4-ULCD-70DT, detailed below) that will be placed within arm’s reach of both users. This screen educates users by contextualizing the power they are generating. Users can see how many phones, laptops, gaming consoles, blenders, toasters, and microwaves can be powered using the power they are generating at that moment. This will allow users to gain an understanding of the power requirements of their everyday activities and be inspired to effectuate their own sustainable energy practices. Mockups of what is displayed on the screen are available in Appendix 7 (Figures 20 to 24).

With regards to entertainment and user engagement, both pedallers’ speeds will be displayed on speedometers, enticing users to push themselves to generate more power. Furthermore, riders will be able to start competitive games such as a race (Figure 22) that allow two users to compete against each other while they are charging their phones.

The screen we selected, the GEN4-ULCD-70DT (Figure 19), has a number of features that make it very suitable to integrate into the existing design:

- Touchscreen: Having a touchscreen allows for user input/selection, creating increased engagement.
- Arduino support: The screen integrates very easily with the Arduino currently running the system, requiring only 2 pins for a serial connection (transmission/receiving) and a reset pin. Additionally, 4D systems have a publicly available library[12] of functions that allows for easy implementation & maintenance of code that communicates with the screen.

- Development tools / documentation: GUIs can be easily created/updated in 4D Systems Workshop 4, allowing for rapid development.

4. Design Recommendations

We believe that the electrical system is complete and should not require further design work. However, the mechanical system design (which does not fall under the scope of this document) has a number of areas that require attention. These include, but are not limited to:

- Realignment and welding of the left bike shaft to the gearbox. It is currently damaged, meaning only one user can currently pedal on the station.
- Seat height adjustment. The present height of the seats makes it difficult for shorter users to enjoy the station.
- The current gearbox used is a differential gearbox. This type of installation is not ideal for this application. In fact, this gearbox is designed to avoid one axle to rotate faster than the other. (This is engineered to work as a traction control) At the moment if one user peddle faster than the other, the axle internally skip turns in order to regulate both axle differential speed. This reduce considerably the output power of the system. Most of the energy produced by the user will be lost within the gearbox instead of being directly transferred to the alternator. In conclusion, the gearbox should not have the ability of traction control. One option would be to add a second alternator and connect each bike directly to an alternator.
- When a mechanical force is applied to the shaft of an alternator, an opposing force called back EMF is generated. This force is proportional to the mechanical force being applied. The faster the user peddles, the bigger the mechanical force on the shaft is and the bigger the back EMF force will be. This phenomenon

cause the biking experience to be very inconsistent. Using a inertia wheel, it would be possible to reduce the direct impact of back EMF. In fact, the inertia wheel will reduce considerably the real time effect of the back EMF on the peddler experience. In other words, once the user has reach a cruising speed using the inertia wheel, the angular momentum of the wheel will absorb the variation in back EMF force that the user would previously experience.

References

[1] WindBlue Power, “DC-520 High Wind Permanent Magnet Alternator,” DC-520 datasheet. [Online]. Available: http://www.windbluepower.com/Permanent_Magnet_Alternator_Wind_Blue_High_Wind_p/dc-520.htm [Accessed: Nov. 1, 2016].

[2] Morningstar, “ProStar Solar Controller,” ProStar 30 Datasheet, 2014. [Online]. Available: http://www.morningstarcorp.com/wp-content/uploads/2014/02/ProStarENG2_11.pdf [Accessed: Nov. 1, 2016].

[3] Surrette/Rolls, “S12-128AGM Deep Cycle Battery,” S12-128AGM Datasheet. [Online]. Available: http://pdf.wholesalesolar.com/battery-folder/Rolls-Surrette-S12-128-AGM-Battery-Specifications.pdf?_ga=1.70449812.589188730.1477952395 [Accessed: Oct. 15, 2016].

[4] Allegro Microsystems, LLC, “Hall Effect Linear Current Sensor with Overcurrent Fault Output for <100 V Isolation Applications,” ACS711 Datasheet, 2013. [Online]. Available: https://www.pololu.com/file/0J497/ACS711.pdf [Accessed: Oct. 24, 2016].

[5] DROK, “Numerical Control Voltage Switching Regulator DC Buck Converter with Red LED Voltmeter 35V to 24V to 12V to 5V Adjustable 5-34V to 0-33V Step Down Variable Volt Power Supply Stabilizers Car Battery,” LM2596 Datasheet. [Online]. Available: http://www.droking.com/wp-content/uploads/2016/04/DATASHEET_090029.pdf [Accessed: Nov. 5, 2016].

[6] DROK, “DC-DC Step-down Constant Current & Voltage Converter 4-38V to 1.25-36V 12V/24V Buck Voltage Regulator 5A 75W High Power LED Constant Current Driver Module for Lithium Battery Electromobile Charging,” Current & Voltage Converter Datasheet. [Online]. Available: http://www.droking.com/wp-content/uploads/2016/04/Datasheet_091016.pdf [Accessed: Nov. 5, 2016].

[7] XLSEMI, “5A 180KHz 36V Buck DC to DC Converter,” XL4015 Datasheet, Rev. 1.3. [Online]. Available: http://www.xlsemi.com/datasheet/xl4015%20datasheet.pdf [Accessed: Nov. 5, 2016].

[8] The Engineering ToolBox, “Wire Gauges - Current Ratings”, http://www.engineeringtoolbox.com [Online]. Available: http://www.engineeringtoolbox.com/wire-gauges-d_419.html [Accessed: Oct. 11, 2016].

[9] M. Kang, A. Kung, A. Liu and G. Merced, “AMS SolePower Station Improvement - The EnerCycle Machine,” Dept. Mech. Eng., Univ. of British Columbia., Vancouver, Proj. Report, Apr. 2016. [Online]. Available: https://sustain.ubc.ca/sites/sustain.ubc.ca/files/seedslibrary/Final%20Report%20-%20T15%20Sprouting%20Solutions_0.pdf [Accessed: Sep. 20, 2016].

[10] V. Bai, S. Car, J. Huang, J. Jiang and L. Woyceshyn, “Mech 45X AMS Seed Project Proposal,” Dept. Mech. Eng., Univ. of British Columbia., Vancouver, Proj. Proposal, Oct. 2014.

[11] Texas Instruments, “LM317L-N 3-Terminal Adjustable Regulator,” LM317L-N Datasheet, 2016. [Online]. Available: http://www.ti.com/lit/ds/symlink/lm317l-n.pdf [Accessed: Oct. 24, 2016].

[12] GitHub, “GitHub - 4dsystems/ViSi-Genie-Arduino-Library: ViSi-Genie - Arduino Library", Arduino Library, 2017. [Online]. Available: https://github.com/4dsystems/ViSi-Genie-Arduino-Library [Accessed: Feb. 14, 2017].

[13] W. Burr, “Guide to the Canadian Electrical Code, Part I — Instalment 6", Electrical Industry Canada [Online]. Available: http://electricalindustry.ca/latest-news/1671-guide-to-the-canadian-electrical-code-part-i-installment-6 [Accessed: Apr. 5, 2017].

Appendices

Appendix 1: Sample Power Measurement Calculations

Suppose that the alternator is outputting 18W, or 12V at 1.5A.

1. We first need to send readings to the microcontroller.

   A) Applying the divider shown in Figure 3, we get an output voltage of

$$12V * \frac{32k\Omega}{(32+100)k\Omega} = 2.91V$$

   B) For current, the ACS711EX will output as shown by the equation found on the datasheet. We use the input current of 1.5A as i.

$$Vout = \frac{Vcc}{2} + i * \frac{Vcc}{36.7A}$$

$$Vout = \frac{4.775V}{2} + 1.5A * \frac{4.775V}{36.7A} = 2.58V$$

2. Once these two values are sent to the microcontroller, we can extract the real values from the measurements in the code

   A) For voltage, we simply rearrange the equation from the voltage divider

$$2.91V * \frac{(32+100)k\Omega}{32k\Omega} = 12V$$

   B) Then current, using the equation found on the ACS711EX datasheet. We use Vout from above and 4.775V as Vcc.

$$i = 36.7A * \frac{Vout}{Vcc} - 18.3A$$

$$i = 36.7A * \frac{2.58V}{4.775V} - 18.3A = 1.5A$$

3. Finally, we simply multiply the two values to obtain the power

$$P = V * I$$

$$P = 12V * 1.5A = 18W$$

Appendix 2 : Mobile Device Charging Diagrams

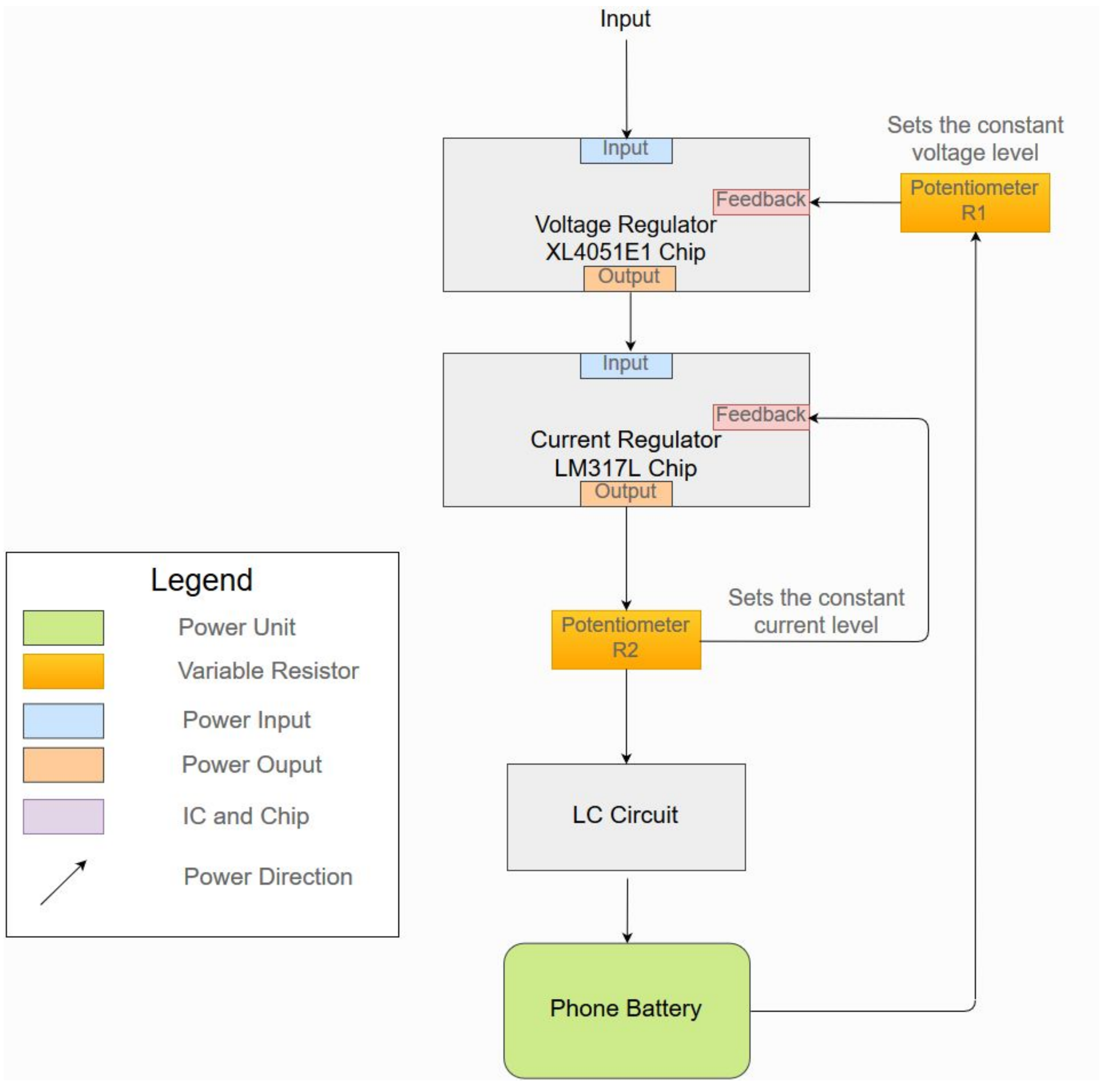


Figure 5 : Block Diagram of Mobile Device Charger

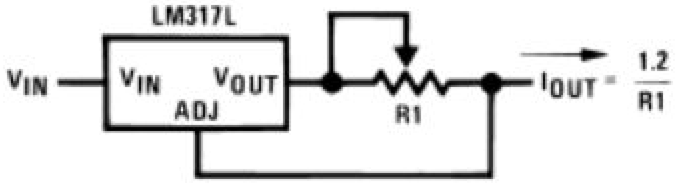
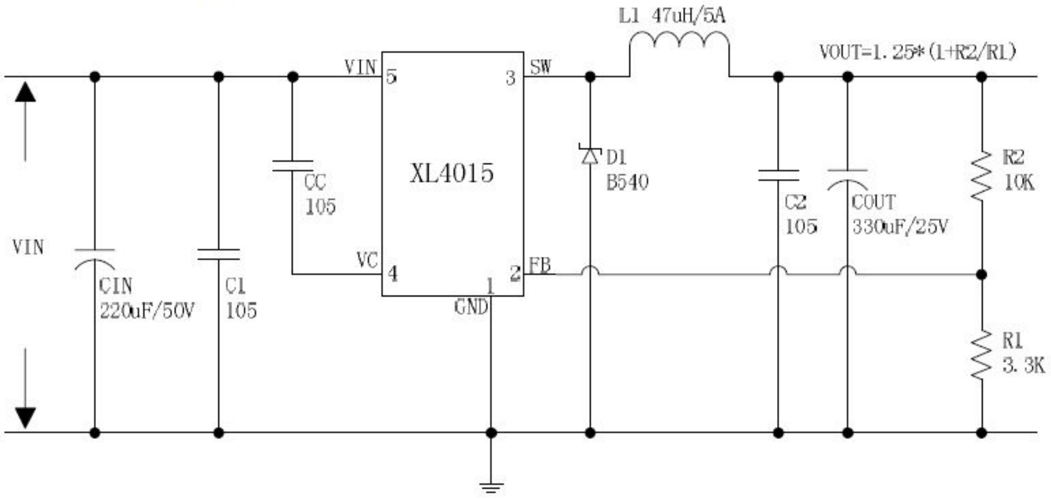


Figure 6 : Adjustable Current Limiter Circuit[11]

Typical System Application (VOUT=5V/5A)



XL4015 System Parameters Test Circuit (VIN=8V~36V, VOUT=5V/5A)

Figure 7 : Adjustable Voltage Regulator Circuit[6]

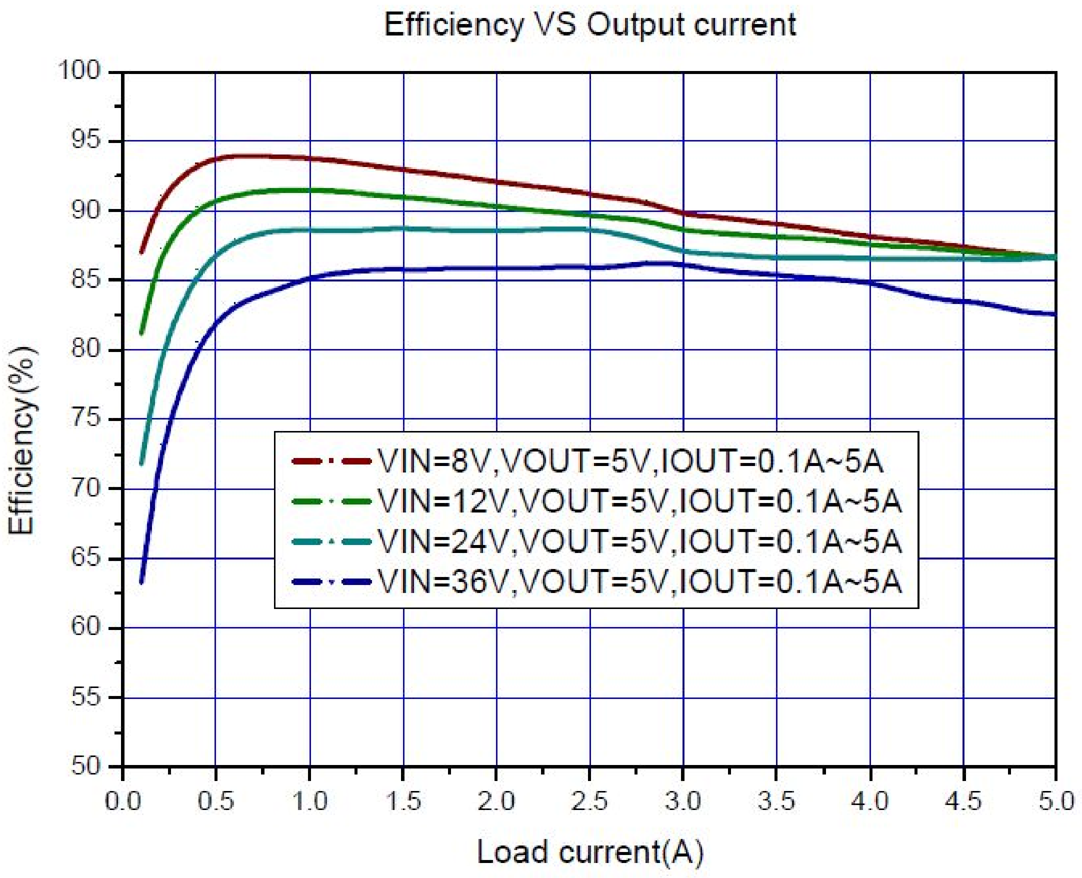


Figure 8 : Efficiency Graph of Buck Regulator[6]

Appendix 3 : Bill of Materials

Quantity	**Description 1**	**Description 2**	**Item**	**Cost**	**Validated**
1	Fuse	1.5A, 3A, 15A	Fuse	$16.00	
1	Relay 2-Channel		Relay	$9.39	✔
1	Relay 16-Channel		Relay	$19.89	✔
1	DC-520 High Wind Permanent Magnet Alternator		Motor	$219	
1	Arduino Uno		Microcontroller	$30	✔
2	DROK® DC-DC Step-down Constant Voltage Converter	Input 5-40V Ouput 1.25-37V	Regulator	$17	✔

2	DROK® DC-DC Step-down Constant Current & Voltage Converter	Input 6-32V Output 1.25-32V 4.5A 35W	Regulator	$20	✔
1	100K Ohm Resistor 1/4W		Resistor	$0.10	
1	30K Ohm Resistor 1/4W		Resistor	$0.10	
1	220 Ohm Resistor 1/4W		Resistor	$0.10	
1	1K Ohm Potentiometer		Variable Resistor	$5.35	
1	Magnetic Sensor		Sensor	$8.98	✔
1	Basic 16x2 Character LCD - Black on Green 5V		LCD Display	$9.49	✔
1	ProStar Charge Controller, 30A, 12/24VDC, PS-30		Controller	$163.93	✔
1	Surrette Rolls S12-128AGM 12V 115 AH AGM Battery		Battery	$357.95	
1	ACS711EX CURRENT SENSOR CARRIER -15.5A TO +15.5A	CURRENT SENSOR -15.5A TO +15.5A		$6.00	✔
1	GEN4-ULCD-70DT DISPLAY LCD TFT 7.0" 800X480		Touchscreen	234.94	
			Total	$1118.56	

Appendix 4 : System Schematics

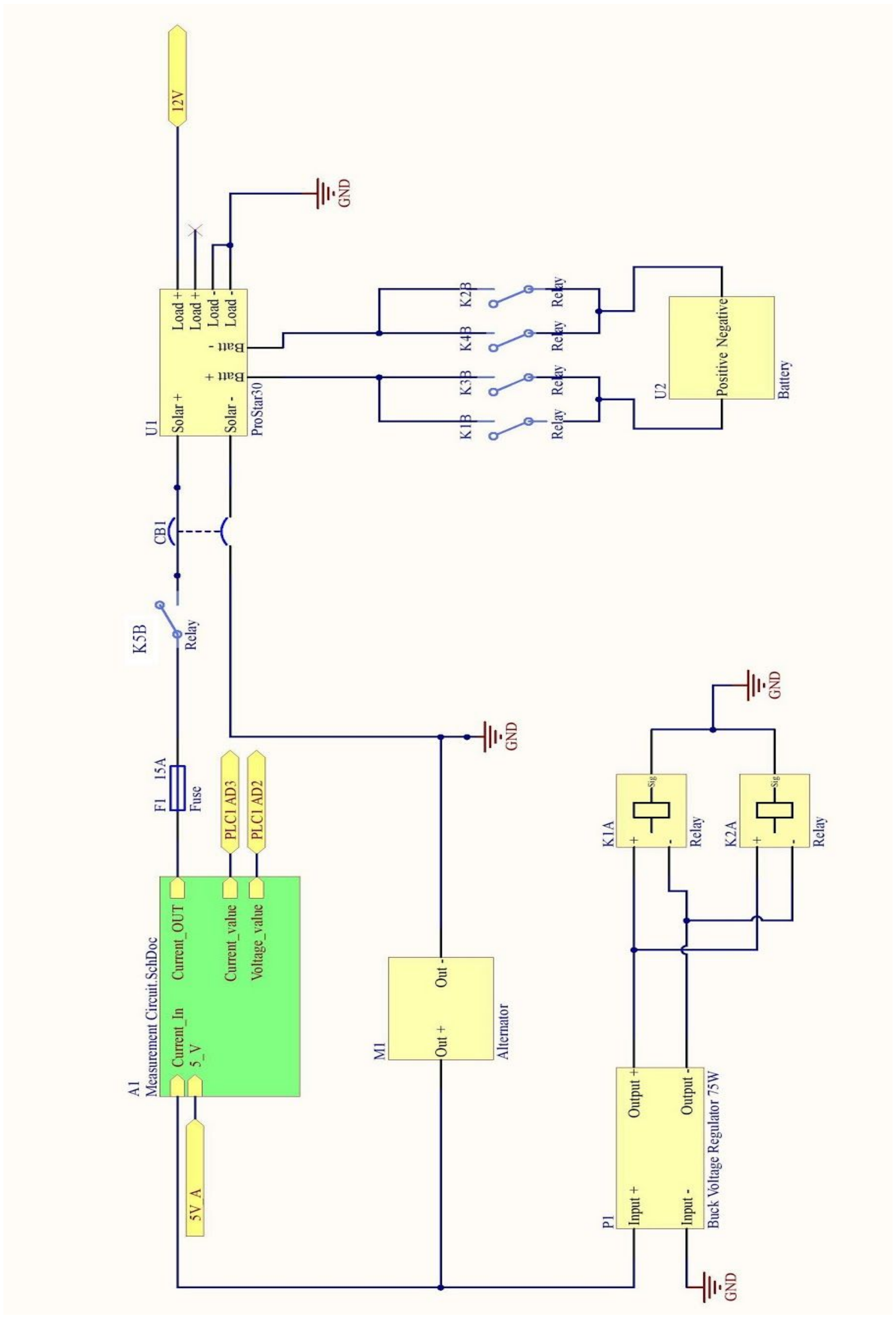


Figure 9 : Electrical Schematic of Mobile Device Charging Station

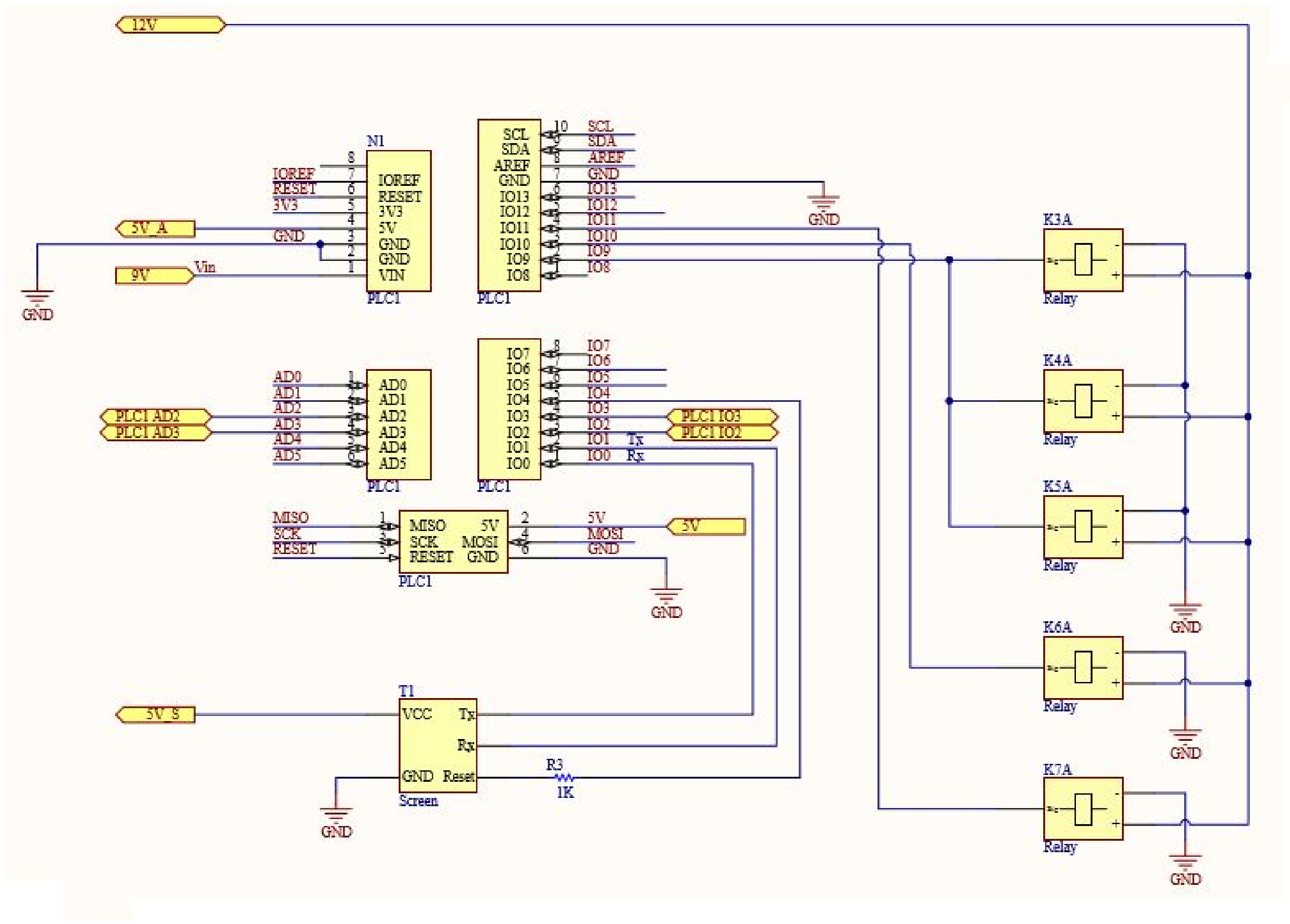


Figure 10: Arduino Board with LCD Schematics

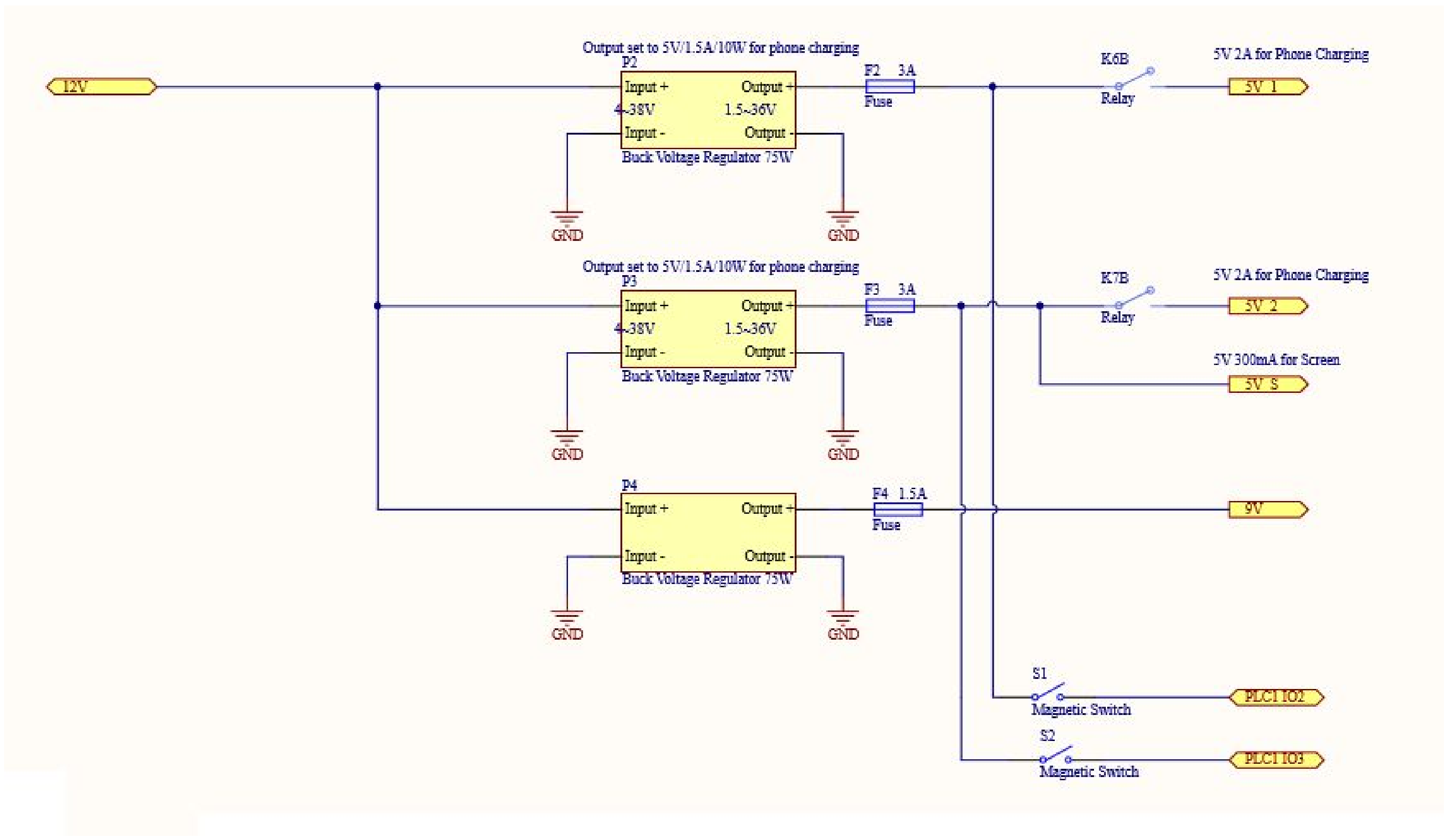


Figure 11: Power Distribution Circuit Schematic

Appendix 5 : System Figures

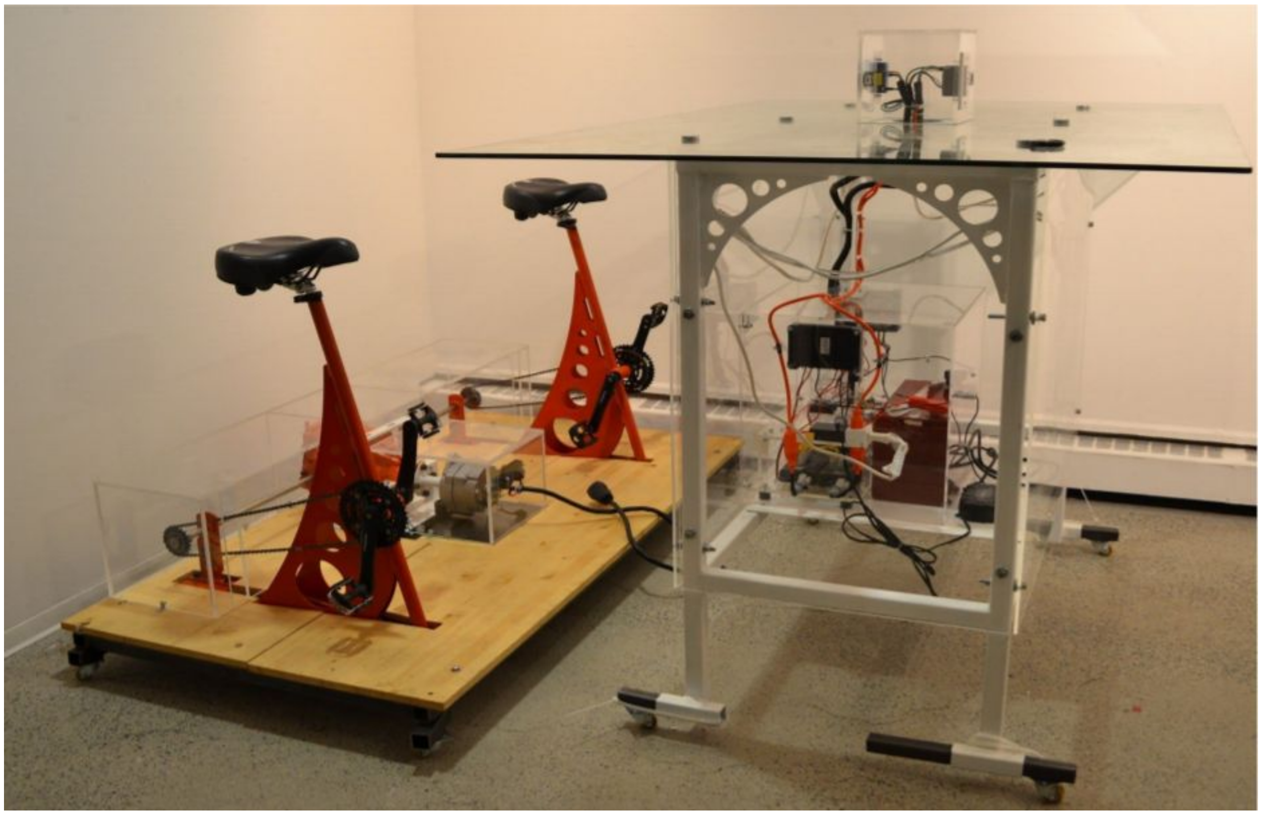


Figure 12: Picture of Overall System[10]

Figure 13: Magnetic Switch Sensor

Figure 14: Windblue 520 Alternator[1]

Figure 15: MorningStar ProStar-30 Charge Controller[2]

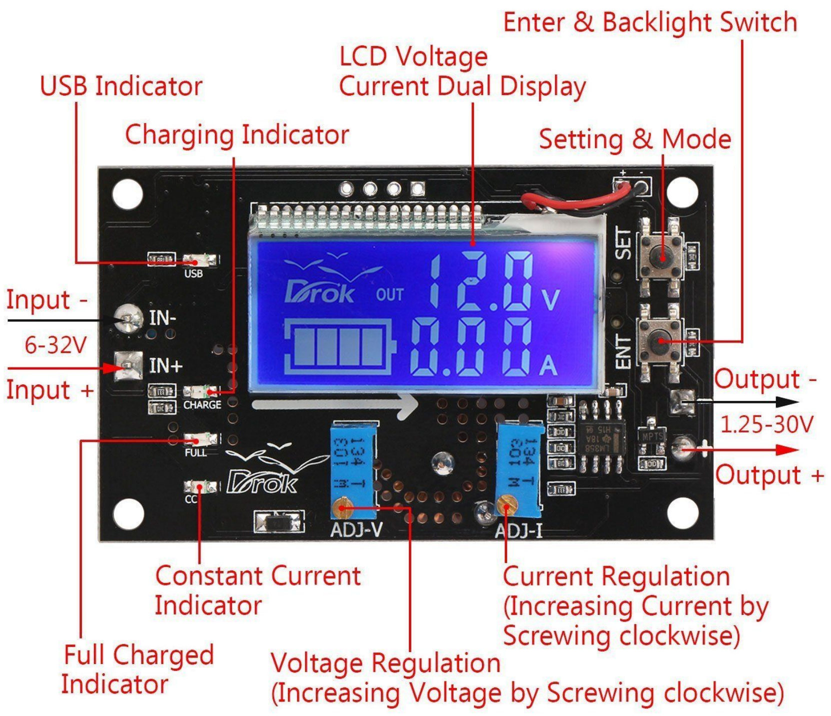

Enter & Backlight Switch
LCD Voltage Current Dual Display
USB Indicator
Charging Indicator
Setting & Mode
Input -
6-32V
Input +
Output -
1.25-30V
Output +
Constant Current Indicator
Full Charged Indicator
Voltage Regulation (Increasing Voltage by Screwing clockwise)
Current Regulation (Increasing Current by Screwing clockwise)
ADJ-V
ADJ-I
12.0 V
0.00 A

Figure 16: DROK CC CV Regulator[6]

Figure 17: Surrette Rolls S12-128AGM 12V 115 AH AGM Battery[3]

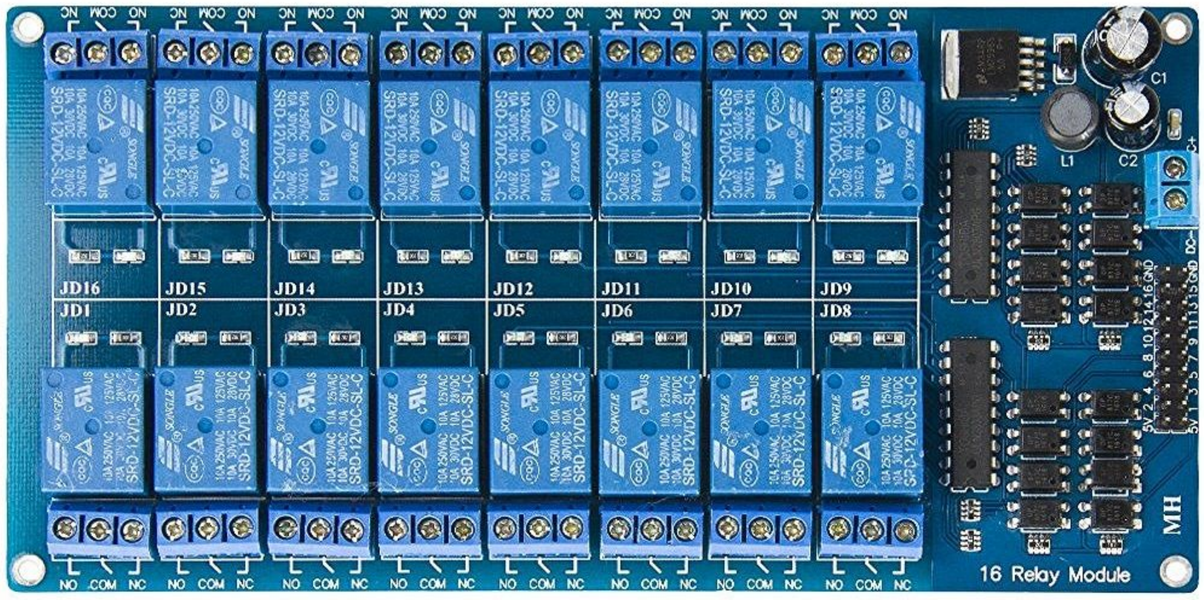


Figure 18: 16-Channel Relay

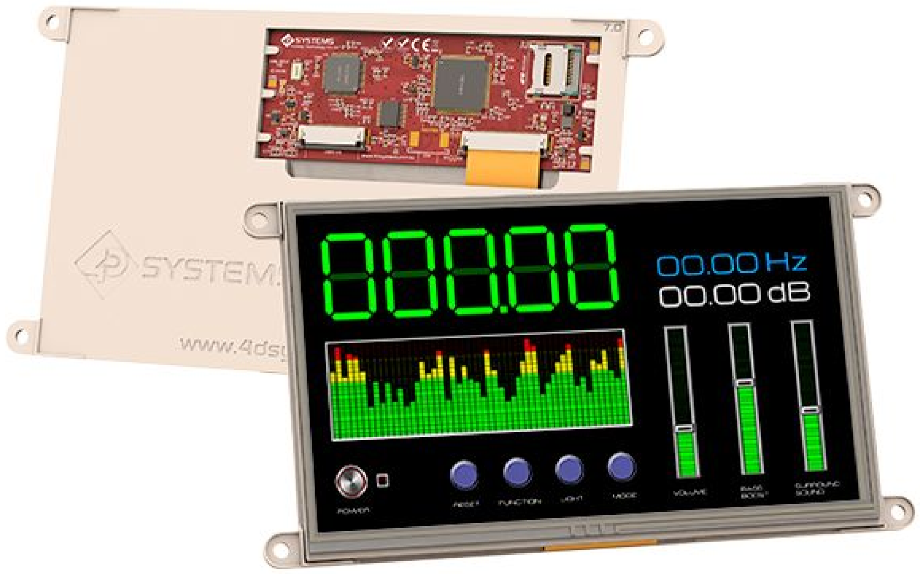


Figure 19: 4D Systems GEN4-ULCD-70DT

Appendix 6: Arduino Code

```
/* Author:      Capstone group PL-23
 * Date:        05.04.2017
 * Description: Main Arduino Framework for Sole Power-Up Project. V3.0.
 *
 * References:  https://www.arduino.cc/
 *
 *
 */

// Libraries
#include <avr/interrupt.h>
#include <avr/power.h>
#include <TimerOne.h>
#include <genieArduino.h> //Library and some code/comments from 4D systems Github
https://github.com/4dsystems/ViSi-Genie-Arduino-Library

// Pins
int sensorPin1 = 2;   // Sensor 1
int sensorPin2 = 3;   // Sensor 2
int chargingPin = 10;  // Phone charging relays        (K6)
int chargingPin2 = 11;  // Phone charging relays       (K7)
int powerPin = 9;     // Power relays
#define RESETLINE 4

// Constants
const int DEBOUNCE = 100;
const int RESET_TIME = 3;
const int STPCHRG = 5; // Power down time constant in seconds
const int PWRDWN = 10; // Power down time constant in seconds
const int PHONE = 5;
const int NINTENDOSWITCH = 39;
const int MACBOOKPRO13 = 61;
const int BLENDER = 300;
const int TOASTER = 800;
const int MICROWAVE = 1000;


// Variables
int rpm;    //This is the value we intend to calculate.
int rpm2;    //This is the value we intend to calculate.
volatile int sensor_count = 0;
volatile int sensor_count2 = 0;
volatile int race_count = 0;
volatile int race_count2 = 0;
volatile int seconds = 0;
int rate_flag = 0;
int power_flag = 0;
int sensor_flag = 0;
int sensor_flag2 = 0;
int charging_flag1 = 0;
int charging_flag2 = 0;
int power_down_count = 0;

// Voltage Measurement Variables
```

```
float sample1;
float sample2;
int record;
float voltage;
float current;
int power;
int power2;


// To control the touchscreen
  Genie genie;
  static long waitPeriod;
  int currentForm;

void setup() {

  // Touchscreen setup
  currentForm = 0;

  pinMode(chargingPin, OUTPUT);
  pinMode(chargingPin2, OUTPUT);
  digitalWrite(chargingPin, HIGH); // Charging relay off
  digitalWrite(chargingPin2, HIGH);

  pinMode(powerPin, OUTPUT);
  digitalWrite(powerPin, LOW);  // Power relay on


  // setup code here
  pinMode(sensorPin1, INPUT_PULLUP);           //Sets the pin as an input
  attachInterrupt(digitalPinToInterrupt(sensorPin1), Rate_interrupt, RISING);  //Configures
interrupt 0 (pin 2 on the Arduino Uno) to run the function "Rate"

  // setup code here
  pinMode(sensorPin2, INPUT_PULLUP);           //Sets the pin as an input
  attachInterrupt(digitalPinToInterrupt(sensorPin2), Rate_interrupt2, RISING);  //Configures
interrupt 0 (pin 2 on the Arduino Uno) to run the function "Rate"

  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);

  // Initialize timer1, and set a 1 second period
  Timer1.initialize(1000000);
  Timer1.attachInterrupt(timerInterrupt);

  Serial.begin(115200);
  genie.Begin(Serial);   // Use Serial0 for talking to the Genie Library, and to the 4D
Systems display
  genie.AttachEventHandler(myGenieEventHandler); // Attach the user function Event Handler
for processing events

  pinMode(RESETLINE, OUTPUT);
  digitalWrite(RESETLINE, 0);  // Reset the Display via D4
  delay(1000);
  digitalWrite(RESETLINE, 1);  // unReset the Display via D4

  delay (5000); //let the display start up after the reset (This is important)

  genie.WriteContrast(10); //0-15 for Brightness Control, where 0 = Display OFF, though to
15 = Max Brightness ON
```

```
}

void loop() {
  unsigned long current = millis();
  static unsigned long previous = 0; // record current and previous interrupt time for
debouncing

  unsigned long current2 = millis();
  static unsigned long previous2 = 0; // record current and previous interrupt time for
debouncing

  interrupts();   //Enables interrupts on the Arduino

  genie.DoEvents();
  UpdateTouchscreen();

  // put main code here, to run repeatedly:

  if (sensor_flag == 1)
  {
    if ( (current - previous) > DEBOUNCE ) // 100ms
    {
      sensor_flag = 0;
      sensor_count++; //Every time this function is called, increment "count" by 1
      if(charging_flag1==1)
        digitalWrite(chargingPin, LOW); // Charging relay on
////////////////////////////////////////////////////

      if(currentForm == 2){
        race_count += 2;
      }
      digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
      power_down_count = 0; // Reset power_down_count
      charging_flag1 = 1;


      previous = current;
    }
  }

  if (sensor_flag2 == 1)
  {
    if ( (current2 - previous2) > DEBOUNCE ) // 100ms
    {
      sensor_flag2 = 0;
      sensor_count2++; //Every time this function is called, increment "count" by 1
      if(charging_flag2==1)
        digitalWrite(chargingPin2, LOW); // Charging relay on
////////////////////////////////////////////////////

      if(currentForm == 2){
        race_count2 += 2;
      }
      power_down_count = 0; // Reset power_down_count
      charging_flag2 = 1;
      previous2 = current2;
    }
  }

  if (rate_flag == 1)
  {
    noInterrupts(); //Disable the interrupts on the Arduino
    rate_flag = 0;
    rpm = 60.0*sensor_count/RESET_TIME+random(-3,3);
    rpm2 = 60.0*sensor_count2/RESET_TIME+random(-3,3);

    //Rate = // some math to include time
    sensor_count = 0;  // Reset the counter so we start counting from 0 again
    sensor_count2 = 0;  // Reset the counter so we start counting from 0 again

  }

  if (power_flag == 1)
  {
    noInterrupts(); //Disable the interrupts on the Arduino
    power_flag = 0;

    sample1=analogRead(A1); // read the voltage from the divider circuit
    sample2=analogRead(A2); // read the current from the current mesuring circuit

    voltage=(sample1*4.0816641*5/1024); // voltage calculation
    current=(((36.7*sample2)/(1024))-18.418); // current calculation
    power= voltage*current; //////////////////////////////////////////////////////////////
    power2 = power * 10;

/*
    Serial.println("Rate:");
    Serial.println(Rate);         //Print the variable Rate to Serial
    Serial.println("Voltage:");
    Serial.println(voltage);         //Print the variable voltage to Serial
    Serial.println("Current:");
    Serial.println(current);         //Print the variable current to Serial
*/

    // Reset samples
    sample1=0;
    sample2=0;

  }

// if (power_down_count == STPCHRG) // if system hasn't been used in PWRDWN seconds
//  {
//    // open IO8
//    digitalWrite(chargingPin, HIGH); // Charging relay off
//  }

  if (power_down_count == PWRDWN) // if system hasn't been used in PWRDWN seconds
  {
    // open IO9
    digitalWrite(powerPin, HIGH);    // Power relay off
  }

}
```

```
void Rate_interrupt()
{
   sensor_flag = 1;
}

void Rate_interrupt2()
{
    sensor_flag2 = 1;

}


// Timer reset ISR
void timerInterrupt(void)
{
  seconds++;

  power_down_count++;

  power_flag = 1;

  if (seconds == RESET_TIME)
  {
    seconds = 0;
    rate_flag = 1;
  }
}

//
//void UpdateTouchscreen(void)
//{
//  genie.WriteObject(GENIE_OBJ_COOL_GAUGE, 0,44); //Rider 1 Speed
//}

// Genie (Touchscreen) event handler
void myGenieEventHandler(void)
{
  genieFrame Event;
  genie.DequeueEvent(&Event);

  if (Event.reportObject.cmd == GENIE_REPORT_EVENT)
  {
    if (Event.reportObject.object == GENIE_OBJ_FORM)
    {

      if (Event.reportObject.index == 0)
      {
          currentForm = 0;
      }
      if (Event.reportObject.index == 1)
      {
          currentForm = 1;
      }
      if (Event.reportObject.index == 2)
      {
          currentForm = 2;
      }
      if (Event.reportObject.index == 3)
      {
          currentForm = 3;
      }
      if (Event.reportObject.index == 4)
      {
          currentForm = 4;
      }
      if (Event.reportObject.index == 5)
      {
          currentForm = 5;
      }
      if (Event.reportObject.index == 6)
      {
          currentForm = 6;
      }
      if (Event.reportObject.index == 7)
      {
          currentForm = 7;
      }
      if (Event.reportObject.index == 8)
      {
          currentForm = 8;
      }
      if (Event.reportObject.index == 9)
      {
          currentForm = 9;
      }
      if (Event.reportObject.index == 10)
      {
          currentForm = 10;
      }

    }
  }
}


void UpdateTouchscreen(void)
{

    if(rpm < 0)
    {
      rpm = 0;
    }
    if(rpm > 200)
    {
      rpm = 200;
    }
    if(rpm2 < 0)
    {
      rpm2 = 0;
    }
    if(rpm2 > 200)
    {
      rpm2 = 200;
    }

      //Need to send int for all WriteObject data parameters
    if(currentForm == 0) //Main screen (green)
    {
      genie.WriteObject(GENIE_OBJ_COOL_GAUGE, 0, rpm2); //Rider 1 Speed
      genie.WriteObject(GENIE_OBJ_COOL_GAUGE, 3, rpm); //Rider 2 Speed
      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 3, power2); //Total watts

      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 0, 10*power/PHONE); //Number of phones (5W)
      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 1, 100*power/NINTENDOSWITCH); //Number of
laptops
      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 2, 100*power/MACBOOKPRO13); //Number of taco13

    }
    if(currentForm == 1) //Main screen (orange)
    {
      genie.WriteObject(GENIE_OBJ_COOL_GAUGE, 1, rpm2); //Rider 1 Speed
      genie.WriteObject(GENIE_OBJ_COOL_GAUGE, 2, rpm); //Rider 2 Speed
      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 7, power2); //Total watts


          genie.WriteObject(GENIE_OBJ_LED_DIGITS, 4, 10*power/PHONE); //Number of phones
(5W)
          genie.WriteObject(GENIE_OBJ_LED_DIGITS, 5, 100*power/NINTENDOSWITCH); //Number of
laptops
          genie.WriteObject(GENIE_OBJ_LED_DIGITS, 6, 100*power/MACBOOKPRO13); //Number of
tacos


    }

    if(currentForm == 9) //Main screen (green)
    {
      genie.WriteObject(GENIE_OBJ_COOL_GAUGE, 4, rpm2); //Rider 1 Speed
      genie.WriteObject(GENIE_OBJ_COOL_GAUGE, 5, rpm); //Rider 2 Speed
      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 11, 10*power); //Total watts

      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 8, 100*power/BLENDER); //Number of phones (5W)
      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 9, 100*power/TOASTER); //Number of laptops
      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 10, 100*power/MICROWAVE); //Number of taco13

    }
    if(currentForm == 10) //Main screen (orange)
    {
      genie.WriteObject(GENIE_OBJ_COOL_GAUGE, 6, rpm2); //Rider 1 Speed
      genie.WriteObject(GENIE_OBJ_COOL_GAUGE, 7, rpm); //Rider 2 Speed
      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 15, 10*power); //Total watts


      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 12, 100*power/BLENDER); //Number of phones
(5W)
      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 13, 100*power/TOASTER); //Number of laptops
      genie.WriteObject(GENIE_OBJ_LED_DIGITS, 14, 100*power/MICROWAVE); //Number of tacos


    }
    if(currentForm == 2) //Race screen
    {
      genie.WriteObject(GENIE_OBJ_GAUGE, 0, race_count2); //Rider 1 bar
      genie.WriteObject(GENIE_OBJ_GAUGE, 1, race_count); //Rider 2 bar

      if(race_count == 100)
        {
         genie.WriteObject(GENIE_OBJ_FORM, 8, 0);
         currentForm = 8;
        }

        if(race_count2 == 100)
        {
         genie.WriteObject(GENIE_OBJ_FORM, 7, 0);
         currentForm = 7;
        }
    }
    if(currentForm == 3) //Press button to start race
    {
      race_count=0;
      race_count2=0;
    }
    if(currentForm == 4) //Countdown 3
    {
      delay(1000);
      genie.WriteObject(GENIE_OBJ_FORM, 5, 0);

      currentForm = 5;
    }
    if(currentForm == 5) //Countdown 2
    {
      delay(1000);
      genie.WriteObject(GENIE_OBJ_FORM, 6, 0);

      currentForm = 6;
    }
    if(currentForm == 6) //Countdown 1
    {
      delay(1000);
      genie.WriteObject(GENIE_OBJ_FORM, 2, 0); //Start race

      currentForm = 2;
    }

    if(currentForm == 7) //Rider 1 wins race, press button to go back to main screen
    {
      race_count=0;
      race_count2=0;
    }
    if(currentForm == 8) //Rider 2 wins race, press button to go back to main screen
    {
      race_count=0;
      race_count2=0;
    }


}
```

Appendix 7: Edutainment System Mockups

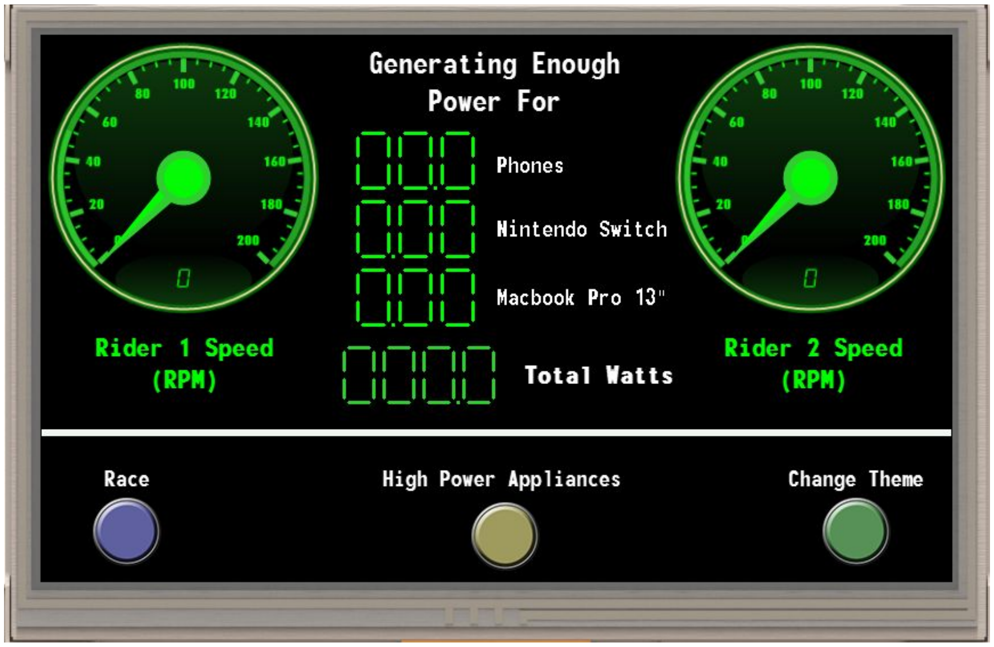


Figure 20: Main entertainment system screen (low power items)

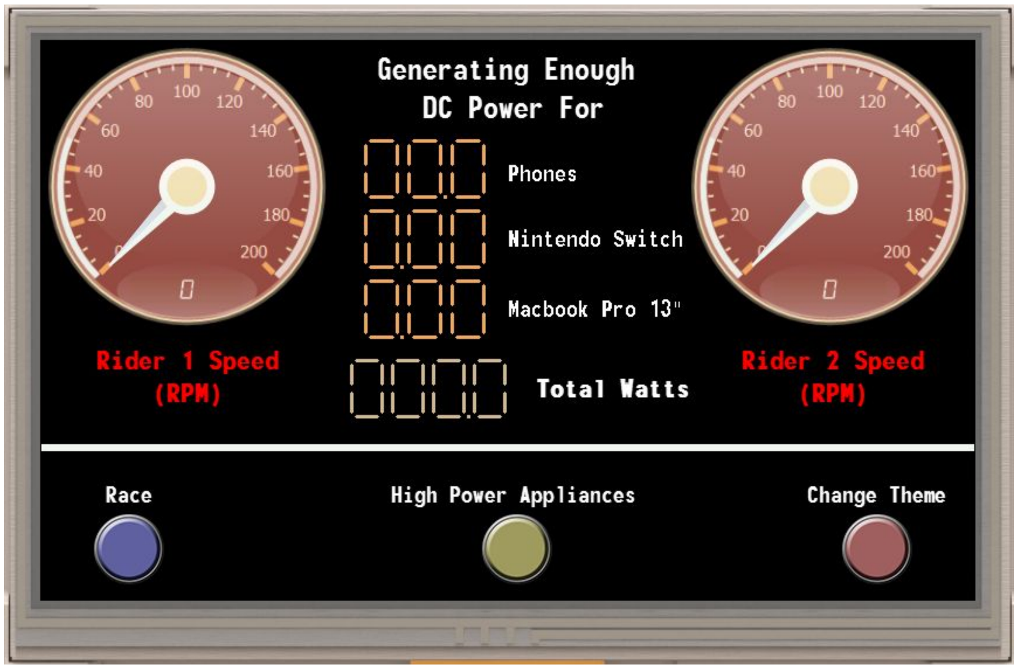


Figure 21: Main edutainment system screen with alternate theme (low power items)

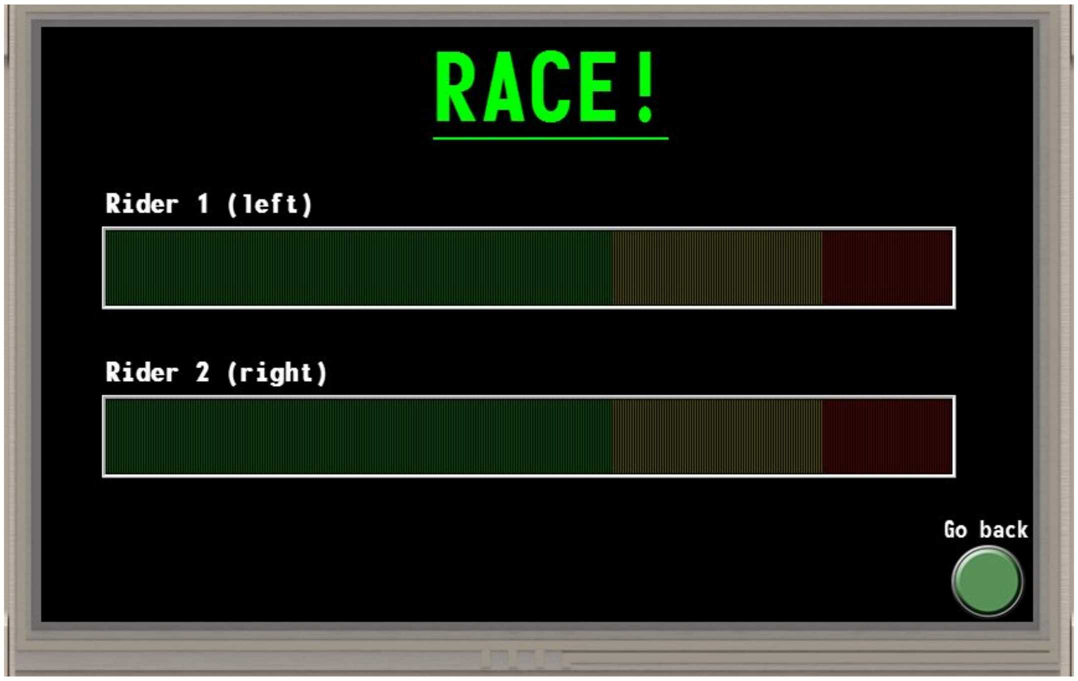


Figure 22: Race between the two users

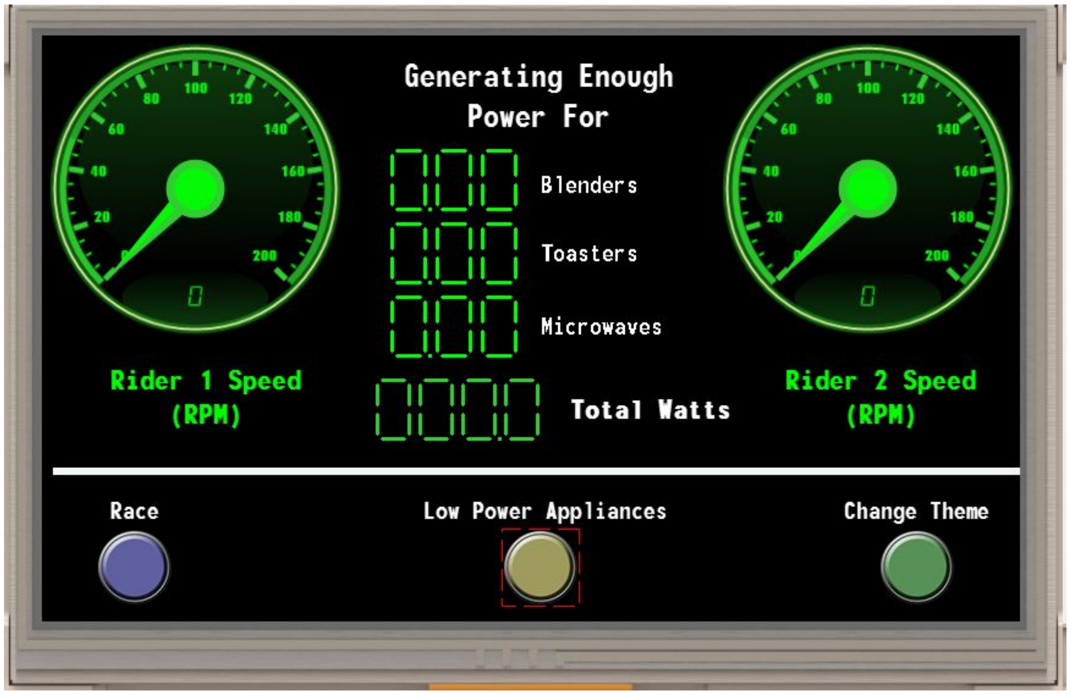


Figure 23: Main entertainment system screen (high power items)

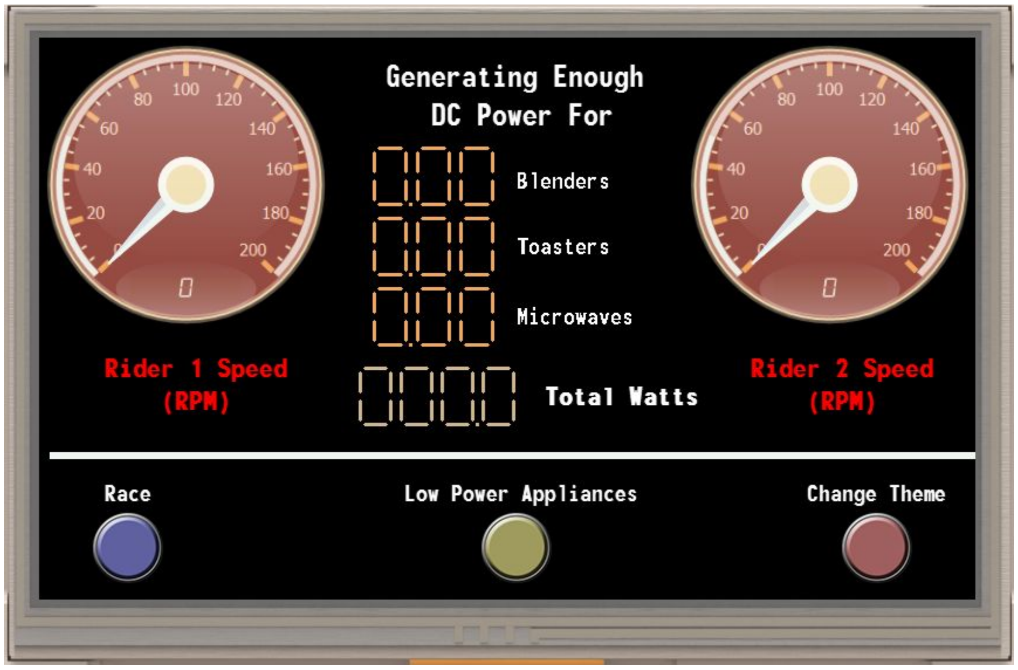
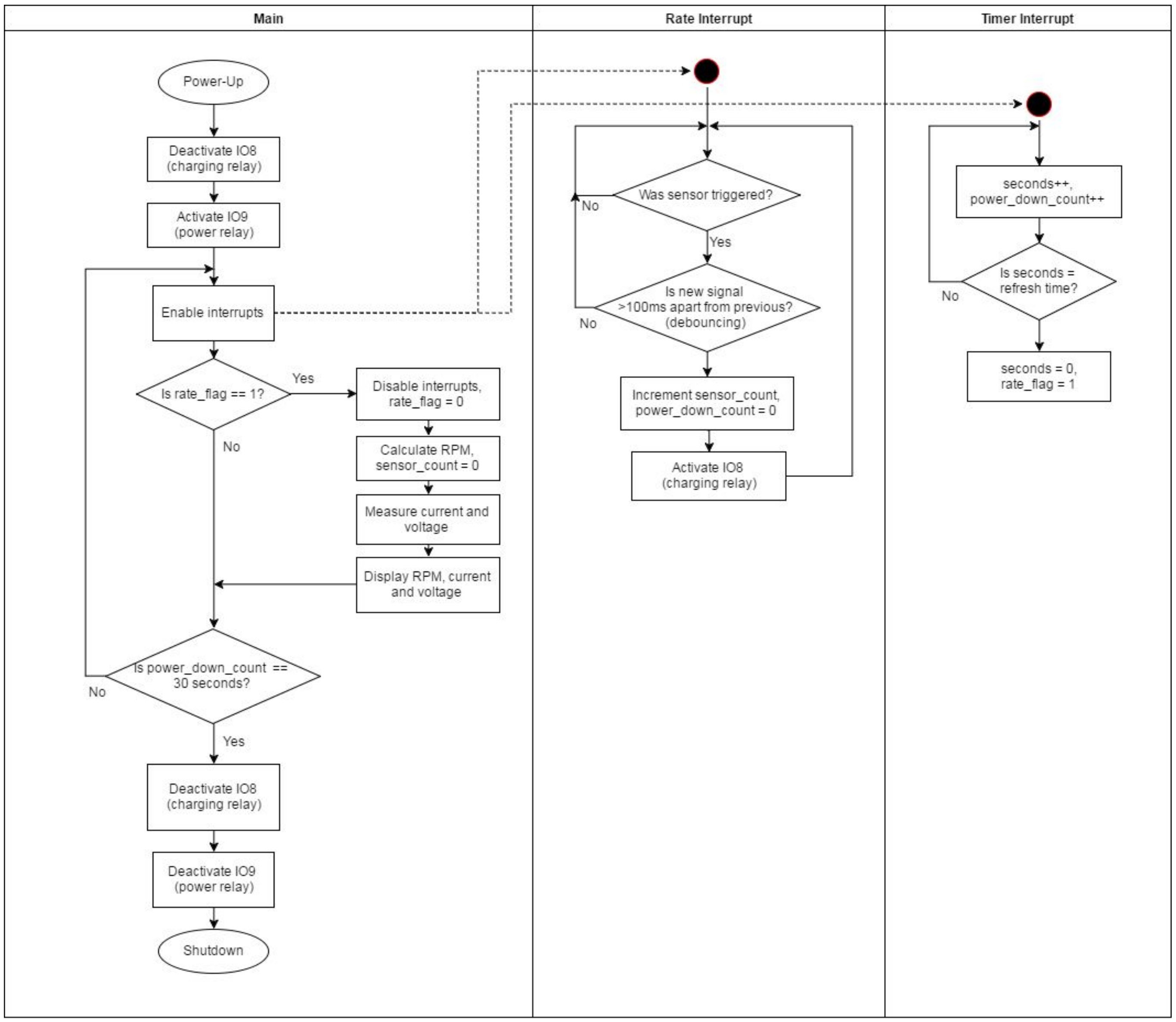


Figure 24: Main edutainment system screen with alternate theme (high power items)

Appendix 8 : Microcontroller Algorithm Flowchart



Main
Rate Interrupt
Timer Interrupt
Power-Up
Deactivate IO8 (charging relay)
Activate IO9 (power relay)
Enable interrupts
Is rate_flag == 1?
Yes
No
Disable interrupts, rate_flag = 0
Calculate RPM, sensor_count = 0
Measure current and voltage
Display RPM, current and voltage
Is power_down_count == 30 seconds?
No
Yes
Deactivate IO8 (charging relay)
Deactivate IO9 (power relay)
Shutdown
Was sensor triggered?
No
Yes
Is new signal >100ms apart from previous? (debouncing)
No
Increment sensor_count, power_down_count = 0
Activate IO8 (charging relay)
seconds++, power_down_count++
Is seconds = refresh time?
No
seconds = 0, rate_flag = 1