**Communication Challenge 3: CAN Bus**



*By: Johnson Domacasse (#4471709)*

*Oktober 2023*

# Abstract:

*A controller area network bus (CAN bus) is a communication network protocol that is used in a variety of sector around the world. Like any protocol it has its advantages and drawbacks. Great aspects revolving around CAN bus is its reliability and easy usage.*

Contents

[Abstract: 1](#_Toc149686064)

[Introduction: 3](#_Toc149686065)

[Procedure: 3](#_Toc149686066)

[Part 1: 3](#_Toc149686067)

[Part 2: 4](#_Toc149686068)

[Results: 5](#_Toc149686069)

[Conclusions: 6](#_Toc149686070)

[References: 6](#_Toc149686071)

# Introduction:

The focus of this document is to hold all of the information that was gathered and processed regarding the CAN bus. CAN stands for Controller Area Network. common digital data network used in automotive, industrial, medical and scientific systems. The CAN bus is used for routing sensor data between pieces of equipment. The main advantages are high resilience to noise, reliability, low cost, simple wiring and ease of use. The disadvantages are that the data packet lengths are small, transmission rates are low and the message transmission cycle time can vary. **[1]**

*NOTE: For the sake of this assignment, I will not be using the provided link as this is outdated. The only thing I will keep is the circuit that was used. Finally,* Due to us not having a soldered joystick, we will be implementing this section using a button and state machines. They will simulate what needs to be done if we were to have a normal joystick. **[2]**

# Procedure:

## Part 1:

In this section our primary goal is to make 2 CAN bus devices communicate with each other. From here we need to decide if we are going to use jumper J1. Once we have the communication ready, all we have to do is make the programs so that the sender can switch an LED on and off on the receiver side. This is done simply by typing in the serial monitor. We will also not be using J1 since we are working with a relatively small system with little to no noise.

For this we will not be needing 8 data bytes so 1 is sufficient. See figure 1 for the circuit.

A diagram of a circuit board

Description automatically generated

*Image 1: Can circuit.*

For this part, a simple state machine was developed to toggle an LED on and off based on the message that is received on the receiver side. See figures 2 and 3 for both state machines.

A diagram of a data flow

Description automatically generated  
*Figure 2: transmitter state machine.*

A diagram of a computer

Description automatically generated

*Figure 2: receiver state machine*

## Part 2:

For this part of the assignment the goal is to change the state of the LEDs depending on what is received from the transmitter. Again, due to the fact that we do not have a soldered joystick, we have opted for another solution. The new implementation is we use a button to change the state that would initially be sent to the receiver using the joystick. We also have to include an additional LED to simulate our “vehicle”. See figure 4 for the extra component circuit.

A circuit board with wires and a few other objects

Description automatically generated with medium confidence

*Figure 4: extra components on circuit.*

The same idea that goes for the joystick would go for the button. If it is pressed, then the state will change. For this implementation We have decided to implement a switch case to deal with the many states that will be handled. The purpose of the switch case for the transmitter size is simply to pass the byte that needs to be sent into the buffer.

On the receiver end we implemented another switch case. This timer, depending on the value that is received, we will change in which manner the LEDs behave. there are 4 states total. 2 blinking states, each for their separate LEDs. One state to turn all of the LEDs off and the last state to turn both LEDs on to resemble the high beam of a car.

# Results:

Initially the link that was provided to us is outdated as stated in the library readme file itself. Once some issues were resolved, we were able to send 8 bytes to the receiver. For the purpose of part A, we only needed to send one. The state machine that was made, made it so that if the transmitter sends a 1, the LED will turn on and if the transmitter sends a 0, it will turn it off.

The second part of the assignment is where things got tricky with data that was being sent. Before, data was being updated before being sent off to the receiver. Now, when a debounced button is pressed, there can be signs that it will not read the value that needs to be passed. In our case we thought this would be solved using debounce or MILLIS instead of delays but this didn’t work. Although in the end, it does have the functionality that we desire.

We even tried disconnecting the serial port or some wiring to see what would happen. The result is that we would have a system in a sort of “pause” state on the receiver side. It will not receive any messages but it would still be reading them. If the problem occurred on the receiver side instead then the transmitter would simply complain that the data was not sent correctly (which is expected).

# Conclusions:

In the end the conclusion is somewhat satisfactory. There were some challenges like the lack of a joystick that were encountered but it was handled using a similar approach to a situation in which we do have a joystick. All the requirements are met.

In terms of efficiency, the only thing that could’ve been done differently is the use of delays. Instead we would opt for MILLIS. However, during testing this did not seem to have much of an effect on the actual working of the system. Performance wise we think this is a good implementation.

If others wish to extend this software it can also be done quite easily. Since we are using a switch case to handle all of the states, adding a new state would simply be creating a new case on each side of the CAN bus.

# References:

**[1]** - Fowler, D. S. (2020, August 18). *CAN Bus*. CAN Bus Wiring Diagram, a Basics Tutorial | Tek Eye. <https://tekeye.uk/automotive/can-bus-cable-wiring>

**[2] -** Administrator. (2023, March 13). *MCP2515 can bus interface tutorial*. ElectronicsHub. <https://www.electronicshub.org/arduino-mcp2515-can-bus-tutorial/>