Digital guage cluster for cars with CANBus Communication

Johannes Melkersson

Automotive Engineering, bachelor's level
2023

Luleå University of Technology
Department of Engineering Sciences and Mathematics
ABSTRACT

Digital guage cluster for cars with CAN-Bus Communication
Johannes H. Melkersson

This thesis presents the development of a digital gauge cluster for automotive and motor-sport applications, taking advantage of the cost effective Arduino micro-controller. CAN-Bus, Computer aided design, 3D-printing and other skills were utilized to create a customizable and adaptable solution.

The study begins by examining the existing solutions of automotive instrumentation. Highlighting the shortcomings of available solutions and the need for innovative alternatives. I describe the integration of the Arduino as the CPU (Central Processing Unit) as well as the other components used. The design process of creating a custom enclosure for the display is also explored, using CAD (Computer Aided Design) and explaining the manufacturing process for the enclosure using 3D-Printing.

My research yields significant findings, showcasing user-friendly interface for monitoring critical engine data. Using the CAN-Bus system ensures a robust and reliable communication link between the Arduino and the car. Using the same connector as the original gauge cluster ensures an easy installation and a plug and play experience.

This work contributes to a broad field in the automotive world, showing the potential for low cost, open-source solutions to complex problems. It inspires people to explore these open-source solutions instead of the limitations of current
proprietary automotive systems.

In conclusion, the development of the digital gauge cluster constitutes a combination of skills attained from the automotive systems program at LTU, combining knowledge of CAD, programming and electrical engineering to create a functional digital gauge cluster. This abstract summarizes the thesis and I invite further exploration and development in the field of automotive instrumentation.
Contents

Abstract 2

1 Introduction 7
   1.1 Purpose ................................................. 7
   1.2 Organization of the thesis .......................... 7
   1.3 Relevancy to ........................................... 8
      1.3.1 The student ...................................... 8
      1.3.2 The industry ................................... 9
   1.4 Available kits ....................................... 9
   1.5 The car ............................................... 9

2 Hardware 10
   2.1 Introduction ........................................... 10
   2.2 The CPU ............................................... 10
   2.3 The display .......................................... 10
   2.4 The power supply .................................... 11
   2.5 Connector cables and PCB’s ......................... 11
   2.6 The CAN-Bus Receiver ............................... 12
   2.7 Sniffer ................................................ 12

3 Software 13
   3.1 Introduction .......................................... 13
   3.2 C++ .................................................. 13
   3.3 Arduino IDE ........................................... 13
   3.4 Nextion Editor ....................................... 14
   3.5 Communication ....................................... 14
      3.5.1 SPI ............................................ 14
      3.5.2 UART .......................................... 15
      3.5.3 CAN-Bus ....................................... 15
      3.5.4 K-Bus .......................................... 16
      3.5.5 A/D Conversion ................................. 16

4 Development setup 17
   4.1 Introduction .......................................... 17
Chapter 1

Introduction

1.1 Purpose

The purpose of this project is to enhance the way that information is displayed to the driver. This is achieved by modernizing to a completely digital system using an LCD-Display instead of a partially digital/analog or fully analog system with dials and indicators.

A fully digital system is more customizable, the driver is able to choose which information, and how it is presented on the display. A fully digital system is able to display more values than an analog gauge-cluster. A Digital system is also able to communicate with other systems in the car more efficiently. By integrating all the signals onto one display, the need for analog dials is reduced. This creates a cleaner, more efficient and more streamline look.

The display is easy to adapt to different cars by changing the design of the enclosure. The control unit for the display also has the ability to be adapted into different cars with minor modification to the software, which means lower variation of production models and therefore lowering the cost of production.

1.2 Organization of the thesis

This thesis is organized in chapters to make it easier to read. The following chapters are:

- Introduction, the current section.
• Hardware, which describes the components and the implementation of the said components in the project.

• Software, which describes the software and programming.

• Development setup, which explains how the components communicate with each other.

• Enclosure design, which explains how the enclosure was design and manufactured.

• Analysis, which describes about the development process and time-frame as well as limitations.

• Further development, which describe what further improvements are to be made to the digital guage cluster.

• Conclusion.

• Appendix, Arduino code.

1.3 Relevancy to

1.3.1 The student

This project allowed me to use the information that I have acquired through the Automotive Systems program. Throughout the courses and personal experience I have learned about CAD(Computer Assisted Design), programming, vehicle dynamics, math and data processing in many different formats. This project served as an opportunity to combine the skills and knowledge that I have gained during my time at LTU. My education shows to be very useful in the development of the digital gauge cluster as well as the design of the enclosure and connectors to adapt it to the specific car in question.
1.3.2 The industry

The trend in automotive and motor-sport industry is showing a transition toward digital interfaces, most prominent in electrical vehicles but also in cars with internal combustion engines. These trends show not only in gauge clusters but also climate control panels and multimedia controls. The purpose for this transition is to allow for a less complex system and a more user-friendly experience. Traditional analog gauge clusters need a separate gauge for every value, such as RPM, Engine temperature or oil pressure. With a digital gauge cluster you have the freedom to change what values you want to display and how you want to display them.

1.4 Available kits

There are multiple companies that make aftermarket digital gauge clusters, such as ID4Motion [1], AEM[2], ECUMaster[3] and many more. These products are well refined and have all the necessary features but the biggest drawback of these is the cost. The lowest cost for a digital dashboard from the previous mentioned companies is 1000 Euro.

1.5 The car

The car used to develop the gauge cluster is a -1999 BMW E46 320i. The car has an aftermarket Arduino-based ECU(Engine Control Unit) to control the turbocharged engine. The data from the engine ECU is transmitted through the CAN-Bus network, this information is then available to be processed anywhere in the CAN-Bus network. The digital gauge cluster does not need an Arduino-based ECU to function, the original ECU of the E46 320i is still able to send signals through the CAN-Bus to the digital cluster.
Chapter 2

Hardware

2.1 Introduction

This chapter explains the devices and hardware that are used in this project.

2.2 The CPU

The CPU (Central Processing Unit) used in this project is an Arduino Mega 2560, which offers 54 digital inputs/outputs, 16 analog inputs, and 4 UARTs (Hardware Serial Ports). I wanted to make sure that the CPU’s capacity for inputs and outputs would not become a limitation. It is worth noting that other options like the Arduino Uno or Arduino Leonardo could also suffice for this purpose.

There are several alternatives to Arduino, including the STM32, Raspberry Pi, and Teensy. While the STM32 features a built-in CAN-Bus module on the chip, my limited experience with STM32 led me to choose the Arduino. Given that Arduino uses C++ syntax, which I became familiar with during high school, it was the more comfortable choice. While all the previous mentioned options have their pros and cons, my decision to go with Arduino was influenced by the experience I have gained throughout the courses at LTU as well as personal experience.

2.3 The display

In this project, the selected display is a Nextion 7” NX8048K070 touch display, recognized for its interactive Human Machine Interface (HMI) capabilities. Typically it is used to oversee operations such as conveyor belts, production machinery,
and various other devices. The Arduino’s inability to process images is the reason for choosing this display. The Nextion display has built in graphical processing capabilities. Communication between the Arduino and the display is transmitted via a UART port, where the Arduino transmits data and the screen renders and presents it visually.

2.4 The power supply

In order to provide power to the devices, a conversion process is necessary to transform the incoming 12-15 volt range from the car’s power source to a 5-volt supply. A buck converter is used to reduce or amplify a voltage. The Arduino’s accepted voltage range is 7-12 volts. This input voltage is then reduced to 5 volts through the integrated voltage divider on the Arduino. It is worth noting that when input voltage exceeds the recommended range, the voltage divider generates heat. To mitigate potential damage to the CPU’s voltage divider and other components, I opted for the lower end of the voltage spectrum.

For the voltage conversion process, a voltage divider, also referred to as a buck converter was used. This component takes 12 volts and ground as inputs on one end and delivers 5 volts along with a new ground on the other end. The voltage generated from the car varies from approximately 12-15 volts. The configuration with the buck converter ensures a safe and stable power supply for the system.

2.5 Connector cables and PCB’s

The connector to the cluster was acquired by disassembling an original analog gauge cluster, this allowed me to seamlessly solder it onto a PCB. The connector is a SIEMENS B-1126-B510 Connector, a component commonly found in both the BMW 3-series E46 and the BMW 5-series E39. The selected PCB is a standard soldering board using a 2.54mm hole spacing. Single stranded cables are used for optimal soldering to the PCB and connecting to the Arduino’s female ports.
This approach simplified the wiring and reduced the amount of cables needed to complete the same tasks. Soldering ensures efficient and effective connections.

2.6 The CAN-Bus Receiver

The MCP2515 is used to receive the signals from the CAN-BUS network and transmit them through the SPI protocol to the Arduino.

2.7 Sniffer

A CAN-Bus sniffer is a component that connects to the CAN-H and CAN-L to receive signal from the CAN-Bus network and then send this information to a computer via USB. The sniffer used to interpret the CAN-Bus signals is a USB-CAN V8.00 CAN-Bus Analyzer.
Chapter 3

Software

3.1 Introduction

In this chapter, I explain the software used in this project, not just the programming aspects but also the programs, communication protocols, and underlying principles that have been utilized.

3.2 C++

C++ is a high level programming language that is evolved from the programming language called C. C++ is an object-oriented programming language which means that software design is easy to structure which allows code to be reused or adapted to different application therefore lowering development time. C++ is widely used today and is one of the world’s most used programming languages. C++ is used in creating Operating systems, games and web browsers as well as programming ECU’s like the Arduino used in this project. [4]

3.3 Arduino IDE

IDE stands for Integrated Development Environment. This is the program that is used to Write code, compile code and update code to the Arduino board. The Arduino IDE also includes a serial monitor which allows the user to monitor the data that is sent through a selected serial port. This is very useful when troubleshooting errors in the code. [5]
3.4 Nextion Editor

The Nextion Editor is the graphical interface for the display. This program allows you to change the layout of different items such as progress bars, gauges or buttons. Each item has an identifier that allows you to name the items such as RPM, ECT or Speed. This makes the code on the Arduino easier to structure since all the data that needs to be send have logical destination names.

3.5 Communication

3.5.1 SPI

SPI (Serial Peripheral Interface) is a standard in synchronous serial communication. SPI has a master and a slave unit, in the case of this project, the slave is the CAN-Bus transciever and the master is the Arduino. Four cables are used for SPI communication:

- SCLK (Serial Clock)
- MOSI (Master Out Slave In)
- MISO (Master In Slave Out)
- CS/SS (Chip Select/Slave Select)

SCLK (Serial Clock): The master/Arduino generates clock pulses on this line to synchronize data transmission between the master and the slave. The clock is used to time the data transmission between the master and the slave.

MOSI (Master Out Slave In): The line that the master sends data to the slave

MISO (Master In Slave Out): The line that the slave sends data to the master
CS/SS (Chip Select/Slave Select): Each slave has its own chip select line, only one slave can communicate with the master at a time. A slave is chosen when the chip select line for the selected slave is set to low. [6]

### 3.5.2 UART

UART (Universal Asynchronous Receiver-Transmitter) is one of the most common communication protocol between devices. In this project UART is used to send data from the Arduino to the Nextion display. UART only uses two lines for communication RX and TX. UART allows both sending and receiving data. The UART protocol does not use a clock signal to synchronize data transmission, unlike SPI, UART transmits data asynchronously. [7]

### 3.5.3 CAN-Bus

CAN-Bus (Controller Area Network) is the standard communication protocol for vehicle applications. CAN-Bus is used in virtually all vehicles and was standardized in November of 1993. CAN-Bus is used for communication between devices in the car, such as the engine, transmission, brakes and gauge cluster. The data from the CAN-Bus is constructed using a binary format but CAN-Bus is usually represented in hexadecimal to make it easier for human interpretation. In this project the CAN-Bus will be used to capture data that is sent from the engine ECU to the gauge cluster.

The CAN-Bus communication is transmitted through a twisted pair of wires called CAN-High and CAN-Low, the twisting of the wires allows the CAN-Bus communication line to be resistant to electro-magnetic interference. CAN-H and CAN-L are mirror signals but inverse of each other, one produces a positive voltage and one produces a negative voltage. If one of the lines is broken there is an internal error check mechanism within the CAN-Bus network. The system will detect the error and react accordingly and communication can in many cases continue without compromising the reliability of the network.
3.5.4 K-Bus

The BMW K-Bus (Karosserie Bus) is a serial communication protocol used in BMW’s from the early 1990’s to the early 2000’s. This serial bus allows devices in the car to communicate with each other such as the light control module, radio and parking distance controller. The K-bus was later replaced with CAN-Bus because of the increased data transmission speed, robustness and reliability as well as standardizing BMW’s vehicles. The K-Bus was not used in this project because of the limited time was not adequate for the advanced circuitry and code necessary to communicate with the K-Bus [10].

3.5.5 A/D Conversion

A/D Conversion is the principle of converting an analog signal into a digital signal. This principle is adapted to measure the resistance of the fuel level sensor. The Arduino does not have the capability of measuring resistance directly. To convert this signal I law, created a voltage divider using a known resistor and the fuel level sensor. I can calculate the unknown resistance of the fuel tank level sensor by measuring a difference in voltage between the known resistor and the unknown resistor. [11]
Chapter 4

Development setup

4.1 Introduction

In this chapter I explain the design layout of the project. Where the devices are located in the communication chain and which communication protocols are between each device. A connection diagram of all the components in the communication chain can be see in figure 4.1.

Figure 4.1: Connection diagram
4.2 Communication

4.2.1 Engine ECU

The ECU (Engine Control Unit), based on an Arduino Mega 2560, is the brain behind efficiently managing the engine’s performance. It collects data, including engine coolant temperature, vehicle speed, and crankshaft position sensor. This, among others, are processed within the ECU, enabling it to execute essential calculations. These calculations ensure optimal fuel injection into the cylinder and correct ignition timing, optimizing the engine’s performance.

For the ECU to read the sensor data, it uses Analog-to-Digital (A/D) conversion. Analog signals from sensors, typically in a range of 0 to 5 volts, are transformed into digital form, necessary for the ECU to comprehend the values from the sensors. Each analog sensor has a range, contingent on the voltage it measures. For instance, a pressure sensor registers 0 volts for zero pressure and 5 volts for a pressure of 10 bar. Inputting the correct sensor range into the ECU settings is a must for obtaining accurate values.

Data from both sensors and the ECU’s calculations are sent onto the E46 CAN-Bus network, which operates at 500kb/s. This transmission is facilitated through a STM32 CAN-Bus module called BluePill.

4.2.2 Arduino

The Arduino gathers vital information from the Engine Control Unit (ECU) via the CAN-Bus network. Simultaneously, it acquires a specific value directly through Analog-to-Digital (A/D) conversion which is the fuel level sensor reading.

The Arduino is the Central Processing Unit (CPU), is able to process both input and output signals. It efficiently processes incoming signals from the CAN-Bus through the MCP2515 CAN-Bus transceiver. The signals from the MCP2515 are initially in hexadecimal format, converting to decimal form is necessary for further calculations.
Moreover, the Arduino serves as a channel for transmitting processed data to an external display unit via UART communication. This process enables the display to show crucial data such as engine revolutions per minute (rpm), coolant temperature, and vehicle speed. In summary, the Arduino has a multifaceted role in extracting, processing, and presenting essential vehicle information in a comprehensible manner.

### 4.2.3 Display

In this project, the display plays a single yet crucial role, presenting the data sent from the Arduino. Only the Rx (Receive) line of the UART interface is utilized. The display does have the ability to send data in the forms of button presses and slider inputs, however in the framework of this project the display is only able to receive data because of the Tx (transmit) line that is not connected.
Chapter 5

Enclosure design

5.1 CAD

The 3D model for the enclosure was created using the SolidWorks CAD (Computer Assisted Design) software. However, it’s worth noting that LTU’s CAD course focuses on teaching NX, there are also other alternative CAD programs available. Due to my familiarity and knowledge of SolidWorks, it was the optimal choice due to the better workflow and enhanced speed I can operate.

5.2 Reverse Engineering

Attaining a model with good fit within its original location is a challenging process that demands extensive trial and error experimentation. In order to get a foundational model, the first step was to capture an image of the original gauge cluster that later got imported into the CAD software. By tracing the outside of the image within a CAD sketch I was able to get a foundational model, I was then able to scale this drawing, thereby generating a baseline for the dimensions.

The resulting model was then 3D-Printed in a 3mm thick prototype, allowing me to visualize an accurate representation of the shape. Differences between the prototype and the original were measured. These variations were then corrected and adjusted within the CAD software, this was an iterative process that continued until optimal fit was achieved.

After achieving the desired fitment, the finalized model was prepared for production. The approach of making prototypes before the final 3D printed version ensured that it aligned flawlessly at the original location.
5.3 3D Printing

Fused Deposition Modeling (FDM) is by far the most popular method for 3D-Printing. This method revolves around the gradual addition of molten plastic, layer by layer, until a complete model is created. The most common plastic that is used is PLA but other plastics such as PETG and ABS are also frequently used. Each plastic has distinct material attributes like hardness, toughness, and varying melting temperatures.

In this particular project, PLA was the chosen plastic due to it being the easiest and cheapest material to 3D-Print. PLA is a brittle plastic but due to the limited stress exerted on the project, the model is not affected by this negative attribute of PLA.

Due to the limited dimensions of the specific 3D-Printer used, the model had to be divided into two separate segments. To create the finished model these two segments were glued together.

5.4 Mounting

Mounting of the display to the model is made with 4 screws, a base-plate is also present for to mount the Arduino and MCP2515 CAN-Bus module as well as the Buck-converter. The model is held in the original location by a tight interference fit.
Chapter 6

Analysis

6.1 Introduction

It is important to analyze and evaluate your work as an engineer. Analyzing and evaluating a project after completion is not just a matter of self-reflection but a chance to develop a better mindset. It enables engineers to learn from their experiences and mistakes, make necessary improvements, enhance their performance, and ensure that future engineering projects are executed more efficiently and effectively.

6.2 Development

6.2.1 Timeframe

The project started in the beginning of May 2023 and the functional version was completed in the end of July meaning it took approximately 3 months from start to finished product.

6.2.2 Research

Starting from scratch means that there has to be extensive research before purchasing components. Research began about a year ago since I was interested in creating a digital gauge cluster outside of this thesis. In the sim-racing world both emulated analog and digital gauge clusters are common to create a more immersive experience for the driver. The gauge cluster does the same thing as in a real car, presenting important to the driver in a comprehensive manner to allow for easier access to information.
6.2.3 Tools and skills

There have been many tools and skills used to complete this project, not only theoretical knowledge but also knowledge and ability of using practical tools. Some of the practical tools used were soldering irons, handheld drilling machines, cable cutters, cable strippers, shrink tubing and many more. There is a lack of understanding about how important practical skills are in the real world. Theoretical education cannot be confused with real knowledge learned by doing. I have achieved this knowledge by myself building and modifying both electric and gasoline powered vehicles including the car used for this project.

6.3 Display Limitations

The display has limitations in the refresh rate as well as the internal memory limits what the display can store. The low refresh rate that causes the display to appear to flicker or appear to lag. The solution for this problem would to upgrade the current Nextion Enhanced display to their top of the line displays called the Intelligent series. The intelligent series has a processor that is twice as fast as the enhanced aswell as more than twice the amount of storage volume. The drawback of the Intelligent display is that the price is 150 Euro while the enhanced version is around 100 Euro.

The display is fitted with a touch display that allows for touch inputs such as buttons or sliders. Due to the limited space in the gauge cluster area, the touch screen had to be removed in order to fit the display. [12]

6.4 UART Limitations

UART has a low data transmission speeds as well as the size of the data frame which is limited to only 9 bits of information. UART has a maximum transmission speed of about 5MB/s, other communication protocols are better for transmitting
more data faster such as SPI at 60MB/s or USB 2.0 at 480 MB/s. If the display was able to receive this much data, it would allow for a much smoother user experience without lag or flickering. [13]
Chapter 7

Further development

7.1 Further development

7.1.1 Introduction

As I acknowledge the significant strides I’ve made in this domain, it is important that I maintain the commitment to improving the system as it’s an alternative method of digital gauge cluster technology, avoiding the proprietary available systems and allowing for more open-source and a more adaptable solution.

7.1.2 PCB

The PCB that is used for this project is a soldering board, PCB stands for (Printed Circuit Board) and is used to reduce and simplify a circuit by making it more compact. A PCB allows for many components on a tiny surface area. Instead of using soldered cables, PCBs have a more reliable connections underneath the insulated part of the board. Creating a PCB requires the knowledge of designing a PCB circuit, there are many sites and companies that offer design and manufacturing of custom PCBs such as PCBway, Digikey and ALLPCB. PCB is a inexpensive and effective way to simplify and improve your project at the same time.

7.1.3 K-Bus

The implementation of K-Bus would be a great improvement to the project but i was not able to achieve it because of the limited time frame for the project. The K-bus would allow me to implement extra functions to the display, such as Turn
signals, high beam indicator, parking sensors as well as the status if a door is open or not. If this were to be implemented it would improve the overall feeling of the project, making it feel more complete.

7.1.4 Faster display

The current display works without any issues but a smoother experience would be preferred in my opinion. Nextion has a more expensive display that offers a higher refresh rate as well as more memory, this would give the display the ability to provide a better experience to the user.

7.1.5 Touch

The Nextion display does have a touch interface but this feature was not able to be used due to the limited space in the gauge cluster area. Implementing a touch interface would allow the driver to make adjustments to the cluster easily, such as chaining the layout or using buttons to control certain features in the display. Using this touch feature would necessitate in removing parts of the frame that the touch was built into or reducing the display size to allow the full Nextion frame to fit into the cluster frame. Alternatively using a different display without a bulky frame.
8.1 Summary

In conclusion, this thesis has summarized the development of a digital gauge cluster for cars with CAN-Bus communication. Using an Arduino, Nextion display among other components as well as CAD Design.

This project shows the adaptability of the Arduino platform for automotive electronics. By leveraging the capabilities of the Arduino I was able to create a system that is adaptable, able to present critical engine data as well as it being a cost-effective alternative to what is available on the market.

Furthermore, this project expands beyond the standard of automotive instrumentation. It shows the potential of open-source hardware, software in the automotive industry. This encourages innovation and experimentation, benefiting both enthusiasts as well as the broader automotive community.

While I have achieved a significant amount of milestones when developing the gauge cluster, there is plenty of room for improvement and expansion. Mainly in implementing the K-bus. Further research could delve deeper in optimizing the circuit using a PCB for greater efficiency and simplicity.

In closing, my journey to develop a digital gauge cluster has been a way for me to combine all my previous experiences as well as my education gained from
the automotive systems program at LTU. I am hopeful that my work will serve as 
an inspiration for further innovations in the field of automotive instrumentation. 
The possibilities for improving the driving experience and streamlining vehicle 
diagnostics are endless, and I enthusiastically await the exciting developments 
that lie ahead.
REFERENCES

[1] ID4Motion. (2023) ID4Motion digital gauge clusters. [Online]. Available: https://id4motion.shop/?gad=1&gclid=Cj0KCQjwy4KqBhD0ARIAsAEbCt6hge5qOmlR6SroXmfCI6UPIVYzFMSBSVOlyhDnrBzkwcB


APPENDICES

Code

```c
#include <SPI.h>
#include <mcp2515.h>
#include <string.h>

int RPM = 0;
float ECT = 0;
int ABS1 = 0;
int ABS2 = 0;
int ABS3 = 0;
int ABS4 = 0;
int FUEL = 0;

int rdm = 0;
int raw = 0;
int Vin = 5;
float Vout = 0;
float R1 = 990;
float R2 = 0;
float buffer = 0;

struct can_frame canMsg;
MCP2515 mcp2515(10);
```
void setup() {
    Serial.begin(2000000);
    Serial1.begin(115200);

    mcp2515.reset();
    mcp2515.setBitrate(CAN_500KBPS, MCP_8MHZ);
    mcp2515.setNormalMode();

    Serial.println("-------CAN Read----------");
    Serial.println("ID DLC DATA");
}

void loop() {

    Serial1.print("rpm.pic=") + 10;
    Serial1.write(0xff);
    Serial1.write(0xff);
    Serial1.write(0xff);

    printToDisplayVal("rpmval", RPM);
    printToDisplayVal("water", map(ECT, 40, 120, 9, 76));  //ECT Guage
    printToDisplayVal("fuel", map(FUEL, 0, 100, 9, 76));  //Fuel Guage
    printToDisplayVal("speed", (ABS1+ABS2+ABS3+ABS4)/4);  //VSS average speed
if (mcp2515.readMessage(&canMsg) == MCP2515::ERROR_OK)
{
    if(canMsg.can_id == 0x316) { //RPM ID
        uint8_t RPMresult[2] = {canMsg.data[2], canMsg.data[3]};
        uint16_t v1 = *((uint16_t *) RPMresult);
        RPM = v1/6.4;
    }
    else if(canMsg.can_id == 0x329){
        uint8_t ECTresult = canMsg.data[1];
        ECT = ECTresult * 0.75 - 48.373;
    }
    else if(canMsg.can_id == 0x1F0){
        uint8_t Wheelspeed1[2] = {canMsg.data[0], canMsg.data[1]};
        uint8_t Wheelspeed2[2] = {canMsg.data[2], canMsg.data[3]};
        uint8_t Wheelspeed3[2] = {canMsg.data[4], canMsg.data[5]};
        uint8_t Wheelspeed4[2] = {canMsg.data[6], canMsg.data[7]};
        uint16_t v1 = *((uint16_t *) Wheelspeed1);
        uint16_t v2 = *((uint16_t *) Wheelspeed2);
        uint16_t v3 = *((uint16_t *) Wheelspeed3);
        uint16_t v4 = *((uint16_t *) Wheelspeed4);
        ABS1 = v1 * 16 / 256;
ABS2 = v2 * 16 / 256;

ABS3 = v3 * 16 / 256;

ABS4 = v4 * 16 / 256;

buffer = analogRead(0) * Vin;

Vout = (buffer)/1024.0;

buffer = (Vin/Vout) - 1;

R2 = R1 * buffer;

FUEL = map(R2, 70, 395, 0, 100);

void printToDisplayVal(String id, int val){
    Serial1.print(id + "\ val=");
    Serial1.print(val);
    Serial1.write(0xff);
    Serial1.write(0xff);
    Serial1.write(0xff);
}

void printToDisplayText(String id, String text){
    Serial1.print(id + "\ val=");
    Serial1.print(text);
    Serial1.write(0xff);
    Serial1.write(0xff);
    Serial1.write(0xff);
}