Technical Memorandum

To:	Mr. Zachary Minter
From:	Mr. Richard Scott
Subject:	ECE 303 Vehicle testbed
Date:	March 14, 2021

Description:

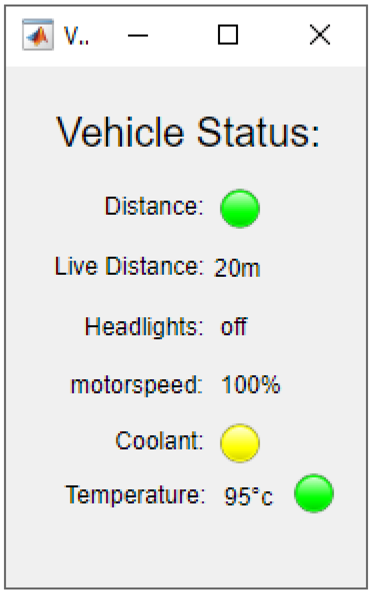
This was a demonstration of the many different sensors and components that Arduino commonly interfaces with. These sensors also comprise a lot of the fundamentals of level one self-driving car system. Including fluid level systems, ultrasonic collision avoidance systems, and temperature sensing systems.

Discussion:

Background
The project had many requirements that a modern-day vehicle would have for its engine bay. Don't let unauthorized people start the car, monitor temperature and fluid levels and disable the vehicle in the event of a catastrophic fault that could cause damage to the vehicle, stop in the event the vehicle is about to hit something. Meanwhile, display live data to the user.

Arduino
The Arduino was used as the central computer, almost ECU (engine control unit), that monitored and controlled the live systems. It reads in data from the water level sensor, temperature sensor, infrared (IR) sensor, and ultrasonic sensor, then automatically controls motor speed through an H-bridge. Meanwhile, it is sending statuses through usb and outputting coolant, temp, and speed to the LCD screen. A passcode, (1234) must be entered on the remote to allow the motor to start.

MATLAB GUI
The MATAB GUI takes in the status codes from the Arduino (list of which is in the text file under the code appendix) and statuses the display accordingly. It is just a verbose HUD (heads up display) that gives live sensor status and data.



V..

Vehicle Status:

Distance: 🟢

Live Distance: 20m

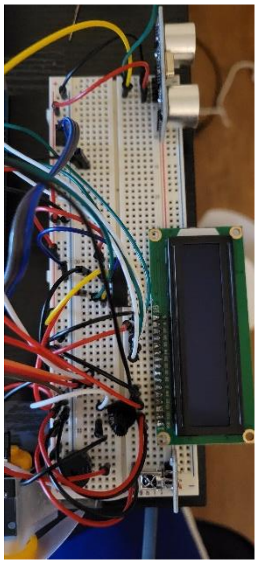
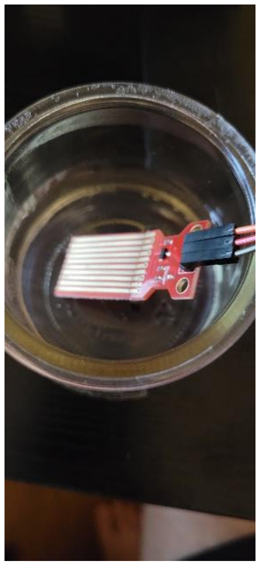
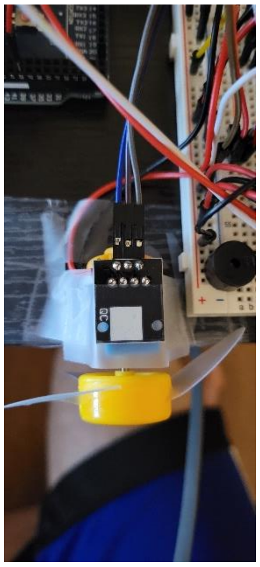
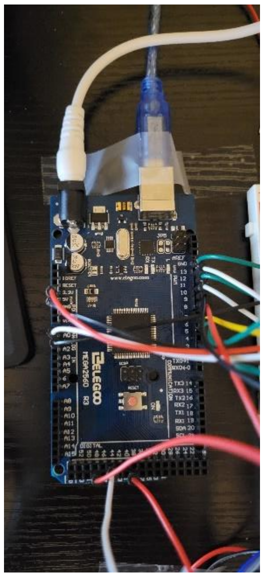
Headlights: off

motorspeed: 100%

Coolant: 🟡

Temperature: 95°c 🟢

MATLAB code
The code that controls the GUI is in main.m and it simply starts a serial connection, then reads and interprets incoming status data. The status codes are in AxWXYZ format where A is the status type (a number) and WXYZ is the status data (numbers) depending on the status type, using 'x' as the separator.

In the above pictures, note LCD, ultrasonic sensor, IR Sensor, Buzzer, H-Bridge, water-level sensor in water, temperature sensor on top of motor and fan, and the Arduino mega.

Many issues and shortcomings are involved with using Arduino with this many sensors and functions. For one, next time and if I were in the lab, I would use a separate power supply for the H-bridge. When the motor turns on, the LCD screen cannot display a second line for the coolant, and the IR sensor cannot draw enough current to work while the motor is running. Even with the Arduino connected to a wall power source, it cannot supply enough current, especially on one tiny 5v pin. Also, when the tone() command was called, the IR receiver (which was on the same timer) stopped working because the tone() command changes timer2's interrupt frequency. Thus, the IR receiver's code had to be edited to use timer 5. The single largest issue with using an Arduino to run all these sensors at once is processing power. For this type of project to be adequately deployed, even on a rover, there ideally should be a system that can handle multiple processing threads to check the sensors simultaneously and work inside of interrupts. If one problem arises on the ultrasonic sensor while the processor is outputting data or checking something else, the processor may not be able to react in time. Like a kid jumping in front to a car suddenly while the processor is waiting for the temperature sensor to respond. Now there where few delays and a lot of work arounds used in this code to avoid any unnecessary delays, so the main loop could cycle through everything in a matter of milliseconds. However, ideally, a better microprocessor and interrupts on separate threads would be used. Maybe even have preprocessors for certain sensors or jobs.

Conclusion:

There must be thought into reducing current draw and making clocks discrete. This way the various sensors and components will not interfere with one another causing significant sensor resolution loss, such as unreliable water levels and IR not even being seen. Still though, many of the primary functions, including stopping and slowing based on obstacle collision avoidance worked, the interlocks do work, and the computer based serial statuses where relayed appropriately.