Short Messaging Control Mechanism for CAN Bus Architecture

EE 6900 – Software Defined Radio and IoT

Andrew Cline, Clifford Gilbert, Goran Novakovic

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 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 can hear and see it clearly 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 to the side. 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.

This project offers the open-source community the opportunity to leverage the principles it develops and simulates. These principles can be applied across various protocols necessitating physical connections, transforming physical layer data transmission into radio wave transmission of data. Serving as a foundational development, this project provides principles for the transmission of simple data packets. By converting and transforming data packets or frames from the physical layer into dependable radio signals, it establishes fundamental principles accessible for learning or development endeavors by anyone. SDR hobbyist may utilize this project for simulating and understanding how outside systems can be attached to the SRD or any radio communication hardware.

COMPLETED WORK

Create the CAN Bus Model – 100%

Using the Vehicle Network Toolbox a simple CAN Bus was created [7]. This CAN Bus model is simulating a real CAN Bus where the user has no access to the CAN Bus data except for the predefined ASE specification of length of frames and the data formatting. For demonstration purposes in the figure 1 the model is displaying a sequence of data FF 0 FF 0 FF 0 FF 0 being converted into binary values of 0 (0) and 11111111 (FF).

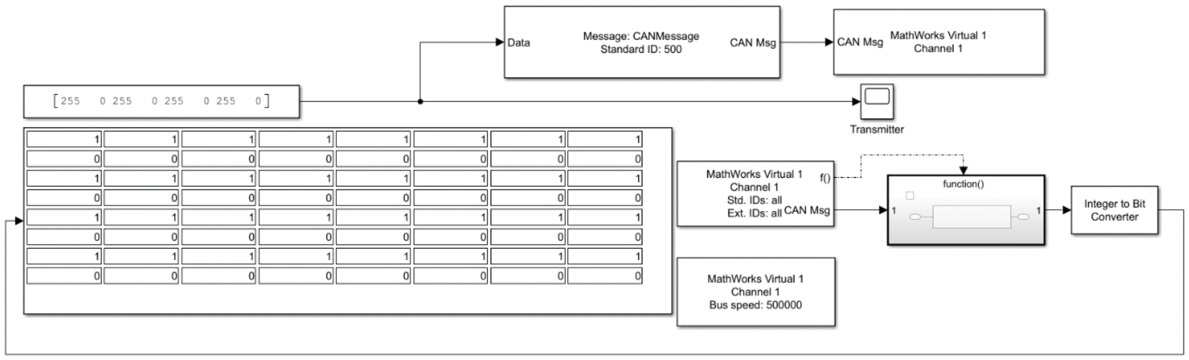


Figure 1. SimuLink CAN Bus Model

Model the Transmitter and Receiver – 100%

For our model to represent the full exchange of data between one user to another, we added transmitter and receiver models. Using the Phased Array System Toolbox, a Transmitter and Receiver models were added [5]. The Receiver model requires the latest Matlab 2024a version to implement the Receiver Simulink model. For our model, we are mainly focused on the conversion of data from CAN Bus to the signal sent through the transmitter. With that being said, a stretch goal of ours is to implement a better noise model on the receiver side to better represent the real world. Currently, we have added a white Gaussian noise model, but there are also other factors within the receiver model we intend to explore.

Model the BPSK Signal and Processing Conversion – 100%

To convert the existing binary signal into a waveform for the transmitter to send out, a BPSK Simulink model is added to our design. The BPSK model utilized is a part of the Communications Toolbox [6]. Utilizing the BPSK model only requires a digital signal input and outputs the converted signal. First attempts at integrating the BPSK model were simple, as shown in Figure 2. A value is entered into a 1-D lookup table, to model the conversion from a manufacturer-specific CAN system to a Universal Common Code value. To test the BPSK conversion was being done properly, the selected integer was converted to a bit, which was fed as the input to the BSPK model. The output of the BPSK modulation model was directly input into a BPSK demodulation model. From here, the received binary values were converted back into an integer. This confirmed that the BPSK model set could be used to convert our digital input into an analog signal for the transmitter.

One portion that we are missing originally in the design was creating a continuous digital input instead of discrete binary values all being sent one after the other. While it was good to verify the binary values were transmitted and received properly, it is unrealistic during real time. To further fix this, we added a job to convert the integer to a continuous data signal. More information can be found under that job.

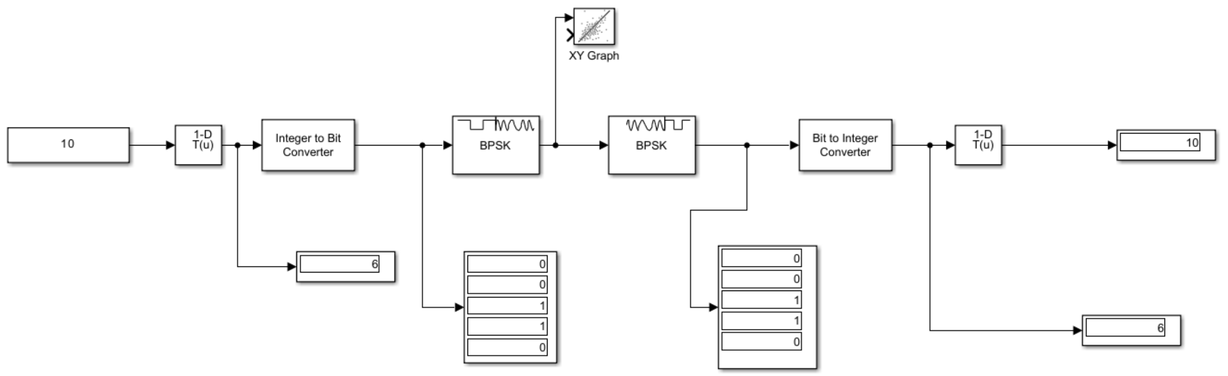


Figure 2. Initial BPSK Modulation Model with Lookup Table and Binary Conversion

Create the Data Encryption Model – 100%

In order to add an encoding and encryption scheme to the model we decided that BPSK or binary phase shift keying as a methodology would be the optimal choice. “Phase-shift keying is a digital modulation process which conveys data by changing (modulating) the phase of a constant frequency carrier wave. The modulation is accomplished by varying the sine and cosine inputs at a precise time.” We decided that it was a good idea to use the phase shifting aspect of BPSK as an encryption scheme. For our next milestone, we will create a method in which the input phase shift to the BPSK modulation block is detected by the demodulation block. We will also determine a specific amount of time we want this change to occur. Think of this as being similar to having an RSA Token. As a visual representation we will be using a constellation chart as seen below:

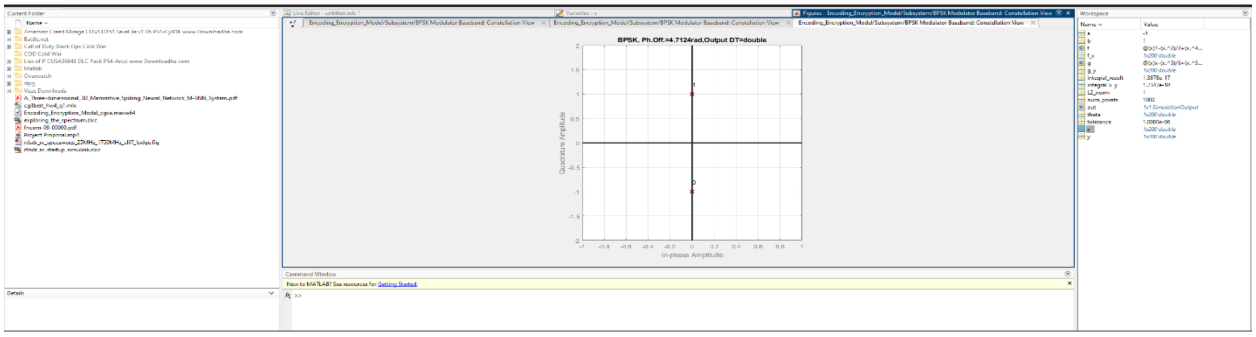


Figure 3.

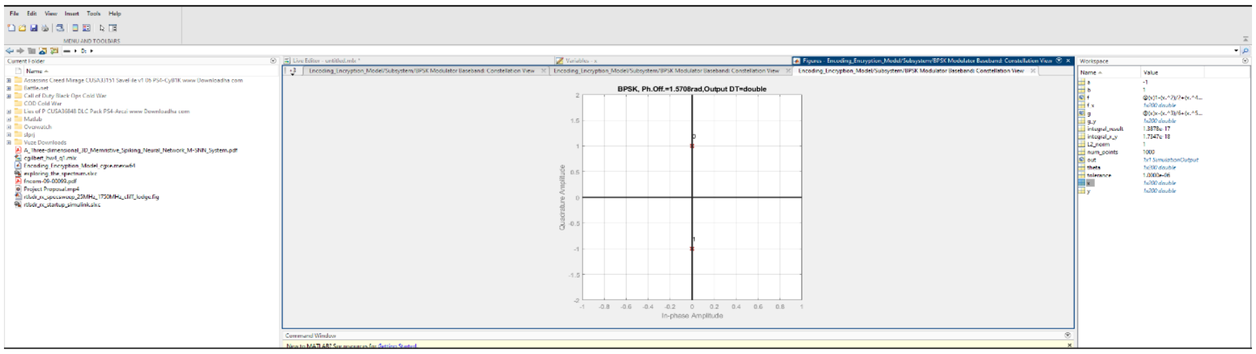


Figure 4.

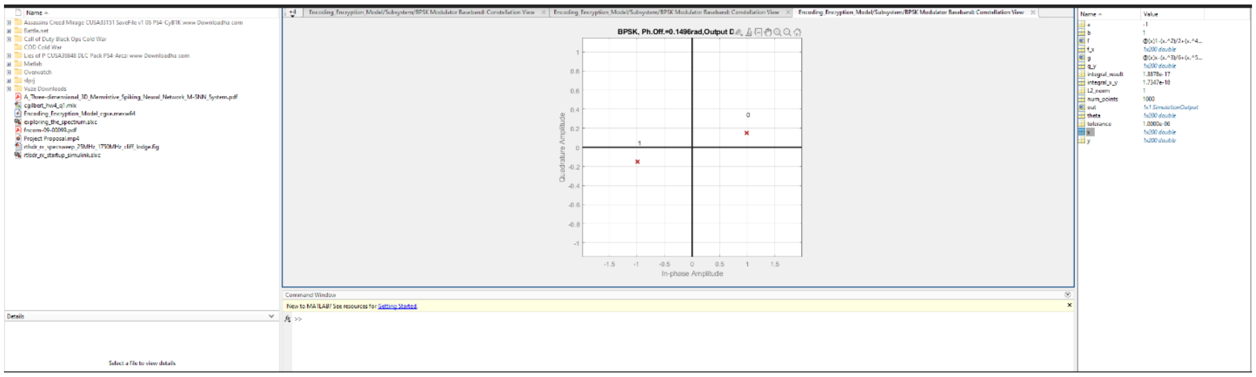


Figure 5.

As you can see in Figures 4 and 5, we decided to shift to just the quadrature components and then we decided to use both in-phase and quadrature components. We can do this in an infinitesimal range from 0 to 2pi, making this a very good random number encryption scheme. We will use a PRNG for the values and decide how many we should implement for maximum data protection and to also mitigate data confusion between vehicles.

Create Data Transfer from Integer to Data Signal – 100%

When we began our research, we were under the assumption that a 1-D linear vector would be sufficient as a means to provide this stream of data. However, we soon discovered that we would need another methodology that could deliver binary data with respect to sim time. Therefore, we decided to use the "Signal From Workspace" in conjunction with the "Sample and Hold" Simulink blocks. "The Signal From Workspace block imports a signal from the MATLAB® workspace into the Simulink® model. Unlike the Simulink From Workspace (Simulink) block, the Signal From Workspace block holds the output value constant between successive output frames (that is, no linear interpolation takes place)." "The Sample and Hold block acquires the input at the signal port whenever it receives a trigger event at the trigger port (marked by). The block then holds the output at the acquired input value until the next triggering event occurs." This in tandem with a pulse generator was enough for us to prove that this concept works.

Given a sample time of 5 seconds for the input data stream of [1 0 1 1 0], triggered on rising edge with a pulse being produced at every second with a pulse width of 1 ms, we were able to get a plot of the following analog signal within a 100 second sim run.

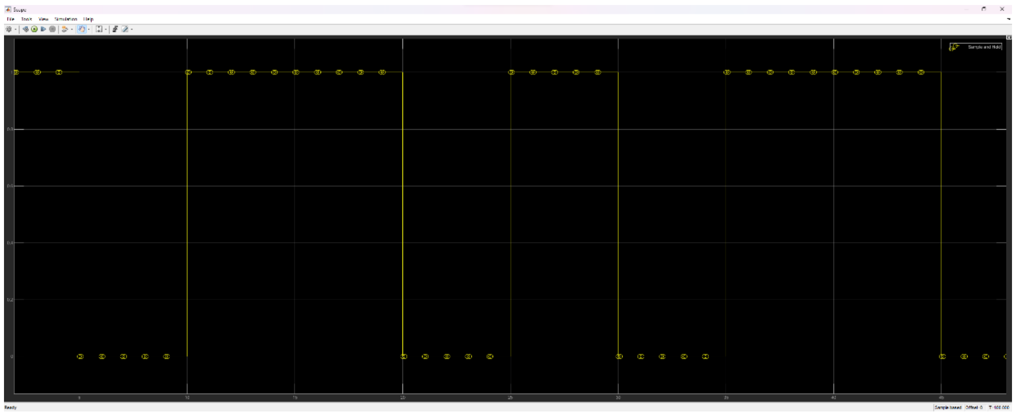


Figure 6. Data Stream Output

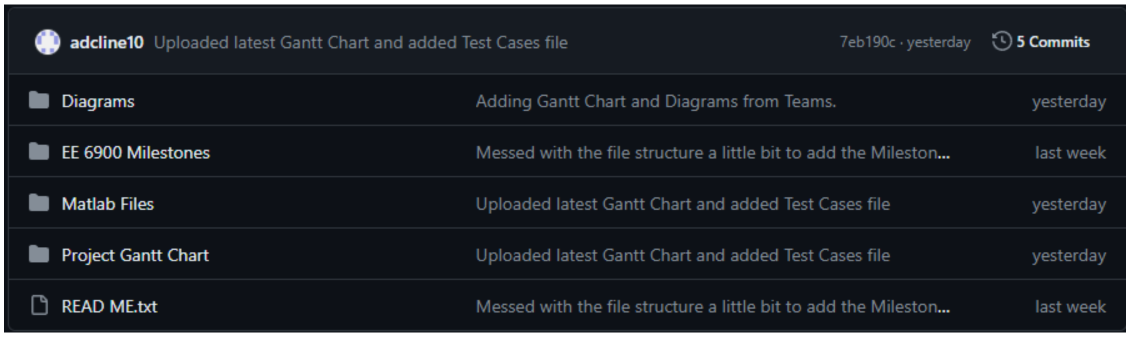
As you can see the data being output is a contiguous stream of the input vector of binary digits [1 0 1 1 0]. Next, we will implement a switching mechanism to incorporate every data signal we want to provide. It is also of important note that we chose this method due to its ability to transfer data at a rate of 4.5 Mbps, which we felt is sufficient enough to account for such small values of data.

Develop Progress Report Documentation – 100%

This specific task in the Gantt Chart is to support development of the Milestone 3 documentation. The Progress Report was outlined per the rubric for Milestone 3. Once the Gantt Chart was updated, each section was added to match the work completed, in-progress, and planned for the future. For the work completed, we go through the development process and the resulting outcome to close the task out. For In-Progress, we discuss where we are in the process of closing out the task. All Future Work is laid out and we further define what the scope of this work will look like at this point in the project.

Create Github Repository – 100%

Based upon comments received on our previous Milestone 2 Project Proposal, we created a Github repository to upload our work. The Github repository allows us to directly work with the Matlab models and upload our progress without transferring files between team members. In Figure 7 the Github repository file structure is shown. All milestones to this point are uploaded under "EE 6900 Milestones." The current versions of our Simulink Models are fund under "Matlab Files." All supporting documentation has been uploaded, such as the Gantt Charts and any diagrams from the previous Milestones.


adcline10 Uploaded latest Gantt Chart and added Test Cases file 7eb190c · yesterday 5 Commits

Diagrams	Adding Gantt Chart and Diagrams from Teams.	yesterday
EE 6900 Milestones	Messed with the file structure a little bit to add the Mileston...	last week
Matlab Files	Uploaded latest Gantt Chart and added Test Cases file	yesterday
Project Gantt Chart	Uploaded latest Gantt Chart and added Test Cases file	yesterday
READ ME.txt	Messed with the file structure a little bit to add the Mileston...	last week

Figure 77. Current Github Repository File Structure

IN-PROGRESS WORK

Test Run Transmitter Conversion Model – 25%

The Transmitter subsystem is running into some issues in its current state. By isolating portions of the system, we know the binary inputs to create the digital signal work and are fed properly to the BPSK model. The transmitter model itself also appears to be working based on the transmitted signal we are seeing. Our current issues are with the conversion from the CAN bus input to the Universal Common Code bits. The CAN bus input is not supported as an integer and will require some more work to convert the output to an integer. We assume the output of the CAN bus is an array, and we may need a model in between to present one integer at a time to the Integer to Bit Converter. The lookup table we implemented in the original BPSK model testing has not been fully integrated. We need to define certain integer values from the CAN bus to a specific Universal Common Code value.

Test Run Receiver Conversion Model – 50%

In its current state, the receiver subsystem models have all been integrated. The receiver model takes in the transmitted signal with additional white Gaussian noise to be closer to the real world. From here, the received signal goes through the BPSK model. The last part of the receiver subsystem converts the binary signal back to the Universal Common Code output. The receiver model still needs more fine tuning, and we want to continue working on the noise modeling in the subsystem. To properly determine if our conversion scheme within the Transmitter SDR works, we need to verify it can withstand real world noise conditions. Since the use case looks at transmitting over short distances, we don't foresee an issue with our current conversion scheme.

Upload PDFs of Milestone Reports (5 Total) - 60%

At this point in time, Milestones 1 through 3 have been completed and submitted. PDF files of all three Milestone reports have been uploaded to the Github repository. The remaining 40% will be completed when Milestone 4 (Presentation PowerPoint) and Milestone 5 (Project Report) are uploaded. Based upon the current schedule for Milestone 4 and Milestone 5 due dates, we should be 80% complete with this task by 4/15 and 100% complete by 4/29.

Upload Supporting Documentation (Gantt Chart, Diagrams, etc.) - 50%

In our Github, we are adding all supplementary diagrams and documents that can be found in our Milestone documents. This source data will allow futures groups to build upon our figures and diagrams in their own designs. The Gantt Charts are baselined for each Milestone and dated to show the progression of our project through the development stages. Currently, we have all Gantt Chart versions and all diagrams, not including Matlab screenshots, uploaded to the Github repository. Once Milestones 4 and 5 are completed, any new or updated supporting documentation will be uploaded to Github and this task will be 100% complete.

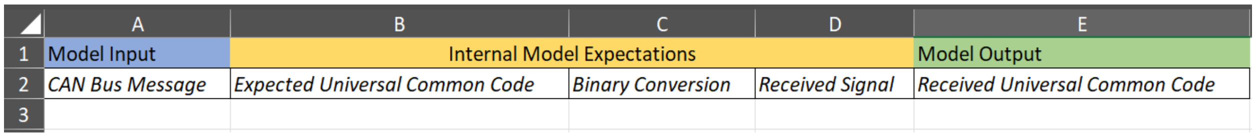
FUTURE WORK

Update the Simulink Model Based on Test Runs – 0%

After all the individual models are fully integrated into one master model, we will conduct test runs with CAN data inputs. Through these test runs, we hope to find errors or weaknesses that can be addressed before going into our formal testing. As issues arise, we will mitigate them as much as possible. Due to time constraints, if there are any issues not fixed, we will note these in our final presentation and report.

Run Test Cases Through the Completed Simulink Model – 0%

Once the Simulink Model has been updated based on the test runs, it will enter formal testing. During formal testing, we will use specific CAN bus messages to verify the validity of our model. In Figure 8, an outline of our test cases is shown. There are three internal conversions that we are tracing as additional test points to know where a breakdown in our output could be found. If the Model Output matches the expected Universal Common Code in column B, then we have a successful test.



	A	B	C	D	E
1	Model Input	Internal Model Expectations			Model Output
2	*CAN Bus Message*	*Expected Universal Common Code*	*Binary Conversion*	*Received Signal*	*Received Universal Common Code*
3					

Figure 88. Test Case Spreadsheet for Formal Model Testing

Create Project Report Draft – 0%

This task is related to Milestone 5. Once we have completed formal testing, there will be enough data to start going drafting our Project Report. The goal of this task is to have the introduction and model details completed, which do not require the testing results to be fully analyzed.

Upload Completed Simulink Models – 0%

After the formal testing is completed, the final Simulink Models will be uploaded to Github. Using the "Read Me," we will describe the different models included for the open-source community.

Review the Results of the Simulink Model – 0%

The results of the testing will be rather quick. As described earlier, if the model correctly outputs the Universal Common Code, then the model succeeded. If the model does not correctly output the Universal Common Code, the internal expected values will be assessed. From these, we hope to capture where our model went wrong and provide these reasonings with our data. In our project report, we will define any areas of our model that should be worked on further for it to succeed.

Calculate the Success Rate of the Simulink Model – 0%

Using the expected outcome of the test cases, we will calculate the success rate of our model. The success rate will be presented alongside our testing results to describe if our model is good or needs additional work.

Develop Diagrams of Results – 0%

Based upon the test results, diagrams will be created. The goal is to present our test results so a reader can assess the effectiveness of our model. For the failures experienced, a diagram will be created to show where in the model the disconnect occurred.

Develop PowerPoint Presentation – 0%

This task represents the deliverable for Milestone 4. A KSU Template will be used for our slides. Using Milestones 2 and 3, the project introduction and model descriptions will be incorporated. Once the test data is analyzed, our results and completed model will be presented. The team will split up the slides evenly for the in-class presentation. One practice run is planned before the formal presentation.

Develop Project Report Documentation – 0%

This task represents the deliverable for Milestone 5. The previous draft will be utilized as our starting point. From here, the test results should be analyzed, and any new diagrams are created. We will also take any feedback from our Milestone 4 presentation and incorporate it into the report.

UPDATES TO PROPOSAL EXPECTATIONS

There are a few changes made to the previously proposed Simulink model. We determined that the conversion to Universal Common Code makes more sense to occur on the transmitter side. By doing so, the lookup table can be pre-loaded based on the CAN bus system type. Another change was to utilize the phase of the BPSK conversion to encrypt the data. Originally, a cybersecurity protocol was going to be used, but most tools require keys much larger than our current data set and seemed like overkill. By utilizing the existing BPSK model, we are cutting down on the needed models in our design. Our updated model block diagram and current Simulink model can be found in Figures 9 and 10.

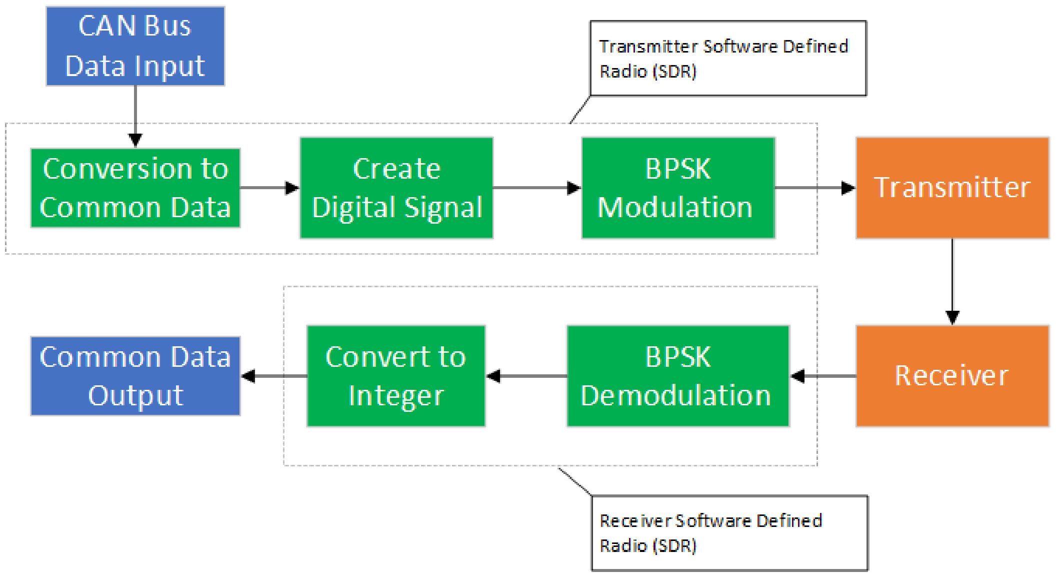


Figure 99. Updated Simulink Model Block Diagram

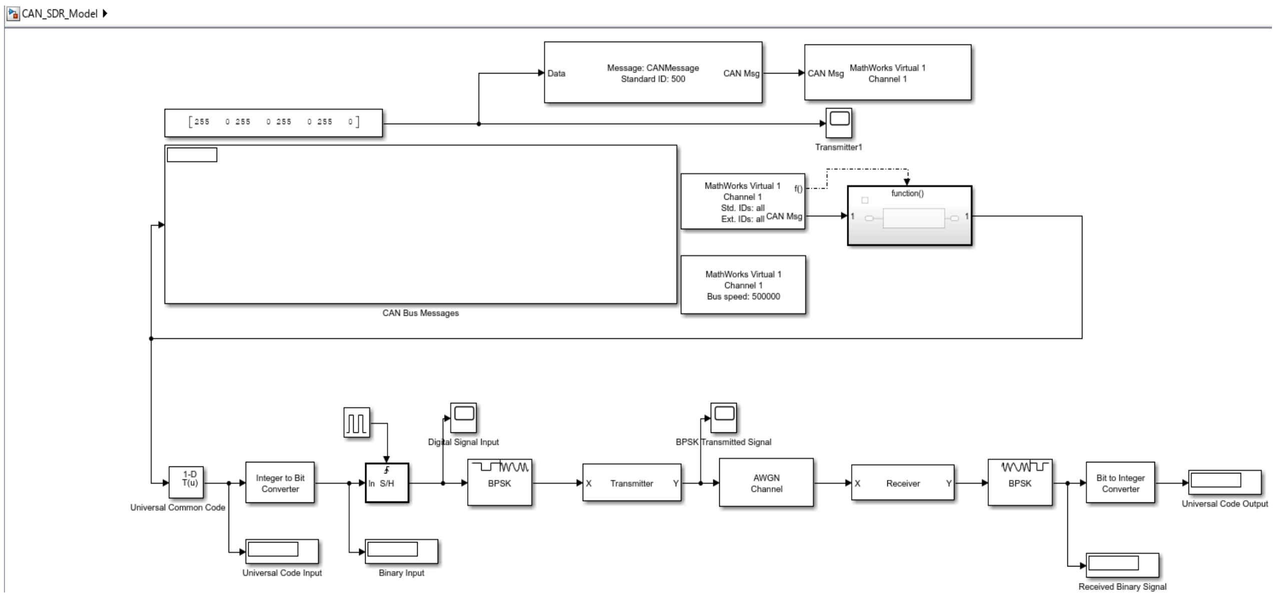


Figure 1010. Current Project Simulink Model

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-creating-a-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-design-understanding-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.

PROJECT GANTT CHART

PROJECT: Short Messaging Control Mechanism for CAN Bus Architecture

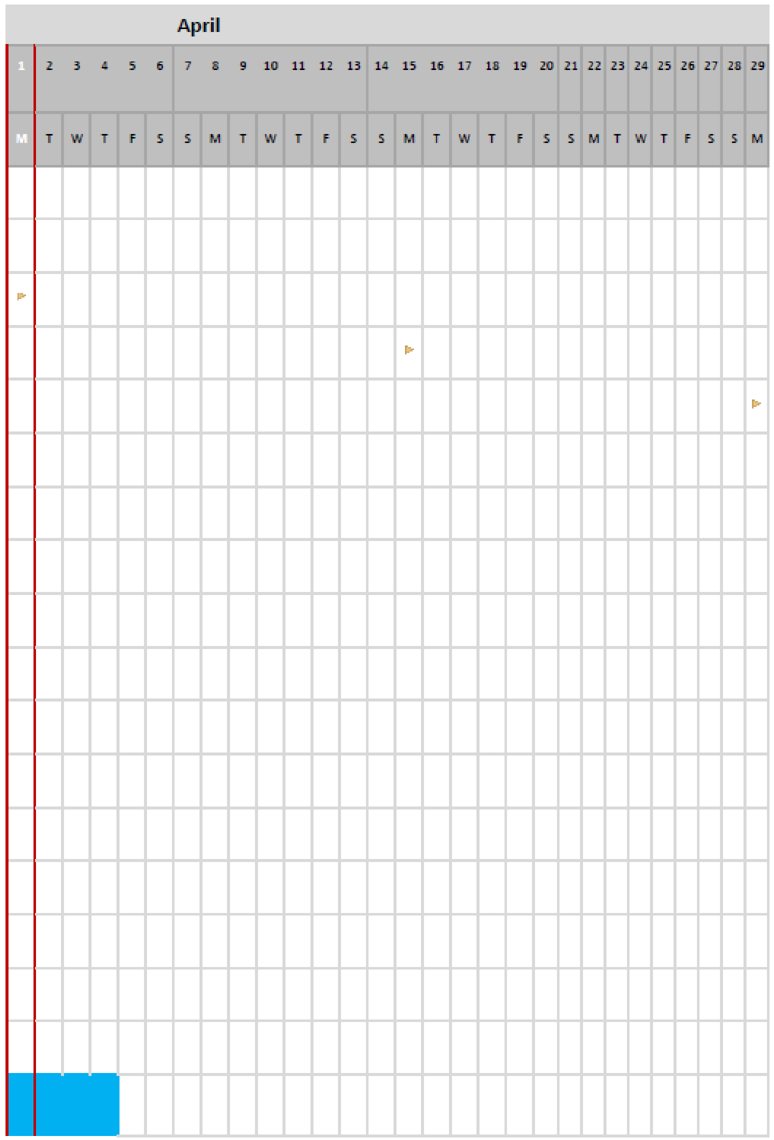
Kennesaw State University

Andrew Cline, Clifford Gilbert, Goran Novakovic

Legend:

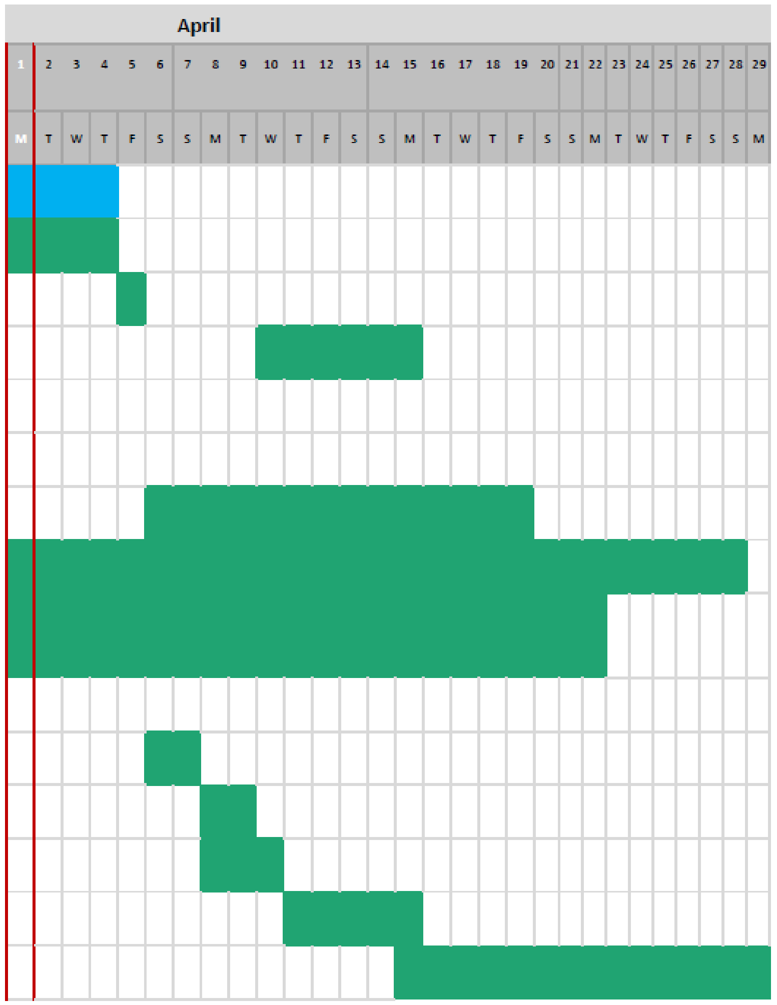
Project start date:	1/31/2024
Scrolling increment:	25

Milestone description	Category	Assigned to	Progress	Start	Days
Milestone Reports					
#2 - Project Proposal	Milestone	All	100%	3/4/2024	1
#3 - Progress Report	Milestone	All	100%	4/1/2024	1
#4 - PowerPoint Presentation	Milestone	All	0%	4/15/2024	1
#5 - Project Report	Milestone	All	0%	4/29/2024	1
Project Selection					
Define Project Goals	On Track	All	100%	2/28/2024	1
Develop Proposal Documentation	On Track	All	100%	2/29/2024	5
Collect References	On Track	All	100%	2/29/2024	5
Simulation Development					
Create the CAN bus model	On Track	Goran Novakovic	100%	3/5/2024	7
Model the Transmitter and Receiver	On Track	Clifford Gilbert	100%	3/5/2024	14
Model the BPSK Signal Processing Conversion	On Track	Andrew Cline	100%	3/10/2024	7
Create the data encryption model	On Track	Clifford Gilbert	100%	3/19/2024	7
Create data transfer from integer to data signal	On Track	Clifford Gilbert	100%	3/26/2024	3
Develop Progress Report Documentation	On Track	All	100%	3/25/2024	7
Simulation Testing					
Test run Transmitter Conversion Model	Low Risk	Andrew Cline	25%	3/26/2024	10



Project start date:	1/31/2024
Scrolling increment:	25

Milestone description	Category	Assigned to	Progress	Start	Days
Test run Receiver Conversion Model	Low Risk	Goran Novakovic	50%	3/26/2024	10
Update the Simulink Model based on Test Runs	On Track	All	0%	3/31/2024	5
Run test cases through the completed Simulink Model	On Track	Clifford Gilbert	0%	4/5/2024	1
Create Project Report Draft	On Track	All	0%	4/10/2024	6
Project Github Repository					
Create Github Repository	On Track	Clifford Gilbert	100%	3/22/2024	1
Upload Completed Simulink Models	On Track	Goran Novakovic	0%	4/6/2024	14
Upload PDFs of Milestone Reports (5 Total)	On Track	Andrew Cline	60%	3/29/2024	31
Upload Supporting Documentation (Gantt Chart, Diagrams, etc.)	On Track	All	50%	3/29/2024	25
Result Analysis					
Review the Results of the Simulink Model	On Track	All	0%	4/6/2024	2
Calculate the success rate of the Simulink Model	On Track	Goran Novakovic	0%	4/8/2024	2
Develop Diagrams of Results	On Track	Andrew Cline	0%	4/8/2024	3
Develop PowerPoint Presentation	On Track	All	0%	4/11/2024	5
Develop Project Report Documentation	On Track	All	0%	4/15/2024	15


April
M T W T F S S M T W T F S S M T W T F S S M T W T F S S M