Designing a Pseudo Tire Pressure Monitoring System Transmitter using Software Defined Radios

Major Qualifying Project completed in partial fulfillment of the Bachelor of Science degree at Worcester Polytechnic Institute

Advisor:

Professor Alexander Wyglinski

Authors:

Stella Banou _____

Felicia Gabriel _____

Adriana Reyes _____

Syed Shehroz Hussain _____

MQP AW1-CPS1

Submitted on: March 4, 2016

Abstract

The purpose of this project is to create a software defined radio based transmitter that can mimic the signals of the Tire Pressure Monitoring System (TPMS) sensors. The team used an amplifying receiver to read signals as well as decode data. The transmitter was built using a USRP N210 software defined radio running MATLAB code. The team conducted a series of tests to verify the functionality of the pseudo transmitter using both computer simulation and over-the-air and with a real vehicle. The results of the tests verified that the pseudo transmitter can communicate properly with the receiver of the previous project as well as a real TPMS receiver in a vehicle. The results of this project are useful in identifying breaches in the TPMS security and offering data for developing a more secure tire pressure monitoring system.

Acknowledgements

The team would like to thank all of those that assisted throughout the span of this project. A special thanks to Professor Wyglinski for all of his guidance and support. Paulo Ferreira, Hristos Giannopoulos, Alexander Arnold, and Travis Collins are acknowledged for their extensive assistance and patience while explaining key concepts and helping with the setup of the software defined radios using MATLAB. In particular, the team would like to thank Sebastián Ortiz for lending his car for the final testing phase of this project. Without their help and support this project would not have been possible.

Authorship

This report was a collaborative effort between all four authors. All members contributed to the research and development of this project.

Table of Contents

List of Figures

List of Tables

List of Terms

ABS - Anti-lock Braking System
ASK - Amplitude Shift Keying
BPSK - Binary Phase Shift Keying
Burst Mode - The *transmission* of data in large continuous *blocks* (bursts), the transmitting *terminal* occupying the whole *channel* during the period of the transmission.
CANbus - Controller Area Network bus
CARB - California Air Resources Board
Center Frequency - The *frequency* used for *broadcast*ing. This normally refers to a radio or television broadcast and the frequencies are tightly regulated to prevent interference between transmitting stations.
CRC - Cyclic Redundancy Check
DAB - Digital Audio Broadcasting
Decimation Factor - An integer or a rational fraction greater than one, which is multiplied by the sampling time. This information is used to down convert the Intermediate Frequency (IF) signal to complex baseband.
DLL - Data Link Layer
ECU - Engine Control Units
EPA - Environmental Protection Agency
Frame Length - The number of bits that are found within a frame.
Frames - A group of *bits* and *bytes* which are collected together in a recognized format for *transmission*.
FSK - Frequency Shift Keying
Gain - A measure of *signal* amplification. It is usually given by the ratio of the output *amplitude* of a signal to its input amplitude.
GPS - Global Positioning System
IP Address - The *address* used to differentiate *users* on the *Internet*. The Internet is designed to use an addressing system based on 32 *bits.*
MQP - Major Qualifying Project
MQP Rx - Arnold and Piscitelli's Receiver designed as part of their MQP last year.
MQP Tx - The Transmitter that the team has designed.
OBD - On Board Diagnostics
Output Data Type - Refers to the type of the output data (e.g. double, precision-floating point, single-precision floating point, 16-bit signed integer).
Platform - Usually refers to the basic *hardware* and *software* elements (e.g. computer, *operating system, relational database*) on which an *application program* is built.
PSK - Pre Shared Key
RDS - Radio Data System
RTR - Remote Transmission Request
RTSA - Real-Time Spectrum Analyzer
Rx - Receiver
SAE - Society of Automotive Engineers

Sample Rate - The number of samples taken in a unit of time.
SDR - Software Defined Radio
TMS - Traffic Message Channel
TPMS - Tire Pressure Monitoring System
TPMS Rx - The receiver that reads the transmitted signals in the vehicle.
TPMS Tx - The four transmitting sensors on the vehicle's tires.
TREAD - Transportation Recall Enhancement Accountability and Documentation
Tx - Transmitter
UDS - Unified Diagnostic Services
USRP - Universal Software Radio Peripheral
Wi-Fi - Wireless Fidelity
WPI - Worcester Polytechnic Institute

Executive Summary

Today's vehicles include more digital components than ever before in order to improve comfort and safety [1]. Examples include infotainment systems, electric windows, key fobs, and even on board Wi-Fi [1]. With such electronic systems as well as an increase in the focus on self-driving vehicles, the threat of hacking these vehicles has increased due to vulnerabilities provided by different access points into the vehicle's electronic network [2]. The most vulnerable access points are the wireless components since they can be intercepted from a distance using the right equipment.

The Tire Pressure Monitoring System (TPMS) is an example of how electronic systems are making vehicles safer. The TPMS consists of four sensors, one on each tire, that send wireless signals to a receiver connected to the vehicle's computer system and provide tire pressure and temperature information. Therefore, the TPMS is one wireless access point to the vehicle's network. Thus, this project aims to expose its vulnerabilities by replicating and decoding the TPMS signal and constructing a Pseudo TPMS Transmitter.

Besides TPMS, the team looked into other potential wireless access points in vehicle's electronic network like the radio data system, key fob technology, in-car Wi-Fi connectivity, and global positioning system. Upon analysis of these potential wireless access points, the team chose the TPMS technology for this project due to the available literature and prior art, ease of testing, sufficient universality, and lack of encryption.

In this project, the team "listened" to the signals transmitted by a TPMS sensor, constructed a fake signal based on the structure of the real signal, and transmitted the mimicked signal using the proposed Pseudo TPMS Transmitter. Initially, the team used a Benchmark SDR (Software Defined Radio) TPMS Receiver constructed by a previous MQP team at WPI in order to eavesdrop on the communication of the TPMS. The team performed a few updates and modifications to the receiver to make it more efficient. The

Benchmark SDR TPMS Receiver used a N210 USRP connected to a computer with the receiver MATLAB code. The signal from this eavesdropping was received through the USRP and decoded in MATLAB to understand the structure of the TPMS signal. After eavesdropping the signal, a thorough analysis was performed to reach some conclusions on the structure of the TPMS signal. For example, the TPMS signal is encoded using Manchester Encoding and transmitted using either ASK or FSK modulation. The team focused on FSK modulation for this project. The knowledge from this analysis of the signal structure was used to create the proposed Pseudo TPMS Transmitter. The purpose of the transmitter is to construct and transmit a “fake” TPMS signal. Ideally that signal would be able to mimic the TPMS signal of an actual vehicle. The signal construction in the Pseudo TPMS Transmitter is shown in Figure A.

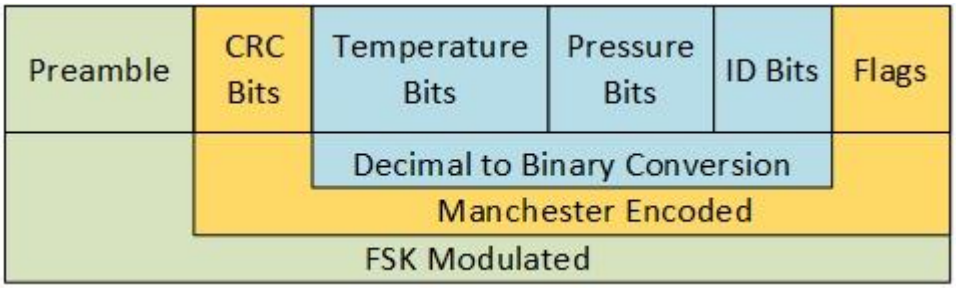


Figure A Representation of the construction of the transmitted signal.

The same N210 USRP was used to transmit the signal which was constructed, encoded and modulated using MATLAB on a computer attached to the USRP. Figure B shows the setup of the Pseudo TPMS Transmitter and the Benchmark SDR TPMS Receiver.

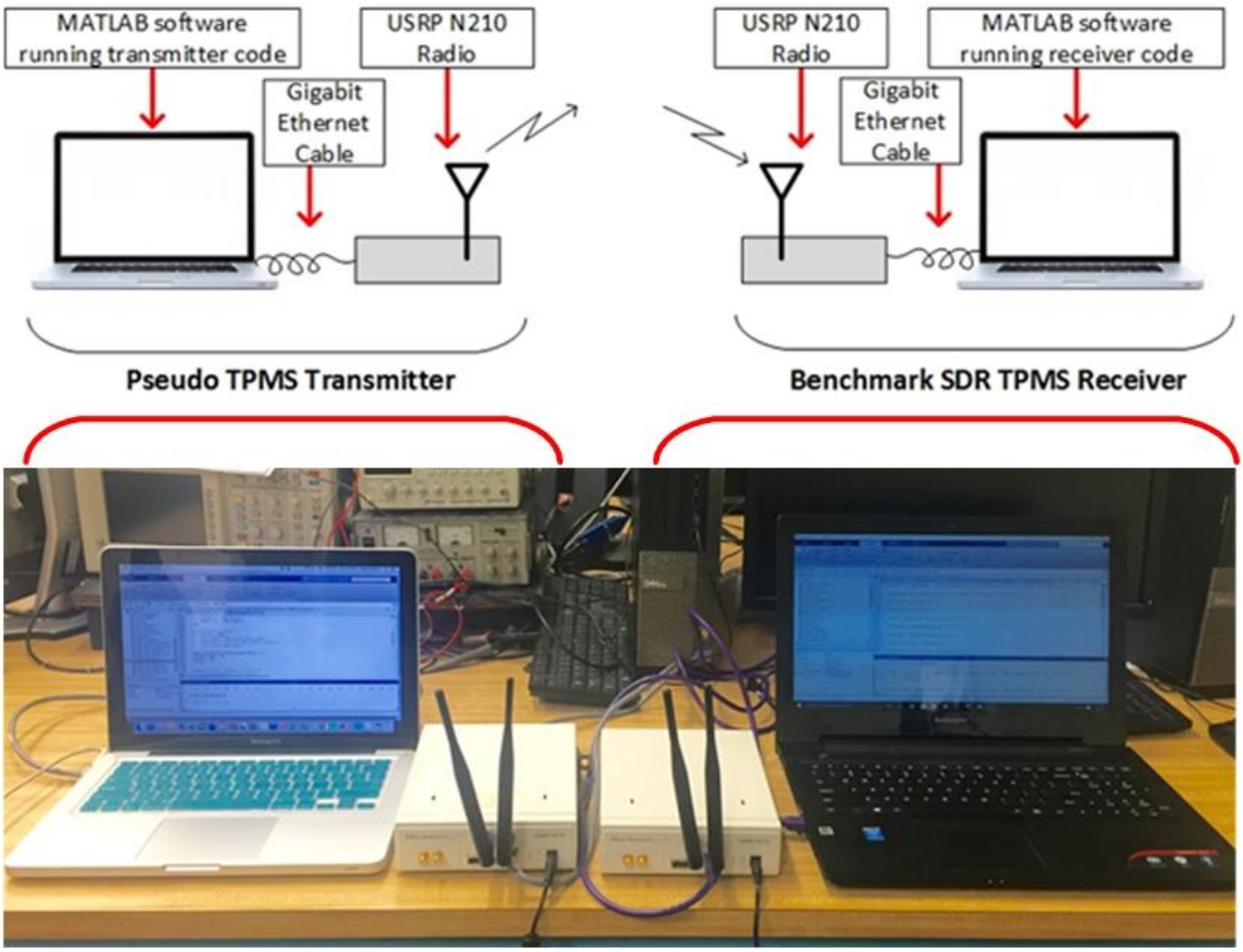
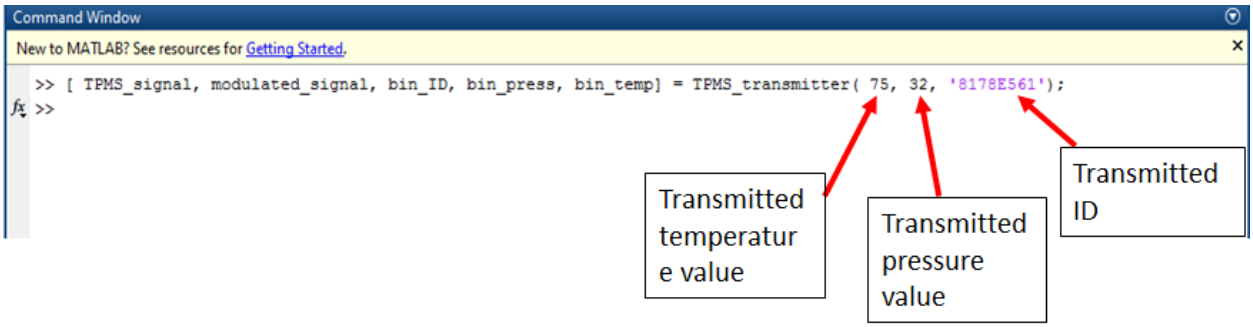


Figure B Schematic and in lab setup for communication of the Pseudo TPMS Transmitter with Benchmark SDR TPMS Receiver testing with USRPs.

Some conclusions that could be drawn from the results of this project were that the Pseudo TPMS Transmitter built by the team was successful in communicating with the Benchmark SDR TPMS Receiver. The Pseudo TPMS Transmitter can send a signal mimicking a real TPMS sensor and that message can be received and decoded by a Benchmark SDR TPMS receiver as shown in Figure C.



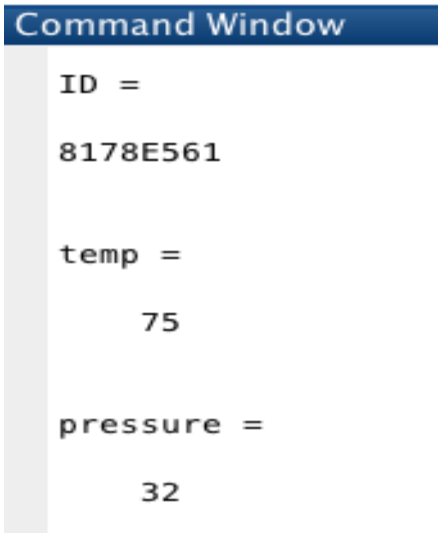


Figure C Shows the signal transmitted and the signal values received, demodulated and decoded.

The same module was tested successful in receiving and decoding an actual TPMS signal as seen in Figure D, proving that the signal sent by the Pseudo TPMS Transmitter is very similar to the “real” signal.

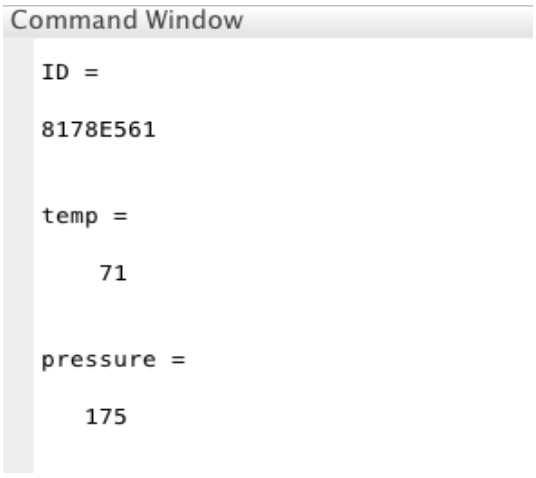


Figure D Inputted values for pressure (32), temperature (75), and sensor ID ('8178E561') transmitted using the Pseudo TPMS Transmitter.

The same signal was tested on an actual vehicle after triggering the TPMS warning light, but it was concluded that the signal construction is manufacturer dependent. Due to this the Pseudo TPMS Transmitter was not able to deactivate the TPMS warning light on the test vehicle's dashboard. Completing the test of successfully sending a Pseudo TPMS signal is a continuation of this project that the team proposes for the future.

To conclude, an important finding of this project related to the TPMS overall is that the wireless communication between sensors and receiver can be intercepted and decoded by using a setup of a USRP with MATLAB code. This fact should be taken into consideration by vehicle manufacturers when designing protection for the electronic systems of a vehicle. If this project managed to decode and recreate a TPMS signal, someone with more advanced equipment and more time and knowledge could take this a step further and insert malicious information in the vehicle's electronic system through the TPMS.

1 Introduction

With the introduction of computers into vehicle technology, vehicles have become more automated and safer. Since their creation, vehicles have been crucial for transportation but nowadays their functionality goes well beyond that. The design of vehicles includes more digital components than ever before. Since vehicles were created, engineers have been finding ways of integrating digital technology into them. Figure 1 shows the growth of the automotive electronic components being integrated into vehicles from 1970 to present day [1]. As seen in Figure 1, the use of electronic components has grown significantly since the 1970s.

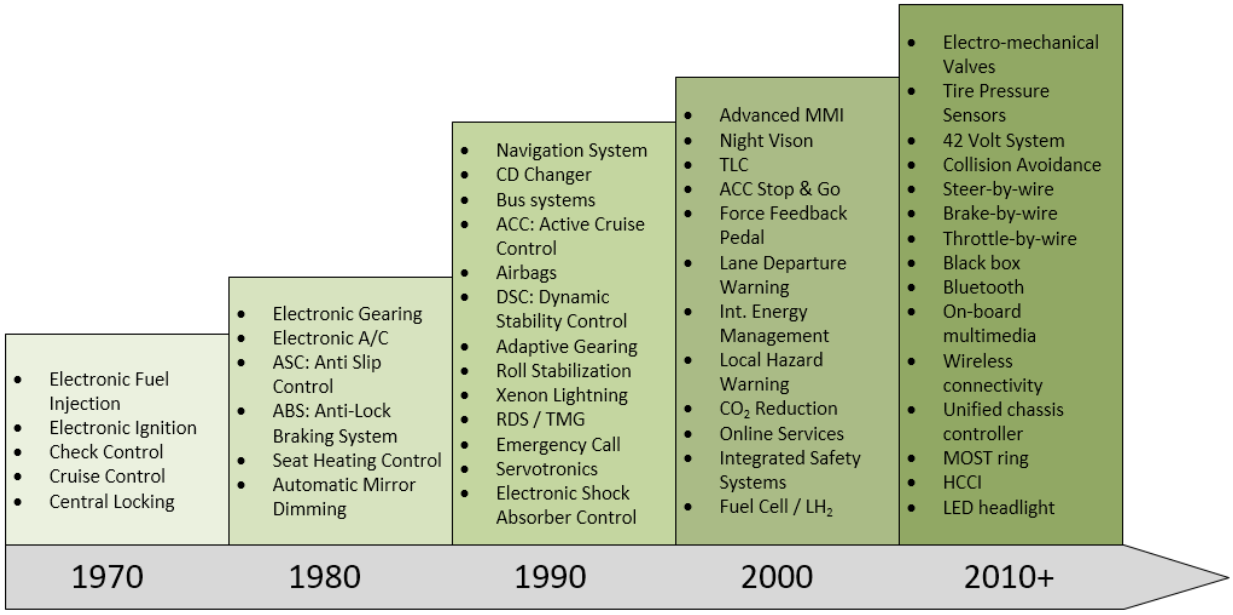


Figure 1 Automotive electronic components integrated into vehicles from 1970 to present day [1].

One of the most important purposes of the added technology in vehicles is the increase of safety while driving. The Tire Pressure Monitoring System (TPMS) is a great example of how technology in vehicles has increased their safety. September 1, 2008 was the last deadline set by the Department of Transportation for car manufacturers to include TPMS in all vehicles released by that date [3]. Undetected low tire pressure was a common factor in 535 fatal accidents between 1995 and 1998 [4]. In 2001, during a study performed by the National

Highway Traffic Safety Administration (NHTSA) in gas stations across the nation, 36% of passenger cars were found with a tire pressure lower than 20% of the required value [4]. In studies performed by the NHTSA, there was a clear correlation between stopping distance when braking and tire pressure [5]. Vehicles with a tire pressure of 30 - 32 psi had the smallest stopping distance in all road materials and conditions tested by the NHTSA. These results showed that accidents can be avoided when vehicles have adequate tire pressure monitoring since their braking stopping distance is directly affected by it.

TPMS uses wireless communication to send pressure and temperature readings from the tires to the vehicle's computer network. Besides TPMS there are a series of systems that use wireless communication in vehicles. Some examples of wireless systems commonly found in vehicles today are Bluetooth, GPS, RDS, OnStar, power locking, and on-board Wi-Fi.

TPMS and the systems mentioned above are great examples of the advancements of vehicle technology. In addition to wireless systems, a recent innovation in electronics includes self-driving cars, such as those from Google [6] and Tesla [7]. Although the introduction of high-end technology enhances the driving experience, it exposes the vehicle to vulnerabilities. The potential access points to the vehicle's computer created by such electronic components could allow a hacker to take control of a modern vehicle. The purpose of this project is to explore possible threats through the TPMS in an effort to gain a better understanding of the vulnerability of such vehicles.

1.1 Current State of the Art

With proper equipment the CANbus can potentially be accessed and manipulated by external sources. This became evident when there was a vehicle hack attempt on a Jeep Cherokee when researchers Miller and Halasek took control of multiple components in the vehicle wirelessly through its entertainment system, including control over the radio, air conditioner, wipers, accelerator, and more [8]. TPMS is another potential access point into the vehicle's CANbus. The TPMS uses sensors to transmit a wireless signal to a receiver, which

can alert the user of low tire pressure [9]. There exists previous literature on accessing a CANbus through the TPMS sensors. For instance, a group of students from Rutgers University performed analyses of the TPMS and they managed to spoof signals in order to send mimicked data [9].

In addition, Worcester Polytechnic Institute (WPI) students Alexander Arnold and Stephanie Piscitelli created a TPMS receiver that communicated with a real TPMS sensor as part of a previous Major Qualifying Project (MQP) [11]. The resulting receiver provides a tool to eavesdrop the wireless signals between the TPMS sensors on the tires and the receiver located in the vehicle from 18 feet away [11]. However, there was not yet an external transmitter capable of sending a TPMS signal to the vehicle that would replace the actual signals from the vehicle's sensors. Arnold and Piscitelli's receiver provides a foundation for this project to build upon.

1.2 Project Contributions

This project focused on creating a TPMS transmitter that could mimic a tire pressure reading and send it to a vehicle's TPMS receiver. The team built a transmitter by using a Universal Software Radio Peripheral (USRP) and coded it using MATLAB software. The transmitter is intended to communicate directly with a vehicle's TPMS receiver and override the real TPMS sensors' transmission. The receiver created by Arnold and Piscitelli was used as a tool to understand the TPMS signal and reverse engineer a transmitter. Once this is achieved the team would be able to send faulty signals that can trigger the tire pressure warning light on the dashboard. The concept diagram in Figure 2 illustrates how the transmitter of this project as well as Arnold and Piscitelli's receiver [11] communicate with a vehicle's TPMS.

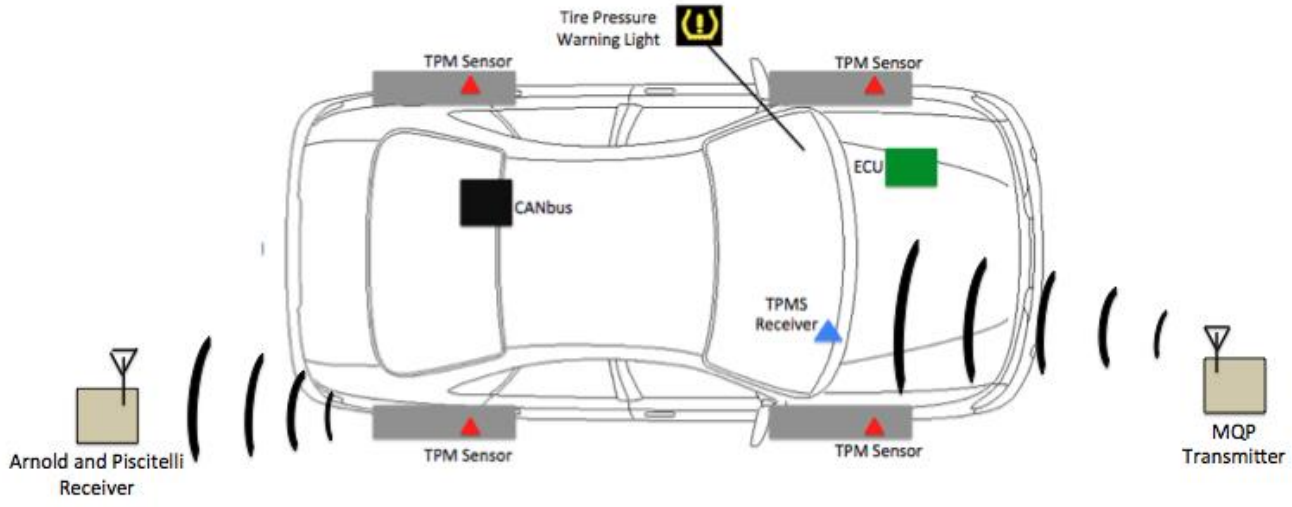


Figure 2 This image is a concept diagram describing the project. There is a transmitter trying to communicate with a real TPMS receiver located in a vehicle's glove box and also Arnold and Piscitellil's receiver picking up signals from real TPM sensors. The image shows the location of the TPM sensors in each wheel, CANbus, TPMS receiver, and ECU.

The project's ultimate goal is to be able to trigger the tire pressure warning light on the dashboard of a vehicle indicating a faulty tire. By accomplishing this, the team would be able to prove that the TPMS sensor wireless technology can be compromised by external forces. Although triggering the warning light does not pose a significant threat to the vehicle's security, it does indicate that other parts of the vehicle's internal network can potentially be accessed. The contributions of this project include:

- Extended research on the communication between an external transmitter and the Tire Pressure Monitoring System of a vehicle.
- A real-world TPMS transmitter.
- A clear understanding of the structure of a TPMS signal.

This project could lead to future research on how to improve TPMS sensor technology in order to prevent external attacks.

1.3 Report Organization

This Section outlines the structure of this report. This report includes the whole process of completing the project such as the research conducted before starting to build the transmitter,

the implementation of the transmitter itself as well as a discussion of the results, thoughts on future work on the topic and conclusions. Chapter 2, immediately following the Introduction, goes over all the background information on different parts of the vehicle electronic systems that the team collected in order to obtain an understanding of the possible paths the project could follow. Chapter 3 discusses the team's proposal for the project as well as an in depth analysis of the potential access points considered. After finalizing the proposed methodology the team went on to the implementation phase of this project. The details of this implementation and the subsequent technical challenges can be found in Chapter 4. The results of the team's implementation are presented and extensively discussed in Chapter 5. Following the results, conclusions are drawn and future work is proposed on the topic of automotive cyber security in relation to TPMS in Chapter 6. References and appendices containing the transmitter and modified receiver code are attached at the end.

2 Information and CANbus Fundamentals

The purpose of this Section is to provide background information in order to be able to understand the contents of this report. This Section covers topics such as CANbus operation, On Board Diagnostics System, potential wireless access points, and software defined radio.

The complexity of the vehicle's communication network inner makings has changed since the first computers were added. All the digital components of a modern vehicle communicate through the Controller Area Network (CANbus). At first computers were added to vehicles in order to make carburetors and fuel injection computerized and hence easier to manage [12]. Today, a CANbus communicates with about 70 computers that manage a plethora of digital components [13]. The CANbus was first introduced to vehicles in 1987 and before CANbus technology automakers had to interconnect every electronic component through wires as illustrated by Figure 3 [14]. The CANbus allows for all electronic components to be interconnected, without the need for much wiring as seen in Figure 4 [14].

Figure 3 This figure illustrates how automakers had to interconnect every electronic component through wires before CANbus technology [14].

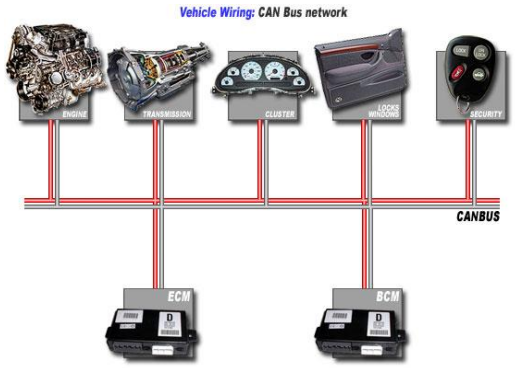


Figure 4 This figure illustrates how the CANbus allows for all electronic components to be interconnected, without the need for much wiring when compared to Figure 3 [14].

The different computers found in vehicles are called Electronic Control Units (ECU). These ECUs manage different digital components including power windows, dashboard warning messages, and cruise control. These components are mostly meant to monitor and analyze the mechanical functions of the vehicle and prompt the user of any impending problems [15], although with the advancement of computers some digital components also have control over the accelerator, airbags, steering, and more [16].

As the CANbus technology evolves, more components use wireless technology to communicate with their designated ECUs. Although the introduction of wireless digital components allows for the reduction of wires and more efficient communication, wireless technology creates vulnerabilities. Introducing wireless digital components to vehicles has made hacking of vehicles popular. The hacking of a vehicle's CANbus can be worrisome due to the control that a person can have of the vehicle's digital components.

One downside to wireless communication is that it creates a potential access point to a vehicle's CANbus [17]. Accessing the CANbus could be dangerous if digital components in the vehicle are altered, thus compromising the safety of the operator. This issue has become more prominent with the introduction of wireless digital components in vehicles and action needs to be taken to understand the technology and find ways to protect vehicles from unauthorized access.

The CANbus operation Section covers the information needed to understand how the CANbus functions and discusses the different frames used in it. The On Board Diagnostics System Section discusses how this protocol works and goes into why it is useful to many mechanics and automakers. The potential wireless access points Section provides information on the tire pressure monitoring system, radio data system, key fobs, in-car Wi-Fi connectivity and satellite. Finally, the software defined radio Section covers how such radios work and also why it would work for the purpose of this project.

2.1 CANbus Operation

CAN Bus (controller area network) is a vehicle bus standard for specialized internal communications network which interconnects different components inside a vehicle. It allows microcontrollers and devices to communicate with each other without a host computer. CAN bus is a message-based protocol designed originally to replace the complex wiring harness with just a two-wire bus [17].

The CAN Bus protocol was developed by BOSCH and officially released at the Society of Automotive Engineers (SAE) congress in Detroit, Michigan, in 1986. Philips and Intel were the first ones to develop the first CAN controller chips, which were made available to the market in 1987. In 2008, SAE required all vehicles sold in USA to use the CAN Bus protocol [14]. The CAN specifications by BOSCH are readily available for free while the ISO standard for CAN has to be bought from ISO.

ISO 11898-1 standard defines the CAN bus data link layer (DLL) protocol and ISO 11898-2 standard defines the CAN bus physical layer protocol. ISO-11898-3 was released later and provides standard for the CAN physical layer for low-speed, fault-tolerant CAN [18]. The layered ISO 11898 standard architecture can be seen in Figure 5.

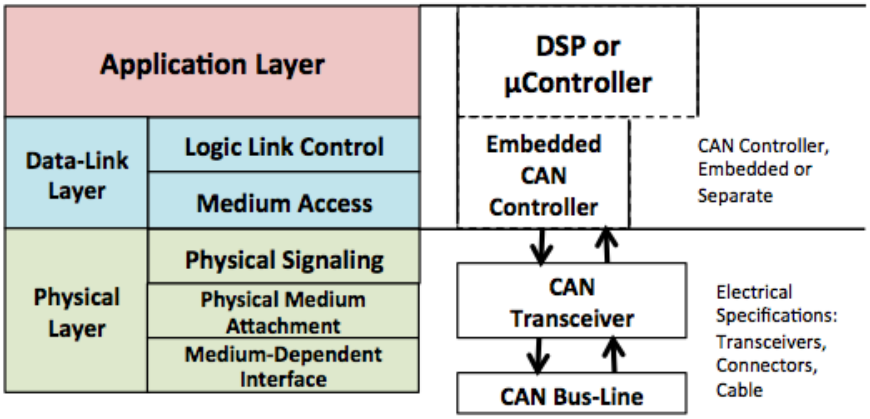


Figure 5 This image shows the layered ISO 11898 Standard Architecture [17].

Following is a summary of the CAN bus protocol defined by ISO 11898-1 standard:

- The physical layer uses differential transmission on a twisted pair wire.

- A non-destructive bitwise arbitration is used to control access to the bus.
- The messages are small (at most eight data bytes) and are protected by a checksum.
- There is no explicit address in the messages, instead, each message carries a numeric value which controls its priority on the bus, and may also serve as an identification of the contents of the message.
- An elaborate error handling scheme that results in retransmitted messages when they are not properly received.
- There are effective means for isolating faults and removing nodes from the bus [18].

In a CAN bus, there are nodes which can be ECUs or microprocessors connected together through two wires. Messages are broadcasted on the bus which go to all nodes and all nodes pick up all transmission traffic. A message cannot be sent to just a specific node but the CAN hardware filters the incoming messages to only react to certain messages. There are four different types of messages according to the CAN standard: the Data Frame, the Remote Frame, the Error Frame and the Overload Frame. A scheme of bit-wise arbitration is used to control access to the bus and each message is prioritized [18].

The most common message type is the Data Frame. It has several parts to it including the arbitration field, data field, CRC field, and the acknowledgement slot. Figures 6 and 7 show the standard CAN data frame and the extended CAN data frame, respectively. The Arbitration Field determines the priority of the message when more than one node is broadcasting message to the bus. The Data Field contains up to 8 bytes of data. The CRC Field has a 15-bit checksum for error detection in the message. Finally, the Acknowledgement Slot is the acknowledgement bit sent at the end of a message when a message is received successfully. If the bit is missing, the message is retransmitted [19].

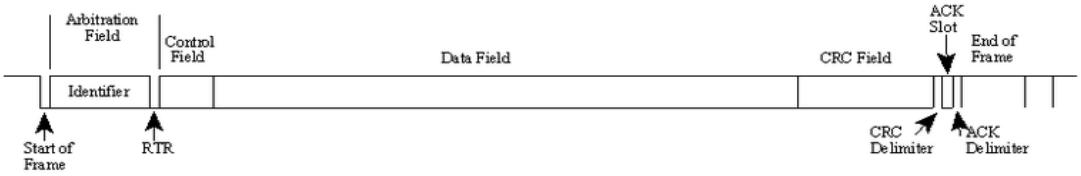


Figure 6 This image shows the standard CAN Data Frame [19].

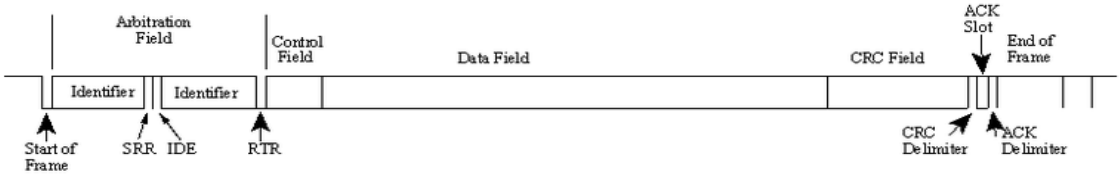


Figure 7 This image shows the extended CAN Data Frame [18].

The difference between a Remote Frame and a Data Frame is that the Remote Frame is marked as a Remote Frame by making the Remote Transmission Request (RTR) bit in the Arbitration Field as recessive. There is also no Data Field in the Remote Frame. The Remote Frame's purpose is to request for a certain Data Frame corresponding to the Remote Frame. This is like a Request-Response type of bus communication [19]. Figure 8 shows a CAN remote frame.

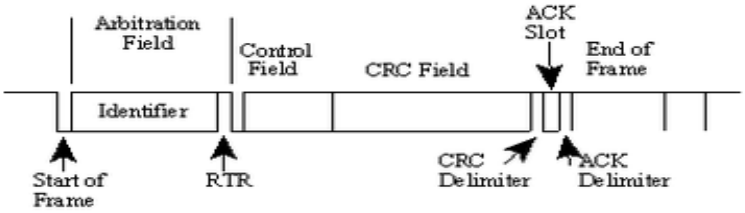


Figure 8 This image shows a CAN Remote Frame [18].

When a node detects a fault, the Error Frame is transmitted on the bus and will cause all other nodes to detect a fault thus, the other nodes will send Error Frames as well. A CAN error frame can be seen in Figure 9. In this case, the transmitter then tries to transmit the message again. It is important to note here that there is a mechanism in place which does not allow the node to ruin the bus traffic by constantly sending Error Frames. There is a 6-bit Error Flag inside an Error Frame with 8 recessive bits of Error Delimiter. When the first Error Flag is detected, the other nodes on the bus can send their Error Flags in the space provided by the Error Delimiter

[19]. The Overload Frame is not used very often. In format, it is very similar to the Error Frame and is transmitted by a node when that node is too busy [20].

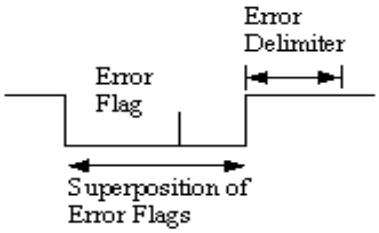


Figure 9 This image shows a CAN Error Frame [19].

2.2 On Board Diagnostics System

The On Board Operating System, better known as OBD-II, is a system used for monitoring emissions and verifying faults occurring in the vehicle. Since January of 1996 it has been mandated that all vehicles have an On Board Diagnostics (OBD-II) system installed in order to meet the emission standards of the Environmental Protection Agency (EPA). OBD-II provides a more universal approach for all manufacturers and technicians to follow. It sets the standards that were approved by the Environmental Protection Agency and the Society of Automotive Engineers to allow for minimum pollution from vehicles. OBD-II provides almost complete engine control while monitoring many different parts of the vehicle including its diagnostic control network through sensors readings [21].

The Environmental Protection Agency required all light duty vehicles and trucks made for sale everywhere in the United States to have OBD-II equipped. They are currently working towards having all vehicles, including heavy duty, to have OBD-II systems equipped. OBD-II allows for monitoring of almost all components that affect the emission performance of a vehicle. It also makes diagnosing and fixing engine control problems easier for technicians. If there is a problem detected, a light will turn on alerting the driver to check the engine [22].

The OBD-II system is a protocol used for reading vehicle parameters and fault codes that provide information on the state of the vehicle. OBD-II is a subset of Unified Diagnostic Services (UDS), which is used by manufacturers and technicians to provide services for

calibration, diagnostics, and flashing firmware. OBD-II uses the First Frame Structure of UDS to send short messages that fit in more than 6 bytes worth of data [23].

OBD-II has 5 different protocols for communication including ISO 9141, SAE J1850 Variable Pulse Width Modulation, SAE J1850 Pulse Width Modulation, and CAN Protocols. Most vehicles will have an OBD-II J1962 connector that you can access the monitored data through. This connector has 16 pins and the different protocols can be read through this connector through different pin configurations. The image and list in Figure 10 illustrate the pin numbering of the OBD-II connector and where the metallic contacts should be for each type of connection [24].

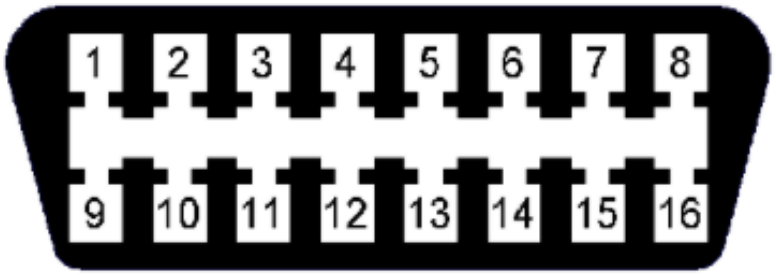


Figure 10 This is an image of the OBD-II connector and it illustrates the locations and sequential order of the 16 pins used to read data [24].

Since 2008, most vehicles follow the CAN communication protocol with their OBD-II systems. An OBD-II CAN connector should have metallic contacts located at pins 4 - Chassis Ground, 5 - Signal Ground, 6 - CAN High, 14 - CAN Low, and 16 - Battery Power. These are the types of connectors the team wants to be reading monitored signals from, particularly those related to the Tire Pressure Monitoring System. OBD-II CAN signals are the same that would trigger an engine light to turn on if something was wrong within the vehicle. Through the data gathered through the connector it is easy to determine exactly where the problem is in the vehicle therefore saving time and money when diagnosing a vehicle. Dealership technicians use the same OBD-II port for diagnostic read-outs including service codes for ignition voltage, cylinder misfires, transmission shift points, and brake conditions [24].

OBD-II was mandated in order to maintain air quality around the world. The California Air Resources Board (CARB) were requiring the use of OBD equipment in vehicles since 1991. By 1994 they had declared that all vehicles 1996 or older will be required to have OBD-II equipment with the Society of Automotive Engineers standards incorporated into it [25].

There are many devices that can connect to an OBD-II port and display the raw codes that indicate the type of problem the vehicle has. Some sort of reference guide for the codes would be needed in order to decipher it. Some devices store the data and it can be transferred to a computer afterwards. The OBD-II port is also being used for other applications such as economy fuel meters and performance computers to access the data being sent by the vehicle during use [26].

2.3 Automotive Wireless Technologies

As mentioned previously, several wireless access points of a vehicle are discussed in this Section, including the tire pressure monitoring system, radio data system, key fobs, in car Wi-Fi connectivity and satellite. The team viewed each one of these as a potential method of hacking into the vehicle and chose the TPMS from amongst them as the most ideal.

2.3.1 Radio Data System

One of the new technologies that create a wireless access point to vehicles is the TMC (Traffic Message Channel) channel of the RDS (Radio Data System). The RDS is the system where all the information displayed on the FM radio are transmitted. Such information includes track name, radio station name, artist name and more. One of the channels, the TMC, includes information on the traffic and accidents that are displayed in the navigation system. Since the information on the RDS are transmitted wirelessly using Radio Frequency [27], it is a possible path for hackers to attack a vehicle. The five steps describing how the RDS system works are shown in Figure 11.

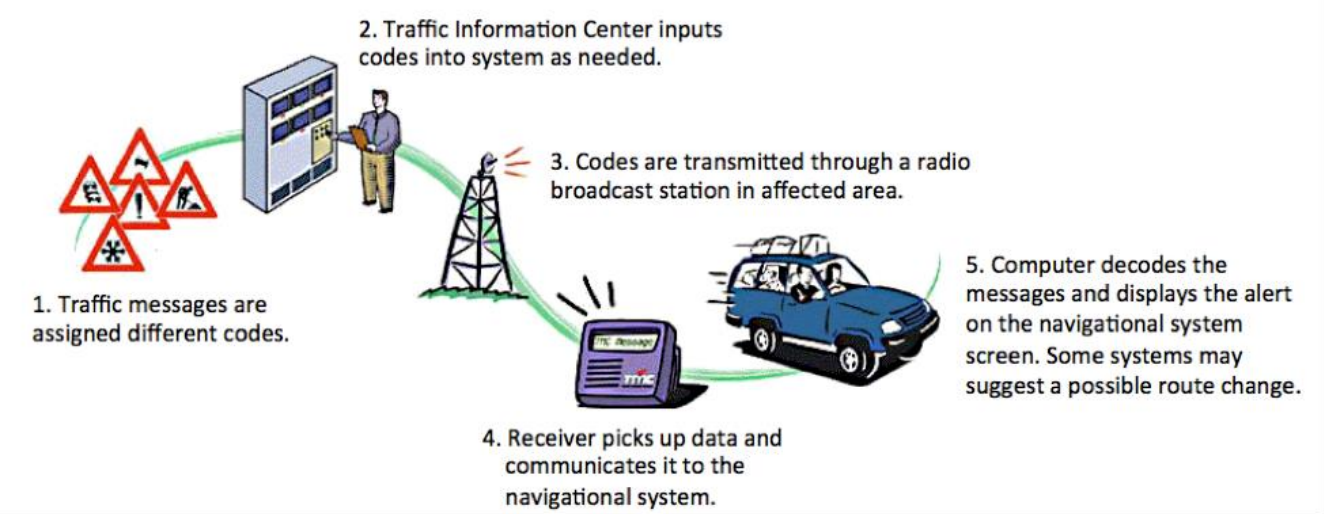


Figure 11 This Figure is a diagram explaining the five steps of how an RDS system works [28].

Such a task was presented in the CanSecWest seminar in 2007 by two Italian hackers [28]. In the presentation, the two hackers from Italy, Barisani and Bianco, talk about how they managed to decode and transmit to the TMC of the RDS system. In the beginning they give some details about how the RDS works (1187.5 bps speed). It is the system that most prominently shows FM channel name, track songs, and other details. It is also used for alternate frequencies, news override and more. It uses frequencies around 57 kHz. TMC uses RDS but can also be transmitted over DAB (digital audio broadcasting) or satellite radio.

Their first step was to sniff a “raw” signal using the FM1216 module by Philips. Then they decoded the RDS subcarrier using a TDA7330B RDS demodulator and a Peripheral Interface Controller. The system then provided them with the binary information of the RDS signal. They managed to decode it using software that they built and managed to find out what bit sequence corresponds to what traffic message. Then they created a transmitter that transmitted their own messages that they coded using the results of the decoded signal.

The two hackers also published a blog post on phrack.com with more detail on their hack [26]. In the post, they go into detail about the structure of the RDS signal (4 blocks of 26 bits each, sampling frequency of ~1118 Hz). They managed to identify in what type of block the

TMC information is stored. In order to translate bits to messages, the Satellite Navigation systems use lookup tables stored in each vehicle's system [27].

The article continues to go over the "sniffing circuit" whose components are: an FM radio with MPx output (PCI video card), an RDS signal demodulator, and an RDS protocol decoder. They use the circuit to read the sequence of bits on their computer and to figure out the series of bits. The bits contain information about the location, the type of message at the location, the direction and more. In the article they publish how each group of bits looks like for many different types of messages.

Barisani and Bianco were not the only ones that encrypted the TMC. One more anonymous hacker claims they have performed the same task in a blog post on windytan.com [29]. The writer of the post explains how the TMC signals are decoded. The messages consist of numerical references to a database with preset sentences and locations. The database is freely available online. The encryption depends on one of 31 keys that changes randomly every night. The key's ID number is transferred in the signal. Then there is an easy algorithm that decodes the bits to a 4-digit number. The way to find the key is by observing persistent messages over days and figuring out how the key changes.

A similar system to RDS is the UConnect, which has also been hacked by Charlie Miller and Chris Valasek in July 2015 [8]. The "Jeep Cherokee hack" caused some controversy since the hackers managed to leverage the UConnect entertainment system, to send commands to mechanical functions of the Jeep, such as the breaks and the engine. They managed to turn off the vehicle, disengage the brakes and perform other more miniscule tasks such as changing the radio station and turning on the windshield wipers. They did all that while the writer of an article on wired.com was inside the vehicle, proving how accessible the Uconnect entertainment channel is.

Overall, TMC-RDS and UConnect have been proven to be wirelessly accessible by hackers that have managed to send messages through them that can cause significant alteration to a vehicle's functionality.

2.3.2 In-Car Wi-Fi Connectivity

Wi-Fi has become a necessity in most American households, workplaces, hotels, restaurants, stores, and more [30]. It was only a matter of time until vehicle manufacturers would seek after introducing Wi-Fi connectivity in vehicles. In-car Wi-Fi has two main functions: personal use of the passengers through their smartphones, laptops and tablets and most important, keeping the car infotainment system up to date. In cases such as Tesla's model S, Wi-Fi is essential for the software updates that keep the electric car running. Manufacturers have introduced built in Wi-Fi capabilities in their newer models, including Audi and GM. Figure 12 shows a visual of a sample screen displaying the Wi-Fi options in an Audi Q5. However, there are more ways to make a vehicle a portable Wi-Fi hotspot, as mentioned in Doug Newcomb's article in the Edmunds website [31].

Figure 12 Sample screen with Wi-Fi settings options in an Audi Q5 vehicle that is currently on the market [31].

The three ways of introducing Wi-Fi in a vehicle are the following: manufacturing a built-in Wi-Fi system, introducing a system that connects in other Wi-Fi networks and uses that connection to create a Wi-Fi hotspot, and connecting a modem that uses cellular networks to

create a Wi-Fi hotspot in the vehicle. The first manufacturer that created vehicles with Wi-Fi capabilities was Audi in 2011 [32]. Their most recent A3, A4, A5, A6 and A7 2016 sedans come with the ability to buy a data plan, similar to that of a smartphone or tablet, that allows for internet connectivity on-the-go using AT&T's 4G LTE network. GM introduced their own Wi-Fi vehicles, followed by Chrysler, therefore opening a whole new market of automotive Wi-Fi connectivity [31]. The second type of Wi-Fi connectivity in vehicles was introduced by Ford and requires the vehicle to be parked somewhere within the range of a Wi-Fi router. The third type is a cellular network to Wi-Fi modem, which is being produced by several manufacturers, can be added in any vehicle's USB or OBD-II port to provide Wi-Fi in the vehicle.

2.3.3 Key Fob

Key fobs are portable devices that allow the operator of vehicles to remotely unlock, lock or open the trunk by pressing buttons located on the fob. Figure 13 shows what a typical key fob looks like. Each key fob contains a controller chip which uses a rolling code. A rolling code means that for each time the receiver and transmitter communicate with each other, there is a different code that is used in order to authenticate that particular transmitter with the receiver in the vehicle. In this particular case, the key fob is the transmitter which contains a bit code stored in the controller chip's memory. This is the same bit code stored in the receiver's controller chip [33].

Figure 13 Example image for what a key fob looks like [34]. Usually these have the standard buttons for unlocking and locking doors, opening the trunk, and panic button.

When the user pushes one of the buttons on the fob, the bit code from the memory in the controller chip is sent to the receiver along with the code telling the receiver what to do (unlock, lock, open trunk, alarm). If the bit code that is sent to the receiver matches the bit code stored in the receiver's controller chip then the vehicle will do whatever the user pressed on the key fob. Once the key fob sends the bit code, a new bit code is chosen by a pseudo random number generator which is then stored in the controller chip's memory. The receiver also uses a pseudo random number generator to store a new bit code in the memory of the controller chip [33].

If the communication between the key fob and receiver is successful, both will have the same new bit code. However, if the key fob is too far away from the receiver, the key fob will get a new bit code because it transmitted a signal even though the communication with the receiver was not successful. To avoid this problem, the receiver will accept any of the next 256 bit codes produced by the pseudo random number generator. The constant changing of codes provides a great deal of security for vehicles. This is due to the fact that hackers would not be able to record one bit code being sent to the receiver and retransmit it to access and unlock the vehicle [33].

Some key fob technology also utilizes passive entry which allows the operator to use the function of unlocking the vehicle wirelessly without physically pressing any of the buttons on the fob. Passive entry systems use a combination of low frequency and radio frequency signals for communication between the key fob and the receiver in the vehicle. The antennas that receive the low frequency signals are located in multiple places on the vehicle including the outside mirrors, door handles, or sometimes on the interior of the vehicle. If the key fob is within two meters of the antennas and the user touches the door handle, the doors will be unlock. This is because when the passive entry ECU receives the message that the user has touched the door handle, the ECU then sends a low frequency challenge signal to the key fob to authenticate it. The key fob then sends a radio frequency signal back to the ECU which if successfully recognized by the antennas, will unlock the doors. This entire communication between the key fob and ECU can be completed within 200 milliseconds [34].

Although rolling codes make it difficult for others to hack into a vehicle through key fobs, it has been proven to be possible. One example of this was done by a digital security researcher, Samy Kamkur. In his experiment, he uses a device called RollJam that is capable of copying the coded signal from the key fob when a button is pressed. This device can be placed under the vehicle so that when the signal is sent from the key fob the device can jam the signal and copy the code. By jamming the signal, the vehicle will not respond to the command from the key fob. This will prompt the user to press the button again. This time the device will record the second signal and send the first one that it had recorded. As a result, the vehicle will respond to the key fob button and the device will still have a valid signal saved so that the hacker can retransmit it later and gain access into the vehicle [35].

2.3.4 Global Positioning System - Satellite Navigation

GPS (Global Positioning System) is a space-based radio positioning system that can offer information about its location in 3 dimensions in real time [36]. It is a wireless technology that has recently been included as part of many vehicles for navigation purposes. Since it uses

wireless technology, it has a wireless access point to the vehicle similar to the ones discussed in this Section. With car manufacturers including GPS technology as an integrated part of the vehicle, and many drivers using it on their phones, it has become a technology targeted by hackers that worked to improve its safety in information transmission.

A team from Chinese Internet Security firm Qihoo 360 led by Lin Huang managed to create a GPS emulator that could transmit their own data to GPS systems in phones and vehicle navigation systems [37]. As the team leader presented in a DefCon conference, the team used inexpensive equipment, such as a Software-Defined Radio, open source code and a HackRF board, a small and cheap board having the capability to interchange between various radio frequencies as well as read and transmit data to a wide range of radio frequencies.

Using the equipment, the team was successful in spoofing the GPS location, showing the wrong location or directing the vehicle to the wrong place. This obviously causes risk hazards for drivers that use GPS since their navigation can be controlled by external transmitters, such as the one designed by Huang's team.

A similar "attack" was performed by Todd Humphreys' team at the University of Texas. They used slightly more costly equipment, including a $3,000 GPS spoofer, a laptop and some antennas [38]. The team managed to take control of a 210-foot luxurious yacht sailing in the Mediterranean by spoofing the on-board GPS with inaccurate information. It was the first time that hackers managed to take control of the GPS driven vehicle and not just interfere with the signal. There is great concern about terrorist attacks that could result from such GPS spoofs. In the same article where Humphreys' spoofing technique is presented, it is discussed that the possibility of controlling vehicles, such as drones which use GPS, would be hazardous for national security.

2.3.5 Tire Pressure Monitoring System

One possible wireless access point into a vehicle is the Tire Pressure Monitoring System. The purpose of the Tire Pressure Monitoring Sensor is to warn the operator if one or

more tires are significantly deflated [39]. In the late 1990's, there was a problem with vehicles that had Firestone tires. When these tires became underinflated, the friction created so much heat that they blew out. The blown out tires caused many vehicles to rollover, which resulted in many fatalities and injuries [21]. In response to this problem the Transportation Recall Enhancement Accountability and Documentation Act was passed as well as a requirement of having a tire pressure monitoring system on all vehicles after 2007 [40].

Tire pressure monitoring sensors are located in each of the four tires of a vehicle and are responsible for monitoring the air pressure in each tire [40]. The diagram in Figure 14 illustrates the location of the tire pressure sensors, emergency light, CANbus, ECU, and the TPMS receiver. When a sensor detects low tire pressure it causes a light to turn on warning the driver of the low readings. The illuminating light is a result of the last steps in the process of an indirect or direct TPMS [39].

This process adheres to the following order of operations:

1. The sensor gathers temperature and pressure readings and communicates these to the TPMS receiver once per minute.
2. The TPMS receiver passes on the data to its designated ECU.
3. The ECU communicates the data to the CANbus.
4. The CANbus analyzes the data and triggers the tire pressure warning light on the vehicle's dashboard if a low tire pressure reading was recorded.

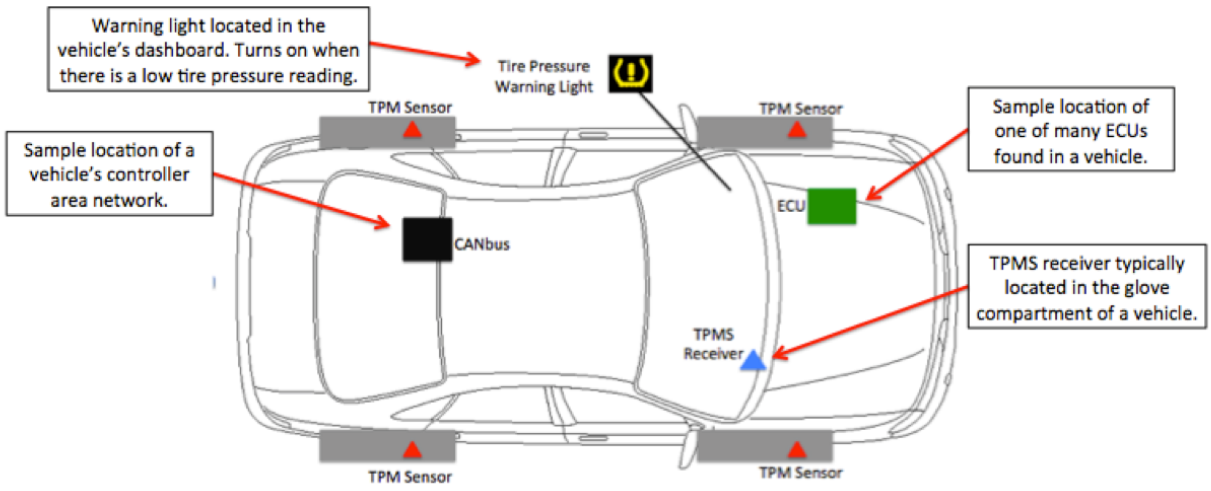


Figure 14 This image is a commented concept diagram describing the TPMS system and the location of its components. It shows the location of the TPMS receiver, TPM sensors in each wheel, CANbus, tire pressure warning light, and ECU.

Indirect TPMS uses speed sensors that are already a part of the Anti-Lock Braking System (ABS) system in a vehicle to analyze whether or not the vehicle's tires have low pressure. As a tire loses air pressure, the diameter of that tire will decrease which will then alter the speed of that tire. These sensors then compare the speed of each tire to each other to determine if the warning light needs to be activated [39].

This approach is cheaper than direct TPMS and causes fewer problems. Another advantage of using indirect TPMS is that when the tires need to be either changed or rotated, there are no special service procedures that are required in order to get the sensors to function properly [41]. One disadvantage to this method is that tire size needs to be taken into consideration every time they are purchased, since consistency is preferred and readings may become inaccurate otherwise [39]. In addition, these sensors are less sensitive than direct TPMS meaning it would take longer for a decrease in tire pressure to trigger the warning light on the dashboard.

Direct TPMS uses pressure monitoring to alert the driver of low tire pressure. In contrast to indirect TPMS, direct TPMS gathers real time data from each of the four tires regarding their precise pressure readings and reports the information as it is gathered [42]. Data is sent from

the sensors to a centralized control module, in other words the TPMS receiver, and once it's analyzed if any tire has low pressure it transmits a signal to turn on the engine light [39]. This data is transmitted wirelessly. Figure 15 shows what a TPMS sensor looks like as well as its location on the tire.

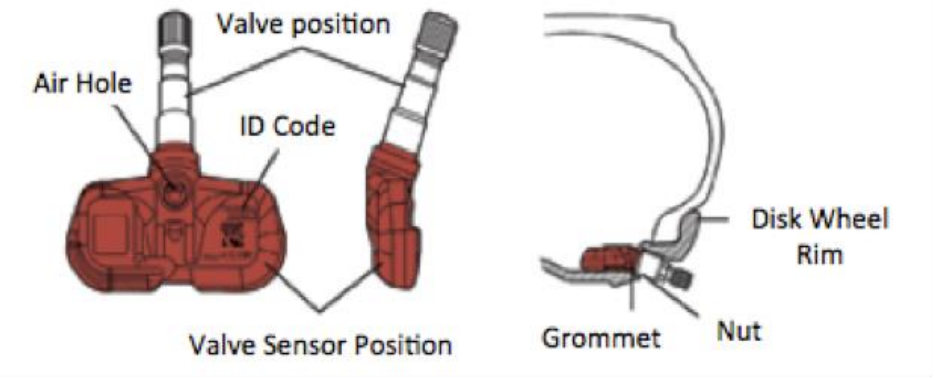


Figure 15 Commented representation of what a TPMS sensor looks like and its location on the tire [41].

Some advantages to using direct TPMS are that this approach is more accurate than indirect TPMS, the battery lasts longer, and it measures actual pressure readings of the tires therefore it's not prone to inaccuracies [39]. Also it is widely preferred by consumers when given the choice between direct and indirect TPMS [42]. The larger preference for direct TPMS makes the chances of a vehicle containing direct TPMS higher, which is more beneficial for the team to explore since there are more vulnerable vehicles with direct TPMS. The main disadvantage to direct TPMS is that it can be very expensive [39].

2.4 Software Defined Radio

An important tool used in Arnold and Piscitelli's TPMS receiver is the software defined radio (SDR). In a software defined radio some or all of the physical layer functions are software defined and its communication technology is based on software defined wireless communication protocols [43]. The SDR cuts down on maintenance costs by making hardware better through software updates instead of physical intervention [44]. A SDR has the following advantageous characteristics:

1. Multifunctional to handle multiple types of radio functions on the same platform.
2. Globally mobile in the sense that it is not confined to standards.
3. Compact and power efficient as it can support several communication standards.
4. It can be easily manufactured because baseband functions are not implemented in the hardware layer.
5. Firmware updates on the SDR platform enable upgrades to support new communication standards [45].

Inside an SDR, several complex interdependent tasks are performed simultaneously for transmitting and receiving data. Figure 16 shows the components of a communication system where you can see that certain components are programmable for a SDR.

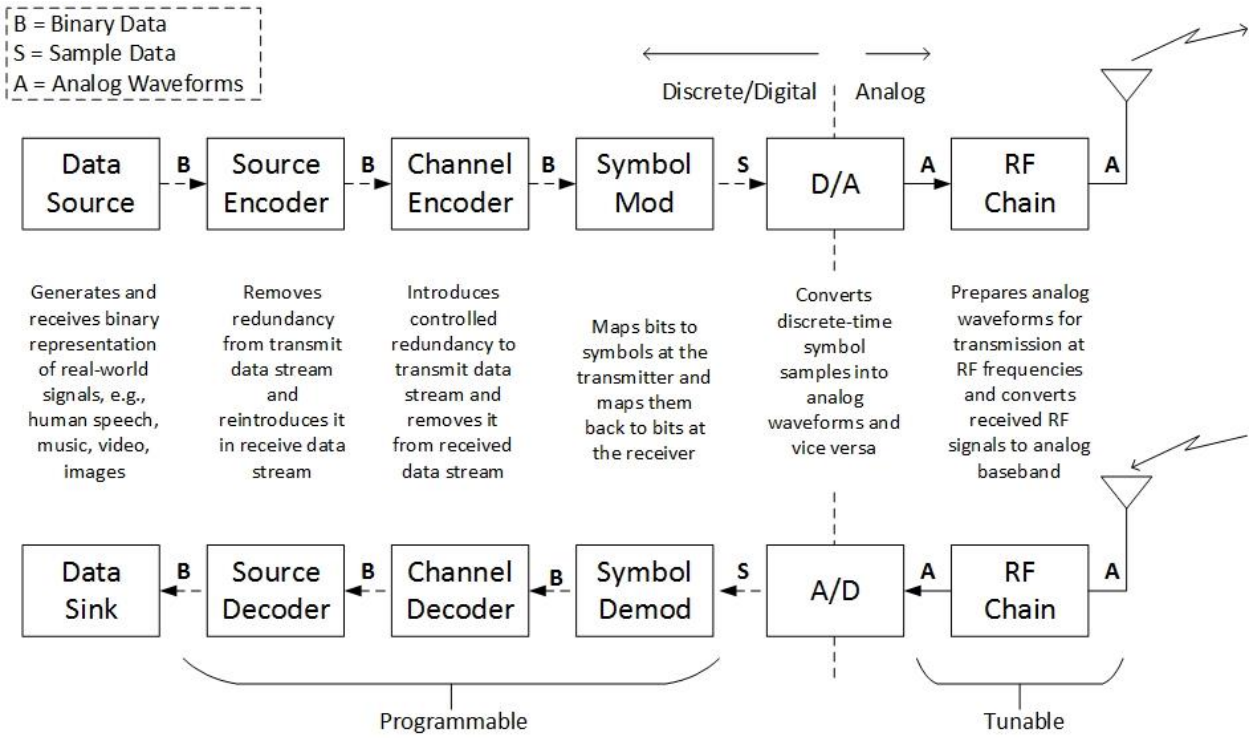


Figure 16 Illustration describing some of the important components that constitute a modern digital communications system [45].

This radio makes the communication between the transmitters and receivers possible and also allows the team to use the 315MHz frequency used for transmitting TPMS signals. Since TPMS involves radio frequency, a real-time spectrum analyzer (RTSA) could be used as

a tool for radio frequency measurements [46]. The RTSA is used to measure and analyze the modulation rate and quality between the transmitter and receiver [46].

For the purpose of this project, an Ettus Research USRP (Universal Software Radio Peripheral) N200 was used with a WBX daughterboard with a frequency range of 50-2200 MHz. This frequency range works for the team's interaction with the TPMS working frequency of 315MHz. The USRP is interfaced with MATLAB on a computer to program the USRP as per the requirements of the application. Figure 17 shows the basic setup of SDR interfaced with a computer with an ethernet port connected to both hardwares.

To use the USRP with MATLAB, follow these steps:

1. Download Matlab
2. Download the "Communications System Toolbox" which includes the support for USRP
3. Connect the USRP to the computer through Ethernet cable
4. Change the computer's IP address to correspond to the USRP's IP address. For example, if the USRP's IP address is 192.168.10.2 then the computer's IP address should be 192.168.10.X where X is any number but 2.
5. In MATLAB, give the command "findsdru" to check if it detects the connected USRP.

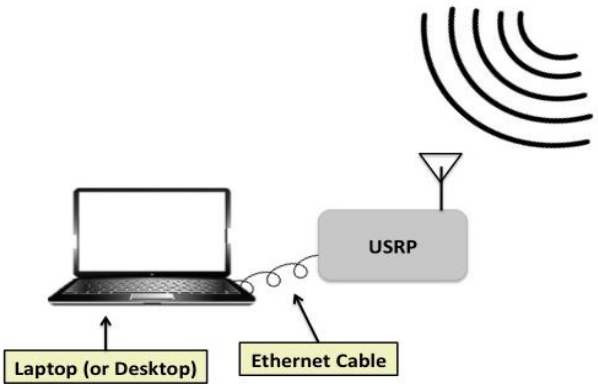


Figure 17 This image is a basic setup of SDR interfaced with a computer through the Ethernet port of both hardwares.

The successful USRP setup will allow for the manipulation of all the important variables of wireless communication. Such variables include gain, center frequency, decimation factor, interpolation factor, IP address, samples per frame, frame length, *etc*. A USRP is a flexible tool

that can be used for any project regarding wireless signals and can act as a transmitter, a receiver, or both simultaneously.

2.4.1 Modulation

Modulation refers to the translation of a binary signal to an analog wave in order for it to be transmitted through a channel. In wireless communications, the channel is always the air and therefore the last step of the transmission will always need to be in the form of an analog wave, even though the majority of the processing is nowadays digital. In order to use the USRPs for wireless communication, one has to have a good understanding of modulation and its different forms.

The most common modulation forms are Amplitude Shift Key (ASK) modulation, Frequency Shift Key (FSK) modulation and Phase Shift Key (PSK) modulation. What differentiates those three modulations is the physical quality of the waveform that changes according to the binary bit sequence. In ASK, different combination of bits translate to different amplitude levels in the wave. In FSK modulation, different frequencies are used to depict the combinations of bits. Finally, PSK uses phase shift in order to translate bits [48]. Figure 18 graphically demonstrates how a sequence of bits is translated in a sine wave using amplitude, frequency or phase shift key modulation.

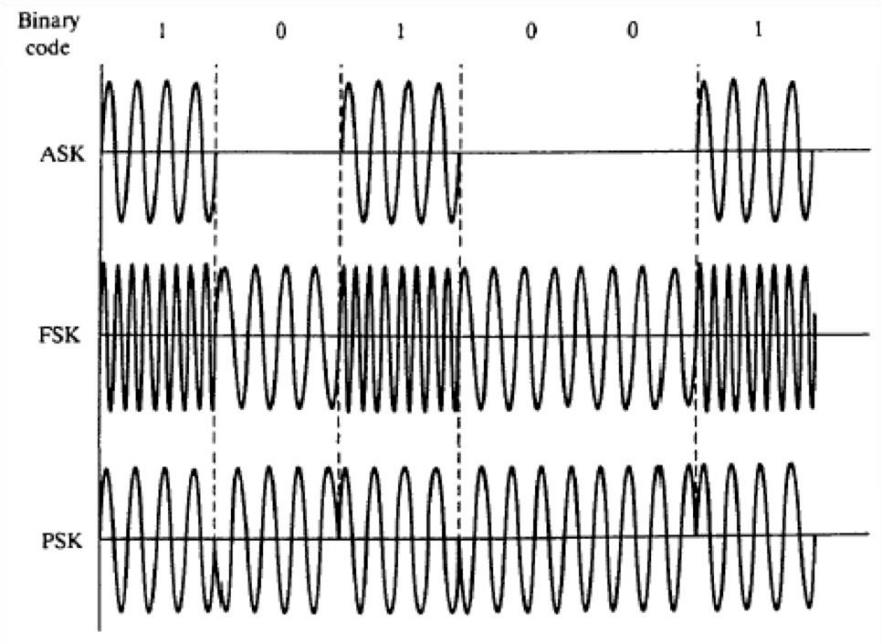


Figure 18 Graphical representation of ASK, FSK, and PSK modulation [49].

Using Figure 18 as an example, as a result of ASK modulation, a sine wave is created where “1” is translated to a specific amplitude and “0” to a wave with an amplitude of 0. FSK modulation uses two different frequencies to create a waveform that translates “1” with a higher frequency wave form than “0”. Finally, PSK uses a waveform where amplitude and frequency are constant and the phase changes when the binary input changes from “1” to “0” or vice versa.

Transmitters can be designed using BPSK (Binary Phase Shift Key) modulation instead of other modulation techniques, such as ASK and FSK, which are typically used by radio frequency identification technologies. BPSK is a form of phase shift keying, which uses two phases that are separated by 180° as seen in Figure 19.

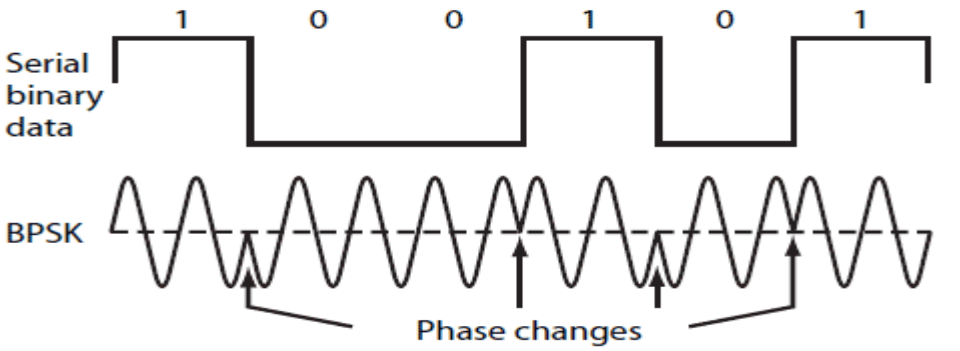


Figure 19 A BPSK modulated signal is displayed in this image for a serial binary data [50].

2.5 Chapter Summary

This chapter contains background information to facilitate the understanding of certain aspects of this project. Car hacking has increased significantly alongside the rapid development of vehicle technology and more specifically the introduction of wireless communications. While researching the wireless access points to determine the one that is the most appropriate for this project, the team encountered a number of attacks that have been mentioned in previous parts of this paper. The attacks are summarized in Table 1.

Table 1 Summary of wireless attacks on different systems of vehicles.

Wireless Access Point	Attack
TPMS	Rouf et al. , Rutgers University [10]
RDS	Barisani and Bianco [28]
Jeep Cherokee Entertainment System	Miller and Valasek [8]
GPS	Lin Huang [37]
GPS	Todd Humphreys [38]

The internal electronics of a vehicle are interconnected through a communication network utilizing the standard called CANbus. The CANbus employs a message-based protocol

without a host with a two-wire bus. The CAN standard specifies four different types of messages: the Data Frame, the Remote Frame, the Error Frame and the Overload Frame. The OBD-II system is a protocol used for reading vehicle parameters and fault codes that provide information on the state of the vehicle. OBD-II is a protocol used to monitor emissions, read vehicle parameters and determine fault codes of a vehicle. Most vehicles' OBD-II systems are compatible with the CAN communication protocol since 2008. This chapter also discussed wireless access points within a vehicle's communication network. Some of the potential access points are the Tire Pressure Monitoring System (TPMS), the Radio Data System (RDS), In-car Wi-Fi Connectivity, Key Fob and Global Positioning System (GPS). A basic understanding of the Software Defined Radio (SDR) was presented in this chapter, in which software is used to define some or all of the physical layer functions. Finally, the chapter briefly outlines a basic understanding of the most common modulation forms: Amplitude Shift Key (ASK) modulation, Frequency Shift Key (FSK) modulation and Phase Shift Key (PSK) modulation. The team utilized the information presented in Chapter 2 to design the methodology and implement the project.

3 Proposed Design

This Section outlines the team's proposal for the project. It covers topics such as the analysis of potential access points and the logistics of the project. The team's ultimate goal is to identify a wireless access point that can allow communication with a vehicle's CANbus. In order to identify the most suitable technology, the team analyzed advantages and disadvantages of all the wireless access points discussed in chapter 2 (TPMS, RDS, Key fobs, In-Car Wi-Fi and GPS). In the end of this chapter, the logistics Section covers the team's proposed testing phase for the chosen wireless access point.

3.1 Analysis of Potential Access Points

This Section discusses the pros and cons of each of the potential access points. These access points all communicate wirelessly with the vehicle CANbus, which is why they were chosen as the potential access points to be evaluated. The access points include TPMS, RDS, key fob, In-Car Wi-Fi connectivity, and GPS.

3.1.1 Tire Pressure Monitoring System

The tire pressure monitoring system was considered as a potential access point because of the large amount of research available on the topic. Several studies have been made available since the tire pressure sensors were made mandatory for all vehicles stating that it is possible to send messages to the CANbus through these sensors. There are known ways of reading TPMS signals using software defined radios, antennas, and surface acoustic wave (SAW) filters. Researchers in Rutgers University and University of South Carolina have achieved spoofing already [9]. The team theorizes that sending a mimicked signal to the CANbus might be possible if the mimicked signal is stronger than the signal being sent by the tire pressure sensor in the tires. Through this method they would be communicating with the CANbus wirelessly.

TPMS works with radio frequency at a frequency of 315MHz [41], which the team can use the available software-defined radio for. These sensors communicate directly with the ECU

which in turn communicates with the CANbus, making these sensors a viable way of achieving the project's end goal of sending a signal to the CANbus. In the schematic below, one can see the layout of the TPMS communications between the sensors, the receiver and the display in the dashboard of the vehicle.

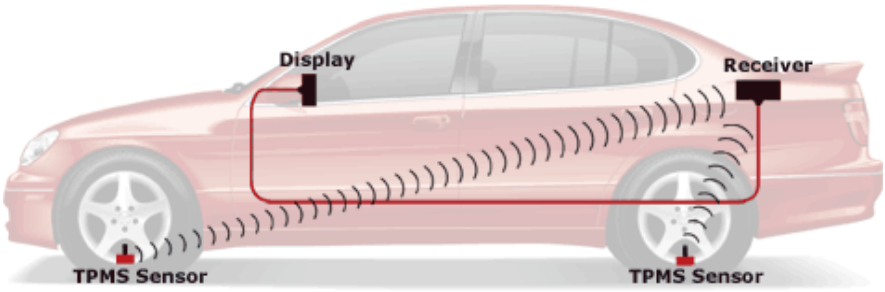


Figure 20 TPMS architecture with receiver at the back of the vehicle [51].

In order to intercept the signal demonstrated in figure 20, the team can build upon Arnold and Piscitelli's TPMS receiver to create a working transmitter.

However, the team believes that the length of the TPMS signals might be too short to allow for complete access to a vehicle's ECU, let alone CANbus. This technology is mandatory for all vehicles but not everything about it is universal. There are many different types of sensors to account for. The team would have to take into account the hundreds of variations in TPMS protocol in order to create a universal technique that would work for all vehicles. The technique chosen for the project might not work for other vehicles or sensors apart from the one that the team programs for.

3.1.2 Radio Data System

The radio data system (RDS) was considered a potential access point because longer signals could be injected into the CANbus than through TPMS or key fobs. As mentioned in the background Section, RDS works over FM broadcast. Since RDS uses FM broadcast the team could gain access into the system by using a frequency that is not currently being used by a radio station in that specific area. The software defined radio needed to achieve this is available for the team to use. Another approach to hack into the RDS would be to go through an

unencrypted traffic message channel (TMC) [27]. From prior work done by Barisani and Bianco the team knows that it is possible to change information in the navigation system of a vehicle [27]. For example, Barisani and Bianco managed to display their own custom made messages in the navigation system of a vehicle they performed tests on as seen in Figure 21.

Figure 21 Barisani and Bianco's "fake" alert about air raid [28].

Their work shows that it would be possible to hack into the navigation system but it is unclear whether it is possible to gain access to other systems.

While researching RDS there were some concerns with the idea of choosing it as the access point. Some of the problems with RDS are that there is less literature available for research, fewer experiments have been conducted compared to TPMS and key fobs, and only newer vehicles have RDS TMC. Since only newer vehicles have RDS TMC, they would be the only ones that could be sent wrong traffic information.

3.1.3 Key Fob

Key fob technology works with radio frequency. They work in several low frequency bands depending on the manufacturer, which the team can use the available software-defined radio for. The technology has an authentication protocol where the receiver sends a challenge signal, which is authenticated by the paired fob [33]. The key fob then sends back another signal to complete the authentication. The authentication process works with rolling codes, which means each interaction is different in terms of codes [33]. This technology is quite universal in terms of the frequency they use and the way they work in different vehicles.

While researching prior work related to hacking through key fob, it was found that generally people have been able to unlock the vehicle doors for many vehicle manufacturers. There is a paper from Netherlands on Megamos Crypto, a main key-fob component used by major car manufacturers, which mentions how the researchers decrypted the codes for many popular manufacturers [52]. The Megamos Crypto transponder chip can be seen in Figure 22.

Figure 22 The Megamos Crypto transponder chip picks up the key fob signal and decodes it [54].

The team would have to sniff the interaction between a vehicle and the key fob and then be present near the vehicle in order to send all possible combinations of the codes until eventually one works. A researcher from Australia successfully used this technique with software-defined radio and equipment costing less than a $1000 [53].

Although the technique is pretty universal, such techniques have been used to only unlock the doors of a vehicle but not hack it. The team decided not to pursue this method. The rolling codes are an issue due to their difficulty of decrypting. This method takes time to complete and the hacker, along with the equipment, needs to be near the vehicle when the owner is locking or unlocking the vehicle. The past literature shows that the vehicles are only being unlocked to be stolen with this access point and not hacked.

3.1.4 In-Car Wi-Fi Connectivity

In-Car Wi-Fi connectivity can be offered in three different ways, as explained in chapter 2. The three ways are by having a built-in Wi-Fi system, by connecting to Wi-Fi hotspots in the surroundings and transforming those to their own hotspot, and by using the cellular network and creating a Wi-Fi hotspot for inside the car. As one can imagine, connecting a vehicle's electronic system to the internet, whether that is through the cellular network or through another Wi-Fi

network, opens up ways for external manipulation of the vehicle. The Radio Data System (RDS), using 3G cellular network, has already been compromised by Barisani and Bianco [27], proving that any connection of a vehicle to a cellular network can make the vehicle vulnerable. In addition, the case of the Jeep Cherokee hacking through their infotainment system [8], also connected to the cellular network, confirms the danger with using cellular networks to connect to the internet in a vehicle.

There has not been research conducted in the vulnerability of cars with built-in Wi-Fi capabilities to show whether that system is prone to hacking or not. The reason is that there are few manufacturers selling cars with built-in Wi-Fi and those are very recent (after 2011). If this wireless access point was to be chosen to explore in this project, the team would have to test on a very new and possibly expensive vehicle. Additionally, the literature on the specific wireless access point is very limited. The method of connecting to Wi-Fi networks in the surroundings is also very immature and lacking literature reviews. For all the reasons stated above, the team decided not to pursue in-car Wi-Fi connectivity as a wireless access point for this project.

3.1.5 Global Positioning System - Satellite Navigation

The global positioning system (GPS) was considered a potential access point because these systems communicate wirelessly with vehicles through satellite. There is some research available on the topic of hacking through GPS. A team from a Chinese Internet security firm created a GPS emulator that could transmit their own data to GPS systems in phones and vehicle navigation systems [37]. Also, a team of students from the University of Texas achieved a similar attack. The team of students was successful in spoofing the GPS location [38].

Although the technique has some documentation behind it proving that it is possible it may not be the best option available for the team. One downside to this method is that most GPS users use this technology on their phones. Data has been released stating that the sale of Garmin navigation systems has decreased by 5% in 2014 alone and had been on a steady decrease since 2009 [55]. This means that the team's attack would only work for vehicles that

have a GPS system integrated into them. Apart from this, the team would need to do more research in order to understand how hacking through GPS is achieved in order to have a better foundation to build a project upon.

3.2 Access Point Selection for MQP

After analyzing the different wireless access points available, the team decided that the most feasible one for this project is the tire pressure monitoring system. The team found that the other access points such as RDS, key fob, GPS, and In-Car Wi-Fi had deficiencies that would be more difficult to address. For instance in TPMS the communication is not encrypted unlike for key fobs, thus making it easier to create a transmitter. Although RDS technology seemed as a great possibility for this project there is not sufficient literature available for the topic unlike for TPMS technology. The lack of literature meant the team would have to invest more time trying to understand the technology as opposed to jumping into the project. GPS technology has shifted towards cellphones as opposed to in vehicle systems, therefore not compromising enough cars. Finally, In-Car Wi-Fi connectivity is very new and only available for a few cars on the market, therefore there is not enough literature on the topic and very few vehicles could be affected if this technology was compromised. Table 2 summarizes the advantages and deficiencies of the 5 evaluated access points.

Table 2 Considered access points advantages and deficiencies.

Access Point	Advantages	Deficiencies
TPMS	• Previous literature • Non-encrypted signal	• Short signal
RDS	• Previous work has accomplished hack	• Only applicable in very new vehicles
GPS	• Documentation shows that it is possible	• Most people use their phones for GPS
Key Fob	• Works for several low frequency bands • Universal	• Encrypted signal • Rolling codes
In-Car Wi-Fi	• Sometimes uses the cellular network, which has been hacked (RDS)	• Only applicable in very new vehicles • No previous literature

The reasoning behind choosing TPMS was mostly influenced by the vast amounts of research available on the topic. Also, since Arnold and Piscitelli have already built a working TPMS receiver, the team feels there is a good foundation for the creation of a TPMS transmitter using MATLAB software. TPMS is present in a vast majority of vehicles today, thus making it a serious vulnerability if the TPMS technology were to be compromised.

3.3 Project Logistics

The following chapter describes in detail the implementation of the project. This Section outlines the tests the team performed to assess the Pseudo TPMS Transmitter's functionality and capacity of interfacing with a vehicle's TPMS Rx.

The team used the structure of Arnold and Piscitelli's TPMS receiver (MQP Rx) [13] in order to reverse engineer a TPMS transmitter (Pseudo TPMS Transmitter). Upon successful completion of building the Pseudo TPMS Transmitter, the team could then begin testing. The first test that was executed was between the Pseudo TPMS Transmitter and the MQP Rx. This test was done to ensure that the signal being transmitted using the Pseudo TPMS Transmitter

could be properly decoded when received by the MQP Rx. The following test was between the Pseudo TPMS Transmitter and Rx with a USRP interface. In this test, the team would essentially replicate the first test but using two USRPs as the transmitter and receiver. Next, the team would run a test between the MQP Rx and a real TPMS sensor (TPMS Tx). Within this test, the team would alter the MQP Rx so that it could successfully decode a real transmitted signal from the TPMS Tx. Finally, the team used the Pseudo TPMS Transmitter and a real TPMS receiver (TPMS Rx) to attempt to successfully use the team's transmitter to communicate with a TPMS Rx in a sample vehicle.

Table 3 Project Logistics Timeline

Date	Milestone	Deliverable by Date
August 29, 2015	Team formation	
September 15, 2015	Decision on TPMS topic	Project Proposal Mid-Term Presentation
October 13, 2015	Preliminary Pseudo TPMS Transmitter Design, Simulation Test Successful	Project Proposal Presentation and Draft
October 28, 2015	Beginning of testing	
November 24, 2015	Test with USRP Interface Successful	Testing Mid-Term Presentation
December 11, 2015	Test with TPMS Tx in tire Successful	
December 15, 2015	Final Transmitter Design Final Receiver Design	Final B Term Presentation
December 17, 2015		Tests Implementation and Results Draft
January 14, 2016	Start Final Paper writing	
January 26, 2016		Introduction, Background, Proposal Drafts for Final Paper
February 3,	Field Test I with Ford Fiesta	

2016		
February 16, 2016		Implementation, Results, Conclusion Drafts for Final Paper
February 20, 2016	Field Test II with Ford Fiesta	
February 26, 2016	Field Test III with Ford Fiesta	
February 27, 2016		Final Paper Draft Submission
March 2, 2016	Field Test IV with Subaru	
March 4, 2016		Final Paper Submission

3.4 Chapter Summary

This chapter discusses the team's reasoning behind picking the tire pressure monitoring system as the access point into a vehicle as well as the logistics of this project. Possible access points such as RDS, TPMS, key fob, GPS, and In-Car Wi-Fi connectivity were analyzed. The testing goals stated in this chapter were used to assess the functionality and capacity of the TPMS transmitter created.

4 Implementation of a Pseudo TPMS Transmitter

This chapter discusses the construction and testing of the proposed Pseudo TPMS transmitter. The first Section focuses on the team's analysis and understanding of Arnold and Piscitelli's TPMS receiver implementation (Benchmark SDR TPMS Receiver) [11]. The team then uses the results of this analysis to reverse engineer the Pseudo TPMS Transmitter. The third Section covers the different tests performed during the second term of the project. These tests are essential in evaluating the functionality of the design of the pseudo TPMS transmitter. This Section includes the testing of the Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver, Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver with USRP Interface, replicating results from the Benchmark SDR TPMS Receiver and the TPMS sensors, and testing the transmission from Pseudo TPMS Transmitter to an actual TPMS receiver.

4.1 Benchmark SDR TPMS Receiver

In [11], Arnold and Piscitelli created a TPMS Receiver using MATLAB software and an SDR platform that can read TPMS signals being transmitted by TPMS sensors. Their Benchmark SDR TPMS Receiver possessed six main steps that it performed in order to decode the data being received. Figure 23 illustrates a block diagram for the Benchmark SDR TPMS Receiver, which consists of the six main blocks: concatenating, reformatting, creating waveform, demodulation with FSK or ASK, and decoding.

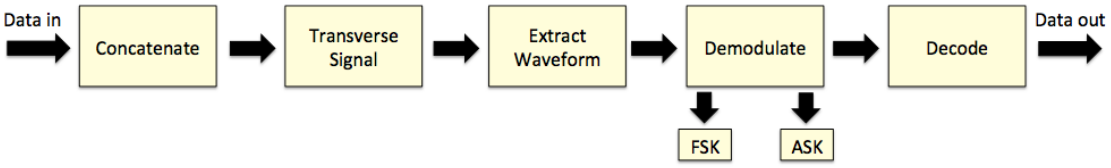


Figure 23 This is an image of a block diagram for the Benchmark SDR TPMS Receiver. It has 5 main blocks, these being: concatenate, transverse signal, extract waveform, demodulate with either FSK or ASK, and decode.

The first block is labeled 'Concatenate', this means that the receiver takes the input and concatenates a multidimensional array into a single dimension. The concatenation takes place by calling the *TPMS_concat()* function, found in Appendix A, Section 7.8. The function takes as

input the signal in form of a multidimensional array, and after a series of commands transforms it to a single dimension array without losing any of the information. This step is essential because the rest of the processing requires a different format of the signal, which starts from converting it to a single dimension array.

After the signal has been concatenated into one line, the next step is to transverse through the received signal. The purpose of this step is to identify the TPMS signal for further processing. This process is performed using power thresholds defined in the beginning of the code based on the bandpower of the received signal. The signal is broken down to sets of 10 points at a time and bandpower of those is compared to the power threshold. If their bandpower is high enough, they are recognized as part of the TPMS signal; otherwise the receiver moves on to the next 10 points. The MATLAB code for this can be found in Section 7.2 of Appendix A.

The next block, 'Extract Waveform', also serves to identify the signal of interest out of the received signal. At this point, the received signal is a modulated waveform. The purpose of this block is to extract the parts of the waveform that correspond to the TPMS signal from the noise that was also received at the same time. Once it extracts the signal, it creates a waveform like the one seen in Figure 27. This process can be found in Section 7.1 of Appendix A. Creating the waveform is essential in recognizing the modulation of the signal in the next block.

The creation of the waveform allows the receiver to move into the demodulation block. In this block the receiver can choose to demodulate using FSK or ASK techniques. To choose between FSK and ASK demodulation the program first takes the Fourier transform of the signal in order to determine the maximum values in the signal. Then it checks to see how many peaks there are in the signal and saves their index values. An FSK signal has two distinct peaks in its waveform, whereas an ASK signal has only one. The peaks of FSK and ASK waveforms can be seen in Figures 38. Once it checks for the peaks, it calculates whether or not the peaks are within range of each other. If two peaks are found within range of each other, the code will

choose FSK demodulation otherwise it will choose ASK. Finally, whether the program chooses FSK or ASK demodulation, it will down sample.

The team focused on FSK demodulation due to the Benchmark SDR TPMS Receiver's capability of only demodulating an FSK signal in MATLAB. For the Benchmark SDR TPMS Receiver, Arnold and Piscitelli looked at the time domain of the FSK signal from the TPMS sensor. As FSK encodes information in the signal using frequency changes, there were two frequency peaks in the frequency domain of the received signal. The two peaks were at -35.645 KHz and 38.089 KHz from the center frequency. The signal was manipulated to make decoding easier by shifting the received signal's frequency to the right to bring the negative frequency to 0 Hz of DC frequency. This resulted in a frequency-domain signal shown in Figure 24 and time-domain signal shown in Figure 25. This gives a signal where the 1 bits are represented by high frequency 73.73 KHz and 0 bits are represented by 0 Hz frequency [11].

All this work is done by the demodulation block of Figure 23. This block finds the frequency separation between the two peaks along with number of samples per symbol and sample rate, and demodulates it into bits using the predefined MATLAB function *fskdemod()*. Then, this block down samples based on repetitive bits and move on to the last phase of the Benchmark SDR TPMS Receiver which is decoding.

In the decoding block, the receiver takes the Manchester encoded signal and converts it into data. Manchester encoding is when data is encoded into a two bit symbol for transmission. There is a negative to positive transition in the middle of a bit period, which indicates a *binary* 1, while a positive or negative transition indicates a *binary* 0 [56]. This block also determines the Cyclic Redundancy Check (CRC) pattern of the signal and outputs packets of data accordingly based on the number of the pattern. The CRC pattern is a technique for error detection during the data transmission and allows to group a number of characters together into a frame. The bits within the frame are combined to give a checksum, which has already been appended

during the transmission. If the checksum calculated in the receiver is different from the one transmitted, the receiver knows that there was an error in the transmission [56].

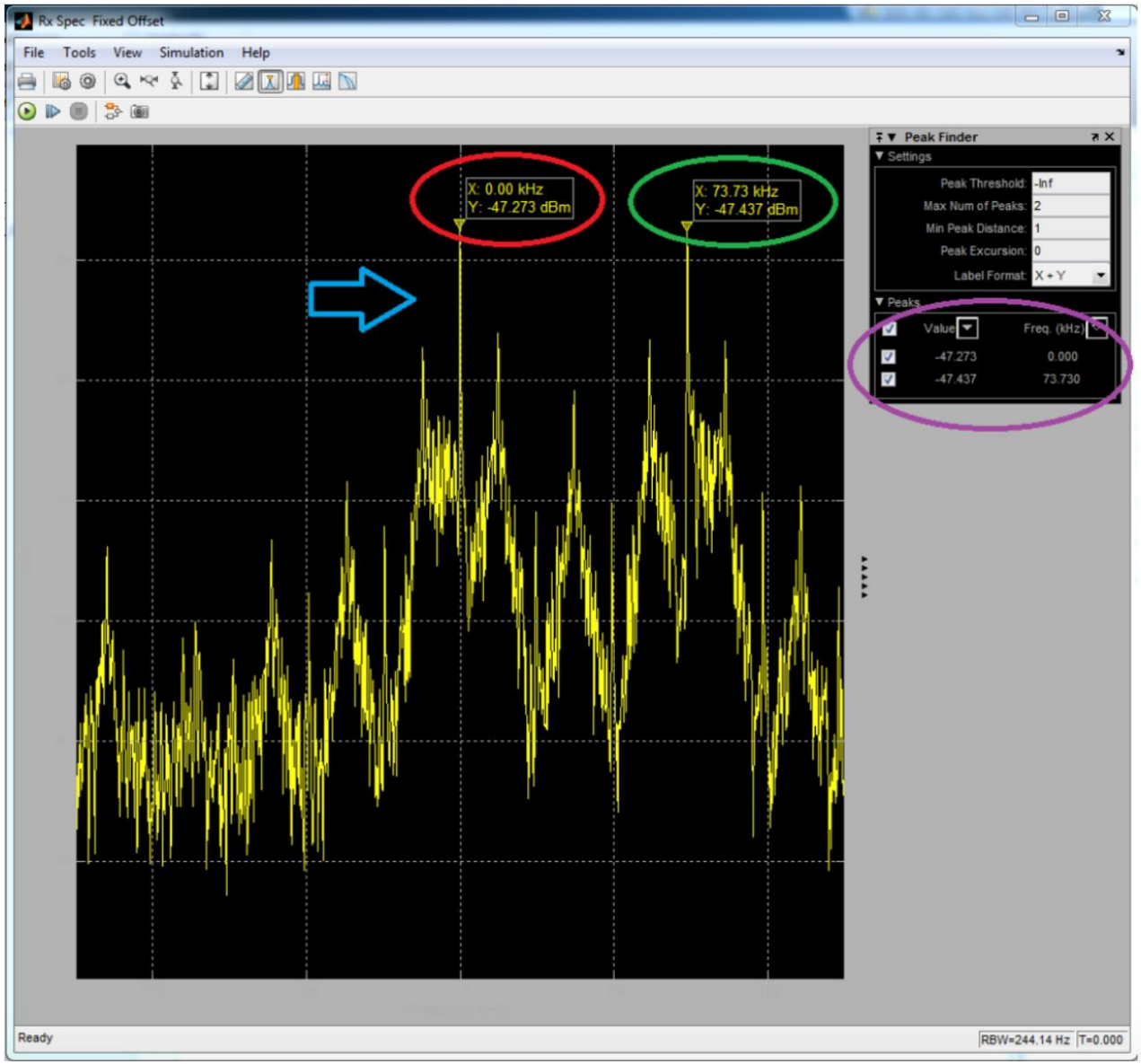


Figure 24 This figure shows the result of shifted FSK spectrum. The blue arrow indicates that the signal was shifted right. The red circle shows that negative signal was shifted to exactly 0.0 Hz. The green circle shows the left signal was shifted to 73.73 kHz. Marked in the purple circle are power and frequencies of the two peaks [10].

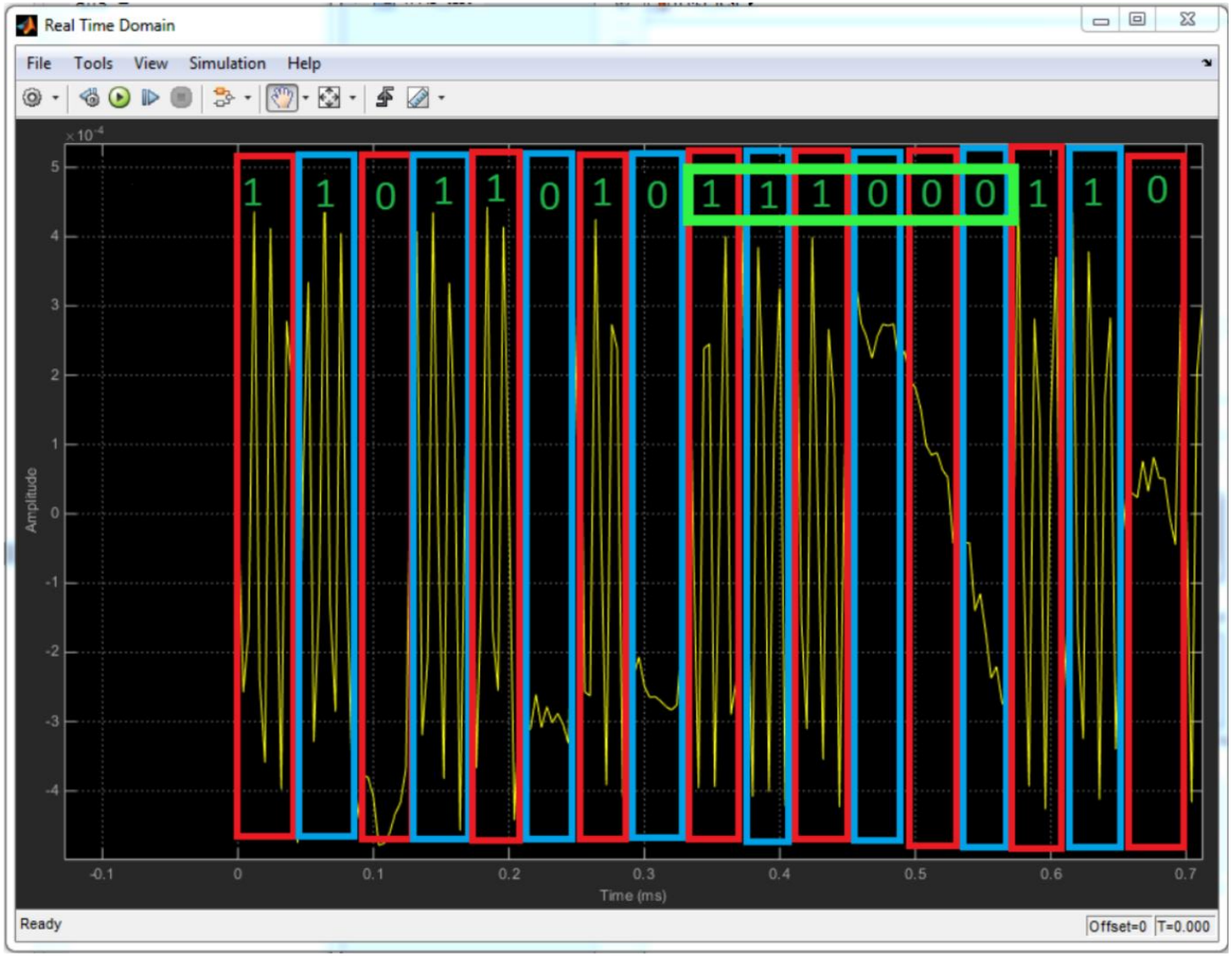


Figure 25 This figure is the time domain spectrum of the frequency shifted FSK signal. This type of waveform is much easier to visually decode by hand. The ones are shown as high frequency signals while the zeros are low frequency. The alternating red and blue rectangles indicate each separate bit. Lastly although the signal use Manchester encoding the green box shows that there are three ones followed by three zeros. It was later determined that this was part of the preamble and therefore it was not included in the Manchester encoding [10].

This receiver's design and methodology influenced the design of the Pseudo TPMS Transmitter. Knowing that the Pseudo TPMS Transmitter had to to take the temperature, pressure and ID values as inputs, the first step would have to be to convert all the information into binary. Since the Benchmark SDR TPMS Receiver decodes the CRC and Manchester encoded data, the transmitter would require a CRC to be generated and added to the data as well as that data to be Manchester encoded. Following the Manchester encoding, a preamble needed to be added to the beginning of the waveform since the Benchmark SDR TPMS Receiver uses the preamble to locate the beginning of the data packet. Using the Benchmark SDR TPMS Receiver as a reference, the next step of the Pseudo TPMS Transmitter was to FSK modulate the data since in the receiver the data is demodulated. The transmitter only includes FSK modulation and not ASK because the receiver only successfully worked with FSK demodulation. Finally, it was decided to transmit the FSK modulated multiple times in a multidimensional array because when the signal is received by the Benchmark SDR TPMS Receiver, the first thing that occurs is that a multidimensional array is concatenated into a single array. Therefore, the Benchmark SDR TPMS Receiver serves as a solid foundation for the design and implementation of a transmitter that can communicate with it. The code for the Benchmark SDR TPMS Receiver can be found in Appendix A.

4.2 Building a TPMS Transmitter

The Pseudo TPMS Transmitter was designed after understanding the flow and logic of the Benchmark SDR TPMS Receiver. The following block diagram shown in Figure 26 outlines the process of the proposed Pseudo TPMS Transmitter which ultimately constructs an FSK modulated signal that can be transmitted to the Benchmark SDR TPMS Receiver.

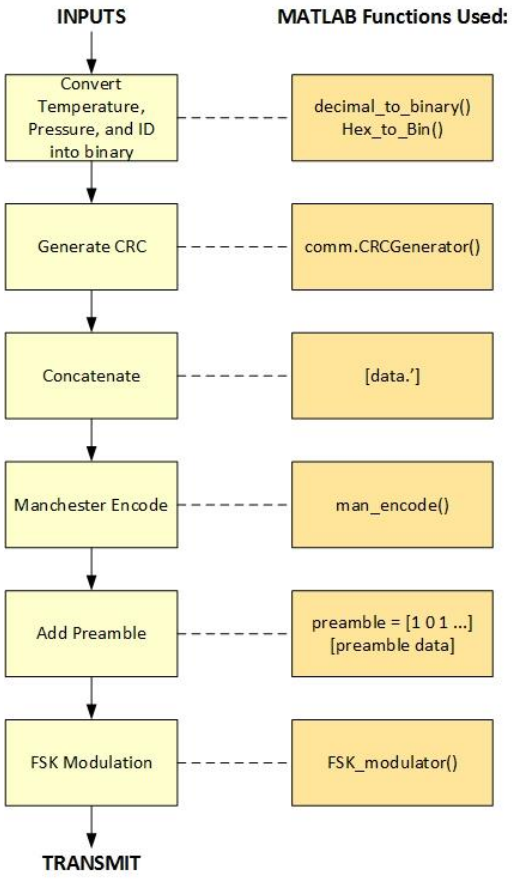


Figure 26 This is an image of the block diagram for the Pseudo TPMS Transmitter created in MATLAB software. It has 6 main blocks, these being: convert temperature, pressure, and ID into binary; generate CRC; concatenate; Manchester encode; add preamble; FSK modulate.

The transmitted signal contains a packet which includes two elements, the first being the preamble and the second being data. To begin, the data of the packet must be constructed, which will include information such as the temperature, pressure, TPMS sensor ID, cyclic redundancy check (CRC), and flags. The constructed signal is illustrated in Figure 27.

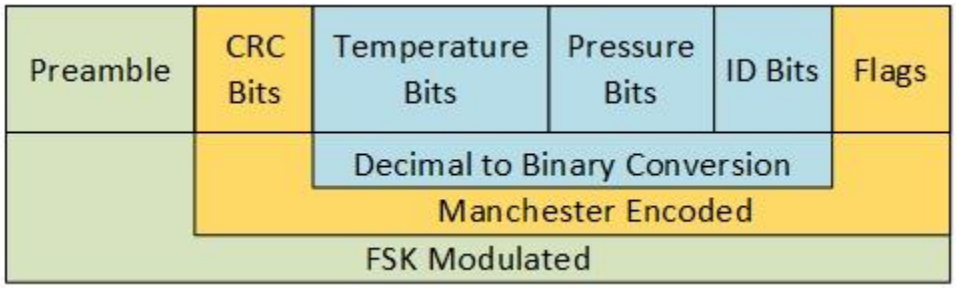


Figure 27 Representation of the construction of the transmitted signal.

The transmitter function accepts 3 inputs from the user: the temperature, pressure and TPMS Sensor ID of the tire. Temperature and pressure are inputted as decimal values while the TPMS sensor ID is inputted as a hexadecimal value. Since the signal must be transmitted as a binary sequence, all of these inputs must then be converted into their corresponding binary values. Next, eight binary bits of both the CRC and flags are required to complete the formation of the packet data. The CRC (Cyclic Redundancy Check) code was created by using a predefined function in MATLAB that generates the code based on a specified polynomial. The purpose of adding the code is to check for errors introduced during transmission. For these testing purposes, the flags code was defined as the same 8 bit binary sequence that was used in Arnold and Piscitelli's MQP report [10].

After all elements of the data were created and converted into binary, they were concatenated to form an array consisting of one row, which was subsequently Manchester encoded. The next step was to create the preamble, which was done by using a Barker Code generator. However, the Barker Code consisted of the values -1 and 1, which is problematic since the whole packet needed to consist of the values 0 or 1. To avoid this problem, a function was created that would change the values of -1 to 0 in the Barker generated code. the function is called *fix_preamble()* and can be found in Appendix B, Section 8.6. The corrected Barker code, or the preamble, was then added to the beginning of the Manchester encoded data, finishing the assembly of the packet.

This packet was FSK modulated, duplicated multiple times, and stacked on top of one another. As a result, there would now be a multidimensional array that would be transmitted, as seen in Figure 28.

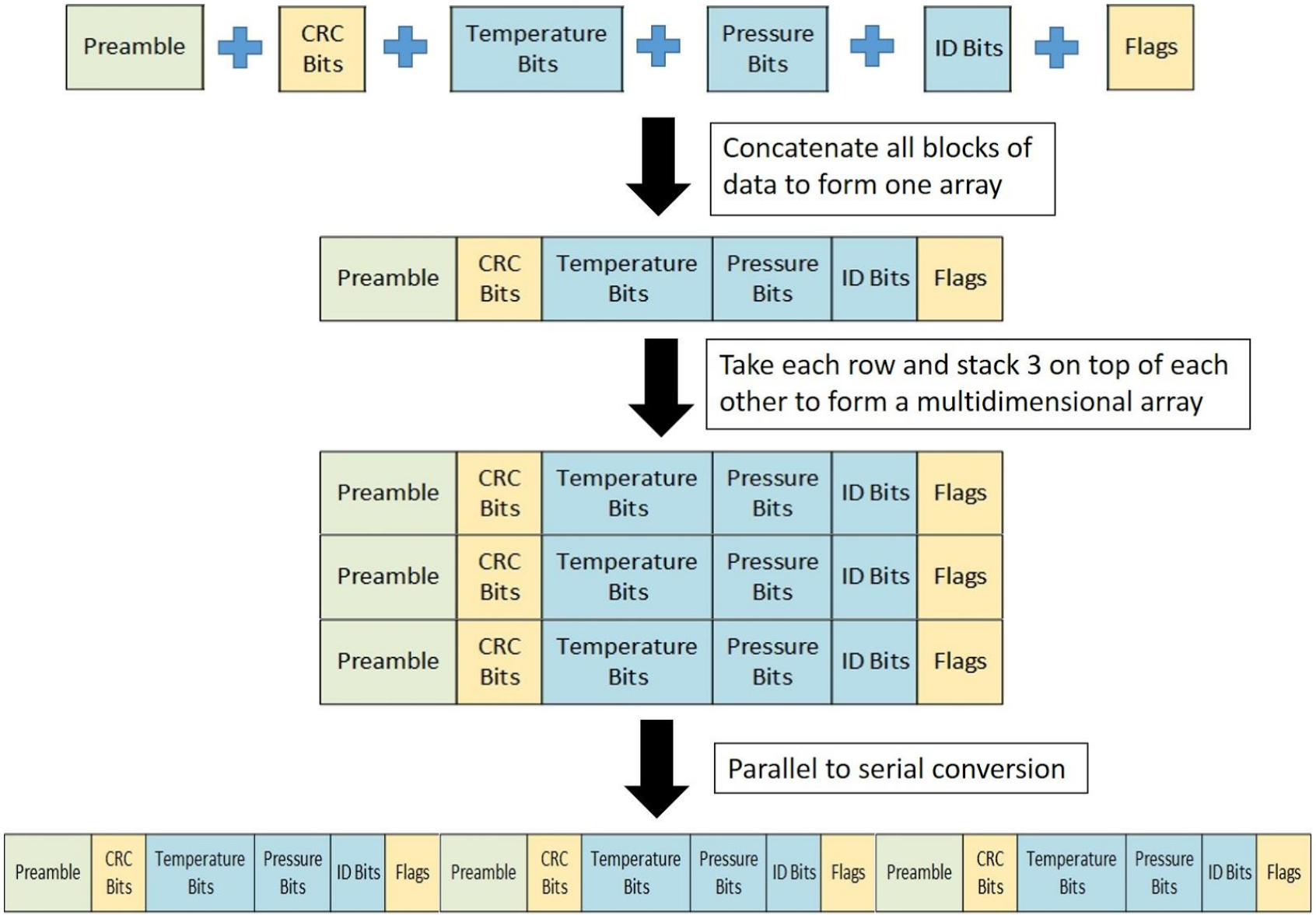


Figure 28 Representation of final construction of the pseudo TPMS signal.

Having a multidimensional array as an output to the transmitter function was necessary because the Benchmark SDR TPMS Receiver is programmed to accept and decode incoming signals that have multiple rows as opposed to a single row array. The Pseudo TPMS Transmitter code can be found in Appendix B.

4.3 Testing

This Section discusses the different test that the team performed in order to assess the functionality of the Pseudo TPMS Transmitter. Such tests include Pseudo TPMS Transmitter with Benchmark SDR TPMS Receiver with MATLAB Simulation, Benchmark SDR TPMS Receiver with TPMS Tx, Benchmark SDR TPMS Receiver with Pseudo TPMS Transmitter with USRP Interface, and Pseudo TPMS Transmitter with TPMS Rx.

4.3.1 Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver with MATLAB Simulation

The first step the team wanted to take was making sure that the signal being transmitted through the Pseudo TPMS Transmitter is equivalent to that of a real TPMS sensor. The team looked at the way the Benchmark SDR TPMS Receiver decoded the received signals from a TPMS sensor and decided to reverse engineer how to create the mimicked signal, as described in Section 4.2. To test if this signal was built correctly, the team simulated transmission through only one computer, without the use of the USRP hardware. The schematic in Figure 29 illustrates the simulation.

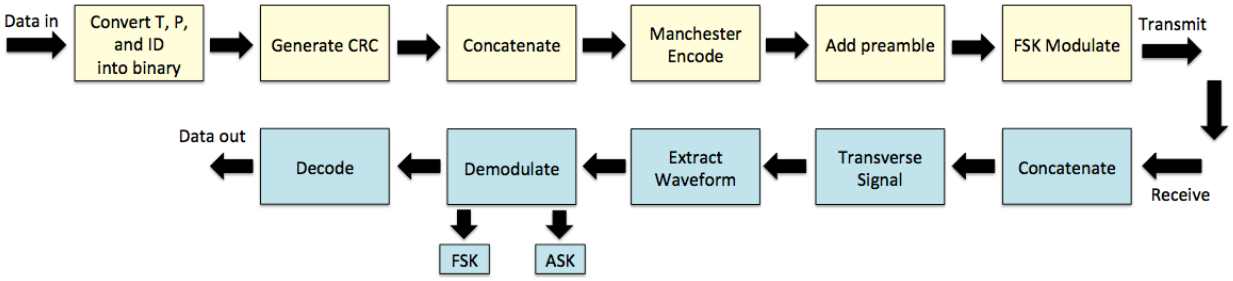


Figure 29 Schematic of MATLAB simulation tests. The yellow blocks represent the transmitter's actions, while the blue blocks represent the receiver's actions.

The data is inserted in decimal values for temperature and pressure, and hexadecimal for the TPMS ID. The steps that follow on the transmitter part of Figure 29 correspond to the different blocks of the pseudo TPMS transmitter created. The final transmitted signal is then passed on to the receiver, with the blue blocks in Figure 29 representing the different steps of decoding and processing as described in Section 4.1. By running the code for the transmitter and saving the output in MATLAB, the team would then feed that output to the receiver and ensure that the decoded values from the Benchmark SDR TPMS Receiver match the inputted values from the Pseudo TPMS Transmitter.

To perform the first test, the team selected two arbitrary values for a pressure and temperature reading as well as a TPMS sensor ID. For the purpose of the example presented below, the temperature was equal to 75 degrees Fahrenheit, pressure equaled 32 psi, and the ID was hex value 'ABCDEF12'. In MATLAB's command window, the team called on the function that runs the Pseudo TPMS Transmitter, *TPMS_transmitter(),* and saved the two outputs under the names *TPMS_signal* and *unmodulated_signal*. The output *TPMS_signal* was necessary because that was the multidimensional array that would be transmitted and used as the input to the Benchmark SDR TPMS Receiver. It is the signal illustrated in Figure 28 in Section 4.2. The output *unnmodulated_signal* was used to ensure that the data including the preamble was constructed correctly before being FSK modulated. The next step was to extract the signal transmitted and place it in the middle of an array with 50,000 zeros. The reason why the signal needs to be extracted is because the Benchmark SDR TPMS Receiver needs to receive information formatted into multiple rows. The team had only created one packet formatted into a single row and stacked the same packet repeatedly into the input provided to the Benchmark SDR TPMS Receiver. Since there is a delay from when the signal is transmitted and received, the padding of zeros ensures that the receiver does not miss the beginning of the signal. The previously described steps can be seen in Figure 30.

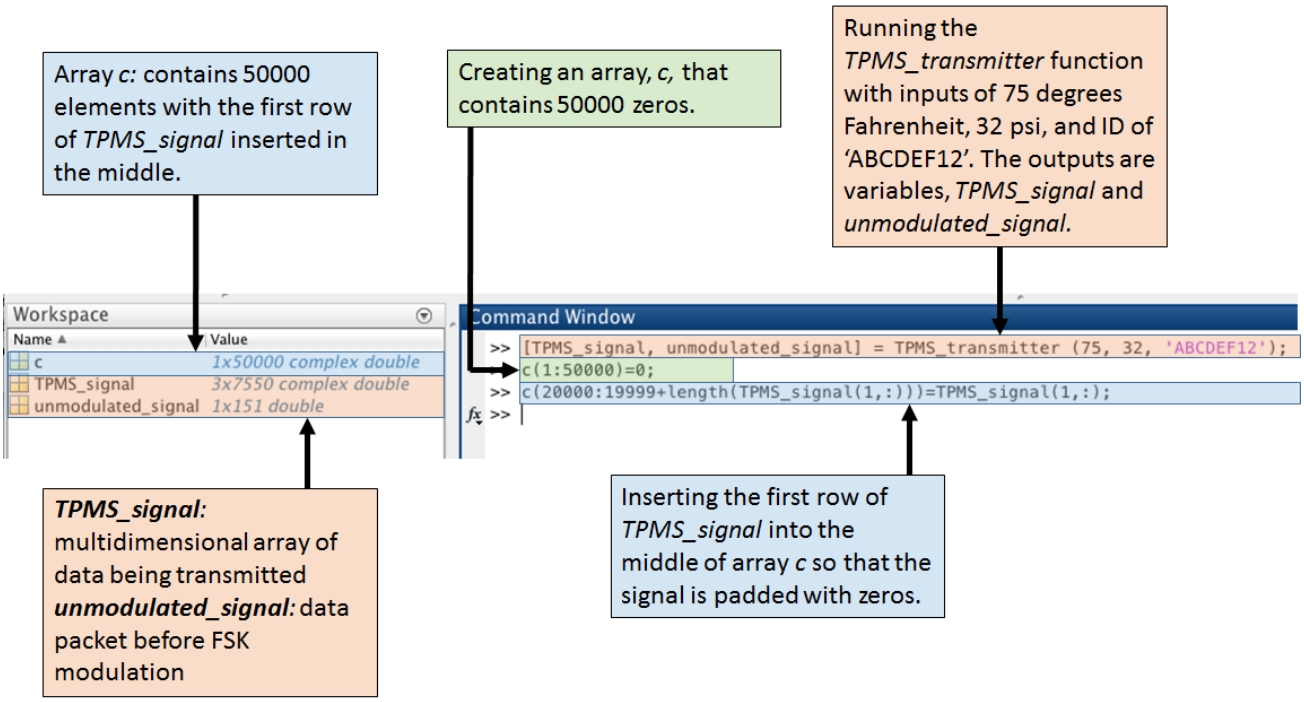


Figure 30 This image shows the first three steps for running the team's TPMS transmitter. These being calling on TPMS_transmitter(), placing signal in the middle of an array with 50,000 zeros, and extracting the signal.

The following step is to receive the transmitted signal into the Benchmark SDR TPMS Receiver using the *TPMS_receiver()* function. This command runs the receiver code created by Arnold and Piscitelli and generates a plot of the magnitude of the received signal's Fourier transform. This plot can be seen in Figure 31. Since this plot shows two peaks, the receiver will use FSK demodulation. As mentioned in Section 4.1, the received FSK modulated signal has peaks at two frequencies, around center frequency, which represent 0 and 1. The whole signal is then shifted to have the lowest peak at 0 and then the highest peak moves further. For this reason, the plot in Figure 31 has one peak close to 0 and one at 6000, since the initial peaks were at -3000 and 3000 Hz.

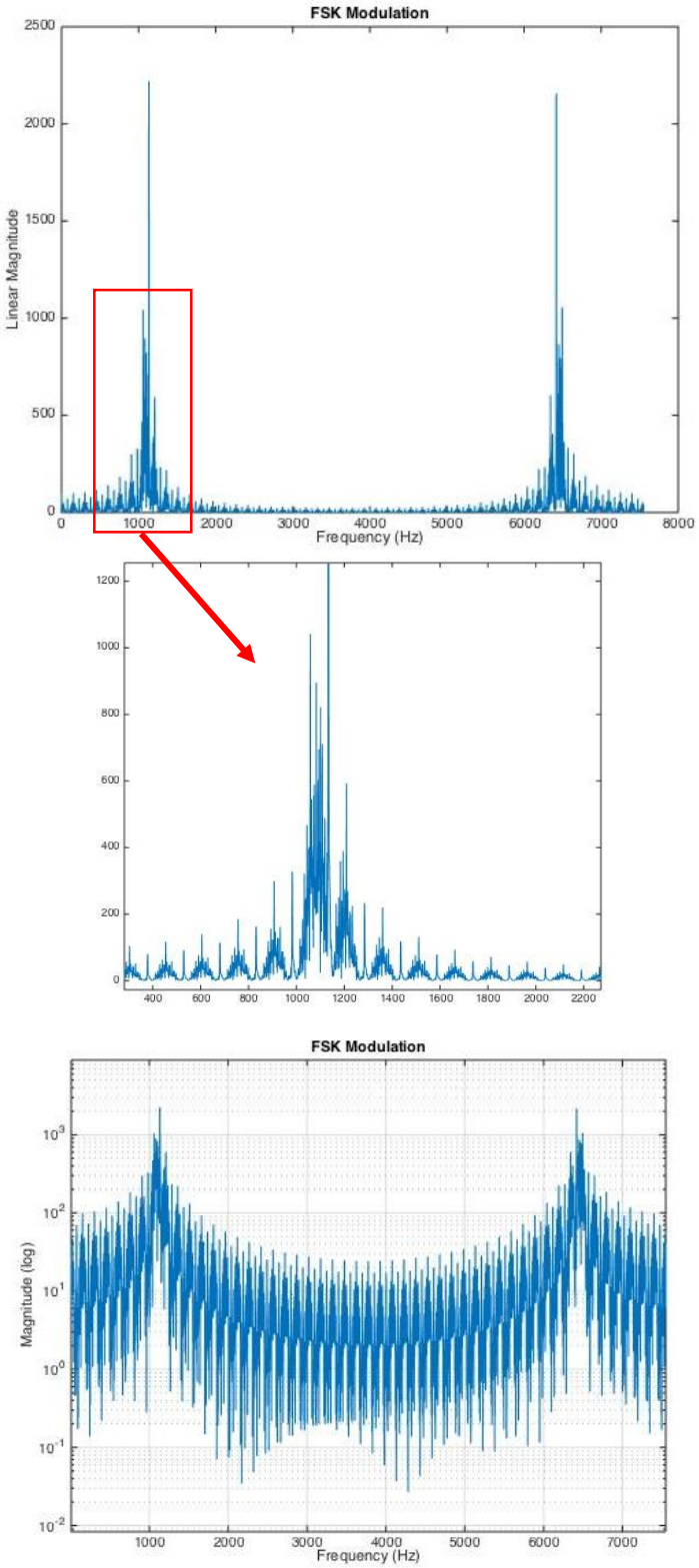


Figure 31 The top image is of a plot generated when running the Benchmark SDR TPMS Receiver with the Pseudo TPMS Transmitter and shows the magnitude of the received signal's Fourier transform as well as a zoomed in look of the further left peak. The bottom image is of the same signal but plotted using log scale for the magnitude showing the peaks of the same two frequencies in the linear graph.

The next step was inverting the output of the receiver, since the first bit to reach the receiver is the last bit of the transmitted signal. The MATLAB command to do this step is shown in Figure 32. The final step of this test is to prove that the transmitted signal is equivalent to the received signal. By calling the function *TPMS_decode_by_ID_second()* the team can validate the results in MATLAB's workspace window. Figure 32 shows the command used to validate the results.

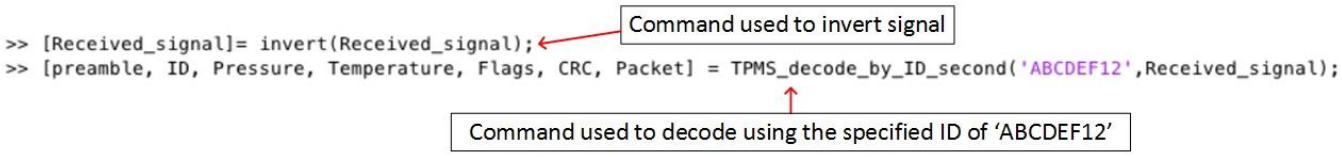


Figure 32 This image shows the command typed into MATLAB's command window in order to verify that the signal transmitted was equivalent to the signal received. The team is checking that the values for the preamble, ID, pressure, temperature, flags, CRC, and packet are the same

The *TPMS_decode_by_ID_second()* function was created by Arnold and Piscitelli and can be found in Appendix A, Section 7.3. The function call takes as input the expected ID and the received signal, ready to be decoded. This function is for TPMS signals that have the ID after the temperature and pressure values. Then, within *TPMS_decode_by_ID_second()*, the function *find_ID()* is called, which correlates a binary version of the ID inputted to the function with the signal to locate the starting point of the ID. Then, the function separates the bits as follows:

- Bits 1 to 8 : Pressure
- Bits 9 to 16: Temperature
- Bits 17 to 48: ID
- Bits 49 to 56: Flags
- Bits from 57 to the end: CRC packet

The function finally converts the bits for each of those parameters to decimal or string and outputs it to the workspace.

4.3.2 Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver with USRP Interface

Based on Arnold and Piscitelli's work [10], as well as the previous literature on wirelessly accessing vehicles, the team decided to use a software defined radio as the hardware part of the Pseudo TPMS Transmitter. As mentioned in Chapters 2 and 3, SDRs are a very useful tool since they can be programmed to have the exact characteristics of a transmitter and receiver needed. The team's acquired knowledge of MATLAB coding also made them the best candidate for this project. Throughout the project, the USRP N210 SDR was used since it was already available to the team in the university's laboratories. In order to use the USRP, its library was loaded on MATLAB and was connected successfully to the hardware through a Gigabit ethernet cable, as shown in Figure 33. Using the *findsdru()* command in MATLAB, the software and hardware interface was confirmed to work properly:

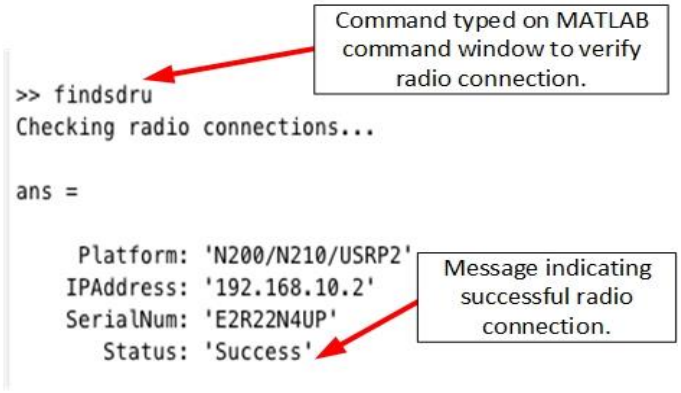


Figure 33 findsdru() function in MATLAB finds the radio connected to the computer and gives information relating to the platform, IP address, serial number, and status of the connection.

The overall setup of the test was two laptops connected to two USRPs; one acting as the Pseudo TPMS Transmitter and one as the Benchmark SDR TPMS Receiver, as shown in Figure 34.

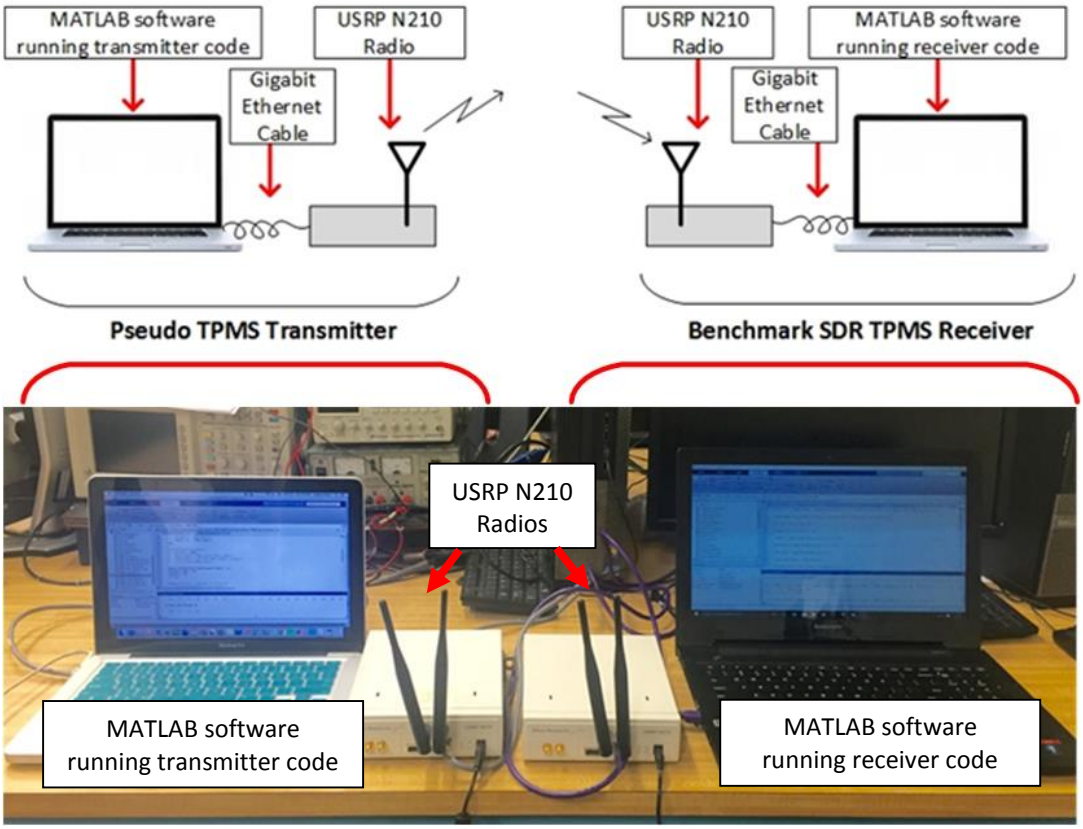


Figure 34 Schematic and in lab setup for communication of the Pseudo TPMS Transmitter with Benchmark SDR TPMS Receiver testing with USRPs.

Having set up the hardware correctly, one of the team's goals was to replicate the Pseudo TPMS Transmitter to Benchmark SDR TPMS Receiver transmission, this time through the USRP Interface. Essentially, the same signal with a specific temperature, pressure and ID would be transmitted and received by the receiver but instead of using commands to code the transmission on one computer, the transmission would be wireless between two URSPs; one acting as the Pseudo TPMS Transmitter and one as the Benchmark SDR TPMS Receiver. Successful communication between the two would mean that the software and hardware of the Pseudo TPMS Transmitter are interacting effectively and the team can continue by testing it with the actual TPMS.

In order to test the Pseudo TPMS Transmitter and the Benchmark SDR TPMS Receiver with the USRP interface, the team had to create a real-time receiver function that could call on the USRP radio and receive data. This real-time receiver function defined the radio's arguments

such as platform, IP address, center frequency, burst mode, output data type, decimation factor, sample rate, gain, frames, and frame length. Definitions for these arguments can be seen in Table 4. Another radio was defined similarly in the transmitter code and called on to transmit the created TPMS signal. The arguments of the transmitting radio had to be compatible with those of the receiving radio.

Over the span of the testing phase, various modifications had to be made to the real-time receiver being used to collect data. Also, the team collaborated with Alex Arnold and implemented some changes to his Benchmark SDR TPMS Receiver in order for it to accommodate both the Pseudo TPMS Transmitter and the real TPMS sensor signals. The team identified that one of the main challenges was the synchronization of the two radios. The previous version of the Benchmark SDR TPMS Receiver ran for a specified amount of time, which was tailored to the length of data being received. The main limitation of that receiver was that its duration was not sufficient to successfully synchronize with the Pseudo TPMS Transmitter. After increasing the Benchmark SDR TPMS Receiver receiving period to a large number, the team concluded that the best chance of successful communication would be when the Benchmark SDR TPMS Receiver is receiving continuously. For this reason the team decided to alter the receiver so that it continuously searches for a signal in the desired frequency of 315MHz and over the designated power threshold value. The modified real-time receiver code was embedded in Benchmark SDR TPMS Receiver code, found in Appendix A. Once the code was ready, the team tried to transmit the mimicked signal from the Pseudo TPMS Transmitter to the Benchmark SDR TPMS Receiver via the USRP radios.

Table 4 Definitions of radio arguments [56]: Platform, IP Address, Center Frequency, Burst Mode, Output Data Type, Decimation Factor, Sample Rate, Gain, Frames, and Frame Length .

Argument	Definition
Platform	Usually refers to the basic *hardware* and *software* elements (e.g. computer, *operating system*, *relational database*) on which an *application program* is built.
IP Address	The *address* used to differentiate *users* on the *Internet*. The Internet is designed to use an addressing system based on 32 *bits*.
Center Frequency	The *frequency* used for *broadcast*ing. This normally refers to a radio or television broadcast and the frequencies are tightly regulated to prevent interference between transmitting stations.
Burst Mode	The *transmission* of data in large continuous bursts, the transmitting *terminal* occupying the whole *channel* during the period of the transmission.
Output Data Type	Refers to the type of the output data (e.g. double, precision-floating point, single-precision floating point, 16-bit signed integer).
Decimation Factor	An integer or a rational fraction greater than one, which is multiplied by the sampling time. This information is used to down convert the Intermediate Frequency (IF) signal to complex baseband.
Sample Rate	The number of samples taken in a unit of time.
Gain	A measure of *signal* amplification. It is usually given by the ratio of the output *amplitude* of a signal to its input amplitude.
Frames	A group of *bits* and *bytes* which are collected together in a recognized format for *transmission*.
Frame Length	The number of bits that are found within a frame.

The team performed several trials in order to identify the adequate values for the radio gain and power threshold. There are two types of power thresholds in this configuration that are important for the functionality of the receiver. The real-time receiver threshold is the average band power of the first few samples received by real-time receiver multiplied by a constant, k, as seen in Figure 35. The purpose of this multiplication is to set a power threshold so that the receiver will be able to differentiate the received signal from noise. Similarly, the decoder threshold is the value that the receiver will use to select the desired portion of the signal to be decoded.

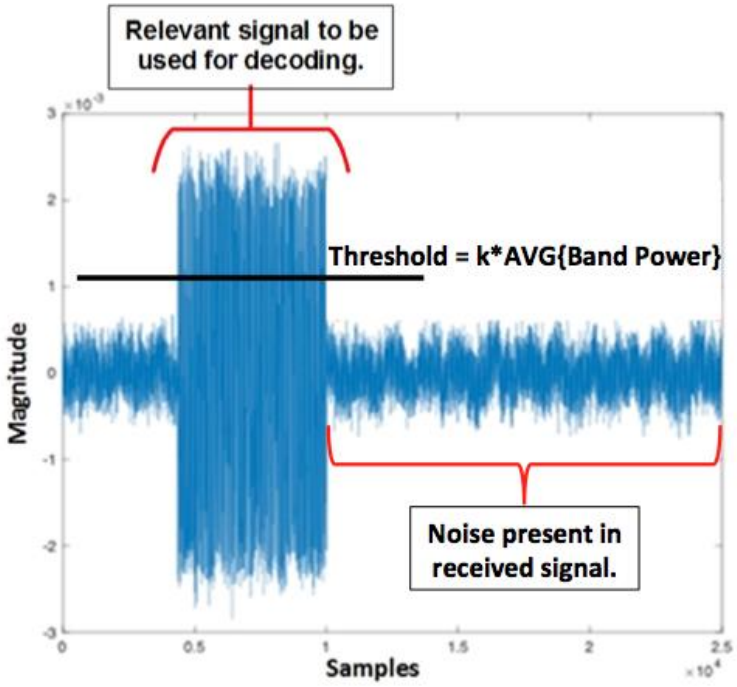


Figure 35 Visual for how the threshold is calculated for both the decoder and the real time receiver.

The first few trials were essential in identifying the errors and malfunctions of the transmitter and receiver code and making the required changes to achieve successful transmission between the Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver with USRP interface. The main issue initially was setting the power threshold for identifying the presence of the signal. In the first few trials, the result was a “no signal was detected” error, which led the team to believe that the power threshold was too high to identify the Pseudo TPMS Transmitter signal. Figure 36 shows the resulting error message when no signal is found. After decreasing the value of bandpower, the signal received had too much noise to be decoded correctly. By trial and error, the team managed to successfully identify the bandpower value that would not be too high to get the “no signal detected” error or too low to get a received signal with too much noise. Figure 37 shows a received noisy signal.

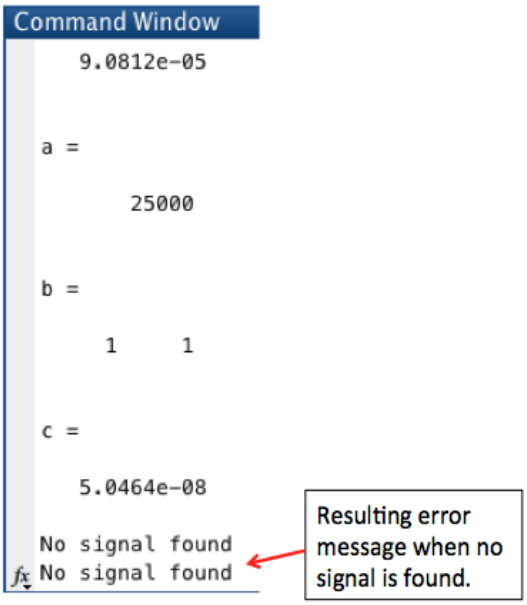


Figure 36 Resulting error message when no signal is found during testing of the Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver with USRP Interface.

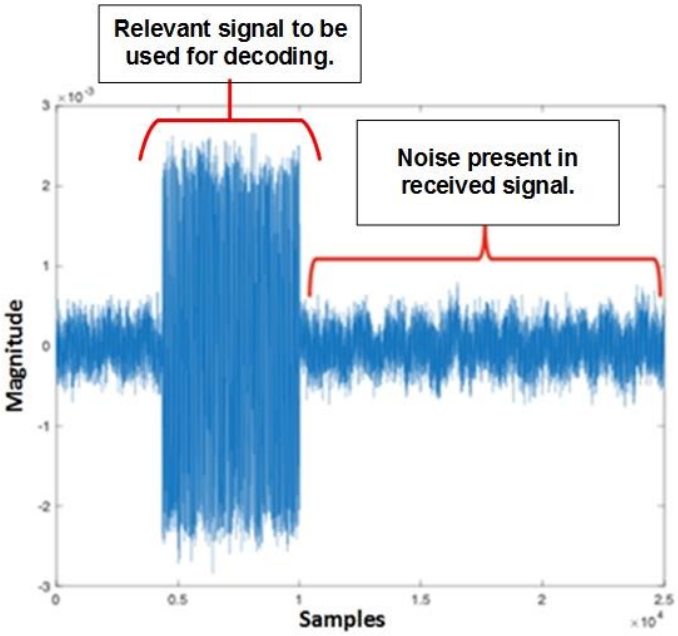


Figure 37 Signal received during testing with noise present.

In a similar fashion, the gain factor of the receiving radio was initially set to 32, then 8 and finally to 16 where the successful communication was achieved. The gain of the transmitter and receiver are equal in this test, however that is not necessary. The reason those values were chosen is because 16 is the minimum gain factor for which the Benchmark SDR TPMS Receiver was able to detect the signal sent by the Pseudo TPMS Transmitter.

After testing with modified values for power thresholds, gain factors and synchronization techniques, the team noted the ones that led to a successful communication between the Pseudo TPMS Transmitter and Rx. The alterations that were made fixed the synchronization issue. The term “real-time receiver” refers to the receiver created by the team to pick up the signal and the term “decoding receiver” refers to the modified Benchmark SDR TPMS Receiver used to further process the received signal. The modifications that worked successfully for the Pseudo TPMS Transmitter can be seen in Table 5.

Table 5 Successful parameter modifications for Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver.

Variable	Modification
Pseudo TPMS Transmitter Gain	16
Benchmark SDR TPMS Receiver Gain	16
Real-Time Receiver Threshold	50 * band power
Decoder threshold	100 * band power (or 20 * band power)
Synchronization	Rx continuously looking for signal. Tx transmits for 5 seconds.

The first step in decoding the signal correctly is for the decoding receiver to identify the correct modulation scheme (FSK or ASK). As seen in Figure 31, the decoder shows a graph of the modulation identified in the signal. The signal transmitted is FSK modulated in the Pseudo TPMS Transmitter, therefore all signals decoded should show FSK modulation. However, in some cases when one of the values shown in Table 5 was not correct the receiver would identify the signal as being ASK modulation and would not be able to continue decoding. Figure 38 shows a plot of what an ASK modulated signal would look like after the receiver incorrectly identifies the modulation scheme in comparison to an FSK modulated signal.

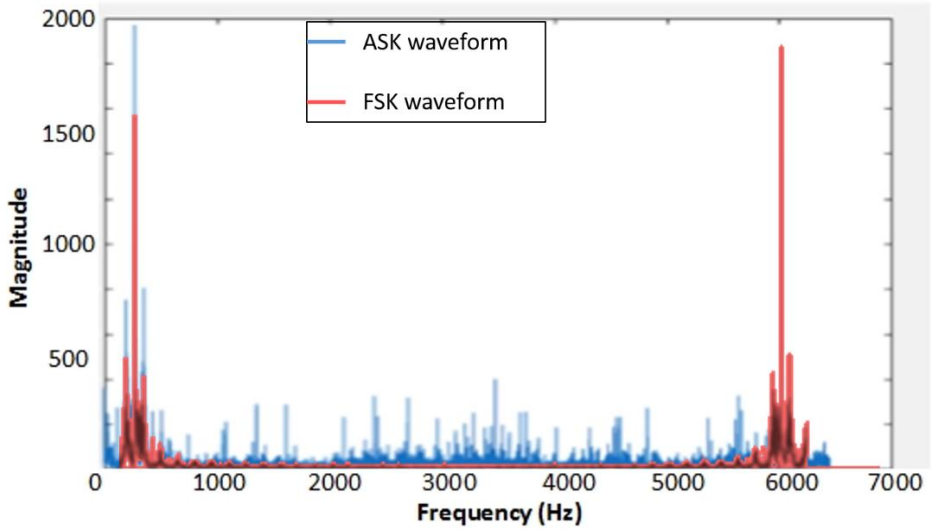


Figure 38 This image shows the difference between an ASK (blue) and FSK (red) signals. A resulting ASK modulation scheme is when the receiver does not properly identify the scheme due to one of the values shown in Table 5 being incorrect.

As long as the modulation scheme is identified correctly, the receiver should be able to move onto correlation. Within the function to find the TPMS sensor ID, *find_ID()*, the MATLAB function for correlation, *xcorr(),* is used. This function takes as input the signal and the ID in binary format, and locates the index where the two signals are identical. The output of the function is an array of indices and their respective arbitrary linear correlation values. The highest the correlation value, the closest the two signals are at being identical at that index. A figure visualizing correlation is outputted by the receiver. In that figure, a high correlation value is obtained at the index of the first bit of the correlated sequence. If the receiver is not able to correlate, a plot like the one shown in Figure 39 is outputted by the receiver.

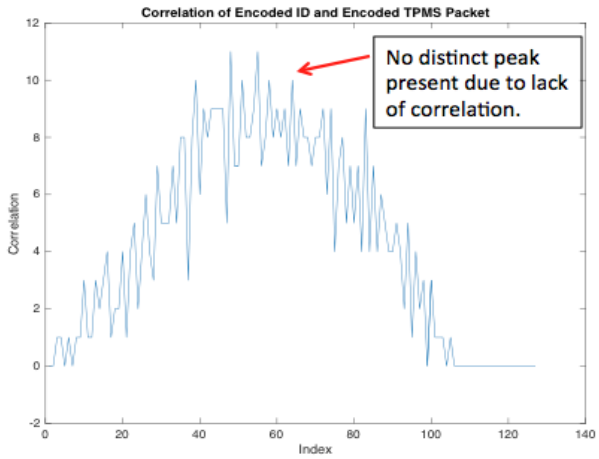


Figure 39 Unsuccessful correlation of transmitted signal's bits with the TPMS ID.

If the signal is received and decoded correctly, the correlation should be successful with a distinct peak that shows the beginning of the TPMS ID, as shown in Figure 40.

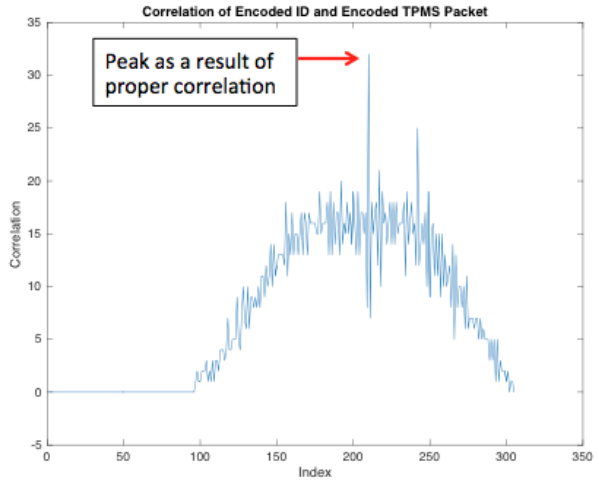


Figure 40 Successful correlation of transmitted signal's bits with the TPMS ID. The peak indicates the first position (index) where the correlated signal begins.

After correlating the signal, the receiver locates the maximum correlation value and its index to locate the beginning of the ID. The index for the beginning of the ID can be used to extract other parameters from the signal. Obtaining a graph of proper correlation in this test is an indication of the correct implementation of the communication between the Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver with USRP interface.

4.3.3 Replicate Benchmark SDR TPMS Receiver with TPMS Tx

Having a functional Benchmark SDR TPMS Receiver, the team proceeded to test the receiver with a TPMS sensor. This test was run almost simultaneously with the Pseudo TPMS Transmitter - Benchmark SDR TPMS Receiver test to ensure that the receiver was functional for both the mimicked signal and the real TPMS sensor signal. The team replicated the results from Arnold and Piscitelli's project [11] using the Benchmark SDR TPMS Receiver to receive the signal transmitted by the TPMS sensor. Using the Benchmark SDR TPMS Receiver, the team verified that the implementation of the communication of Benchmark SDR TPMS Receiver with the Pseudo TPMS Transmitter was successful. The test was conducted in a laboratory to ensure maximum control. A photograph of the setup for this test can be found in Figure 41.

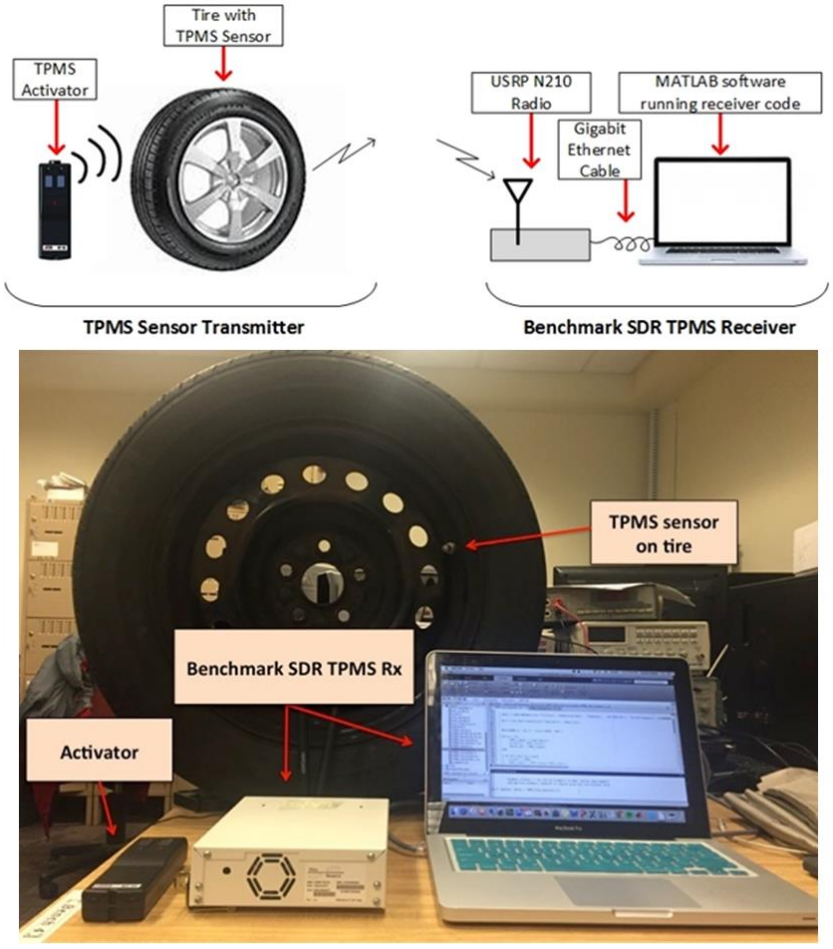


Figure 41 Schematic and in lab setup for communication of the TPMS Tx with Benchmark SDR TPMS Receiver testing with a USRP.

The TPMS sensors used were two DORMAN TPMS modules, part numbers 974-063 and 974-026, one with ASK modulation and the other with FSK modulation for control purposes. A photograph of the FSK modulation sensor (part number 974-063) is shown in Figure 42.

Figure 42 DORMAN TPMS module, ID number 8178E561.

This is the sensor that was installed in the tire from Figure 41a. The TPMS sensors were placed on top of the USRPs and then activated using the ATEQ VT15 activator. This activator has two buttons, one to begin activating the sensors and one to cancel the activation. By pointing the activator at the sensor and pushing the activation button, the sensor will then begin transmitting TPMS signals. The signal transmitted by the TPMS sensors were read by the USRP and analyzed using the Benchmark SDR TPMS Receiver MATLAB program.

The next step was to test the transmission of a TPMS sensor from inside an actual tire. To run this test, the team installed the FSK TPMS sensor into a tire to read live pressure and temperature values. The real-time receiver needed to be run in MATLAB such that the team could activate the TPMS sensor using the activator. The real-time receiver would continue to collect data until it detected a signal. It then called on the Benchmark SDR TPMS Receiver to decode the detected signal. If the decoded values were representative of the surrounding temperature and actual pressure value (in kPa), then the team would have replicated the results

of the previous MQP correctly and would have an Benchmark SDR TPMS Receiver that works for both the Pseudo TPMS Transmitter signal and real TPMS sensor signal.

Within the first few trials it became clear that the threshold and gain values used to receive the Pseudo TPMS Transmitter signal were not appropriate for receiving the TPMS sensor signal. In an attempt to explain that, the team assumed that one explanation could be the lower amplitude of the TPMS signal, that did not allow it to get picked. In order to prove that the low amplitude was the issue, the team increased the values for gain and lowered the power threshold, until the signal was successfully transmitted and received. For this reason, the values for gain and power thresholds were altered to those mentioned in Table 6.

Table 6 Successful parameter modifications for the Benchmark SDR TPMS Receiver when testing with the TPMS sensor.

Variable	Modification
Benchmark SDR TPMS Receiver Gain	8
Real-Time Receiver Threshold	50 * band power
Decoder Threshold	20 * band power
Synchronization	Rx continuously looking for signal

As one can notice from Table 6, the Benchmark SDR TPMS Receiver can only decode the TPMS sensor signal when its decoder threshold is 20*band power, which is significantly lower than the one used for the Pseudo TPMS Transmitter with Benchmark SDR TPMS Receiver communication. The reason is because the TPMS sensor signal has a lower amplitude and therefore, when there is a threshold of above 20 multiplied by the average band power, the decoder does not identify the signal to be decoded. Figure 43 shows the difference in amplitude between a real TPMS sensor signal and the Pseudo TPMS signal.

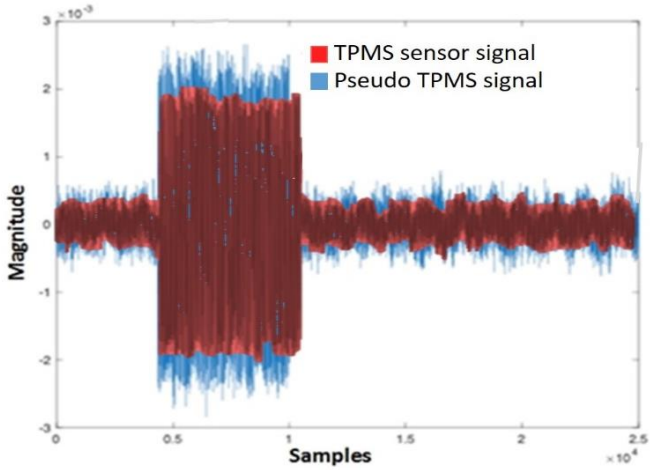


Figure 43 This image shows the difference in amplitudes between a signal from a real TPMS sensor (red) and the Pseudo TPMS Transmitter signal (blue).

Similar to the challenges faced in implementing the Pseudo TPMS Transmitter - Benchmark SDR TPMS Receiver communication in Section 4.3.2, the team also experienced issues in this testing phase that led to “no signal found” error, wrong modulation used in decoding, or errors in correlation. Those were attributed again to synchronization issues, which were present if the sensor activator button was not pressed at the correct time. In order to overcome the synchronization issues again, the team used the same Benchmark SDR TPMS Receiver as in the previous test, which received continuously so that the timing of pressing the activator button would not matter.

4.3.4 Transmit using Pseudo TPMS Transmitter to TPMS Rx

The project’s goal is to communicate with a vehicle’s CANbus through the Pseudo TPMS Transmitter. To corroborate this, the team purchased a TPMS Rx to see if communication with it was possible with the Pseudo TPMS Transmitter. Following that, it was necessary to investigate the layout of the receiver, gain more insight on how it works and how the information from the transmitter is received and propagated to the rest of the CANbus [57]. It was discovered that the TPMS Rx relies on multiple specific messages that prompt it to communicate with the TPMS sensors. The team does not have access to such messages in order to determine whether or not the Pseudo TPMS Transmitter communicated effectively with

the TPMS Rx. Therefore, the team reached the decision to perform the test between the Pseudo TPMS Transmitter and TPMS Rx on a vehicle.

For this test, the Pseudo TPMS Transmitter was taken outdoors to be tested on a real vehicle. The vehicle used for this test was a 2012 Ford Fiesta. In order to implement this test, the TPMS sensor IDs of the vehicle needed to be known. The car was taken to a mechanic to read the tire IDs. Figure 44 shows the values read through the OBD-II port of the tires in the Ford Fiesta. Table 7 shows the tire IDs converted from decimal to hexadecimal of each tire in the Ford Fiesta.

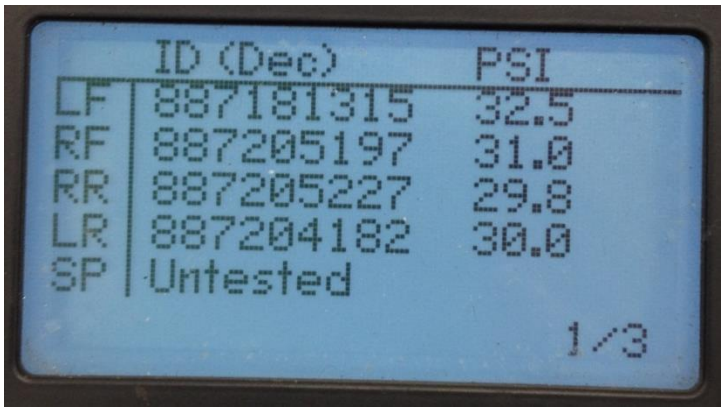


Figure 44 Tire IDs read through OBD-II port of the Ford Fiesta used for testing.

Table 7 TPMS IDs for each of the tires on the Ford Fiesta converted into Hexadecimal.

Tire Position	TPMS ID (in HEX)
Front left (LF)	34E128F3
Front right (RF)	34E1AD4D
Rear left (LR)	34E1A956
Rear right (RR)	34E1AD6B

When it was time to physically test on the Ford Fiesta, the first step was to shield the front-left tire using a metal sheet to disable the communication of the TPMS with the vehicle's receiver to trigger the TPMS light to turn on. The metallic shielding used for this purpose is

Cobaltex Near Field Magnetic and Electric Shielding Fabric which is a nickel, copper and cobalt plated polyester. The team acquired 5ft x 3.5ft of the metallic shielding fabric from Amazon. This metallic fabric provides RF shielding in the range of 30 MHz to 1 GHz which encompasses our TPMS communication at 315 MHz. To confirm that the metallic shielding provided by this fabric worked properly, the team wrapped a cellphone with this fabric and called that cellphone. The wrapped cellphone did not receive any calls.

After covering the vehicle's tire with this metallic shielding fabric, the vehicle was kept with the ignition on for 25 minutes but the TPMS light never turned on. The setup and schematic for the test is demonstrated in Figure 45.

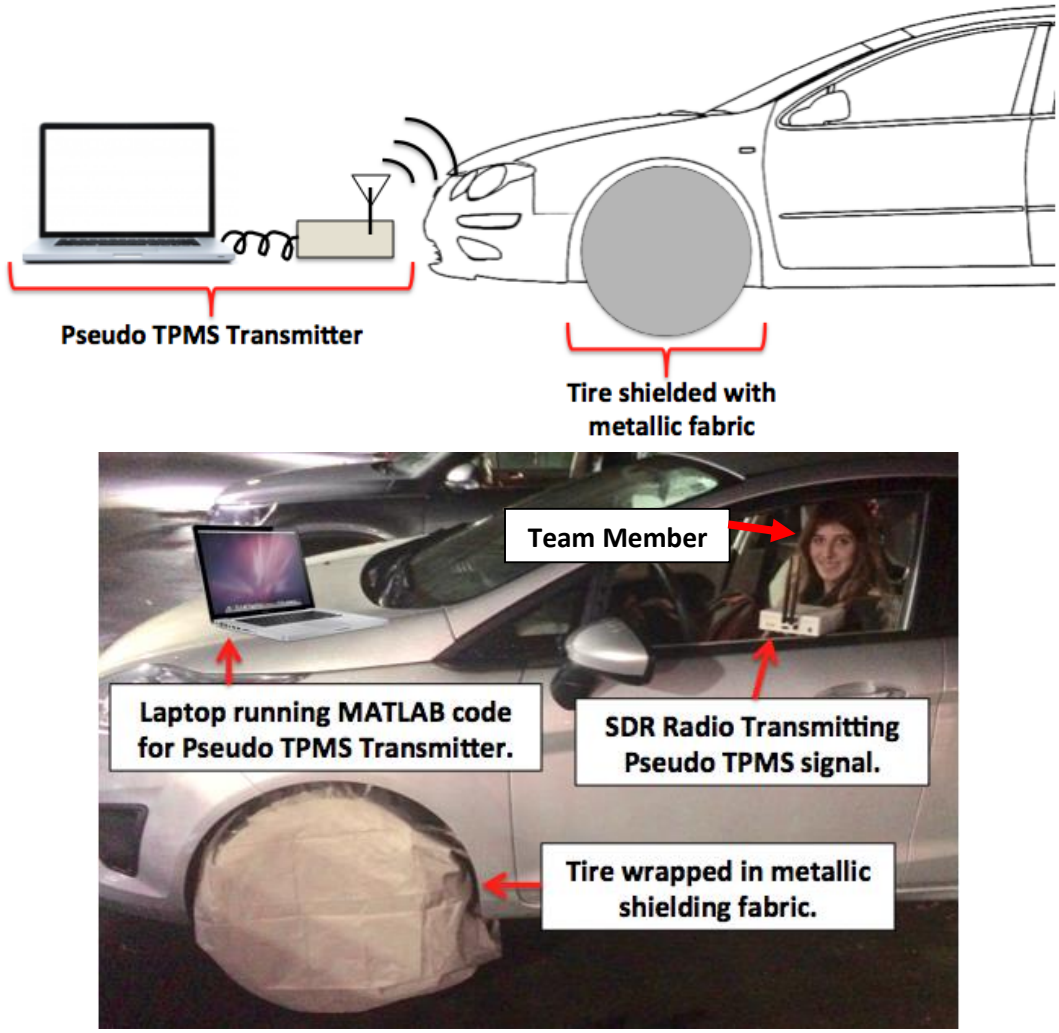


Figure 45 Schematic and setup for Pseudo TPMS Transmitter and TPMS Rx test using a metallic fabric to shield the tire.

It was speculated that maybe the vehicle needs to be driven to activate the TPMS light to turn on. For this purpose, the front of the car was raised using two car jacks, put on the e-brake and made sure that the front-right tire is fully covered with metal sheet. One of the teammates slowly pressed the gas pedal to make the front tires rotate. This was done for ten minutes but the TPMS light still did not turn on. Therefore, the test was not completed in the first attempt either because the metal shielding was not blocking the TPMS sensor signal appropriately, or the sensors needed to activated by actually driving the car for a few miles instead of having the wheels rotate in the air for a few minutes.

Since the TPMS light did not turn on, the team decided to send a signal from the Pseudo TPMS Transmitter with a low pressure value to trigger the TPMS light to turn on. The chosen tire's ID was inserted in the *TPMS_transmsitter()* function as well as the desired values for temperature and pressure. This approach still did not turn on the TPMS warning light.

The second attempt was to take more drastic measures and remove the front left tire completely so that the TPMS sensor is not in range. In order to be able to drive the car and activate the TPMS, the spare tire was installed, which was assumed not to have an activated TPMS sensor. The team concluded that the spare tire did not have an activated TPMS sensor because when the sensor IDs were read, it did not return an ID for the spare tire. The reason behind removing a tire with the TPMS sensor was that the TPMS receiver would trigger the TPMS warning light if it is not receiving any signal from a TPMS sensor. Figure 46 shows the Ford Fiesta with the spare tire installed, right before it was driven around.

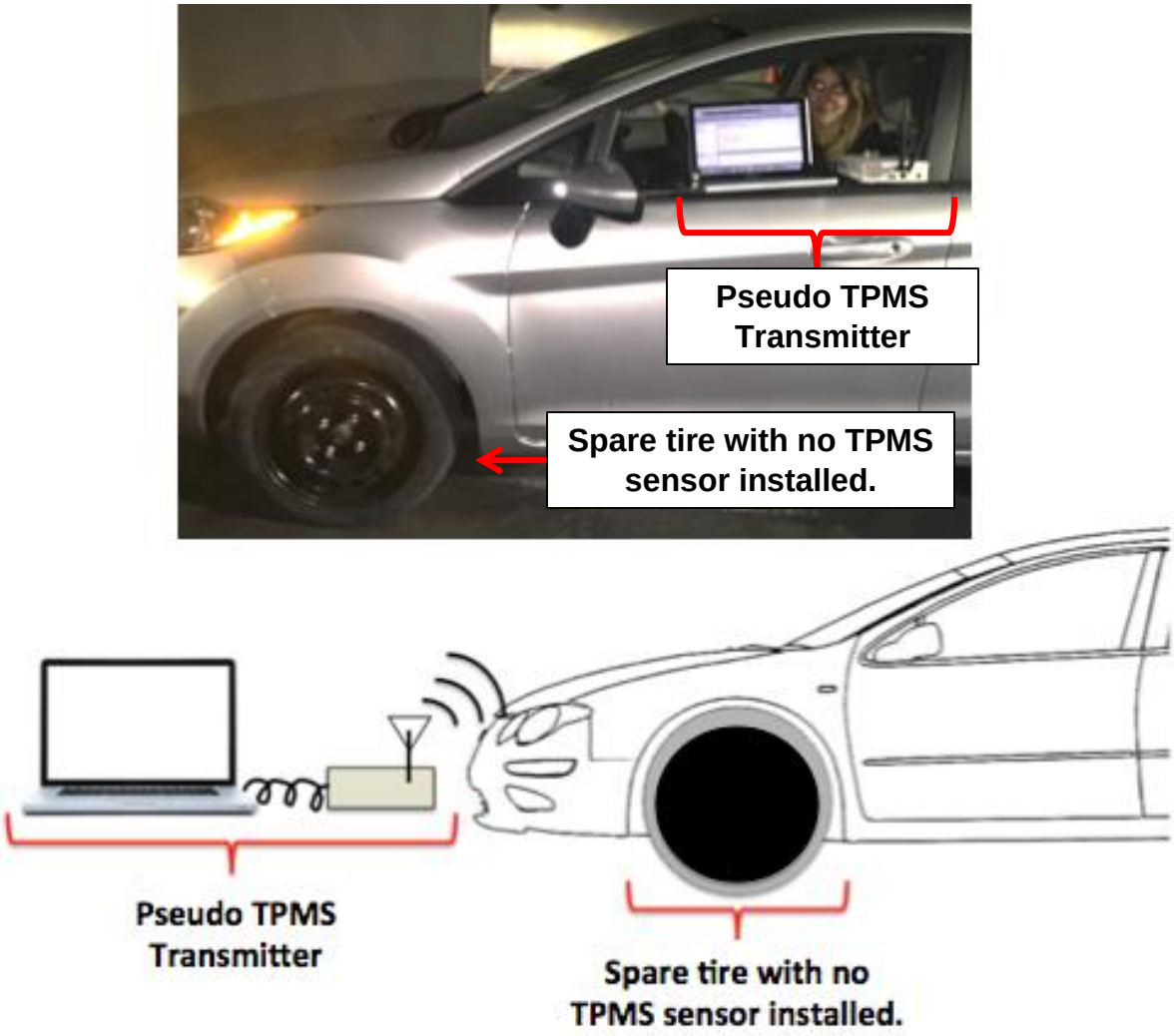


Figure 46 Schematic and setup for the second attempt of testing with Pseudo TPMS Transmitter on a car. Installation of the spare tire to activate the TPMS light on the dashboard.

While driving with the spare tire on, waiting for the TPMS light to turn on, one USRP was connected to the cigarette lighter of the car using an inverter and the Benchmark SDR TPMS Receiver was run on it to collect data of the TPMS sensors in the car. Unfortunately, the light did not turn on after 50 minutes of driving with the spare tire. The signal in Figure 47 was a false reading of either a signal or noise, while Figure 48 shows two of the signals received that has the expected shape of a TPMS signal. The signals in Figure 48 are speculated to have come from different vehicles since the TPMS warning light was never triggered during this part of the testing.

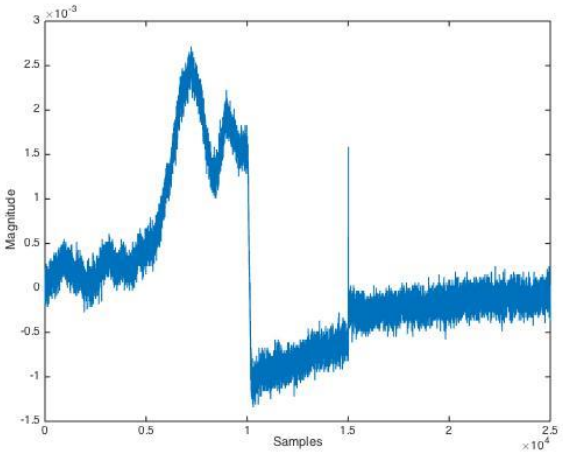


Figure 47 Signal received while driving Ford Fiesta with spare tire installed. This signal is most likely noise because it does not have the shape of TPMS signals encountered so far throughout testing.

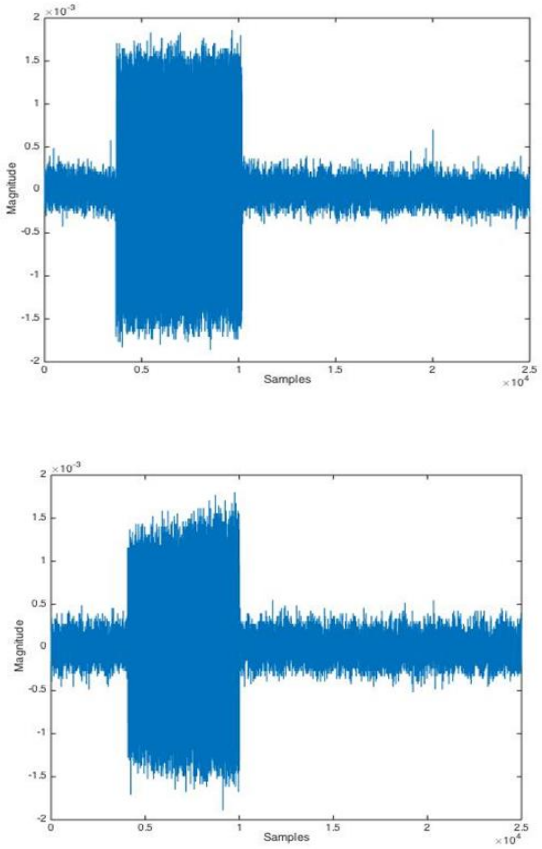


Figure 48 TPMS signals received while driving Ford Fiesta with spare tire on. The signals have the shape of a TPMS signal but the team believes that they belong to other vehicle's TPMS sensors that were passing by since the TPMS warning light was never triggered on the Ford Fiesta during the first attempt at this test..

The TPMS signals received were not decoded using the Benchmark SDR TPMS Receiver. That could be because the signals were of other cars with different IDs as the

vehicle's TPMS light never turned on proving that the vehicle's TPMS system is not active, therefore the decoding based on the Ford Fiesta's TPMS IDs could not happen. Another explanation could be the structure of the signal; if the ID was not in the location expected by the decoder functions, the receiver would not be able to decode it properly. Overall, the interaction between the Benchmark SDR TPMS Receiver and the TPMS of the Ford Fiesta was not successful.

One possibility for not having the light turn on might be that the car wasn't driven in a high speed for an extended period of time. The maximum was 35 mph for a few seconds and the rest of the time, the speed was lower than 25 mph. For this reason the next step was to perform this test while driving at least 25mph, for a minimum of 2 minutes. After driving at 50mph for 10 minutes the vehicle did not trigger the TPMS warning light on the vehicle's dashboard. While driving the Benchmark SDR TPMS Receiver was running to see if any signals were picked up or decoded but it is speculated that none of the signals received belonged to the Ford Fiesta since they were not successfully decoded. Such signals were similar to those shown in Figure 48.

In order to draw some conclusions on the communication of the Benchmark SDR TPMS Receiver and the Pseudo TPMS Transmitter with a vehicle's TPMS, the team performed one last test on a 2014 Subaru Outback. The same procedure as in the previous test was followed. Similarly to what is depicted on the schematic of 46. The front left tire was removed and replaced by the spare tire. Following the tire change, the car was driven for approximately 20 to 25 minutes with an average speed of 35 mph until the TPMS light turned on. Then, the team proceeded to turn on both the Pseudo TPMS Transmitter and the Benchmark SDR TPMS Receiver connected to laptops and ran the codes for both while continuing to drive the car around Worcester.

The Pseudo TPMS Transmitter was used to send a signal with tire pressure of 227 kPa (33 psi) and the outside temperature of 47 degrees Fahrenheit, using the tire ID of the front left

tire which was replaced by the spare. The purpose of this transmission was to mimic the "missing" tire's signal and turn the light off. At the same time, the Benchmark SDR TPMS Receiver was used to intercept the signal of the remaining three tires using their IDs. The tire IDs were provided to the team by Professor Wyglinski, the owner of the 2014 Subaru Outback, and are shown on Table 8.

Table 8 Subaru Winter Tire IDs

Tire Position	ID in hexadecimal
Front Left	00AA1BB4
Front Right	00AA1B49
Rear Right	00AA15E7
Read Left	00AA1CB5

4.4 Summary of Implementation Challenges

The team's ultimate goal was to manage to transmit a signal to the TPMS Rx in a vehicle. In order to do that, it was essential to have a clear understanding of the TPMS transmitter and receiver, how they communicate, what kind data their signals exchange as well as their architecture. Throughout the implementation phase of this project, the team faced a series of challenges while familiarizing with the equipment. Those challenges are analyzed in this section.

One of the challenges that the team faced early on in the implementation of Tx to Rx communication was synchronization. Within the first few tests between Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver as well as TPMS Tx and Benchmark SDR TPMS Receiver, the team noticed that communication was not stable and many times the Rx side would not recognize the transmitted signal. The team attributed this miscommunication to synchronization issues. The approach taken to overcome the synchronization challenge was to make a real-time receiver that would search continuously until a signal above a specific

threshold is identified. This method was successful in overcoming the challenge of synchronization.

Another aspect of the TPMS that the team needed to gain expertise on was the TPMS receiver architecture. One of the main research focus was to figure out how the receiver connects to an ECU in the vehicle. That includes whether it connects to an ECU that controls other functions of the vehicle, whether the receiver itself has an embedded computer or whether the TPMS ECU has exclusively that function. More specifically, if the TPMS ECU performs other tasks besides monitoring tire pressure, a false signal sent to it could cause complications to other systems. This challenge became more complicated when the team realized that analyzing the connection between the TPMS ECU and the TPMS receiver cannot be done simply by a model [57]. There are a wide variety of differences in CANBus connections from one vehicle manufacturer to another; therefore in order to study the TPMS connections within a vehicle the team decided to test on a whole vehicle rather than a TPMS Rx by itself. In addition, when the team attempted to get information on how a TPMS Rx is connected to an ECU from a car dealership, it was explained that that information is proprietary and therefore unable to gain access to.

To get a better understanding of the interaction between a vehicle CANBus and the role of TPMS Rx in relation to an ECU, the team also attempted to gather information from car manufacturing companies in USA. The companies refused to share their proprietary information regarding particular vehicle's CANBus network and how the TPMS Rx functions within this network. This response was expected as the companies tend to keep such information confidential due to competitive reasons but this fact became an obstacle for the team which had to rely on information and understanding gathered through experimentation.

To summarize the challenges of implementing the Pseudo TPMS Transmitter and how the team worked on identifying and resolving them, Table 9 is provided:

Table 9 Summary of implementation challenges throughout the testing phases of the Pseudo TPMS Transmitter.

Issue	Identification	Solution
Synchronization	"no signal found" error message	• Continuous signal detection with real-time receiver • Setting appropriate power thresholds
TPMS Receiver Variation	• Research on TPMS Receivers • Field testing with Ford Fiesta	Personal communication with expert [57]

4.5 Chapter Summary

This section discusses the methods and approach used to achieve successful communication between the Pseudo TPMS Transmitter and the Benchmark SDR TPMS Receiver. In order to ensure proper functionality of the Pseudo TPMS Transmitter, the team devised tests to evaluate the communication between both the TPMS sensor and the Pseudo TPMS Transmitter with Benchmark SDR TPMS Receiver. First of all, the team simulated the Benchmark SDR TPMS Receiver and Pseudo TPMS Transmitter through MATLAB by using the Benchmark SDR TPMS Receiver code and created a corresponding Pseudo TPMS Transmitter code. The simulation was used to confirm the construction of the signal and communication was established within MATLAB without any hardware. Then the team moved on to fully implementing and establishing communication between the Benchmark SDR TPMS Receiver and the TPMS Tx with the USRP hardware. Simultaneously, the team worked on developing the Pseudo TPMS Transmitter to successfully communicate with the Benchmark SDR TPMS Receiver wirelessly using the USRP. A series of modifications and updates were made to the original Benchmark SDR TPMS Receiver MATLAB code to resolve the issues of synchronization and signal power thresholds that arose with the use of hardware. Finally, the team used the Pseudo TPMS Transmitter to mimic and replace an actual TPMS sensor's signal

in a vehicle. Throughout the testing phase, the team managed to identify and resolve a series of challenges before achieving a successful implementation of this project.

5 Experimental Results

In this chapter, the results of the implementation of the four different tests described in Chapter 4 are presented. The first test is a simulation of the communication between the Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver through MATLAB. The following three tests are performed over-the-air using USRPs. Each section of chapter 5 is dedicated to one of the four tests with screenshots and pictures of the results the team obtain during each test.

5.1 Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver with MATLAB Simulation

The first milestone that the team achieved in terms of implementing and testing parts of this project was performing a successful MATLAB simulation of the communication between the Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver. The reasoning behind starting from this simulation is to obtain some results without the interference of noise and other issues that might result from wireless communications. Without the SDRu hardware implemented in the code, the receiver created by the previous MQP group successfully decoded the signal generated by the Pseudo TPMS Transmitter. This indicates that the mimicked TPMS signal was constructed correctly and is equivalent to that of the real TPMS sensors.

Initially, a signal was transmitted with values for temperature and pressure, as seen in the first line of Figure 49. Then, those values were compared with the output of the simulation.

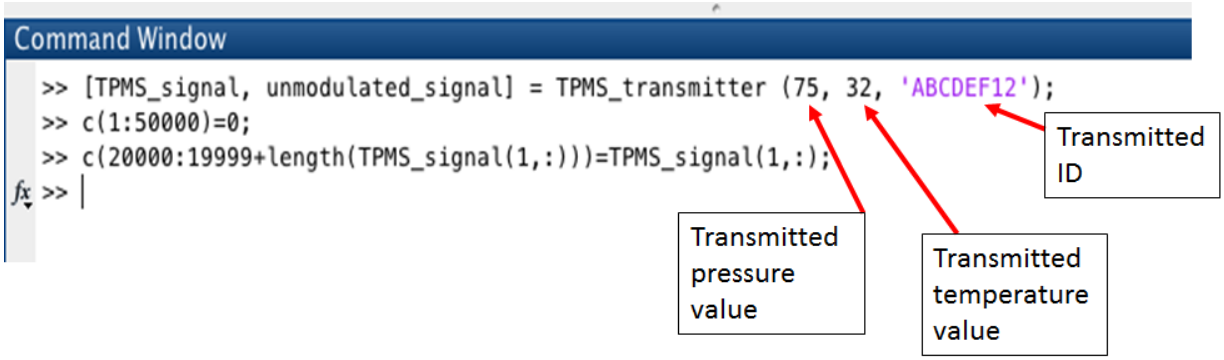


Figure 49 The first line demonstrates the transmission of a TPMS signal in simulation. Temperature is 75 degrees and pressure is 32 psi.

The results are illustrated in Figure 50, where the team calls on the decoding function which returns the values of pressure and temperature that it has decoded. As seen in the workspace, the temperature and pressure readings decoded are equal to those transmitted in Figure 49. This means that the mimicked signal has been built correctly and is able to be decoded by the Benchmark SDR TPMS Receiver.

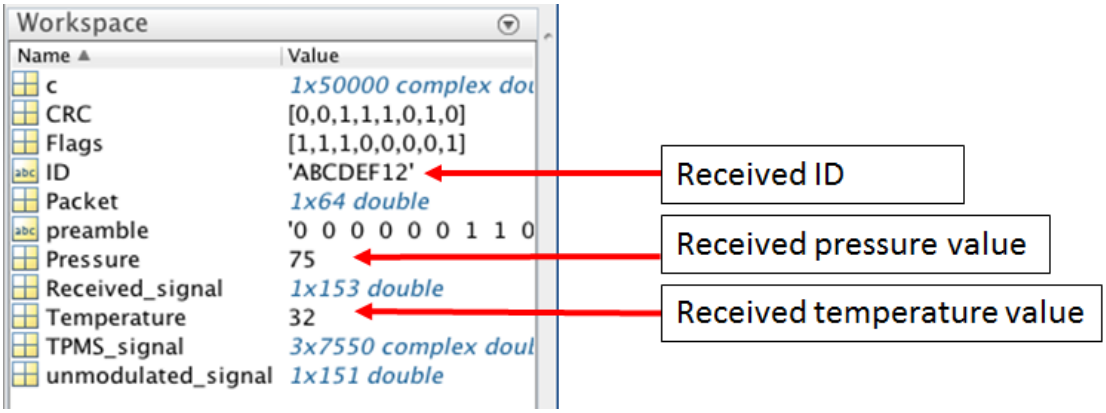


Figure 50 This image shows the values of the received signal's parameters, therefore validating that the signal transmitted is equivalent to the signal received during simulation.

Those results are important because they verify that the signal transmitted by the Pseudo TPMS Transmitter can be decoded by the same receiver that decodes the “real” TPMS signals. Therefore, the two signals should be very similar if not identical.

5.2 Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver with USRP Interface

With the use of the correct gain, power threshold, and synchronization explained in Section 4.3.2 the team was able to receive a clean signal, as illustrated in Figure 51. This implementation resulted in successful transmission and decoding of the signal transmitted by the Pseudo TPMS Transmitter.

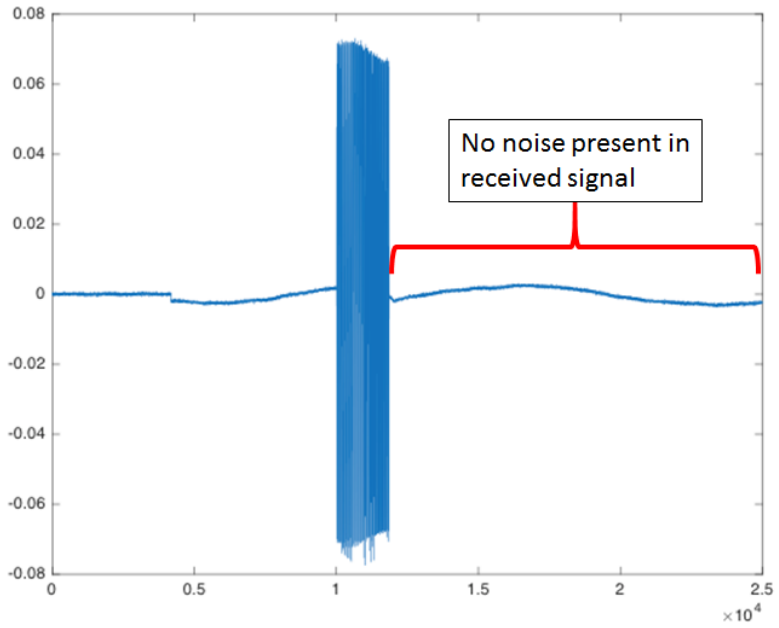


Figure 51 Clean signal received during testing with no noise present.

Once the Benchmark SDR TPMS Receiver had correlated the data, it was able to decode the values for pressure and temperature specified in the Pseudo TPMS Transmitter. These values matched those received by the Benchmark SDR TPMS Receiver perfectly. Figure 52 illustrates the values that were inputted in the Pseudo TPMS Transmitter, those being 75 for temperature, 32 for tire pressure, and sensor ID '8178E561'. The received and decoded values can be seen in Figure 53. These values for pressure, temperature, and sensor ID are also 32, 75, and '8178E561', respectively.

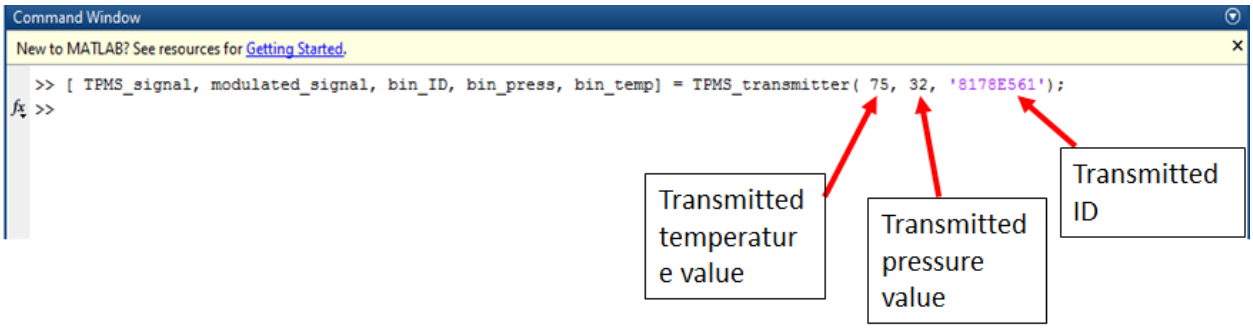


Figure 52 Inputted values for pressure (32), temperature (75), and sensor ID ('8178E561') transmitted using the Pseudo TPMS Transmitter.

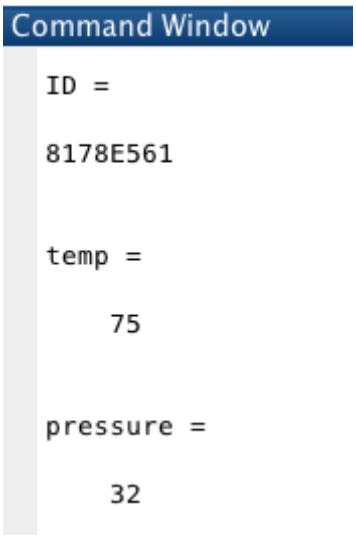


Figure 53 Received values for pressure (32), temperature (75), and sensor ID ('8178E561') in the Benchmark SDR TPMS Receiver.

5.3 Replicate Benchmark SDR TPMS Receiver with TPMS Tx

With the alterations to the values of gain and thresholds shown in Table 6, the team achieved a successful reception and decoding of the TPMS signal from the sensor in the tire. As seen in Figure 54, the Benchmark SDR TPMS Receiver received the real temperature and pressure readings of the TPMS as well as the sensor ID. Those values are 71 degrees (room temperature), 175 kPa, and and the correct TPMS sensor ID of '8178E561'.

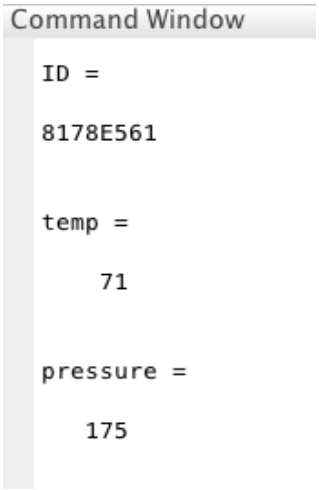


Figure 54 Received values for pressure (175kPa), temperature (71 F), and sensor ID ('8178E561') transmitted through the TPMS sensor.

To ensure that the Benchmark SDR TPMS Receiver could decode information from both the TPMS sensor and the Pseudo TPMS Transmitter, the team ran again the Pseudo TPMS Transmitter - Benchmark SDR TPMS Receiver test (Section 4.3.2 - Pseudo TPMS Transmitter

and Benchmark SDR TPMS Receiver with USRP Interface), this time with the updated values presented in Table 5. Using the modified values from Section 4.3.3 the team achieved successful communication between the Benchmark SDR TPMS Receiver and Pseudo TPMS Transmitter, thus proving that the signal from the Pseudo TPMS Transmitter is equivalent to that of the TPMS sensor, since it can be received and decoded by the exact same Benchmark SDR TPMS Receiver.

5.4 Transmit using Pseudo TPMS Transmitter to TPMS Rx

Two different approaches were utilized to get results for communication between Pseudo TPMS Transmitter and TPMS Receiver. In the first approach, the team covered one of the tires on the vehicle with a metallic shielding to turn on the TPMS warning light. The result was that after 25 minutes of having the car on, the TPMS light did not turn on. Even the Benchmark SDR TPMS Receiver did not receive any signal from any of the tires of the vehicle. The team got a “no signal found” error message in MATLAB when the Benchmark SDR TPMS Receiver did not pick up any signal from the vehicle’s tire as seen in Figure 55.

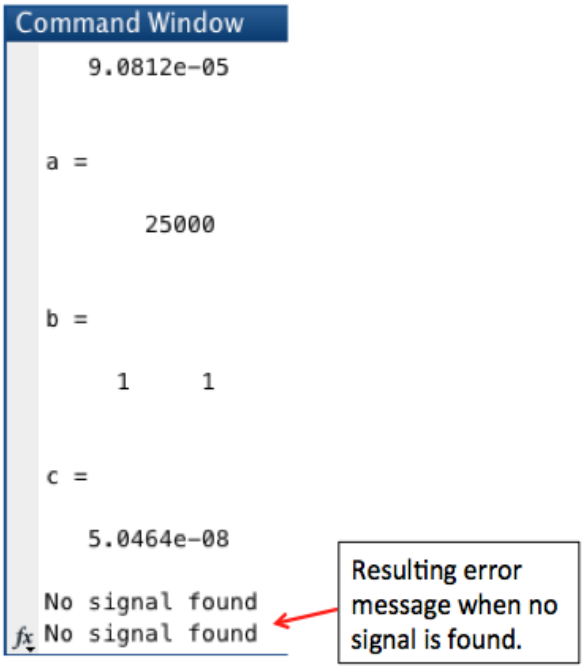


Figure 55 No signal found error message when no signal was picked up by Benchmark SDR TPMS Receiverfrom the TPMS Transmitter.

In the second approach, one of the vehicle's tires was replaced with the spare tire, which did not have a TPMS sensor installed. In an attempt to realize if the TPMS sensors and receiver were on, the team connected a Benchmark SDR TPMS Receiver in the car's cigarette lighter plug using an inverter and received a series of signals at 315 MHz. The team drove at a speed higher than 20 mph continuously for a time duration of 10 minutes.

Unfortunately, this test resulted in no triggering of the TPMS warning light. After 10 minutes on the highway going at an average speed of 50mph, it was expected that the warning light would be triggered due to the spare tire being installed and having the other tire stored away far from the reach of the TPMS receiver located in the vehicle. The signals received by the Benchmark SDR TPMS Receiver are similar to those of Figure 48. It is speculated that such signals belonged to other vehicles driving by since, had they belonged to the Ford Fiesta, the TPMS warning light should have been activated during that drive if the sensors in the vehicle had been transmitting data and communicating with the vehicle's TPMS receiver.

The test with the Subaru yielded some results that helped the team draw conclusions about the communication of the Pseudo TPMS Transmitter and an actual vehicle's TPMS. While driving around the Subaru and transmitting using the Pseudo TPMS Transmitter, the light that had originally turned on because of the spare tire did not turn off. Therefore, the team could not prove successful communication between the Pseudo TPMS Transmitter and the Subaru's TPMS Receiver. However, by using the Benchmark SDR TPMS Receiver, the team managed to receive the vehicle's TPMS signals from the three tires, using the identified tire IDs from Table 8 in Section 4.3.4. The received TPMS signal of the Subaru's active TPMS sensors is demonstrated on Figure 56, generated by the MATLAB script running the Benchmark SDR TPMS Receiver.

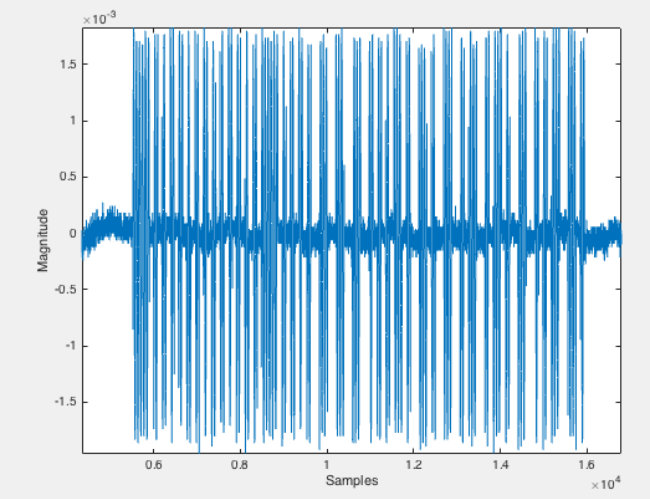


Figure 56 Received signal of Subaru's TPMS sensor using Benchmark SDR TPMS Receiver.

The signal is composed of "bursts" of high magnitude signal samples and then large gaps in between. Compared to the signal received by the same receiver when transmitting using the Dorman TPMS sensor on the tire in lab (Figure 51), the signal has significant differences. The Dorman TPMS signal is a lot more compact, unlike the Subaru's signal with the bursts and gap layout.

The Benchmark SDR TMPS Receiver also managed to receive the Pseudo TPMS Transmitter's signal in a similar manner as in Section 5.2. The signal sent by the Pseudo TPMS Transmitter aiming to communicate with the Subaru's receiver was picked up and decoded by the team's receiver and the results can be observed on Figure 57.

```
Command Window

ID =

AA1BB4


temp =

    47


pressure =

   227
```

Figure 57 The Pseudo TPMS Transmitter signal mimicking the missing tire's information as received by the Benchmark SDR TPMS Receiver.

The tire ID received is the one that corresponds to the front left tire of the Subaru, proving that the Pseudo TPMS signal was, indeed, transmitted. The temperature and pressure values are those inserted to the transmitted and transmitted.

The same receiver was used with a random ID (“ABCDEF12”), used for testing purposes in lab, in an attempt to understand why the Benchmark SDR TPMS Receiver was decoding data from the Pseudo TPMS Transmitter when given a similar ID to decode. The Benchmark SDR TPMS Receiver successfully received and decoded the following signal from a passing car with a tire ID similar to the test ID “ABCDEF12”. This came as a surprise to the team but it revealed that the Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver were designed to work according to specific manufacturer standards for TPMS signal construction. The received signal and the information decoded are demonstrated on Figures 58 and 59.

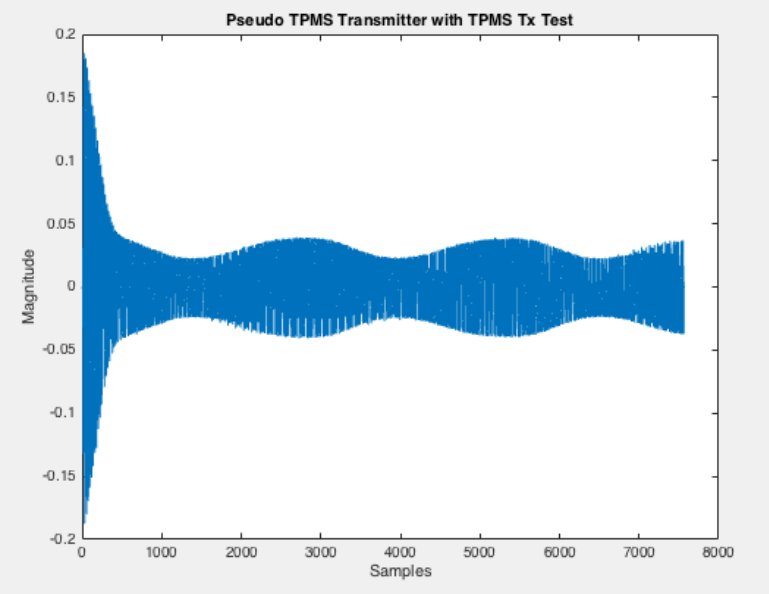


Figure 58 Signal of passing vehicle’s TPMS sensors as received by the Benchmark SDR TPMS Receiver.

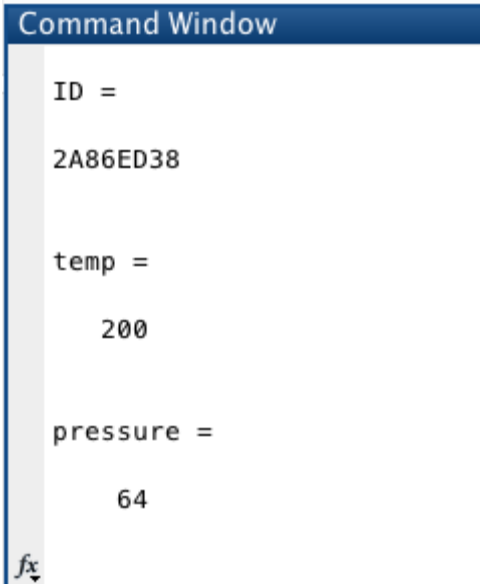


Figure 59 Decoded information from passing vehicle's TPMS signal.

The signal received was most likely decoded because the ID has some similar characters as the test ID ("ABCDEF12"). The temperature and pressure readings were switched in the specific signal but the rest of the architecture was the same as the one used to create the Benchmark SDR TPMS Receiver, therefore the signal was successfully decoded. The actual temperature received was 64 degrees Fahrenheit probably due to the fact that the vehicle had been in motion and caused the tires to record a higher temperature. The tire pressure of the passing vehicle's tire was 200 kPa (29 psi), which is a normal value for tire pressure. These findings prove the Benchmark SDR TPMS Receiver's interaction with an actual vehicle's TPMS sensors is possible. However, it was not possible with the TPMS of the 2014 Subaru Outback, possibly due to the fact that the manufacturer of the TPMS sensors and receiver are different and therefore the signal architecture is different (as proven by Figure 56). In the future, the Pseudo TPMS Transmitter could be modified to successfully replicate the signals of more TPMS sensor manufacturers. This would expand its capabilities to communicate with more vehicles and make it universal.

5.5 Chapter Summary

This chapter discusses the results gathered during the testing of the Pseudo TPMS Transmitter. Throughout the four testing phases the team ensured that the created transmitter was functioning properly. For the first test, Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver with MATLAB simulation, the team concluded that the transmitter created was encoding the data correctly and that Arnold and Piscitelli's receiver [11] was decoding the values properly. For the Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver with USRP interface test the team found that the transmitter and receiver were communicating properly and that the data was successfully decoded. The third test, replicate Benchmark SDR TPMS Receiver with TPMS Tx, showed successful reproduction of previous results from Arnold and Piscitelli's project using a real TPMS sensor placed inside a tire. For the final test on the Ford Fiesta involving a real TPMS receiver the team found that activating the TPMS light was not as straight-forward as one would think. Both attempts to turn on the light did not yield the results needed to test the Pseudo TPMS Transmitter. Finally, the Subaru test managed to turn on the TPMS light but not to communicate with the vehicle's receiver.

6 Conclusion

With the advancement of electronic technologies in vehicles, the use of such electronics is becoming a commonplace in vehicles. These technological developments have allowed vehicles to include a myriad of safety and comfort features that have improved the overall driving experience of users. These features use different sensors, which communicate and interact with each other through a network of ECUs in a CANBus and therefore leave the vehicle exposed to many hacking vulnerabilities through such sensors. The team evaluated different wireless access points for the purpose of this project and chose the Tire Pressure Monitoring System (TPMS) because of the prior literature and research readily available.

The outcome of this project has exposed the team to a significant amount of information on the TPMS technology and its vulnerabilities. The results of the testing between this team's transmitter, Alex Arnold and Stephanie Piscitelli's receiver and the TPMS sensor revealed the structure of the TPMS signal, which lead to the construction of a semi-successful Pseudo TPMS Transmitter. The final test conducted in the real environment of a vehicle was catalytic for the conclusion of this project about the vulnerability of the TPMS system and therefore the team hopes that this phase of testing is continued by a following MQP project.

The goal of the initial testing was to verify the communication between the Pseudo TPMS Transmitter and Benchmark SDR TPMS Receiver within a MATLAB simulation. The successful results of this test allowed the team to conclude that the coding part of the transmitter was correct, as well as the decoding of the receiver into the correct information. This first test was essential in the continuation of the project since it verified that the software logic was correct. The next step was to test the same software over a USRP interface.

As one can imagine, over-the-air communication brought a lot more challenges than the simulation. Once those challenges were overcome, the successful communication between the team's transmitter and Arnold and Piscitelli's receiver with USRP interface led to the conclusion

that the software is correct and it was interfaced correctly to communicate wirelessly. This step was crucial since direct TPMS is a system that is based exclusively on wireless communication between the TPMS sensors and the TPMS receiver in the vehicle. By establishing good communication between the team's Pseudo TPMS Transmitter and the modified Benchmark SDR TPMS Receiver, the project was one step closer to communicating with the real TPMS of a vehicle.

In order to establish communication between the Pseudo TPMS Transmitter and the vehicle's TPMS, the team first decided to test the Benchmark SDR TPMS Receiver with the vehicle's TPMS sensors. This test and its results was important in order to verify that the software and USRP interface used in this project could communicate successfully with the "real" TPMS sensors. Through Arnold and Piscitelli's receiver, the team managed to read tire pressure and temperature from a tire used for testing purposes. These results were very encouraging since they let the team conclude that the system built could communicate with a real, commercial TPMS sensor.

Once the team was sure that the receiver created by Arnold and Piscitelli was working properly with both the team's transmitter and real TPMS sensor, the final test involving communication between team's transmitter and a TPMS receiver was conducted. In order to test the capabilities of the Pseudo TPMS Transmitter, the team first attempted to trigger the TPMS warning light in a 2012 Ford Fiesta, so that then a signal from the transmitter would turn the light off. However, after trying to trigger the light by sealing one of the tires and then replacing one of the tires with a spare tire, the light did not turn on. Therefore, there were no conclusions from performing this test since the Pseudo TPMS Transmitter signal was never sent to an actual vehicle's TPMS Receiver.

After the failed testing on Ford Fiesta, the team tested the Pseudo TPMS Transmitter on the 2014 Subaru Outback provided by Professor Alexander Wyglinski. After replacing one of the tires with a spare tire and driving for more than 20 minutes, the TPMS warning light turned on

but the Pseudo TPMS Transmitter failed to turn off the light. The team concluded that the Pseudo TPMS Transmitter signal was based on the signal construction of DORMAN TPMS sensors and different TPMS sensor manufacturers use different signal construction. This is the reason why the TPMS receiver of the Subaru did not recognize the signal of the Pseudo TPMS Transmitter. The Benchmark SDR TPMS Receiver did not pick up any signal from the Subaru's tires as well, which supports the conclusion that the Subaru TPMS sensors are from a different manufacturer that construct the TPMS signal differently than the DORMAN TPMS sensors.

The results and conclusions of this project are only the beginning of the possibilities to explore a vehicle's CANbus through the Tire Pressure Monitoring System. The team has shown that there is a relation between signals sent from a USRP coded in MATLAB and those of an actual vehicle, and has provided a good ground for further research on the possibilities once someone can access the TPMS receiver. This project may have only gone as far as replicating the TPMS signal and testing it against a Benchmark SDR TPMS Receiver, but this has opened the way for further exploring the other pathways from the TPMS receiver to different parts of a vehicle's digital network.

One of the recommendations for future work would be to further investigate the architecture of the CANbus and more specifically, that of the TPMS receiver and its interaction with other parts of the network. Knowledge of the setup of the TPMS receiver would give more ideas of how to use the wireless access achieved in this project to modify different parts of the CANbus, alter information, activate and deactivate features and more. In order to achieve that, part of the research should be to further explore the capabilities of the TPMS receiver in receiving longer signals, and what kind of signals would be needed to modify different parts of the vehicle's digital systems.

Naturally, a good continuation of this project would be to interface the Pseudo TPMS Transmitter with the TPMS receiver of a vehicle in a real environment and test this interaction successfully. In other words, completing the test on an actual vehicle that the team was not able

to finalize. Also, another way to continue this project is to come up with a system to protect the TPMS from being “hacked” or the signal from being “sniffed.” Successful communication between the Benchmark SDR TPMS receiver and the TPMS signal transmitted by a vehicle’s sensor in a tire showed that there is no encryption in the wireless signals between TPMS sensors and the TPMS receiver. This, of course, can be dangerous if a system similar to this project is developed to attack commercial vehicles and introduce malicious signals that could harm the vehicle and its passengers. It is recommended that the TPMS is protected further and the results of this project can be used as a basis for developing a protection system for TPMS.

The pseudo TPMS transmitter developed in this project sends an FSK modulated signal to the receiver. However, market research has shown that there are also TPMS sensors that use ASK modulation in their signal processing. For this reason, one more recommendation for future work in this field would be the construction and testing of an ASK pseudo transmitter, that would mimic the real-life ASK TPMS sensors. Also, different manufacturers construct the TPMS signal differently and future work can be based on exploring other manufacturers of TPMS sensors as well. Development of such a transmitter would enhance knowledge on the TPMS and its vulnerabilities, since all types of sensors would be explored equally.

This project is solely based on the direct TPMS by exploring its technology, the types of signals the system is based on and its components (transmitter and receiver). However, there are numerous vehicles that have indirect TPMS instead, which is based on the calculation of the tire’s diameter based on the rotational speed. Indirect TPMS is part of the Anti-Lock Braking System and was not explored in this project. The team recommends an implementation of a similar project that would mimic the behavior of the indirect TPMS, in order to offer more in depth knowledge of the vulnerabilities of TPMS.

Conclusively, the team’s Pseudo TPMS Transmitter leads to many more possible investigations that can take place to better understand TPMS and its vulnerabilities. It is important to explore all facets of the TPMS in order to protect vehicles from malicious

interception and alteration. Future work does not have to stop at wireless TPMS, but can continue on multiple access points of a vehicle's electronic system. An enhanced understanding of automotive security will bring together a future of safe vehicles without having to compromise all the comfort that digital systems offer.

7 Appendix A - Benchmark SDR TPMS Receiver

7.1 TPMS Receiver

```
function [ packet] = TPMS_receiver2( TPMS_signal, power_threshold )
%TPMS_receiver
% This function takes in the received signal from a TPMS and will
% demodulate and decoded the packet
%close all
%clc
% Sample rate used on USRP
Fs = 100e6/128;
%These are the thresholds used to determine if a signal is present
threshold = 20*power_threshold
%% Reformatting Signal
rx = TPMS_concat(TPMS_signal);
%% Traverse the entire received signal looking for TPMS signal
lower_ind = 1;
%lower_ind:lower_ind+10
a = length(rx)
b = size(bandpower(rx(lower_ind:lower_ind+10)))
c = bandpower(rx(lower_ind:lower_ind+10))
dataRx = rx(lower_ind:lower_ind+10);
while (bandpower(rx(lower_ind:lower_ind+10)) < threshold) && (lower_ind < length(rx))
  lower_ind = lower_ind + 1;
  if lower_ind >= 24991
     disp('No signal found');
     break;
  end
end
%locating the end point
upper_ind = length(rx);
while bandpower(rx(upper_ind-10:upper_ind)) < threshold && upper_ind > 1
  upper_ind = upper_ind - 1;
     if upper_ind <= 10
     disp('No signal found');
     break;
  end
end
%holds the wave form of just the packet (no noise)
packet_waveform = rx(lower_ind:upper_ind);
%demodulate the packet
figure
plot(1:length(packet_waveform),packet_waveform)
packet = demodulator( packet_waveform, Fs );
%decode the packet
%decode_packet(packet);
%figure(2)
%plot(1:length(packet_waveform), real(packet_waveform));
end
```

7.2 Real-Time Receiver

```
function [packet, data] = TPMS_live_receiver_1()
packet = 0;
close all
```

```
clc
radio = comm.SDRuReceiver('Platform', 'N200/N210/USRP2', 'IPAddress', '192.168.20.2',
'CenterFrequency', 315000000, 'EnableBurstMode', true, 'OutputDataType', 'double', 'DecimationFactor',
128, 'SampleRate', 100e6/128, 'Gain', 16, 'NumFramesInBurst', 5, 'FrameLength', 5000);
%hss = dsp.SpectrumAnalyzer('SampleRate', 100e6/128);
data(5000,1) = 0; % = zeros(10000, 2001);
for m = 1:5
    TPMS_signal = step(radio);
    %step(hss, TPMS_signal);
    data(:,m) = TPMS_signal;
end
% %% Reformatting Signal
%   columns = 5000;
%   TPMS_signal = TPMS_concat(TPMS_signal);
%   TPMS_signal = reformat(TPMS_signal, columns);
%   [r,c] = size(TPMS_signal)
power_threshold = 50*bandpower(data(:,1))
packet(1,500) = 0;
packets = 0;
i = 2;
while 1 %i<r-1   %changed to inf while
   %add a buffer of 4 to 5 columns
   TPMS_signal = step(radio);
   %step(hss, TPMS_signal);
   %data(:, 1:4) = data(:, 2:5);
   data(:,i+3) = TPMS_signal;
   if (bandpower(data(:,i)) > power_threshold)
       low_ind = i-1
       upper_ind = i+3
        while i < upper_ind
            i = i + 1;
        end
       figure
       [a,b] = TPMS_concat(data(:, low_ind:upper_ind)');
       plot(b,a)
       packets = packets + 1

       d = size(data(:, low_ind:upper_ind)')
       e = data(:, low_ind:upper_ind)';
       test = TPMS_receiver2(data(:, low_ind:upper_ind)', power_threshold);
       packet(packets,1:length(test)) = test;
       [preamble, ID, temp, pressure, flags, crc, packet] =
TPMS_decode_by_ID_second('8178E561',packet(packets,1:length(test)))
       break;
   else
        i = i + 1;
        if (i == 5000)
            i=2;
        end
   end
end
end
```

7.3 Decode by ID

```
function [preamble, ID, pressure, temp, flags, crc, packet] = TPMS_decode_by_ID_second( ID,
TPMS_bits )
%this function assumes that the ID comes after the packet information
% this function takes the ID as a Hax string and the packet
%finds the start of the packet
[m, ind] = find_ID(ID, TPMS_bits);
ind = ind - length(TPMS_bits) + - 31;
%takes the preamble
preamble_bits = TPMS_bits(1:ind-1);
preamble_bits = TPMS_bits(1:ind-1);
%decodes the rest of the packet
packet = man_decode(TPMS_bits(ind:end));
%fills up each field with the bits then calculates the values
pressure_bits = packet(1:8);
temp_bits = packet(9:16);
ID_bits = packet(17:48);
flags = packet(49:56);
crc = packet(57:end);
count = 7:-1:0;
preamble = num2str(preamble_bits);
temp = sum(temp_bits .* 2.^count);
pressure = sum(pressure_bits .* 2.^count);
ID = dec2hex(sum(ID_bits .* 2 .^ (31:-1:0)));
%solves for a CRC pattern
CRC_pattern(packet, 9);
End
```

7.4 CRC Pattern

```
function [ good_patterns ] = CRC_pattern( packet, p )
%CRC_pattern
% this function takes in the packet and the length of the pattern and
% solves for a CRC pattern using brute force
%% init vars
dec_nums = 0:2^p-1;
pattern_str = dec2bin(dec_nums);
pattern = zeros(2^p, p);
%% initializing the patterns array that will be checked
for x = 1:2^p
  for y = 1:length(pattern_str(x,:))
     pattern(x,y)= str2double(pattern_str(x,y));
  end
end
n = length(packet);
k = n - p + 1;
pattern(2^(p-1)+1,:);
good_patterns = zeros(1, p);
good = 1;
%% cycles through all potential patterns
for y = 2^(p-1)+1:2^p
  y;
  x = 1;
  div = packet(1:p);
  a = 0;
```

```
  while div(1) == 0 && x+a < k
        div(1:end-1) = div(2:end);
        div(end) = packet(x+a+p);
        a = a + 1;
  end

  x = a + x;

  while x < k
     div = xor(div, pattern(y,:));
     a = 0;
     while div(1) == 0 && x+a < k
        div(1:end-1) = div(2:end);
        div(end) = packet(x+a+p);
        a = a + 1;
     end
     x = a + x;

  end
  % If the pattern works put it in the good patterns array
  if sum (div == zeros(1,p)) == p || sum(div == pattern(y,:)) == p
     good_patterns(good, :) = pattern(y,:);
     good = good + 1;
  end
end
end
```

7.5 Find ID

```
function [  m, ind  ] = find_ID( ID, signal )
%find_ID
%This function takes the ID in Hex and the packet and solves for the
%location of the ID within the packet using corr
% The function outputs the max value and its index
%% Convert the ID to bin and then encode it
ID_binary = Hex_to_Bin(ID);
encoded_ID = man_encode(ID_binary);
%% perform corr and take the max val and index
acor = xcorr(signal, encoded_ID);
figure
plot(1:length(acor), acor)
xlabel('Index')
ylabel('Correlation')
title('Correlation of Encoded ID and Encoded TPMS Packet')
[m, ind] = max(acor);
End
```

7.6 Manchester Encode

```
function [ encoded_signal ] = man_encode( signal )
%man_encode
% this function takes in a packet and manchester encodes it
%encoded signal is twice the length
encoded_signal = zeros(1, 2*length(signal));
```

```
for x = 1:2:length(encoded_signal)
  if signal((x+1)/2) == 1
    encoded_signal(x) = 1;
  else
    encoded_signal(x) = 0;
  end
  encoded_signal(x+1) = xor(encoded_signal(x), 1);
end
end
```

7.7 Manchester Decode

```
function [ decoded ] = man_decode( encoded )
%man_decode
%This function takes in a manchester encoded signal and decodes it
x=floor(length(encoded)/2)
decoded = zeros(1, x);
%takes every other value starting at 1
for x = 1:2:length(encoded)
  %making sure two succesive bits are not the same
  if encoded(x) ~= encoded(x+1)
    decoded((x+1)/2) = encoded(x);
  else
    decoded = -1;
    break;
  end
end
end
```

7.8 Concatenate Signal

```
function [ out, samples ] = TPMS_concat( in )
% This function takes in the input signal in the form of a multidemensional
% array and concatenates them into a single demension
[x, y] = size(in);
out = zeros(1, x*y);
samples = 1:x*y;
for a = 0:x-1

  out(a*y+1:(a+1)*y) = in(a+1,:);

end
end
```

7.9 Demodulator

```
function [ packet] = demodulator( rx, Fs )
%demodulator
% Takes in the packet and the sample rate
%This function determines if it uses ASK or FSK and then demodulates the
%signal and passes the demodulated packet out
%perform the FFT and get magnitude
freq = abs(fft(rx));
```

```
figure(3)
plot(1:length(freq), freq)
%range to prevent side bins from influencing results
range = 10;
[max1,ind1] = max(freq);
%Determines the two max bins that are not closely adjacent
if ind1 - range > 0 && ind1 + range <= length(freq)
   freq(ind1-range:ind1+range) = 0;
elseif ind1 - range < 0
   freq(1:ind1+range) = 0;
   freq(length(freq)+ind1-range:length(freq)) = 0;
else
   freq(ind1-range:length(freq)) = 0;
   freq(1:(ind1+range)-length(freq)) = 0;
end
[max2, ind2] = max(freq);
%determines if the second frequency is large enough to be FSK
if max2 > max1/2
   [packet] = FSK_demodulator(rx,Fs);
else
   [packet] = ASK_demodulator(rx,Fs);
end
end
```

7.10 ASK Demodulator

```
function [ packet ] = ASK_demodulator( rx )
%ASK_demodulator
% Function performs ASK demodulation of the TPMS packet
%% Variable setup
interp_val = 2;
rx_interp = interp(rx, interp_val);
lenRx = length(rx_interp);
rx_rect = abs(rx_interp);
threshold = max(rx_rect/2);
%figure
%plot(1:lenRx, rx_rect)
%% Demodulation
rx_square = zeros(1, lenRx);
for x = 1:lenRx
   if rx_rect(x) > threshold
      rx_square(x) = 1;
   end
end
%figure
%plot(1:lenRx, rx_square);
%% Down sampling
packet = down_sample(rx_square);
end
```

7.11 FSK Demodulator

```
function [ packet] = FSK_demodulator( rx, Fs )
%FSK_demodulator
%This function performs FSK demodulation on the signal and outputs the
```

```
%packet
%% variable intialization
lenRx = length(rx);
t = (1:lenRx)/Fs;
interp_val = 20;
bits_per_packet = 750;
samp_per_sym = floor(interp_val * lenRx / bits_per_packet);
down_samp = interp_val * lenRx - bits_per_packet * samp_per_sym;
interval = floor(interp_val * lenRx / down_samp);
%% Adjusting the offset frequency
freq = max_frequencies(rx, Fs,2);
offset = -(freq(1)  + freq(2))/2;
mod_sig = exp(j*2*pi*t*offset);
rx = rx .* mod_sig;
%% Calculating the freq separation
freq = max_frequencies(rx, Fs, 2);
freq_sep = abs(freq(1)) + abs(freq(2));
%% downsamples so there is an symbols per bits divides evenly
down_samp_sig = zeros(1, bits_per_packet * samp_per_sym);
rx_interp = interp(rx, interp_val);
for y = 1:down_samp
   down_samp_sig((y-1) * (interval-1) + 1:y * (interval - 1)) = rx_interp((y-1) * interval + 1:y * interval - 1);
end
%% Demodulates packet
over_packet = invert(fskdemod(down_samp_sig,2,freq_sep,samp_per_sym, Fs*interp_val));
%% Because more bits were output than needed a downsample is used
packet = down_sample(over_packet);

%figure
%plot(1:lenRx, abs(fft(rx)))
End
```

7.12 Down Sample

```
function [ down ] = down_sample( packet )
%down_sample
% This function takes in a packet with excess bits from demodulation. and
% downsamples based on the small number of consequtive bits
%% Starts a litte in because sometimes the first bits are compromised
ind = 50;
val = packet(ind);
lenPacket = length(packet);
%% finding the start of the next change so it does not throw off count
while packet(ind) == val && ind < lenPacket
   ind = ind + 1;
end
%% Calculates the min number of consequtive bits and is the default single bit
min_count = lenPacket;
while ind  < lenPacket-20

   val = packet(ind);
   count = 0;
   while packet(ind) == val && ind < lenPacket
      count = count + 1;
      ind = ind + 1;
   end
```

```
  if count < min_count
     min_count = count;
  end
end
%% performs the down sampling based on the min count above
down = 0;
down_ind = 1;
ind = 1;
while ind  < lenPacket

  val = packet(ind);
  count = 0;
  while packet(ind) == val && ind < lenPacket
     count = count + 1;
     ind = ind + 1;
  end

  count = floor((count)/min_count);
  if count == 0
     count = 1;
  end
  a = 1;
  while a <= count
     down(down_ind) = val;
     down_ind = down_ind + 1;
     a = a+1;
  end

end
end
```

8 Appendix B - Pseudo Transmitter

8.1 Transmitter Code

```
function [ TPMS_signal, modulated_signal, bin_ID, bin_press, bin_temp] = TPMS_transmitter(
dec_temperature, dec_pressure, ID )
[bin_temp, bin_press] = decimal_to_binary(dec_temperature, dec_pressure);
bin_ID = Hex_to_Bin(ID);
flags = [1 1 1 0 0 0 0 1];
[data] = [bin_temp bin_press bin_ID flags];
checksum = comm.CRCGenerator('Polynomial', [1 1 1 0 1 0 1 0 1]);
CRC_data = step(checksum, data.');
unencoded_data = [CRC_data.'];
encoded_data = man_encode (unencoded_data);
preamble = step(comm.BarkerCode('Length', 13, 'SamplesPerFrame', 23)).';
fixed_preamble = fix_preamble(preamble);
unmodulated_signal = [fixed_preamble encoded_data];
modulated_signal = FSK_modulator(unmodulated_signal);
TPMS_signal = [modulated_signal; modulated_signal; modulated_signal];
end
```

8.2 Live Transmitter Code

```
function [] = TPMS_transmitter_live(modulated_signal)
radio = comm.SDRuTransmitter('EnableBurstMode',true,'NumFramesInBurst',5,'Gain', 16,'Platform',
'N200/N210/USRP2', 'IPAddress', '192.168.10.2', 'CenterFrequency', 315000000, 'InterpolationFactor',
128, 'UnderrunOutputPort', true);


output = modulated_signal.';
output = padarray(output, [10000 0]);
%hss = dsp.SpectrumAnalyzer('SampleRate', 100e6/128);


for counter = 1:5
  step(radio, output);
  %step (hss, output);
end
end
```

8.3 Decimal to Binary Conversion Code

```
function [ bin_temperature, bin_pressure ] = decimal_to_binary( dec_temperature, dec_pressure )


bin_temperature = de2bi(dec_temperature, 8, 'left-msb');
bin_pressure = de2bi(dec_pressure,8, 'left-msb');


end
```

8.4 Hexadecimal to Binary Conversion Code

```
function [ bin ] = Hex_to_Bin( ID )
```

```
%Hex_to_Bin
% this function takes in the ID in a hex string and converts it to a
% binary array


binStr = dec2bin(hex2dec(strcat('A', ID)));
bin = zeros(1, length(binStr));


for x = 1:length(bin)
   bin(x)= str2double(binStr(x));
end
   bin = bin(5:end);
end
```

8.5 Manchester Encoding Code

```
function [ encoded_signal ] = man_encode( signal )
%man_encode
% this function takes in a packet and manchester encodes it


%encoded signal is twice the length
encoded_signal = zeros(1, 2*length(signal));


for x = 1:2:length(encoded_signal)


   if signal((x+1)/2) == 1
      encoded_signal(x) = 1;
   else
      encoded_signal(x) = 0;
   end
   encoded_signal(x+1) = xor(encoded_signal(x), 1);


end
```

8.6 Fix Preamble Code

```
function [fixed_preamble] = fix_preamble( preamble )
preamble(preamble < 0) = 0;
fixed_preamble = preamble;
end
```

8.7 FSK Modulator Code

```
function [FSK_signal] = FSK_modulator (unmodulated_signal)
mod_order = 2;
freq_sep = 73737;
Fs = 100e6/128;
nsamp = 50;
FSK_signal = fskmod(unmodulated_signal, mod_order, freq_sep, nsamp, Fs);
```

end

References

[1] Chong, A. (2015). THE GROWTH OF AUTOMOTIVE ELECTRONICS IN APAC, THE NEXT FRONTIER. 13-27. Retrieved February 2, 2016, from http://www.infineon.com/export/sites/default/cn/drivingasia_chap1.pdf

[2] Markey, E. (2015). Tracking & Hacking: Security and Privacy Gaps Put American Drivers at Risk. Retrieved March 3, 2016, from http://www.markey.senate.gov/imo/media/doc/2015-02-06_MarkeyReport-Tracking_Hacking_CarSecurity%202.pdf

[3] Federal Motor Vehicle Safety Standards; TPMS; Controls and Displays. Final Rule. (2005). Retrieved February 2, 2016, from http://www.nhtsa.gov/cars/rules/rulings/TPMSfinalrule.6/TPMSfinalrule.6.html

[4] NHTSA (2006). Safety Problem. Retrieved February 2, 2016, from http://www.nhtsa.gov/cars/rules/rulings/TirePresFinal/safetypr.html

[5] NHTSA (2006). TIRE PRESSURE SURVEY AND TEST RESULTS. Retrieved February 2, 2016, from http://www.nhtsa.gov/cars/rules/rulings/TirePressure/LTPW3.html

[6] Google Self-Driving Car Project. (n.d.). Retrieved January 21, 2016, from https://www.google.com/selfdrivingcar/how/

[7] Gibbs, S. (2015, November 20). Tesla hits the gas on self-driving car tech. Retrieved January 20, 2016, from http://www.theguardian.com/technology/2015/nov/20/tesla-self-driving-car-tech

[8] A. Greenberg. (2015, July 21). *Hackers Remotely Kill a Jeep on the Highway—With Me in It* [Online]. Available: http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

[9] J. Heary. (2010, July 31). *Defcon: Hacking Tire Pressure Monitors Remotely* [Online]. Available: http://www.networkworld.com/article/2231495/cisco-subnet/defcon---hacking-tire-pressure-monitors-remotely.html

[10] I. Rouf et al (2015). *Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study* [Online]. Available: http://www.winlab.rutgers.edu/~Gruteser/papers/xu_tpms10.pdf

[11] A. Arnold and S. Piscitelli, “TPMS Receiver Hacking”, Worcester Polytechnic Institute, Worcester, MA, 2015.

[12] E. Laurens. (2015) *Car Computer History* [Online]. Available: http://www.ehow.com/about_5082250_car-computer-history.html

[13] N. Chandrashekara. (2015). *Basics of Automotive ECU* [Online]. Available: http://www.etas.com/data/group_subsidiaries_india/20140121_ETAS_Webinar_ECU_Basics.pdf

[14] Canbuskit. (2015).*What is Canbus* [Online]. Available: http://canbuskit.com/what.php

[15] B. Wojdyla (2012) *How it Works: The Computer Inside Your Car* [Online]. Available*:* http://www.popularmechanics.com/cars/how-to/a7386/how-it-works-the-computer-inside-your-car/

[16] J. Ferris. (2015). *Car hacking is real and dangerous* [Online]. Available: http://www.komando.com/tips/318332/car-hacking-is-real-and-dangerous-protect-yourself/all

[17] S. Checkoway et al. (2011). *Comprehensive Experimental Analyses of Automotive Attack Surfaces* [Online]. Available: http://www.autosec.org/pubs/cars-usenixsec2011.pdf

[18] National Instruments. (2014, August 01). *Controller Area Network (CAN) Overview* [Online]. Available: http://www.ni.com/white-paper/2732/en/

[19] Kvaser. (2015).*CAN Protocol Tutorial* [Online] Available: http://www.kvaser.com/can-protocol-tutorial/

[20] Texas Instruments. (2015). *Introduction to the Controller Area Network (CAN)* [Online]. Available: http://www.ti.com/lit/an/sloa101a/sloa101a.pdf

[21] AutoTap. (2011). *OBD-II Background.* Available: http://www.obdii.com/background.html

[22] Office of Transportation and Air Quality. (2015, March 16). *OBD-II Basic Information* [Online]. Available: http://www3.epa.gov/obd/basic.htm

[23] E. Evenchick. (2013, October 29). *CAN Hacking: Protocols* [Online]. Available: http://hackaday.com/2013/10/29/can-hacking-protocols/

[24] AutoTap. (2011). *OBD-II The Connector.* Available: http://www.obdii.com/connector.html#which%20cars

[25] Automotive IQ (2011, June 8), *Automotive Diagnostic Systems: History of OBD* [Online]. Available: https://automotiveiq.wordpress.com/2011/06/08/automotive-diagnostic-systems-history-of-obd/

[26] A. Goodwin. (2010). *A Brief Intro to OBD-II Technology* [Online]. Available: http://www.cnet.com/news/a-brief-intro-to-obd-ii-technology/

[27] A. Barisani and D. Bianco. (2007, May 27). *Hijacking RDS TMC traffic information signal.*[Online]. Available: http://phrack.org/issues/64/5.html

[28] Inversepath. (2007). *Unusual Car Navigation Tricks: Injecting RDS-TMC Traffic Information Signals* [Online]. Available: http://dev.inversepath.com/rds/cansecwest_2007.pdf

[29] Windytan. (2013, 04 May). *A determined 'hacker' decrypts RDS-TMC* [Online]. Available: http://www.windytan.com/2013/05/a-determined-hacker-decrypts-rds-tmc.html

[30] Evans, C. (2015, March 24). 98 Percent of Americans Are Connected to High-Speed Wireless Internet. Retrieved January 21, 2016, from https://www.whitehouse.gov/blog/2015/03/23/98-americans-are-connected-high-speed-wireless-internet

[31] Newcomb, D. (2014, December 29). Tapping Into In-Car WiFi | Edmunds.com. Retrieved January 21, 2016, from http://www.edmunds.com/car-technology/tapping-into-in-car-wifi.html

[32] Audi connect | Audi USA. (2015). Retrieved January 21, 2016, from https://www.audiusa.com/technology/intelligence/audi-connect

[33] M. Brain. (2015). *How Remote Entry Works* [Online]. Available: http://auto.howstuffworks.com/remote-entry2.htm

[34] Trw. (2015). *Remote Keyless Entry: Passive Entry* [Online]. Available: https://www.trw.com/sites/default/files/TRW_ge_rkepe_en.pdf

[35] R. Sorokanich, 'Hacker's $30 Device Unlocks Just About Any Keyless Entry Car', *Road & Track*, 2015. [Online]. Available: http://www.roadandtrack.com/new-cars/car-technology/news/a26327/hacked-keyless-entry/

[36] E. Ngah (2006, August 15). *GPS Technology: Optimizing Car Navigation* [Online]. Available: https://www.math.vu.nl/~sbhulai/theses/werkstuk-ngah.pdf

[37] F. Hussain. (2015, August 9). Hacking GPS Signals of Smartphones and In-Car Navigation System. Retrieved October 5, 2015, from https://www.hackread.com/hacking-smartphones-gps-in-car-navigation-system/

[38] R. Neal. (2013, July 29). *GPS Terrorism: Hackers Could Exploit Location Technology To Hijack Ships, Airplanes* [Online]. Available: http://www.ibtimes.com/gps-terrorism-hackers-could-exploit-location-technology-hijack-ships-airplanes-1362937

[39] Bridgestone Tires. (2015). *What is TPMS & How Does it Work?* [Online]. Available: http://www.bridgestonetire.com/tread-and-trend/drivers-ed/tire-pressure-monitoring-system-how-tpms-works

[40] E. Baxter. (2015). *How Tire Pressure Monitoring Systems Work* [Online]. Available: http://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/tire-pressure-monitoring-systems.htm

[41] Carley, L. (2008, October 1). The Ins and Outs of Indirect and Direct TPMS. Retrieved January 21, 2016, from http://www.underhoodservice.com/the-ins-and-outs-of-indirect-and-direct-tpms/

[42] TPMS Made Right. (2010). Direct TPMS Versus Indirect TPMS. Retrieved January 21, 2016, from http://tpmsmaderight.com/direct_vs_indirect.php

[43] Texas Instruments. (2015). *Software Defined Radio* [Online]. Available: http://www.ti.com/solution/software-defined-radio-sdr-diagram

[44] Wireless Innovation. (2015). *What is SDR?* [Online]. Available: http://www.wirelessinnovation.org/assets/documents/SoftwareDefinedRadio.pdf

[45] Pu, D., & Wyglinski, A. M. (2013). *Digital communication systems engineering with software-defined radio*. Boston: Artech House.

[46] D. McCarthy (2015). *Evaluating and optimizing RFID and NFC systems using real-time spectrum analysis* [Online]. Available: http://defenseelectronicsmag.com/site-files/defenseelectronicsmag.com/files/archive/rfdesign.com/mag/604RFDF2.pdf

[47] MATLAB. USRP® Support from Communications System Toolbox. Retrieved January 28, 2016, from http://www.mathworks.com/hardware-support/usrp.html?refresh=true

[48] National Instruments. (2014, November 13). *Amplitude-Shift Keying, Frequency-Shift Keying, and Phase-Shift Keying* [Online]. Available: http://www.ni.com/white-paper/7824/en/

[49] Rahmani, M. (n.d.). Digital Modulation. Retrieved January 28, 2016, from http://ee.mouloudrahmani.com/Electrical/Communication/DigitalModulation.html

[50] Frenzel, L. (2012, January 23). Understanding Modern Digital Modulation Techniques. Retrieved January 28, 2016, from http://electronicdesign.com/communications/understanding-modern-digital-modulation-techniques

[51] TPMS - Tire Pressure Monitoring System. (n.d.). Retrieved January 28, 2016, from http://www.discounttire.com/dtcs/infoTPMSArticle.do

[52] R. Verdult, F. Garcia, B. Ege (2013). *Dismantling Megamos Crypto: Wirelessly Lockpicking a Vehicle Immobilizer* [Online]. Available:https://www.usenix.org/sites/default/files/sec15_supplement.pdf

[53] MEGAMOS CRYPTO ID48 CAN AUDI (MG10/A4/TP25/EL30) GLASS TRANSPONDER CHIP. (n.d.). Retrieved January 28, 2016, from http://key-code.com/tiendaeuropa/en/3483-megamos-crypto-id48-can-audi-mg10a4tp25el30-glass-transponder-chip.html

[54] Rouse, M. (n.d.). What is transponder? - Definition from WhatIs.com. Retrieved January 28, 2016, from http://internetofthingsagenda.techtarget.com/definition/transponder

[55] S. Mattera. (2014, October 29). *Garmin Ltd. Tumbles As Market for GPS Units Declines* [Online]. Available: http://www.fool.com/investing/general/2014/10/29/garmin-ltd-tumbles-as-market-for-gps-units-decline.aspx

[56] Mazda, X., & Mazda, F. (1999). *Focal Illustrated Dictionary of Telecommunications*. Oxford: Focal.

[57] Technology Training Limited (2015). BPSK - Binary Phase Shift Keying [Online]. Available: http://www.technology-training.co.uk/bpsk.php

[58] Hristos Giannopoulos. Student Researcher on CANBUS, WPI. Interview. Worcester, MA. December 9, 2015.

[59] Sloka (2012). *Binary Phase Shift Keying* [Online]. Available:

http://www.sloka.in/resources/glossary/66-binary-phase-shift-keying-bpsk

[60] N. Amin et al., “A BPSK Backscatter Modulator Design for RFID Passive Tags”, presented at the Radio-Frequency Integration Technology, 2007. RFIT 007. IEEE International Workshop, Singapore, 2007.

[61] Infineon, *Infineon.com*, 2015. [Online]. Available: http://www.infineon.com/dgdl/SP37_LF_v1.0.pdf?fileId=db3a30433899edae0138be4e8e2931af

[62] Arts Automotive. (2015). *Tires – TPMS (Tire Pressure Monitoring System)* [Online]. Available: http://artsautomotive.com/publications/8-automotive/81-tires-tpms-tire-pressure-monitoring-system/

[63] Embedded Systems Academy. (2015).*CANbus* [Online]. Available: http://www.canbus.us/

[64] A. Francillon et al. “Relay attacks on passive keyless entry and start systems in modern cars”, ETH Zurich, Switzerland, n.d.

[65] Icpdas. (2015). *PISO-CAN 200/400: Linus SocketCAN CAN bus manual* [Online]**.** Available FTP: ftp.icpdas.com/pub/cd/fieldbus_cd/can/pci/pcm_piso-can_series/driver/linux_can_driver/socketcan/linux_socketcan_can_bus_manual.pdf

[66] Instructables. (2015). *Programming the Arduino to accept messages from the CAN-BUS* [Online]. Available: http://www.instructables.com/id/Hack-your-vehicle-CAN-BUS-with-Arduino-and-Seeed-C/step2/Programming-the-Arduino-to-accept-messages-from-th/

[67] Kernel.org. (2015). *Readme file for the Controller Area Network Protocol Family (aka SocketCAN)* [Online]. Available: https://www.kernel.org/doc/Documentation/networking/can.txt

[68] K. Nice (2015). *How Car Computers Work* [Online]. Available: http://auto.howstuffworks.com/under-the-hood/trends-innovations/car-computer2.htm

[69] Obdiicom. (2015). *OBD - II Acronyms and Jargon* [Online]. Available: http://www.obdii.com/acronyms.html

[70] Opengarages.org. (2014). *Car Hacker's Handbook*. : ThelaLabs Publication.

[71] Openxcplatform. (2015). *Openxcplatform.com* [Online]*.* Available: http://openxcplatform.com/

[72] M. Szczys (2011). *CAN sniffing for steering wheel button presses* [Online]. Available: http://hackaday.com/2011/03/08/can-sniffing-for-steering-wheel-button-presses/

[73] Usenix.org. (2015). *Supplement to the Proceedings of the 22nd USENIX Security Symposium* [Online]. Available: https://www.usenix.org/sites/default/files/sec15_supplement.pdf

[74] Vector. (2015).*Controller Area Network (CAN)* [Online]. Available: https://vector.com/vi_controller_area_network_en.html

[75] W. Xu, “Analyzing the security and privacy vulnerability of emerging wireless networks via software defined radio: a Tire Pressure Monitoring System case study,” Dept. Comp. Science and Eng., UMass, Amherst MA, 2015

[76] Gibbs, S. (2015, November 20). Tesla hits the gas on self-driving car tech. Retrieved January 20, 2016, from http://www.theguardian.com/technology/2015/nov/20/tesla-self-driving-car-tech

[77] McHugh, M. (2015, October 14). Tesla’s Cars Now Drive Themselves, Kinda. Retrieved January 20, 2016, from http://www.wired.com/2015/10/tesla-self-driving-over-air-update-live/

[78] Fleming, B. (2014, December). Advances in Automotive Electronics. *Vehicular Technology Magazine, IEEE*, *9*(4), 4-12.