Dashboard Display

Andrew Smith
IFE 2018

Introduction

The state of charge, speed, and temperatures are all very important metrics for the driver to be able check while driving. With previous iterations of the Illini Formula Electric formula car there was no way to display these things to the driver. This poses a problem in competition where a driver might not be able to exploit the full potential of the car due to estimating the state of charge and thermals. In order to increase the potential of the car and driver a display system was a necessity.

Design

The Low Voltage Team has experimented with displays in the past. We found was that standard off the shelf solutions like arduino controlled displays were not powerful enough so we determined that a custom solution would be necessary to achieve the performance we wanted. The custom solution also gives us the advantage of creating a system that is easy to integrate onto the car. We can use our own connectors, and design the hardware to be easily mounted in a box. There are three main design fronts: electrical hardware, software, and mechanical.

Hardware Overview

The display board is a fairly complex board. It has many different modules and its main purpose is to drive the display and support CAN networking with the rest of the vehicle. We decided to support a 5 inch backlit display. In order to provide readability in direct sunlight and at night a backlit display is a must. There are three main systems that must be supported, the display itself, the Field Programmable Gate Array (FPGA) to drive the pixels, and the microcontroller to control the FPGA (Figure 1).

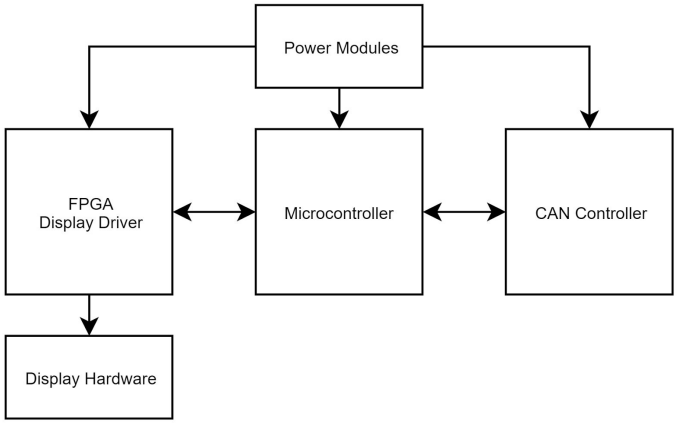

Figure 1: Hardware Modules Block Diagram

Power Supply

There are three different supply voltage levels required by the display system: 5V, 3.3V, and 1.2V. The 5V is required for the CAN chipset, microcontroller, and the boost converter for the display's 24V backlight. The 3.3V is required for the output logic of the FPGA chip as well as powerign the display's internal circuits. The 1.2V is strictly for the internal logic and memory inside the FPGA. To generate all of these power supply levels there were two options: linear voltage regulators or switching supplies. Linear regulators are inefficient at high current loads but are very simple and require very little effort to integrate. Switching supplies on the other hand are very efficient at high loads but they require an advanced background in power to integrate properly and require many components. I decided to use the linear regulators because their simplicity would mean less time debugging and more time to spend on writing the software.

In order to filter the power line noise a simple low pass filter is included on the input to the power supplies. This should attenuate any noise from the high voltage induction motor. The electromagnetic interference (EMI) generated from three phase induction motors are in the 100+KHz range[1]. The cutoff frequency of a low pass filter (Figure 2) is given by equation 1 where the L is the value of the inductor in Henrys and the C is the capacitance in Farads.

$$f_c = \frac{1}{\sqrt{LC}}$$

Equation 1: Cutoff frequency of Low Pass Filter

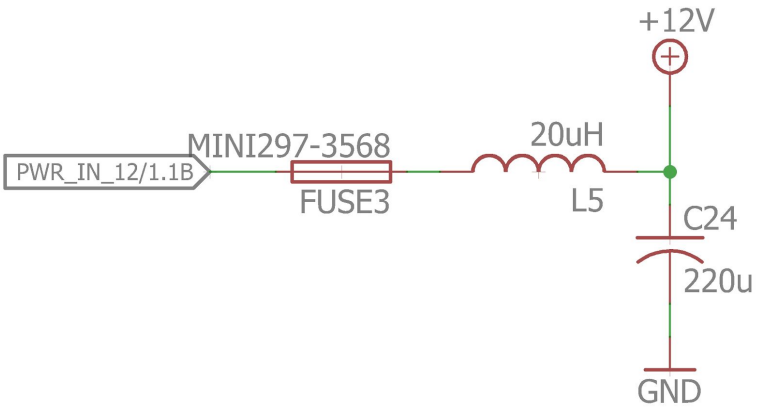

Figure 2: Low Pass Power Filter

Choosing a 20uH inductor and a 220uF inductor will attenuate noise above 15KHz which will filter out the EMI from the motor.

I found three linear voltage regulators that are small surface mount and can handle up to 500mA of current which is more than enough for any of the systems on the board. Figure 3 is an

example of the 12V to 3.3V DC-DC circuitry and the other linear regulators are all set up in similar fashion.

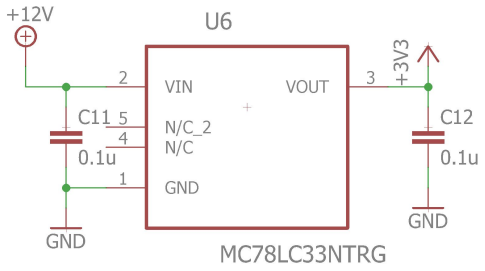

Figure 3: 12V to 3.3V DC-DC regulators

Controller Area Network Interface

In order for the display to receive the car's information it must have a Controller Area Network (CAN) interface. The CAN network is a two wire communication bus that is a very reliable and robust system. I chose to use the MCP 2515 CAN controller and MCP 2551 as the transceiver because they are very well documented chips and have a simple SPI interface to communicate with the microcontroller (Figure 5). The transceiver is what generates and interprets the differential signal on the CAN bus and converts it to a signal that the CAN controller can understand (Figure 4).

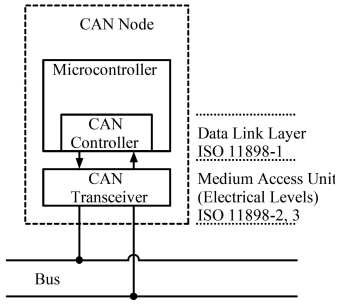

Figure 4: CAN Interface Layers[2]

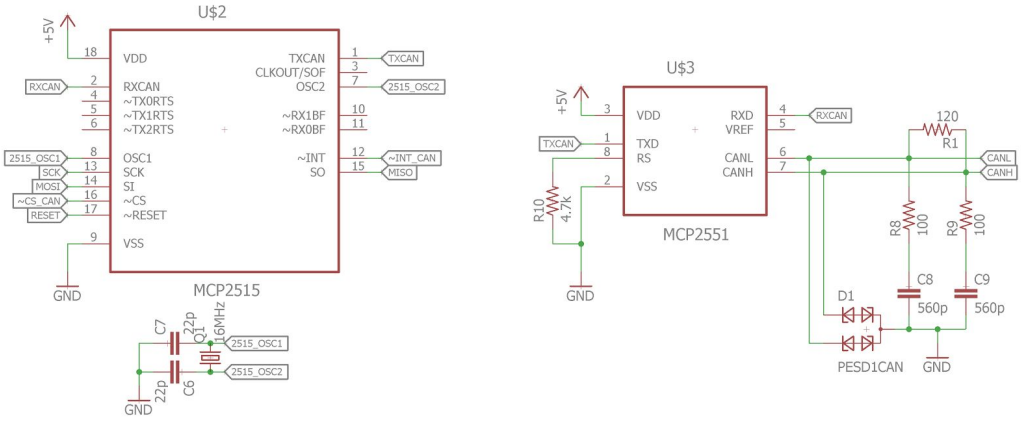

Figure 5: CAN interface with MCP2515 and MCP2551 chips

Field Programmable Gate Array

One problem that the display poses is that it needs powerful hardware to drive even a low resolution screen like the one we are using. It has large memory requirements, a high frequency pixel clock, and requires many I/O for each of the red green and blue channels. Previously the basic microcontrollers we had attempted to use in the past fell short on the pixel clock, ram, and I/O ports. Without greatly increasing cost and the complexity of incorporating an unfamiliar more capable microcontroller I decided that a FPGA would be a viable option for driving the display. With the FPGA display specific hardware could easily be written and with the Altera Quartus design suite debugging is simple with modelsim. The microcontroller will be able to write into registers on the FPGA to control which images are displayed as well as the control all of the different level bars on the display. I chose to go with a Cyclone II FPGA from Altera because they are an older generation and are low cost for their abilities. Unfortunately FPGAs are by nature volatile devices thus they must have an external flash that they read on power up to restore their configuration. The Cyclone II and configuration flash setup are shown in figure 6.

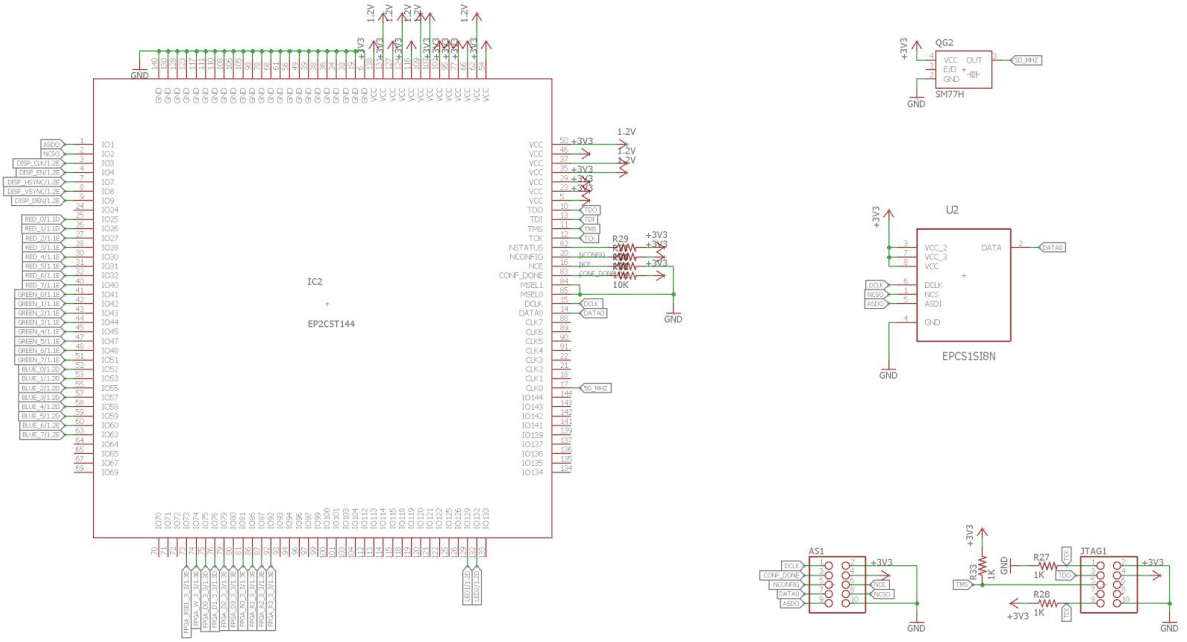

Figure 6: Cyclone II and serial configuration flash

Microcontroller

In order to control all of the high level features of the board I decided to use a basic microcontroller. The microcontroller is responsible for reading and parsing the data from the CAN bus. It then formats the data to what the FPGA is expecting and then writes to one of the FPGAs registers. I chose to use a Atmel ATmega328p because of their low cost and thorough documentation and support. The FPGA and the microcontroller run at different voltage levels so

in order to properly interface the two devices a level shifter serves as a buffer between the two devices (Figure 7).

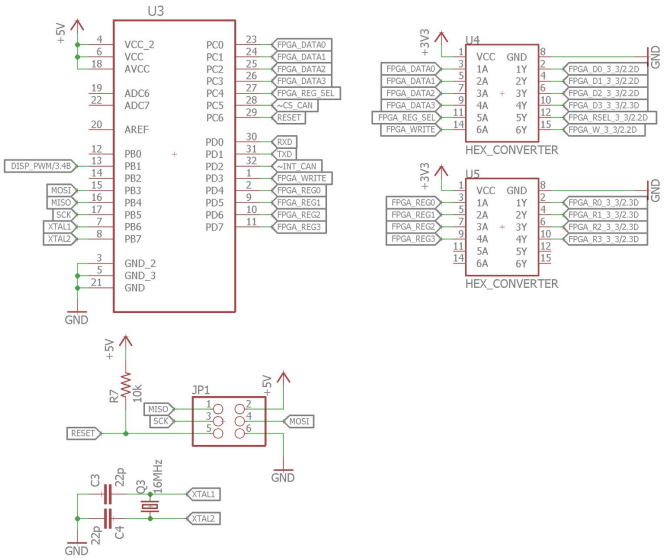

Figure 7: Microcontroller and Level Shifters

Display Backlight Driver

The display backlight has a forward voltage of ~24V and requires a current of 50mA. In order to achieve this a boost converter is needed. I decided to use a common boost converter design for LED backlights. A boost converter works by storing energy in the magnetic field of an inductor while the switch is closed. When the switch is opened the inductor resists the change in current and the load sees a source voltage of the voltage across the inductor given by equation 2 and what the original DC source voltage is. When the switch is open the load capacitor is charged which maintains the voltage across the load while the switch is closed.

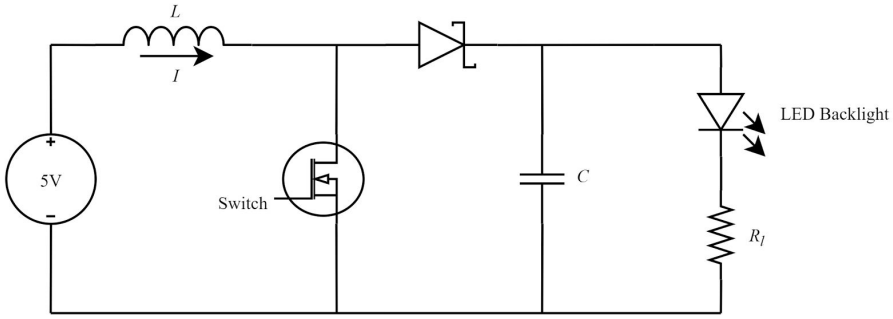

Figure 8: Basic boost converter

$$V = L\frac{dI}{dt}$$

Equation 2: Voltage across an inductor

I decided to replace the switching MOSFET with a general purpose LED driver chip. The chip has closed loop control of the load current to maintain a constant 50mA through the LED backlight. Figure 9 is the complete LED driver circuitry for the display system.

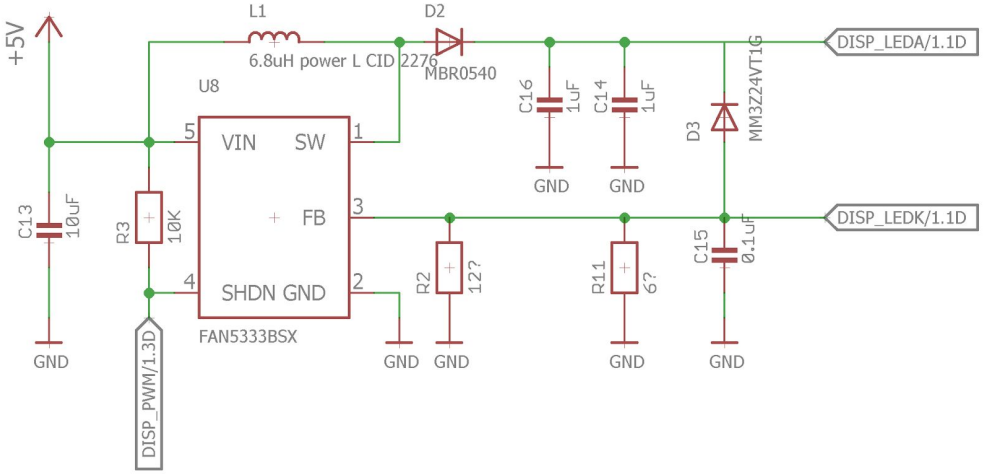

Figure 9: LED Backlight Driver

Display

The display itself is a 5.0 inch TFT display with a LED backlight. These displays are very common on appliances and often used in hobby applications. It can be configured to use a serial bitstream or a parallel interface for pixel information. I decided to support the 8 bit RGB parallel interface because it is clocked much slower than the serial bitstream so it would be easier to meet timing requirements. The display has 8 bit color support and a resolution of 480x272.

Figure 10: 5.0 Inch TFT Display

Printed Circuit Board

The printed circuit board (PCB) is a two layer board and is 3.3x4.25 inches. I chose to use easy to solder surface mount components to make routing easier and make layout simpler. All of the different subsystems are highlighted on the PCB (Figure 11). The power filter and the voltage regulators are in purple. The boost converter for the LED is in light blue. The CAN chips

are in green and the microcontroller is in white. The FPGA is highlighted in orange and the configuration headers for the FPGA are highlighted in yellow.

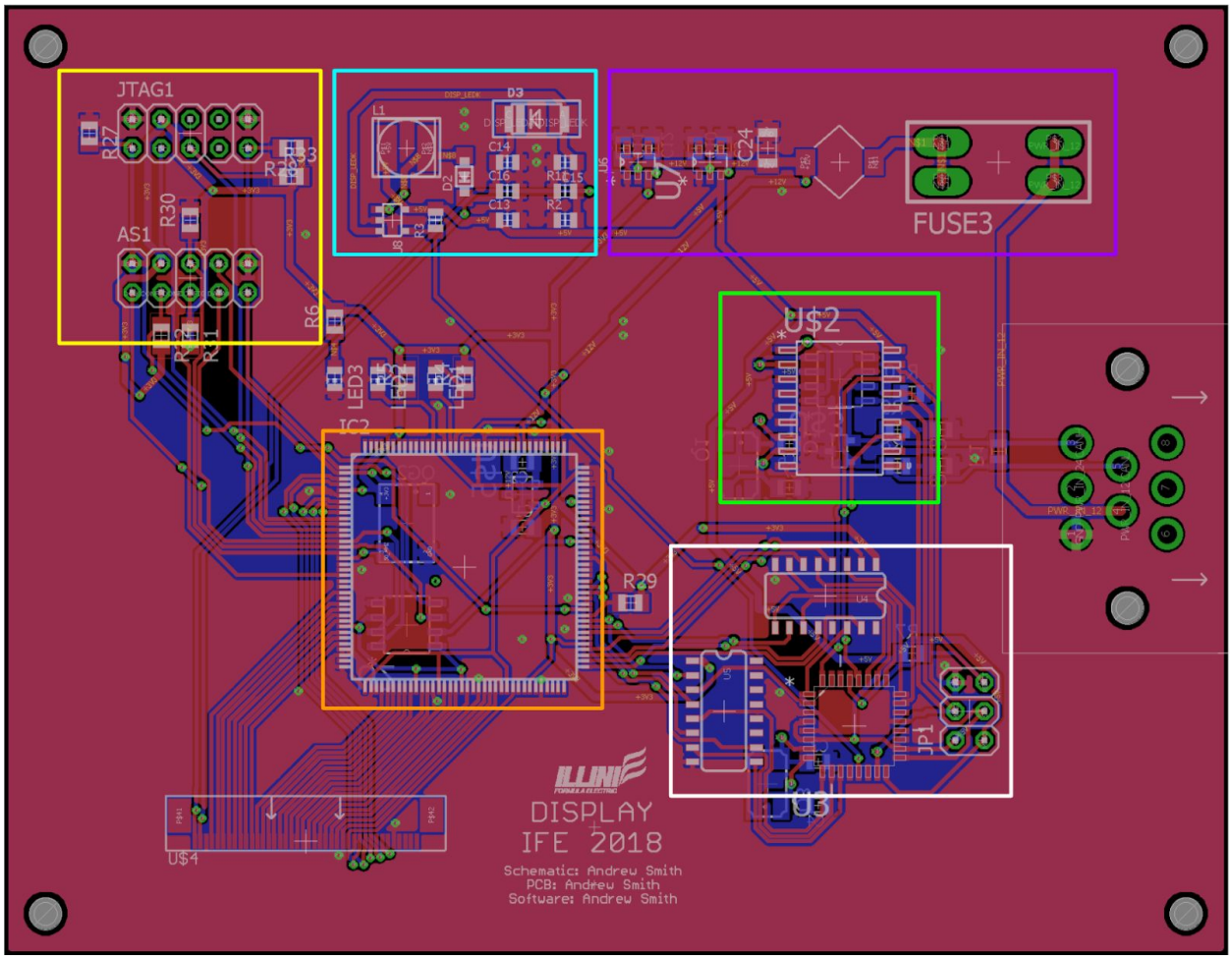

Figure 11: Display PCB

Software

Currently I am in the software development stage. I have some basic frameworks set up for both programmable components. I have written a program to convert the concept art to the format that the FPGA requires. I have tested this out by displaying the IFE logo on the display.

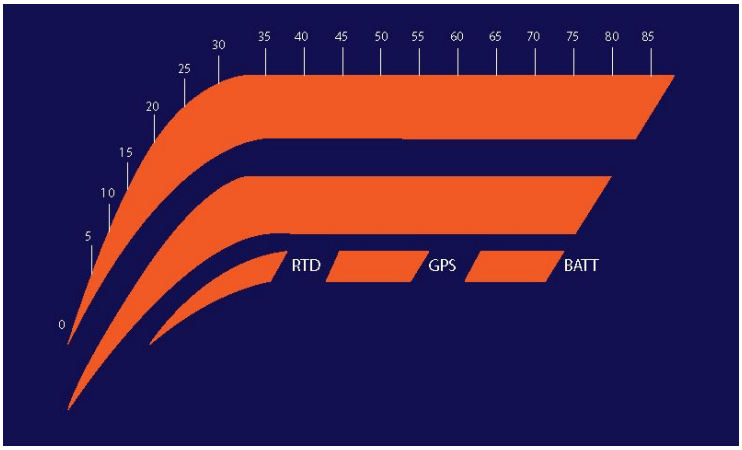

Figure 12: Display graphic concept

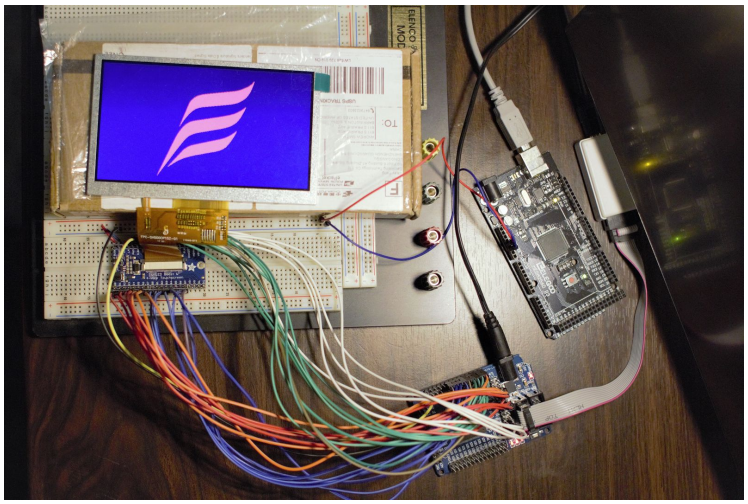

Figure 13: Prototyping with the display and FPGA

Mechanical

I have sent my board designs to the Electromechanical Integration team to design a box that will house the pcb board and display itself.

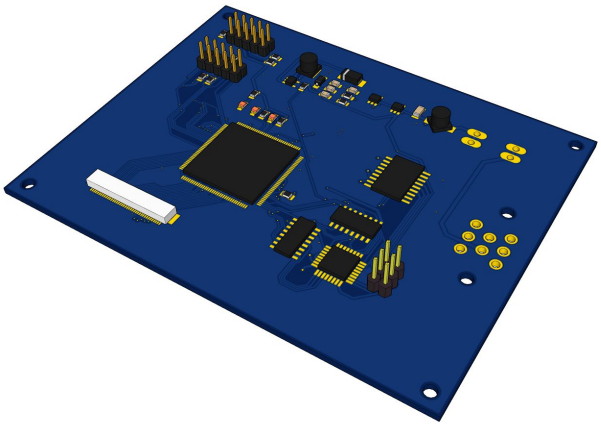

Figure 14: 3-D Render of the board

Testing

In order to ensure that both the software and hardware are working, I have devised two testing schemes. I intend to test the different hardware units individually by analyzing voltage levels with a multimeter as well as running by basic display framework code on the FPGA to ensure the display interface is correct. I also intend to create a dummy wiring harness with a CAN enabled arduino connected to send the CAN messages as if it were connected to the vehicle to make sure that the software is working properly (Figure 16). There are also a series of debug LEDs on board to make debugging the embedded software easier.

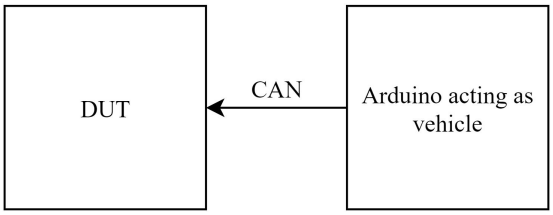

Figure 16: Test configuration

Bibliography

1. Pulomena, Joseph. “Mitigate EMI/EMC in Today's Electric and Hybrid Vehicles.” *Mitigate EMI/EMC in Today's Electric and Hybrid Vehicles*, EPCOS Inc., a TDK Group Company, 29 Sept. 2016, electronics360.globalspec.com/article/7405/mitigate-emi-emc-in-today-s-electric-and-hybrid-vehicles.

2. By EE JRW - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=35496960