**Address:**

To: Dr. Christopher Peters and Mr. David Cinciruk

From: Philip Mak and Sancheet Hoque

Date: 5/31/2019

Re: Final Lab

**Introduction**

The purpose of the final lab was to have a GUI/Interface control the entire electric car system and display values such as RPM, speed, battery life, distance traveled, temperature, and coolant level. This data can be displayed using an speedometer, gauges, and LED lights as well. Besides these requirements, the designer is free to implement any other data into their graphical display as well. Such as cruise control, time run, and other data values. This display could have been created using Python’s Tkinter or Matlab App Designer. For this memo, the students used Matlab App Designer as the rest of the previous implementation has been done in Matlab thus far.

**Method/Analysis**

Code in both Arduino and Matlab was implemented to ensure success in the graphical control display. The Data Acquisition board contains all of the values needed, so only this board was sending data to Matlab. This data was sent by using serial print in a certain order and delay, so the designers could parse it better in Matlab. In Matlab, the students first implemented a base structure of all gauges, speedometers, text boxes, LEDs, and buttons using App Designer’s. This was a simple drag and drop into an interface screen. In App designers there are two views: design view and code view.

The code was implemented inside of the Start/Stop button call back. This is a button with two states which made the handling a lot smoother. Upon the press of a button, all of the hardware is opened: oscilloscope, power supply, and Arduino. Tic is also started to measure time while the system is active. The code flows into a while loop that stays triggered while the button is pressed. SCPI code was used to set the power supply to 12 volts into the system at the beginning of for loop. The code also parses the Arduino data into a data structure and places them into the right area for the display.

Speed and distance traveled was calculated through RPM conversion and tic toc given a measured diameter of the wheel of about 2.625 inches.

Battery life % was calculated by using a formula based on voltage which was created by utilizing the data from lab 3. The oscilloscope was used to measure in the voltage and used in the formula. Our lab 3 data was based off a battery that started at around 8.4V, so since the power supply is outputting 12V, it will show a value over 100% until it goes below 8.4V

Time run was simply the toc value that was started at the beginning of the loop. So num2str was used heavily to convert when passing the variable into display.

Temperature and coolant level of the system was obtained through the Arduino. The LEDs attached to them lit up based on an if statement which triggered the lights depending on the values.

All the gauges attached onto the display (speed, temperature, coolant level, and battery life) took the same value as the text value counterpart except it stayed as a numerical value rather than using num2str to for the text.

Cruise control is also a constant number read in from the Arduino. If it is turned off, it will display ‘Turned OFF’. If it’s turned on a value will be present in there. After the while loop, the power supply is set to zero volts and all hardware is closed.

Figure I: Gui Display

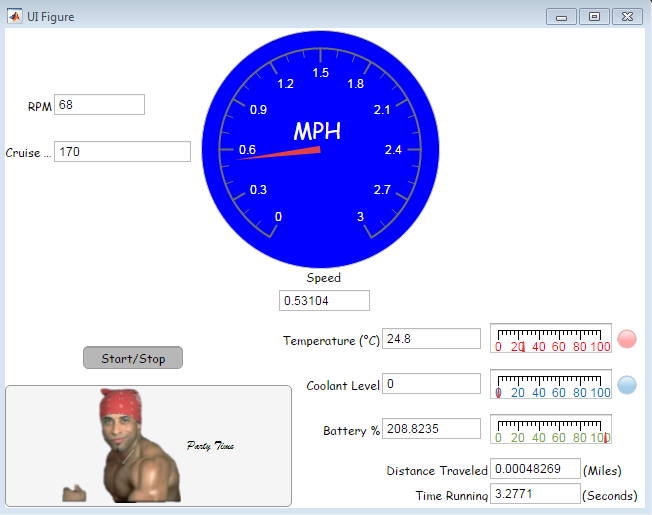
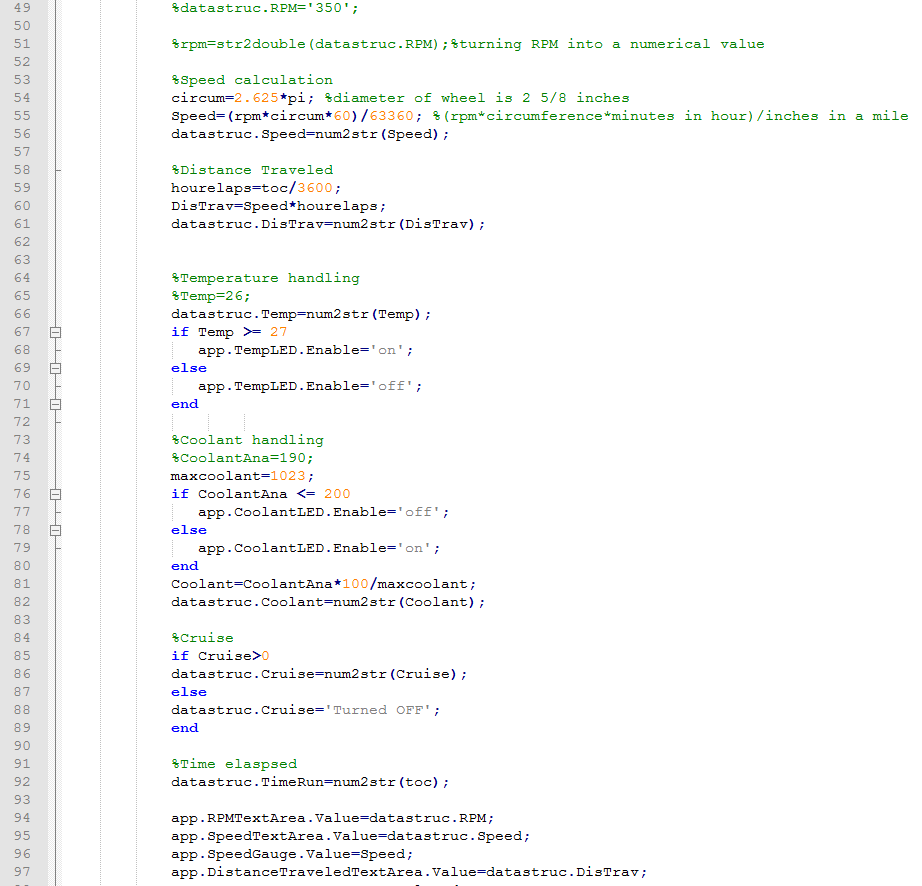
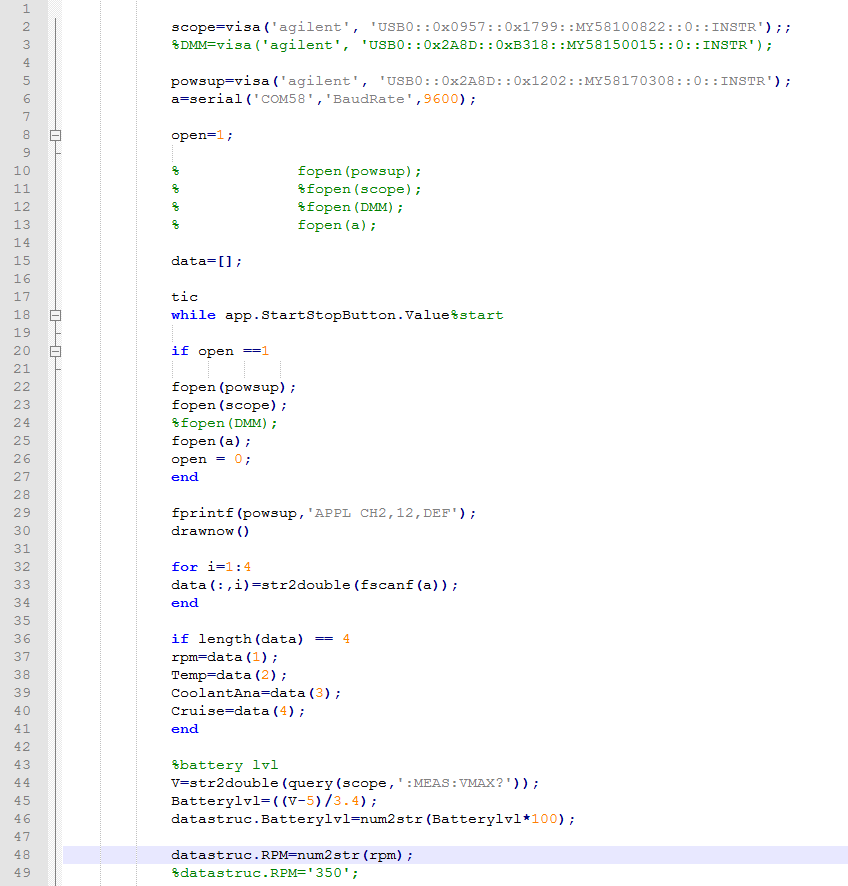


Figure II: Gui



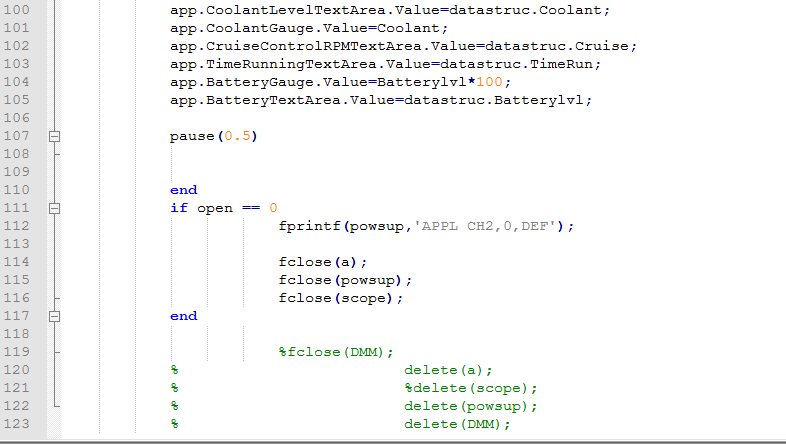


Figure III. Gui Code

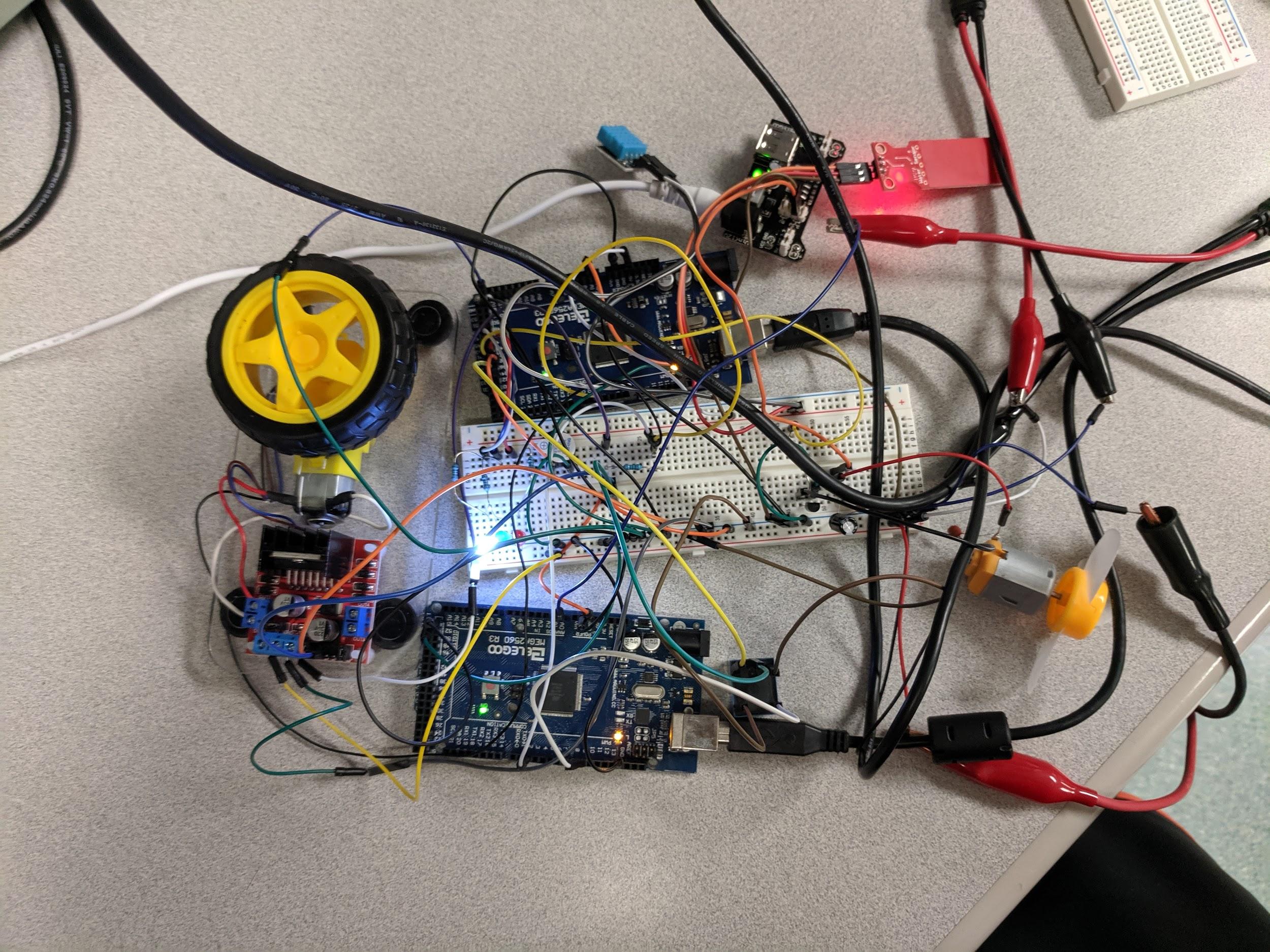
****

Figure IV. Image of final setup

**Discussion and Conclusion**

App designer has a pretty interesting coding methodology, as students were forced to put most the programming in the callback of the start and stop button. There were a lot of issues regarding opening all the lab equipment communications to keep the GUI running after multiple starts and stops. A lot of innovations in the Gui design were cancelled or adapted due to the restrictions in place by the serial communication channels and the App Designer methodology.