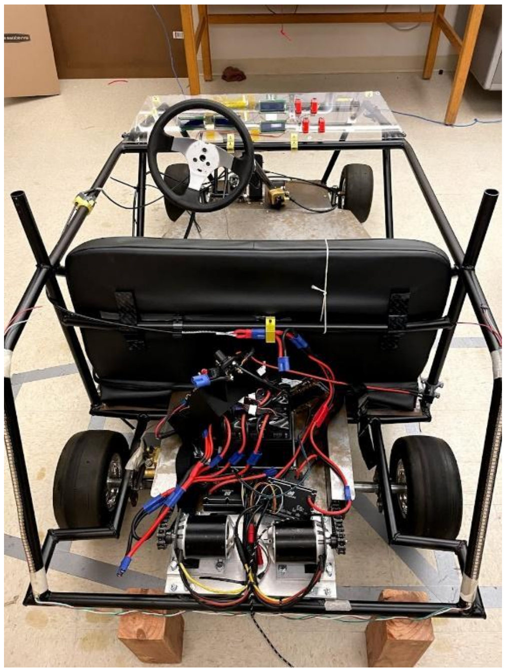
ELECTRICAL AND COMPUTER ENGINEERING
COLORADO STATE UNIVERSITY

E-Kart E-days Owner's Manual:

Everything you need to know about the Go-Kart from '21-22 Team



Author:

Vani Kapoor Systems Engineer

Table of Contents

Subsystem Explanation:

This first section highlights what each subsystem before we get into operation. Following the explanation will be the schematics that further show the design of a few of the sub systems. It is important to understand thoroughly how and what everything is and the vitality before we operate the vehicle to prevent damages.

Whole system overview: The Go-Kart is a highly modular, multidisciplinary electric vehicle advancing towards very modern technology used in every day EVs*. Starting up front is the power system for most small electronics, single board computers, and microcontrollers. This is powered by a camping battery while at the same time has solar panel charging capabilities. Moving to the dash we see our UI*. This UI is very involved. We have a full touch screen interface for the driver which reads in all of our data and outputs in real time speed, tachometer readings, and has buttons to stop the vehicle control as well as serving as the interface for our APD. Clicking on the camera button shows the already active APD detecting pedestrians and vehicles through a machine learning module.

UI:

The user interface was first designed in inkscape to create the display that will be put onto the raspberry pi display. This design is then being developed in PyQT 6 designer which will give this design more functionality and work as the sole user interface controlling much of the components on the Go-Kart.

The purpose of this dashboard feature is to display many features that the team is deciding to implement and take measurements from. This first screen will include information such as speedometer, tachometer, and battery charge levels. Since this will be displayed on a touchscreen, the next biggest feature to be included is that students will be able to click the screen and then begin viewing the camera output as talked about in the next section. Along with selection options, this will also feature the ability to control all of the fun gimmicks that the team plans on adding next semester. We intend on including a color wheel to change the colors of the light strips that we are adding as well as displaying more data that is coming from new sensors that the team plans on adding next semester.

A key component of this dashboard feature is also to display when there will be any warning or faults in the control system. These faults will range from faults that the devices we have choses already give warning too as well as faults not yet fully chosen such as “battery critical” and motor and ESC* faults.

CAN-SPI:

With a Raspberry Pi 4B running our user interface (UI), we wanted a way for the electronic speed controllers (ESCs) controlling the kart to be able to communicate with the interface. Following noise and reliability issues when last year’s team used I2C for this communication, we pursued the CAN protocol. CAN is widely used in the automotive industry due to its reliability, so we figured it would be a perfect fit for our go-kart.

The Raspberry Pi does not natively support the CAN protocol though, so we had to look into ways to make this possible. We settled on using MikroElectronika’s CAN-SPI Click boards, as they fulfilled our use case well and our EiR could vouch for their design. These boards work in our system by setting one up as a CAN network interface within the Pi’s Linux-based operating system, then writing to it using the SPI protocol. The CAN-SPI Click board then uses its SN65HVD230 CAN transceiver and

MCP2515 CAN controller to communicate over the CAN bus we established between the ESCs and the CAN-SPI Click board. For now, we are just reading from this bus to retrieve data such as motor RPM, but it would definitely be possible in the future to expand this network and add on other devices that the Raspberry Pi could address frames towards. See Figure 7 in the **Important Schematics** section for a drawing of the CAN connections.

APD:

The Autonomous Pedestrian Detection (APD*) system is a machine learning model monitoring a webcam feed to perform object detection on a variety of vehicular classes, most notably pedestrians. This system was developed in partnership with Andrew Helmreich of the ML* on the Edge senior design team. The main goal of this feature is to provide a real-time demo of bounded-box object detection on the kart. While this data could theoretically be used to implement autonomous braking functionality (and we consistently evaluated the system's performance in this capacity), we decided this was too complex of a scenario to put our model into. Based on the overall goal of the kart, to demonstrate ECE* applications to students, we decided it was more important to have a live demo of this feature, rather than have it actually make decisions for the driver.

The base model of APD* is YOLOv5*, a machine learning model developed by Joseph Redmon and Ali Farhadi for real-time object detection. This model was selected due to its impressive performance in the real-time object detection area, where it consistently matches the accuracy of competing models while doing so in much less time. This was important for our application, as we theoretically want to keep the kart's stopping distance as low as possible in a braking scenario, and decreased model latency results in a shorter stopping distance. As for the data used to train the YOLOv5* model, we went with the KITTI* dataset. Developed by the Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago, the KITTI* dataset is heavily geared towards autonomous driving. With built-in classes for pedestrians, cars, cyclists, and other vehicle and traffic-oriented objects, this was the perfect choice as we didn't have to edit the dataset's built-in classes. We also added some of our own data using the Roboflow tool. By importing a few hundred of our own pictures of people, we used the tool to draw bounding boxes around them and label them as pedestrians, which added some extra data to the main class we were targeting with the APD* system.

To deploy this model, we are using a Raspberry Pi 4B with a Coral USB* accelerator and a Logitech C920X USB* webcam. The Raspberry Pi 4B was selected due to our familiarity with the Raspberry Pi line of single-board computers and the versatility they allow in these types of applications due to their small size, low power consumption, and relatively high performance. The CPU* of the Raspberry Pi 4B isn't quite powerful enough for our needs in machine learning inference performance though, so we purchased a Coral USB* accelerator to help in this category. By just plugging this accelerator in over USB*, making slight modifications to our deployment code, and re-compiling the model for Coral operations, we went from inference times per frame of ~3s to ~0.03s. Before we had the Coral, all 283 operations in our model's inference process had to be run on the Raspberry Pi's CPU*, but after re-compiling for the Coral, just 21 now run on the CPU*, with the remaining 262 operations being executed on the Coral accelerator. In addition to the Coral's custom Edge TPU* accelerating inference by being specifically designed for that application, the re-compilation also employs quantization on our model. Quantization approximates our original model from using 32-bit floating point numbers down to 8-bit integers, which only slightly hurts accuracy while drastically improving inference times. We also purchased a Logitech C920X USB* webcam to use as video input to our model due to its high resolution and good performance in autofocus and lighting correction.

Table #1: APD Performance at Various Resolutions

Input Resolution	Inference Time (s)	FPS*	CPU Ops*	TPU Ops*	mAP* (%)
256x256	0.021	47.62	0	285	45.7
320x320	0.032	31.25	21	262	53.7
384x384	0.048	20.83	21	263	54.2
448x448	0.092	10.86	21	263	56.4
512x512	0.163	6.13	21	263	58.1
640x640	0.275	3.64	40	246	61.3

As shown in Table 1, we evaluated several model resolutions for their performance in both inference time and mAP*. We settled on the 320x320 option as the best balance between FPS* and mAP*, as it maintains a respectable mAP score while maxing out the highest possible FPS* we can achieve with our deployment (since our webcam is capped at 30 FPS*). The 30 FPS* (due to the webcam limit) of this model means that if the kart was traveling at 20 MPH*, our system is able to make detections in each frame every 11.73 inches, which we believe is certainly fast enough for our demo. When testing this model in the lab and on footage we recorded from a car, the bounded boxes are fairly accurate. The image shows some inaccuracies in certain boxes, as well as missing a few cars, but it does notably well on the pedestrians and detecting the van in the background.

Back Power System and Motors:

The design of the power systems in the kart had to change quite a bit from what it was last year. After documenting the wiring schematic from last year we found design choices that were causing issues with starting the kart this semester and also may have been causing issues for last semester's team.

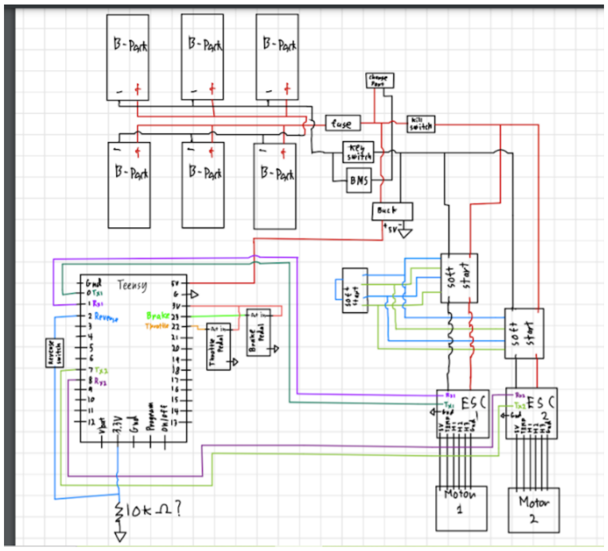


Figure 1: Wiring diagram of last semesters Go-Kart Electronics

From the wiring diagram and testing we found issues with grounding and connection points. The grounding issues were mostly due to the buck converter. With the buck converter the ground for the TEENSYv.4 controller and the rest of the battery power system were shared. This caused power voltage spikes to be present on the control side of the power system which broke the chips of the TEENSYv.4 as well as the Transistor on the throttle pedal. The buck converter could have also been the cause of signal noise issues from last year's team due to it being linked to the control ground. The duty cycle from the buck converter was linked to the signal ground of the TEENSYv.4, which we speculate contributed to unnecessary noise. The other grounding issues happened on the power side of the system with the power switch and the soft starts. The power switch being on the ground had a potential to have a current spike through the buck converter when switched off which could potentially break our controller chips. The problem with the soft starts took a lot of testing to find out what the problem was. It turned out the soft start used a mosfet with a PWM* to create a slow rising voltage differential between the positive and negative wires. The problem was that the mosfet was on the ground side making it so that the ground voltage for the ESCs* was higher than the control ground on the TEENSYv.4 until the soft starts hit steady state. This is a problem because the ESC* and TEENSYv.4 are supposed to share a ground and when the grounds have a voltage differential they cause a voltage larger than the ESC* and TEENSYv.4 chips could handle. Other than grounding difficulties we had a problem with the ESCs losing power at random times for no apparent reason. At first we thought that the connection problem was due to poor soldering but upon closer inspection it was the connector male plugs getting deformed with use that was causing them to be loose with the capability to become open circuited while plugged in.

To fix the grounding issues we had to completely redesign the power systems of the Kart. As mentioned earlier in the report we moved the control power system to a separate power source in the front of the kart to avoid all the issues with having the ground tied to the large voltage at the front of the kart. The redesign of the power side of the kart made it so things were less complicated and streamlined to avoid potential errors due to unneeded complexity.

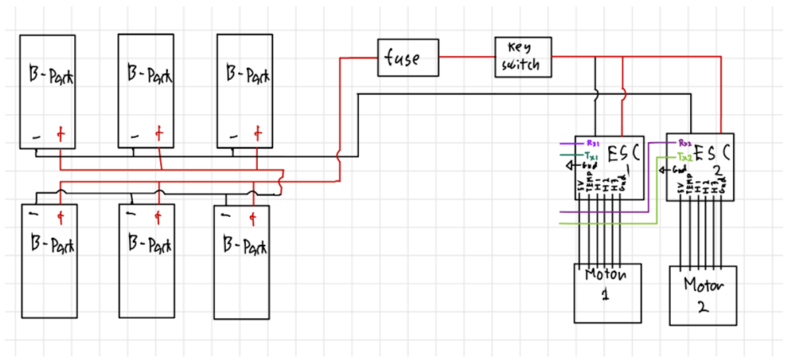


Figure 2: Redesigned Power system for the Motors

As seen in Figure 1, the battery is a lot less complicated than the previous system. The main changes made were the moving of the key switch, and the removal of the charge port as well as the soft starts. The key switch was moved so that the ground is never broken, making for a more reliable circuit. The soft starts were removed to completely take out the ground voltage differential error and we have not had any problems since removing them. The charger port and BMS were removed as we could plug them into the batteries directly at any point to make the physical wiring less cluttered and easier to maneuver. The connection issue took a long time to find because the initial thought was that the wires themselves were faulty. As replacements were being made it showed that the male banana plugs in the connectors had the possibility of collapsing in on themselves and not putting pressure on the female banana plug. This was easily fixed by pushing the male plugs out with a screwdriver at all loose connection points.

Front Power System and Solar Panel Charging:

The idea behind the solar panel charger first came from the decision to have a separate power system for the electronics being placed up front. The team felt as though the voltage rails needed if we were to over use the 5V supply voltage from the buck converter would not supply enough power to our control system which would then render the Go-Kart unusable in many aspects. To combat this issue the design came to add a simple 12V lead acid battery to the front of the vehicle that is easily accessible, and something we are able to buck down to a steady 5V voltage supply. The choice to use a lead acid battery in the front of the vehicle verses using the same LiPo* batteries that are used in the back are because of the way these batteries charge and how they react to overcharging. LiPo* batteries are very sensitive to overcharging, they have a threshold voltage that they cannot exceed and the same case for the voltage when discharged. This would cause the battery permanent damage and therefore we foresaw a far more need for replacement with these batteries. Lead acid is much more tolerable, they do not over charge to the point to where they would break and have no issue being recharged if the battery voltage drops too low. With this understanding the design became more feasible for the semester.

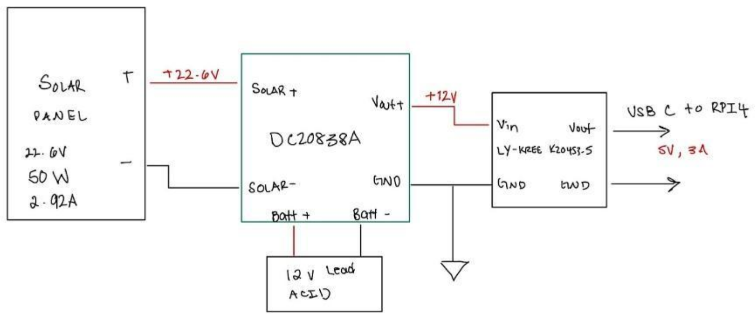


Figure 3: Solar Panel Charging Schematic

The above design includes a 22.6V 50W solar panel which was given to us by the Outreach team, a DC20838A BMS* a 12V-13V lead acid battery and a 5V regulator. The solar panel will be providing anywhere from a 18.75V to the max rated 22.6V input voltage to the BMS* which takes this voltage and supplies charging voltage back to the battery. As well as charging the battery, this device will also be outputting 12V which will serve as the input to our 5V regulator which will then be powering our Raspberry Pi 4B controller hub and 7inch display that goes with it.

This simple design is currently in the process of fabrication. Due to supply chain shortages, we have faced a lot of latency on our orders which has pushed back some of the assembly that has needed to take place. This circuit is placed on a plywood board which is fitted on mounts that we have created to create the look and feel of having electronics under the hood of the car. The solar panel that we have been given is flexible, therefore this will serve as the hood needed to hide the electronics until it is taken out for demonstration with the Outreach team.

Important Schematics:

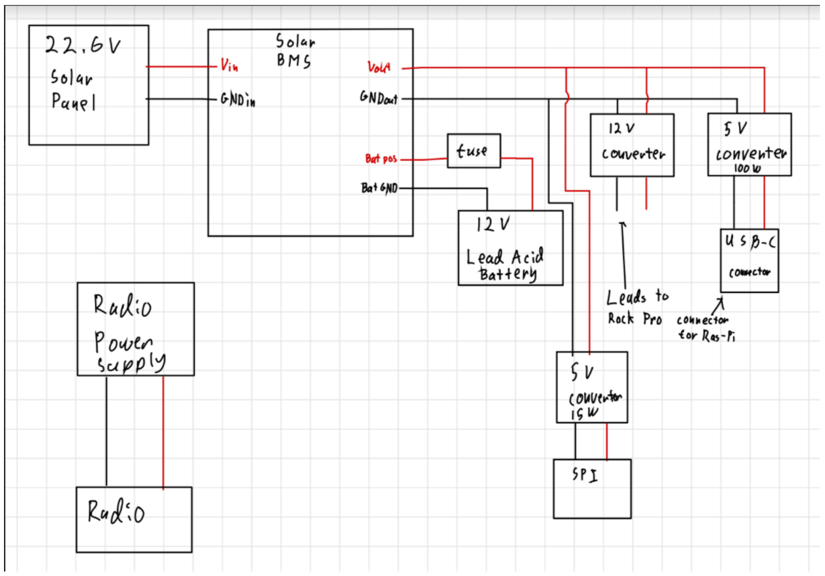


Figure 4: Front Power System Schematic:

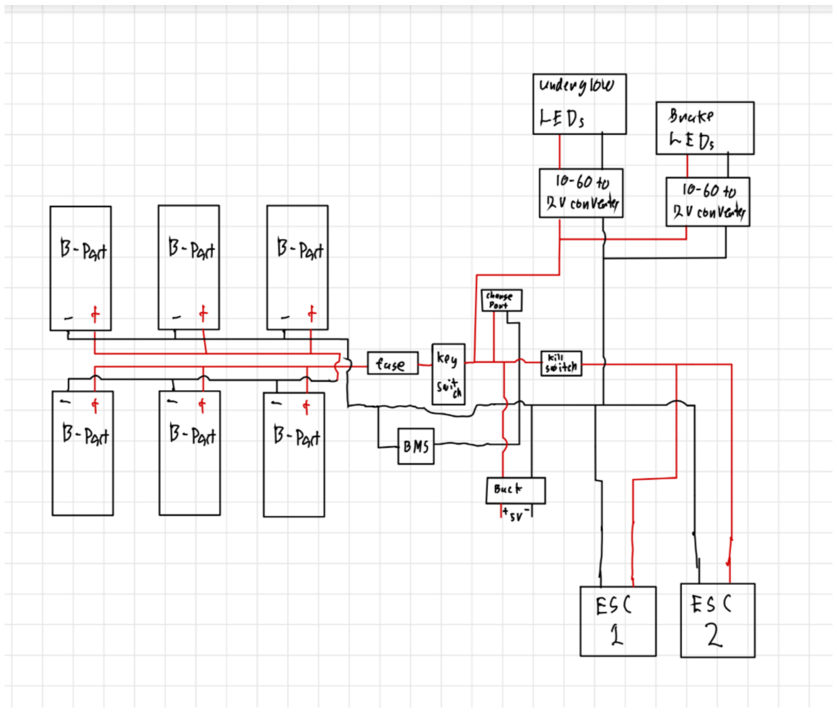


Figure 5: Back Power System:

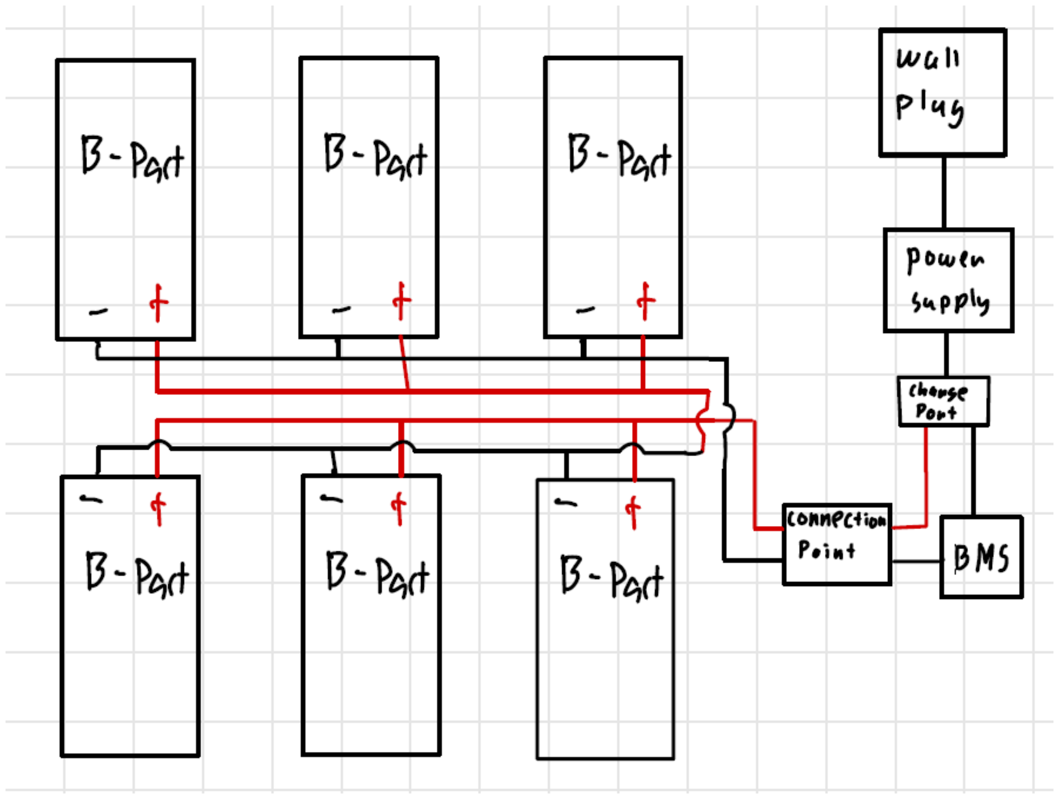


Figure 6: Battery Charging Hook up Schematic:

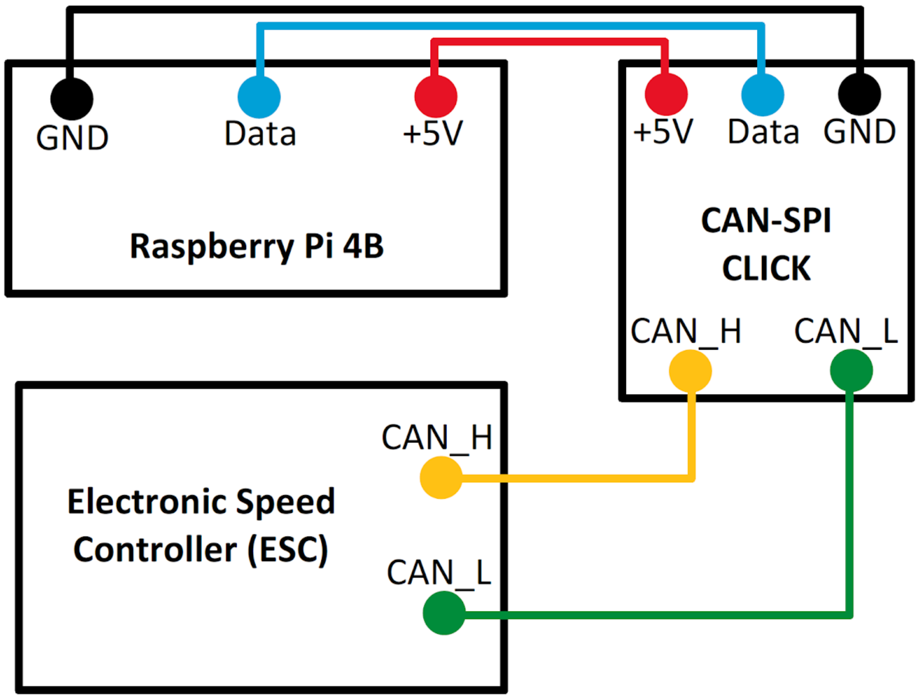


Figure 7: SPI-CAN Connections:

Requirements

- Power Delivery:
  - Front and back systems must be separated all electronics must be powered individually from the main power system
  - Raspberry PI must be powered by 5.1V and no more than 3A by USB type C
  - Solar Panel Charger must include 12V battery for all converters to be connected to
  - Must have at least 30W power tolerance for all front electronics due to down stream current
- SPI-CAN Communication:
  - Main control system and UI must be able to take in and read CAN data from the ESCs
  - Raspberry PI must be able to take in and read CAN data
  - All communication lines must be shielded and twisted to for noise reduction
- UI:
  - UI must fit the 7in touch screen display to its full dimensions
  - Must be able to switch from main screen to APD in real time
  - Must provide data taken in from any added electronic or sensor
- APD:
  - Must be able to demonstrate detection of objects with bounded box
- Motor Control
  - Go-Kart Must be powered by 2 motors at the push of the throttle pedal
  - Motors must implement regenerative braking suitable for stopping entire vehicle

Risk Analysis:

- Big risk was not having a mechanical engineering student had to weld own chassis
  - Caused structural issues and need for new chassis we cannot supply
- We have to be mindful of younger students and risk around children
  - Every aspect of this vehicle has to be safe for children to interact with
- Electronics: Grounding issues
  - Imposes safety issues at the end of the day if not dealt with has been solved but posed a real risk for voltage arc during a demo
- Machine learning may not be suitable yet for autonomous braking
  - We have chosen to look more into adding data and therefore making the system more reliable before we start this task

Testing:

EE Testing Requirements:

Electronic components will mostly be tested on the large kart with the exception of sensors which will first be tested on the small kart. Having access to a small scale of the vehicle makes it easier to troubleshoot before integrating the components into the final design. An additional feature of the smaller kart for testing is the ability to move it to different locations easily, so engineers will be able to take it with them, if the need arises, for more testing.

Electronics Testing Strategy		
Componen t	**Testing Strategy**	**Expected Result**
Solar Panel	-Use a multimeter to read the initial output of the panel and check each cell is working -Find the current incident radiation flux in the area and use the solar panels surface area to determine the theoretical maximum power output -Being as there is constant variation in the light observed, we need to use an artificial light force to determine the acceptable minimum power from the panel to still operate	The solar panel will be within an acceptable range of the theoretical maximum power and we can conclude it is operating well enough for demonstrations and operation(This value will never be the same and is calculated by the tester when using the panel). The solar panel does not need to produce power at all times, it is for charging and is not critical to any components of the vehicle. Minimum power to operate the solar panel is 600W, anything less in an hour will not charge the battery or power any electronics.
Batteries	-Connect a multimeter to the load and simulate low power operation on each battery -Check the readings periodically and ensure they are within expected ranges for 3+ hours -Connect a 1M ohm resistor to the leads of each battery, one-by-one, and observe the current and how long the voltage takes to drop. Do not perform this test for long, the resistor could be damaged	There should be 8 amps output and a slow decline until the batteries reach low safe voltage controlled by a PCB in the battery. With a large load connected, the current should be less than 1mA and voltage should be steady.
Buck Booster	-Use TI's GUI and adjust the voltage control -Connect a power supply to the input terminals send the voltage command in software, use multimeter to read output voltage -A function generator and oscilloscope can be used to flick the booster on and off between functions, the oscilloscope will show output and a multimeter can	There should be the desired output voltage with the given input from either the GUI or code. This is a situational value depending on what inputs are used in the GUI or code. Final design will step down to 5V or 50V +/- 1V.

	confirm the correct step down or up voltage.	
LCD Display	-Connect the RPi to the display loaded with raspbian or other linux tool -Configure the screen	Once powered on, the screen will immediately display. Brightness will be adjustable and the touch screen can be calibrated. Display was purchased over the counter and will need very little testing, other than power on, from the team.
ESC	-Connect a motor to the ESC off the kart, on the motor mount -Connect the ESC to VESCtool and configure the PWM -Simulation software can be used in VESCtool, or CAN data can be collected to ensure proper communication -Placing a 120ohm resistor to terminate the CAN bus can force a constant data stream	The VESC tool will allow us to read the output from the motor to the ESC, we expect adjustable PWM within spec and adjustable speed from 0-9000rpm. The PWM should be configurable to allow the motors to operate together on the single rear axle, it is an automated process in VESCtool to configure the PWM. CAN data should be hex, expect [ff] if powered on at 5v with 120ohm load, expect CANopen protocol
LED lights	-Power on the lights on the test bench -Use controller to run through colors	Lights will turn on and be able to change to RGB depending on what color we send. Lights were purchased over the counter, they will need no testing other than a power on test from the team.
Transformer	-Mathematically compute the expected output before simulating for expected results -Apply a voltage to the transformer and measure it’s output -Compare results and decide if simulated results are optimal for the kart	The transformer should double the voltage that is being inputted into it (12V-24V or 24V-48V). This depends on the number of windings that was previously solved for. For our kart we need a transformer to output roughly 50V using a 24V input.

Future Work

- Fabricate new chassis with functioning mechanical brakes
    - The current chassis is compromised because of a cut made to center the steering column. This is bound to fail and for future teams there needs to be safety measures taken. This includes complete refabrication of the chassis and adding mechanical brakes back on the vehicle in case the system fails
- Further software integration
    - Automated configuration of ESC software on start-up
        - This will solve a lot of issues and make things a lot easier when performing further development of the vehicle
- Addition of more sensors for further data collection and ML development
- Increased user interface integration with drivetrain, LED, and radio control systems