Short Messaging Control Mechanism for CAN Bus Architecture

Andrew Cline
EE 6900 Software Defined Radio & IoT
Kennesaw State University
Kennesaw, GA, USA
acline11@students.kennesaw.edu

Clifford Gilbert
EE 6900 Software Defined Radio & IoT
Kennesaw State University
Kennesaw, GA, USA
cgilbe26@students.kennesaw.edu

Goran Novakovic
EE 6900 Software Defined Radio & IoT
Kennesaw State University
Kennesaw, GA, USA
gnovakov@students.kennesaw.edu

***Abstract*— This paper describes development of a simulated Software Definer Radio (SDR) implementation for use with CAN bus devices. The simulation environment used was Simulink, a MATLAB-based graphical programing and modeling software. This project is open source and thus open for contributions and expansions. Future expansions to consider may contain expansion of other "bus" style protocols that are used in industry like LIN bus, and many more. The project results show that this implementation of SDR can prove to be valuable and further research and physical testing is recommended to prove its effectiveness. Once it's proven to be effective in real-world applications it may be deployed in many industrial environments before it's available to the consumer space devices.**

Keywords—SDR, CAN bus, Simulink, MATLAB, BPSK

I. Introduction

The focus of the project is to create a Simulink model that will reliably transmit and receive a short length message that can be applied to a variety of use cases. Use cases for a Software Defined Radio (SDR) in these applications might range from safety control systems inside of an Oil rig or refinery, Military, Medical, Home, Emergency Responders, Automotive Civilian, Automotive Commercial and many more industries or consumer quality of life improvements. Example for these messages in an Emergency Responder case might be the siren, civilians near the emergency responder vehicles can hear and see them but every second matters when responding to an emergency. Thus, an emergency responder could be transmitting a short radio signal that the civilian cars can interpret and then over the sound system or instrument panel warn the drivers of an emergency vehicle approaching their position and to move over. The use cases are only limited by the integrator's imagination. Furthermore, the use of an SDR can lower the cost, remove proprietary barriers, and increase freedom of design and implementation only limited by the designer's knowledge.

II. Methodology

To develop the Simulink model, the project was broken up into three main parts: CAN Bus, SDR Digital Signal Processing, and Transmitter & Receiver. Each part is defined by how the input is manipulated to achieve the overall goal of transmitting CAN bus data. Figure 1 shows an overall block diagram of the Simulink Model.

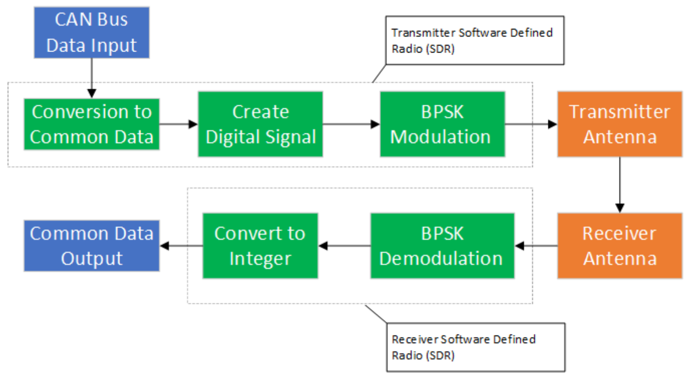

Figure 1: Model Block Diagram with SDR Functionality Outlined

A. Controller Area Network (CAN) Bus

The main data input to the SDR will come from a CAN bus system. While the model shows proof of concept for any input, a CAN bus system is widely utilized in automotive, industrial, medical, and maintenance applications. The payoff in these applications is the reduction in overall wiring required to route messages throughout the system. The CAN protocol utilizes serial half-duplex data transmission through CAN high and CAN low buses. A bus architecture allows connection to multiple sensors and processors through two wires. It also allows for additional sensors or processors to be added to the existing architecture without requiring increased wiring impacts. For our model, we are only focused on transmitting the extracted data field from the CAN message. A CAN protocol data field can be a maximum of 8 bytes. To show the model can handle a CAN bus, we decided to utilize all 8 bytes of the data field as our main input to the model.

B. SDR Digital Signal Processing

A major cornerstone in this design is the SDR. Due to its flexibility in protocols and tuning, the SDR makes the most sense to utilize for developing a short messaging control mechanism. In our model, the SDR is responsible for converting, developing and modulating the incoming digital integer from the CAN bus. Converting the CAN bus input into a common data value is important since each system is different. For example, in the automotive industry, each car manufacturer has different CAN bus values assigned to the brakes, steering, acceleration, etc. In order to transmit this data from one car to the next, a conversion has to be made so two different CAN bus systems can understand the same message. After the conversion, the SDR creates a digital signal based on the 8 byte data input by utilizing a carrier digital waveform. The digital signal is sent to a modulator that uses Binary Phase-Shift Keying (BPSK). The function of BPSK modulation is twofold: to encrypt the data based on a specific phase value and to create a transmittable signal by the transmitter. For the encryption, BPSK utilizes a phase value to shift the binary input. By creating an algorithm, the phase value can change to encrypt the data for transmission. Since the data is a digital value, the transmitter requires a complex baseband signal as an input. BPSK takes the binary values and outputs a complex baseband signal directly to the transmitter. The difference in a logical 1 and logical 0 is set by the phase shift value. All of this is done in software, minimizing the hardware requirements of the design.

C. Transmitter & Receiver

Lastly, the model needs to behave as close to the real-world environment as possible. One of the hardest parts of this project was determining what parameters needed to be added to the overall transmission of the signal. Due to time constraints, the group focused on providing noise to the simulation between the transmitter and receiver. To do this, the model uses additive white Gaussian noise (AWGN) to distort the transmitted signal before it reaches the receiver. The noise provides a more realistic output to the receiver that can help determine if signal degradation can create issues in our design.

III. Simulink Model Design

Our group decided to use Simulink as the CAN Bus tools were readily available in the Vehicle Network Toolbox. It was decided that as opposed to using a CAN DBC file or ARXML file we would create an input signal with a "***Signal From Workspace***" block. This was decided because we also wanted to implement a BPSK (Binary Phase Shift Keying) block as a method of modulation for our CAN Bus data messages. To create a signal that is a 1:1 replica of a BPSK signal we would need to modulate the signal we created with a sinusoid block. This process proved to be impossible, however, due to the datatype restriction on the input of the CAN Pack block, which was UINT 8. Due to this restriction we decided to skip this process and proceed to incorporate a binary signal that changes with respect to sim time. To add realism, we opted for cyclic repetition so the model would be able to run indefinitely with the code being transmitted in a loop. We named the different signals according to commonly checked functions such as A/C, steering, ABS, TPMS, Smart Key, Airbag and Speed. A manual switching system was then created that provided the ability to change test parameters during model runtime. It is set up in a ladder style meaning that only one block is allowed to propagate its data after being switched. The associated signal is then propagated into a "***Sample and Hold***" block that samples the data at a period of five seconds per bit, which then gets converted into a UINT8 signal from a double. After the data passes through the UINT8 block, it is passed to the "***CAN Pack***" block. This block associates a message name, identifier type (Standard 11-bit), CAN Identifier and length field for the input signal. Once processed, we pass the signal to a "***CAN Transmit***" block which sets a channel for the data to be sent at a rate of 0.025s with a corresponding bus speed of 500 kbps. The message is then received and unpacked to be passed to the "***BPSK Modulator***" block. The phase offset parameter was set to (-pi/5) to account for both the in-phase and quadrature component portions of our input signal. After the modulation has been performed it is now time to transmit the signal wirelessly. Once again, we wanted to replicate a real-world scenario as closely as possible, therefore the implementation of the "***AWGN***" block (Additive White Gaussian Noise) was used to add an element of ambient disturbance. The signal is then sent to the wireless receiver block and then processed by the "***BPSK Demodulator***" block which has a phase offset of (pi/5) to correct the phase shift. The phase shift can be seen in Figure 2 as a constellation diagram. It should be noted that Simulink blocks handle several details that wouldn't be handled in real life. We would have needed to implement a high-pass filter to account for the introduced noise as well as an ADC to account for the transition from analog to digital data transmission to the receiving hardware.

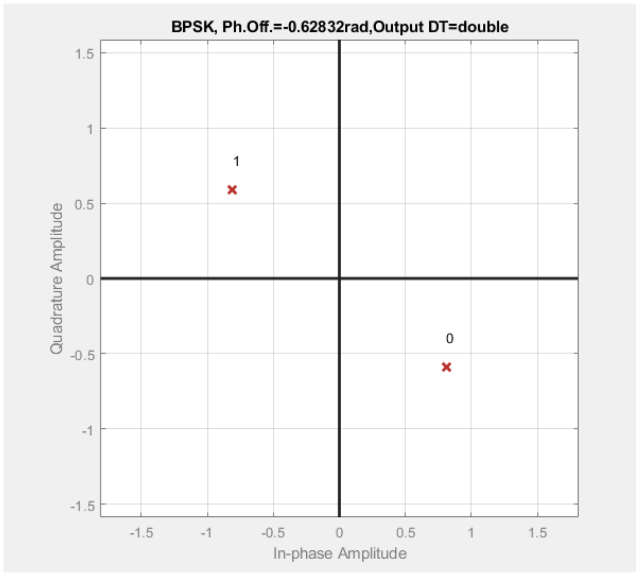

Figure 2: BPSK Phase Shift Constellation Diagram

IV. Test Cases and Results

As described in the previous section, we utilized seven vehicle specific subsystems to simulate real data being sent from the vehicle. Each one is represented by a different 8 bytes and is input to the model as an array. The ladder style switch system was utilized to pick which signal is sent. To test the model, we selected an input then ran the simulation for at least two full periods or 80 seconds. A period in our case is defined as one full array being sent. For each simulation, we utilized the recorded output to assess how the signal is transformed from start to finish. There are five main signals we looked at: CAN Bus output, BPSK modulated signal, transmitter output, receiver output, and the BPSK demodulated signal. To verify a success, the BPSK demodulated signal must match the exact same bytes that were input to the model.

The results of our testing concluded with a 100% success rate. To capture all of the testing data, a test results spreadsheet was made. One area we expected more variation in is the receiver output due to the AWGN that was introduced to the signal. As seen in Figure 3, the received signal does have a variation due to noise that is less exact than the transmitted signal. However, the noise did not play a large enough roll to impact the BPSK demodulation. With this data, we showed our proof of concept for the model design can successfully be achieved.

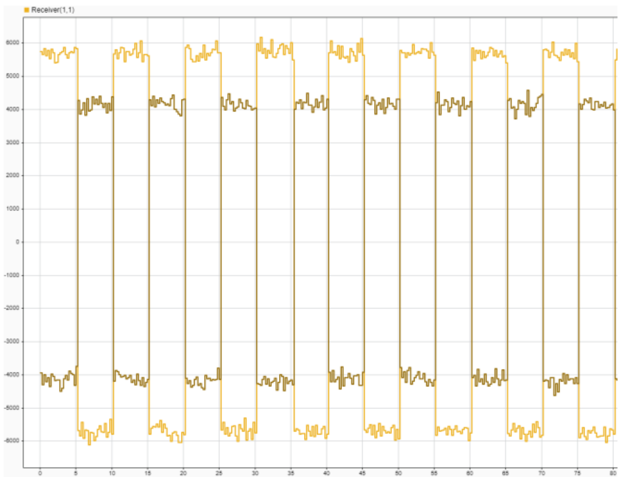

Figure 3: TPMS Receiver Output Result from Simulation

V. Next Steps

Future developments for this project include expansion of current work that was performed into the physical domain. Using a CAN bus device generating predictable CAN messages and translating them into RF signals with a transmission capable SDR. A FPGA would prove to be the most effective way to decode messages from the CAN bus in a speedy manner and then over the physical interface send the signals that the SDR needs to transmit. A SDR receiver would then need to send that signal into another FPGA or the same FPGA for cost effectiveness and retrieve the initial FPGA output that was generated from the CAN bus message. Further works may include additions of other bus style protocols that can be translated into RF data using SDR devices. Another addition that would improve and solidify this implementation for industry is the use of some modulation and encryption as to prevent easy access to the data from malicious actors.

VI. Conclusion

The current project work and goals were accomplished. Due to this projects open-source nature any student or professional may expand this idea into their own respective interests. As previously stated this projects idea is only limited to some physical device constraints but outside of that it can be utilized in many different projects and product improvements. This projects idea is not limited to just CAN bus or the automotive use case. Thus the project shows a high expansion possibility outside of the scope in this paper and code library.

Below is our GitHub repository link for our model, milestone reports, and all additional source files: https://github.com/cgilbert43/SDR_KSU/tree/main/Matlab%20Files.

Acknowledgment

We would like to recognize the work of Vicente Matus, Nicolas Maturana, Cesar A. Azurdia-Meza, Samuel Montejo-Sanchez, and Javier Rojas from the University of Chile. Their paper was very influential in utilizing an SDR to transmit CAN bus data. Their project set the foundation for our model design.

References

[1] Matus, Vicente & Maturana, Npossible,, Azurdia-Meza, Cesar & Montejo Sánchez, Samuel & Rojas, Javier. (2017). Hardware Design of a Prototyping Platform for Vehicular VLC Using SDR and Exploiting Vehicles CAN Bus. 10.1109/SACVLC.2017.8267606.

[2] Design and Create a Custom Block - Matlab & Simulink, www.mathworks.com/help/simulink/ug/tutorial-creatinga-custom-block.html. Accessed 4 Mar. 2024.

[3] Frenzel, Lou. “Understanding Modern Digital Modulation Techniques.” Electronic Design, Electronic Design, 12 Jan. 2023, www.electronicdesign.com/technologies/communications/article/21798737/electronic-designunderstanding-modern-digital-modulation-techniques.

[4] “Data Encryption Methods & Types: Beginner’s Guide to Encryption.” Splunk, www.splunk.com/en_us/blog/learn/data-encryption-methods-types.html. Accessed 4 Mar. 2024.

[5] “Phased Array System Toolbox — Blocks.” Reference List - MATLAB & Simulink, www.mathworks.com/help/phased/referencelist.html?type=block&s_tid=CRUX_topnav. Accessed 4 Mar. 2024.

[6] “BPSK Demodulator Baseband.” Modulate Using BPSK Method - Simulink, www.mathworks.com/help/comm/ref/bpskmodulatorbaseband.html. Accessed 21 Mar. 2024.

[7] “Build CAN Communication Simulink Models.” Build Can Communication Simulink Models - MATLAB & Simulink, www.mathworks.com/help/vnt/ug/build-can-communication-simulink-models.html. Accessed 11 Mar. 2024.

[8] McHugh, Brendon. “SDRs to Deploy Vehicular Networking for AVs.” EE Times Europe, 31 Aug. 2022, www.eetimes.eu/using-sdrs-to-prototype-and-deploy-vehicular-networking-for-avs/.