Pi-Top Telemetry Standard Operating Procedure

All Saints' Episcopal School Solar Car Team

1 Integrating the Arduino into the Solar Car

The Rasberry Pi inside the Pi-Top communicates with an arduino which must be integrated into the solar car to receive power and data.

1.1 Power Supply

What we did...

We powered the arduino using the auxiliary battery.

How to do it...

- Attach a wire from the negative side of the auxiliary battery terminal bus into the negative port of the power supply on the arduino.
- Attach a wire from the positive side of the auxiliary battery terminal bus bar into the positive port of the power supply on the arduino.
- When the auxiliary battery system is turned on, the arduino will power up, and a green light on the board will turn on.

Our reasoning, problems we encountered, and other potential approaches...

This will consume the auxiliary battery power faster than when it was not attached, but this is not a pressing issue. It would be interesting to see how power much the arduino pulls from the auxiliary battery, and this may be necessary for some teams to do depending on what batteries they are using. We tested the arduino's effects on our auxiliary batteries through experimentation, and it did not seem to have a large effect on the battery's life. The other approach to powering the arduino would be to integrate it into the propulsion battery system. This would require some method to lower the current running through the arduino, which is further explained in [1.2 Current Readings].

Other approaches may include installing some third battery system (besides the auxiliary battery system and propulsion battery system) that is powered by means other than a battery. Though this approach is much more experimental and may violate a rule, it would be interesting to see other ways it could be powered.

1.2 Current Readings

What we did…

We used a Hall effect sensor and a converter given to us. Hall effect sensors give voltage readings which are directly proportional to the magnetic field going through it induced by the flowing current of a wire. The converter was a 3 digit LED display that was programmed so that, when wired to the Hall effect sensor, rather than showing the voltage readings, it showed current readings.

How to do it…

- Attach the Hall effect sensor to any part of a 4 gauge wire used in the propulsion battery system. We attached it using tape, but there are most likely other ways to do it that are more reliable.
- Attach two of the wires coming from the hall effect sensor to the converter/monitor, and attach the other two into the ADC inputs.

Our reasoning, problems we encountered, and other potential approaches...

Figuring out how to measure the current of the car was the most difficult task we encountered. The pi-top-helios only has analog inputs which measure voltage. Our first plan was to use the shunt already in the system as a known resistance, measure the voltage across that, and use ohm's law to calculate the current in out code. This is how our previous battery monitor measured our current. However, this was not feasible because the shunt had such a low resistance (to avoid power loss in the system), creating such a small voltage drop the pi-top-helios could not pick up the voltage.

Our second approach was to use the hall effect sensor which measured the flux around a wire to measure current. This produced a signal by voltage which we fed into the arduino which measured that voltage. We then had to calibrate the voltage with the actual current going through which we did by plotting points of the voltage to the known current, creating a graph and an equation, and reflecting that equation in the code which translated the voltage read into current.

1.3 Voltage Readings

What we did…

There are two voltage readings we were gathering: auxiliary and main voltage. Both readings are obtained by running wires from either side of the batteries (+ & -) into the ADCs on the board.

How to do it…

- For the auxiliary battery, lead two wires from each of the battery's terminals into two separate ADC inputs on the arduino. The negative lead is actually not imperative, because the board is powered by the aux battery, so the ground of the board is on the voltage level of that negative lead. Effectively, the negative terminal on the power acts as the negative lead for the ADC input.
- For the propulsion batteries, lead two wires from either side of the battery box. It is not important where they come from within the system, just they must be on either side of the battery terminals, or either side of the load, so it can read the voltage drop.
- NOTE: The way the board is designed, with both the auxiliary and main battery systems being read by the board, they share a similar ground. This connects the two systems which is in direct violation to the rules. However, at the race last year, they made us prove that that was the only grounding point between the two systems, and let us pass because it was not exactly our fault they were grounded together.

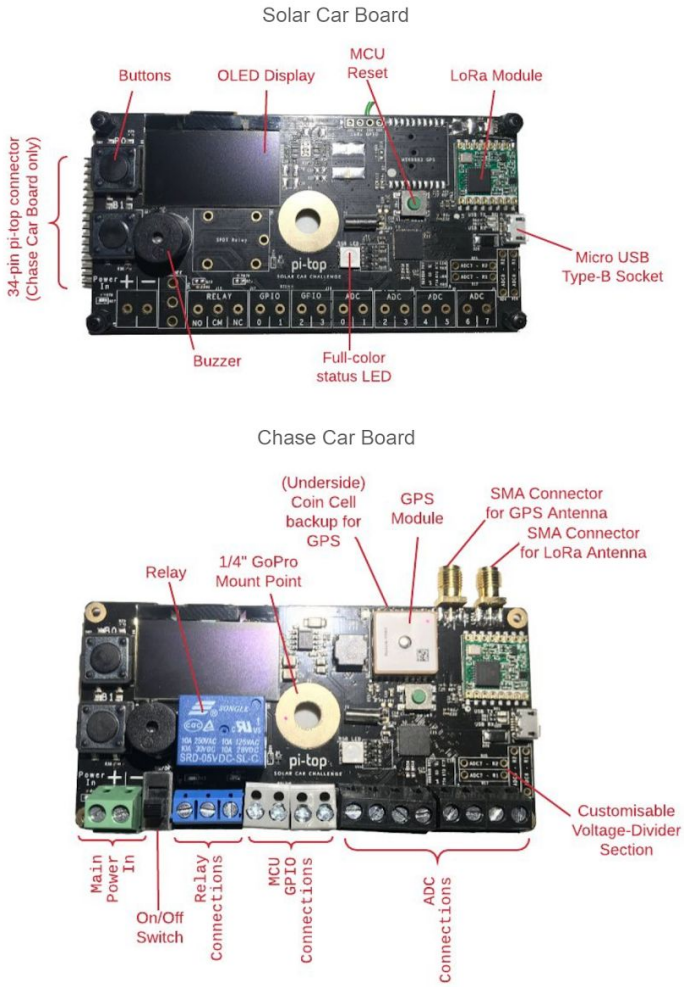
1.4 Other Applications

1.4.1 Current Indicator

This was a project we started playing with at the race last year, but were never able to finish. The idea is to have three LED lights on the driver's dashboard, side by side. When the driver is running the car at (or at least near) the desired current, the middle light (any color, we had blue) turns on. If the driver is running the car at too low of a current, the left light (any color, we had green) turns on. If the driver is running the car at too high of a current, the right light (any color, we had red) turns on. This creates a feedback loop for the driver without him/her needing to constantly take his/her eyes off the road.

1.4.2 Battery Monitor System

Although I am unsure if the arduino has enough ADC inputs to do this completely on the arduinos and pi-tops, perhaps a BMS is a good idea to monitor all of the batteries within the system at the same time. Knowing the voltage of all of the battery's helps identify earlier if one is going bad, and would have saved a lot of issues on last year's race.



**Titles on image are opposite. Bottom is solar car board, top is chase.

2 Codes / getting data to pi-top and presenting visually

The whole system is composed of 3 codes, an arduino code ran on both boards, a python code which takes the data from the chase card board, and transfers it to the third code which the second one calls upon to display the dashboard.

NOTE: This is all very well spelled out at this site. And these are great start up instructions. Follow those instructions first.

2.1 Software for the two boards (pi-topHELIOS.ino)

The two boards run of arduino IDE, and each will have the same code downloaded on them. Before downloading each code to the boards, the code "pi-topHELIOS.ino" must be edited to the correct team number and the correct board number.

The board on the car collects data from adc connections (aux, main, and current voltages). In addition, it collects data from the GPS module and sends these data points out via its LoRa Antenna.

The chase card board receives the the data over its radio antena, and is plugged into the rasberry pi of the pi-top. It transmits its data over using the same code into the computer.

2.2 Python Software (main.py)

This is the program that is called upon to run on the pi-top, and it calls upon the next visual program [2.3], making it the master program. It takes in the data bytes off of the arduino board, and converts them into usable data. In addition, this program creates the excel sheet "History" which logs data at certain intervals. This needs to be edited to make sure the correct serial port is selected, the correct rows and columns are defined and named, as well as any adjustments to the pure voltage coming in on the car board (for example the equation we got to convert the voltage from the hall effect sensor into current). The history excel should make a new page each day (although we had intermittent results with this), and will not update live, it must be closed and reopened to present new data. The transformation of data to the excel sheet is an area which we had some trouble with overwriting different days or dropping data after some time, and could likely be cleaned up for this race.

2.3 QML Dashboard code (main2.qml)

This code is called upon to display the visual dashboard which gives live feedback to the operators in the chase vehicle. It is made up of a few gauges which are fairly easily editable, and images which are likewise easy to swap out. These changes are done by editing the code to call upon different images instead of what is originally done, the pi-top used in last years race will have the already edited version of this with an allsaints logo in the background. This code is called upon from main.py, so when that is ran, the dashboard will come up. The gauges can be calibrated by equations found within the code. There are instructions on ow to do this on the

github page, and a excel file on that page which helps you calculate what those equations are. On the code we used for the last race, I believe the equation for the current was a bit off, so the gauge did not always display what the number was.

2.4 Other Notes

Again, most of this information and more can be found on the github page. I recommend reading all of the start up guide thoroughly, and looking up any language you do not understand. I would try to start with the clean program downloaded directly off that page, and then look at the codes on the pi-top which have been edited. Although little has been edited on the codes we ran off at last year's race, I think there were some issues we ran into and had since been resolved within the codes.