SYSTEM INTEGRATION OVER A CAN BUS FOR A SELF-CONTROLLED, LOW-COST AUTONOMOUS ALL-TERRAIN VEHICLE

by

Karim H. Erian

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering

Charlotte

2019

Approved by:

_____
Dr. James M. Conrad

_____
Dr. Hamed Tabkhi

_____
Dr. Aidan Browne

ABSTRACT

KARIM H. ERIAN. System Integration over a CAN Bus for a Self-Controlled, Low-Cost Autonomous All-terrain Vehicle. (Under the direction of DR. JAMES M. CONRAD)

This thesis presents research for making an All-terrain Vehicle (ATV) autonomous. The objective of this research is to have a standardized platform for autonomous ATV modules having the basic controllers integrated and working. This research includes standardizing data collection modules and control modules, and integrating all of these functional modules in one working system. The work involved building new modules, enhancing existing modules, and using already working (and standardized) modules. This effort includes electronic hardware, software, and mechanical designs. The overall system consists of actuators modules including: a steering controller, a throttle controller, and a braking system. The system also contains various sensor modules including a GPS sensor and an IMU sensor. All modules are attached to a CAN Bus which is connected to a central controller. The system design and implementation was demonstrated by the vehicle autonomously driving in a straight line, in a circle, and in a figure '8' path.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Prof. Dr. James M. Conrad for being very supportive during this research and for providing his time to support me with the needed advice to make my research a success.

I would like also to thank my committee members Dr. Hamed Tabkhi and Dr. Aidan Browne for their guidance and support.

Most appreciated is for the Fulbright Bi-national Commission, the Department of State, and the Amideast team for providing me with this opportunity to do my Master's degree by offering me the prestigious Fulbright Scholarship and giving me the guidance and support during my study in the States.

I would like also to thank Eddie Hill from the lab team for providing his time and support with all the lab equipment needed to reach these results, and the ECE graduate school staff for there continuous guidance and support.

Certainly my lab mates are very appreciated for their technical support and continuous help, especially Shantanu Mhapankar for his work and support in providing me the libraries needed and the help during the testing, and Stuart Gambill for his contribution in the mechanical parts.

I mainly would like to acknowledge my parents for the kind support they provided me to accomplish my thesis.

Least but not last, I thank My Lord and God for His precious gifts for making everything possible. I would not be able to reach results without His support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ATV An abbreviation for All-Terrain Vehicle.

AutoSAR An abbreviation for AUTomotive Open System ARchitecture

CAN An acronym for Controlled Area Network.

ECE An acronym for Electrical and Computer Engineering.

ECU An abbreviation for Electronic Control Unit.

EPS An abbreviation for Electrical Power Steering.

GPS An acronym for Global Positioning System.

H/W An abbreviation for Hardware.

HWI An abbreviation for Hardware-Software Interface Layer.

I2C An abbreviation for Inter-Integrated Circuit.

IMU An acronym for Inertial Measurement Unit.

K15 An acronym for Ignition Car Key.

LIN An abbreviation for Local Interconnect Network.

OEM An acronym for Original Equipment Manufacturers.

PCB An abbreviation for Printed Circuit Board.

PS An abbreviation for Power Steering.

PWM An abbreviation for Pulse Width Modulation.

SAE An abbreviation for Society of Automotive Engineering.

SPI An abbreviation for Serial Peripheral Interface.

UART An abbreviation for universal asynchronous receiver-transmitter.

UNCC An acronym for University of North Carolina At Charlotte.

CHAPTER 1: INTRODUCTION

The worldwide automotive industry is moving toward autonomous vehicles and self-driven cars. The studies started a few years ago, and the competition between inveterate mechanical cars Original Equipment Manufacturers (OEMs) and the new high technological electrical cars OEMs is increasing daily. Autonomous vehicles are still in an early phase of research, and those produced by some OEMs are very expensive to be afforded by average employees [1].

Self-driven cars have many benefits and applications. These state-of-the-art machines may contribute to saving non-productive hours for drivers commuting daily to and from their work area to be used in more useful tasks. They may also contribute to public transportation and shipments deliveries. More essential features are traffic safety and a massive reduction in the percentage of car accidents. All autonomous vehicles will be programmed to follow traffic rules by the law. They will keep correct lane placement, respect the speed limits and the traffic lights. They will also be able to detect objects like human or animals passing the streets or other road obstacles [2].

Autonomous vehicles will lead to a significant reduction in the number of car accidents [3]. Similar machines may be used in more prestigious jobs than just commuting. They may be used to deliver emergency supplies in hurricanes areas, earthquake zones, and others. Those areas are not reachable by conventional vehicles, and the roads may be dangerous for human drivers. Another usage to be considered would be the discovery of new locations in the woods or deserts. While having those machines moving around, recording everything in their ways, human operators can escape the danger of encountering wild animals, fallen trees, quicksand or others.

This research focuses on one type of autonomous vehicles which is how to make an All-Terrain-Vehicle (ATV) autonomous. The main goal is to make the ATV understudy move from an initial position to a final destination and stop without any human interface.

The University of North Carolina at Charlotte started working with Zapata Engineering in 2009 on a project to make an Autonomous ATV. Zapata Engineering participated by giving the University a Honda TRX420FE 2009 ATV. Since then, a lot of research effort has been conducted to make this ATV autonomous.

1.1 What was already there?

In the previous research, the ATV had the actuators throttle controller, braking system, and steering controller attached to the ATV. It also had to read from stationary sensors to identify the road. The vehicle was controlled using a remote control over Controlled Area Network (CAN) messages [4, 5, 6].

- **The Throttle Module:** The ATV in hand is equipped with a wire-driven throttle. The throttle module implemented, as a part of previous research, was benefiting from having a wire driven throttle initially controlled by a lever. The previous study had an electrical servo motor that simulates the wire movement. The servo motor is directly attached to the fuel pump. The motor is controlled through 3 wires used as power source, Ground, and Control signal. The 3 wires were connected to the Brain module. In this research, the same servo motor is used with an enhanced motor control circuit.

- **The steering module:** The Honda ATV was originally equipped by an electrical steering assistant module made by the OEM. Steering Assist module is not a full power steering system. This steering assistant is only triggered when a torque sensor detects a higher torque on the steering axis while trying to move

the steering wheel. The main idea of the steering module is to simulate high torque situation in one direction to make the steering assistant enter the scene and move the wheel. In previous research, a PCB is fabricated with a PIC micro-controller, a CAN transceiver, and an analog switch that controls the value of resistors that simulate the Torque Sensor output. The main idea of the circuit was to connect and disconnect some parallel resistors in order to obtain a certain resistance value corresponding to a steering angle value sent on the CAN bus [7, 8].

- **The Braking System Module:** On the other side, the ATV is equipped with 2 brakes controllers, one is a lever to control the front brakes, and the other is a pedal to control the back brakes. The braking system used in the previous research used to be a hydraulic actuator that is connected to the braking pedal. Only the hydraulic actuator is re-used in this version with a new controller.

- **The Stationary on-road sensors:** The ATV in the last research used to communicate with on-road sensors used as SLAM solution for locations without Global Positioning System (GPS) signals. The on-road sensors used to draw the path with respect to the vehicle, and the vehicle was controlled using a remote control. This sensor is not used in this research.

- **The Speed Module - Wheel Encoders:** As a way to communicate the car speed to be used with the steering module, a set of two wheel encoders was implemented and connected to the 2 rear tires. They were used to count the number of holes in the piece of plastic connected to the tire and using the count of holes, it detects the number of turns, which helps to calculate the rotational speed. Knowing the Tires dimensions, and the rotation speed, it is simple to get the ATV speed.

- **The Brain and the Remote Control:** All the previous modules were connected

to "The Brain" which was a Renesas evaluation and demonstration board called RX63N. The Brain was connected to a remote control receiver, and used to control the Throttle Module, the Braking System, and send CAN messages to control the steering module. There was no physical connection between sensors and actuators at this point except for the steering module that was getting readings from speed sensors and the steering wheel encoder that determine the current position of the axis [5].

1.2 Problem Statement

This study addresses the problem of transforming the old existing system into a fully autonomous ATV without the need for a human interface. Instead of controlling the ATV using a remote control, to make it self-controlled.

Besides, this study also addresses how to fix existing problems in the old system, like rebuilding damaged or missing boards, standardizing connections, communications, boards, power supply, the code used, and integrate the whole system together.

The initial state of the ATV at the beginning of the study: At the start point of this study, the ATV was parked in the university garage for a long period and needed some service to be able to reach the running state. A servo motor was attached to the vehicle for the throttle, and the hydraulic hand was connected to the braking pedal and fixed in the ATV. Also, the speed module was attached to the vehicle. However, those modules did not have controllers connected to them. Also, the steering module ECU was detached from the vehicle and may or may not be working. The steering module wires connected in place of the torque sensor, and to the steering angle encoder were existing and attached to ATV with all wires labeled.

1.3 What is my contribution?

The current research aims to full system integration to have an autonomous ATV without human interaction. In this thesis, in order to reach this goal, some milestones had been achieved first. It was necessary to standardize some of the existing modules, ameliorate and enhance other modules, have minimal re-use from the old modules, and build new modules as needed. By utilizing all lessons learned from the past 9 years, the final goal is to have the ATV self-controlled without the usage of the remote control, and without having any human interface during the ATV trips.

This version of research targets to re-use the steering module with minor adjustments with the assumption that it is in a good situation, knowing that the last attempt to connect it before this study begins had a case of wrong connections that may lead to module damage. In this case, a new module will be needed. In addition, this study also includes enhancement and rebuilding the throttle module and the braking system to avoid existing problems like breaking of the braking pedal due to overload. Moreover, one of the goals of this thesis is to standardize the CAN bus and power distribution network all over the ATV to avoid wrong connections which may lead to the damage of already existing modules which would result in losing more time and money trying to re-build them.

The new design for the system is to have a central processing unit connecting many small processing units, through CAN bus. This idea is used to distribute the processing power on different nodes in order to have a fast response system.

- **CAN Bus and Power distribution:** Following the standard way of communication in the automotive industry, the different modules should be communicating using CAN bus, and each module - if needed - should communicate internally using Local Interconnect Network (LIN) bus. In some cases, some OEMs may use FlexRay which is faster but more expensive than the CAN bus or Ethernet communication. In the current case, the communication is only using CAN

bus as it is used to connect all modules to the main processing unit together. As the CAN Bus is connected to all available modules, it is a perfect way to use the same bus to distribute power. This part of the study will be about how to physically implement the bus and the power distribution network while preventing the wrong connections. Besides, it will discuss the needed hardware and the software configuration.

- **Throttle module:** As mentioned before, the throttle module already has an electric servo motor attached to the vehicle. This part of the thesis will discuss how to control this motor using a new controller, the main processing unit, and the CAN bus.

- **Braking System:** In this section, a demonstration will illustrate a better mechanical way to connect the hydraulic actuator to the braking pedal to protect the pedal from braking. It will also discuss the process of controlling the brakes, the hydraulic actuator setup, and calibration, and all needed communications between all available hardware, the controller, the motor driver, and the hydraulic actuator. Besides, the software algorithm used will be presented.

- **Steering Module:** The old steering module is used but with a change of connector to match the whole system. In this section, a description of the interface needed to use the old steering module with more details on how to control it.

- **GPS and Compass Module:** While working on this thesis, another study is taking place in parallel and the results are going to be used in the final integration. The plan is to utilize an Adafruit GPS sensor and Adafruit 3-D compass sensor for location and direction purposes. The module should be read by a micro-controller which should send the readings as messages using CAN bus.

- **The Main Processing Unit:** This unit is the main Brain for the ATV, it will

be controlling all actuators according to the vehicle position and direction in order to reach the final destination. Working as a maestro, it does not directly control the actuators or decode the sensors, but it will communicate the orders to the specific controllers that control the actuators and get the data from the corresponding controllers that decode the sensors. This part of the thesis will include the hardware used and the algorithm for controlling the ATV.

- **Overall System integration:** The most important part of the thesis is integrating all previous modules together. Mainly, all sensors and actuators, are connected to an MSP430 either directly or through a middle controller like in the case of the Braking System. The MSP430 is connected by SPI signals to a CAN shield equipped with the CAN transceiver MCP2515. Except for the old steering module which uses a PIC micro-controller DSPIC30f4011 and MCP2551 CAN transceiver. The CAN shields are all connected to the CAN bus. Also, the CAN bus will serve as a power supply to other modules. The CAN bus is also connected to the central processing unit, which will read the position and the direction from the GPS and the 3-D compass, and accordingly communicate with the steering module, throttle module, and braking system. The ATV should only run on the first gear with a maximum speed of 10 mph.

- **System Testing:** This study includes three different types of testing with different strategies. Mainly at first, a module testing is used for each module to make sure it works alone without integrating it in the vehicle. Then, an integration testing will take place, by integrating two modules and adding at a time. Then integrating 3 modules for testing and so on. All integration testing shall be monitored and controlled for safety as well. Some of the test cases will be done first while lifting the vehicle from the ground. Other test cases will be done with a person riding the vehicle to take control in case of an emergency.

Finally, after all the modules are integrated, a system validation test shall take place, by inserting coordinates to the system and monitor the ATV reaction. System Validation testing shall be done in an empty parking lot in the University of North Carolina at Charlotte with the knowledge of the University Police department.

1.4 Limitations

This thesis is limited to move the ATV from one position to another autonomously, to reach the final destination despite initial location and orientation. This project does not provide obstacles handling at this time.

Also, this ATV under test has a manual gear transmission ATV. This study does not cater for automation of the manual gear transmission, since the ATV is planned to run on the first gear only with a maximum speed of 10 mph.

1.5 Significance of this Study

This thesis shall contribute to a proof of concept that it is possible to transform an already made ATV to an autonomous ATV with low-cost materials. This may encourage taking this study to further levels and integrating more sensors to have better control of the ATV.

As mentioned earlier, it is important to have autonomous ATV for many reasons including but not limited to the need to deliver emergency supplies to locations suffering from any catastrophic circumstances like hurricanes, earthquakes or others that destroyed or blocked rural roads or when the roads are in unsafe condition for human drivers to reach the final destination. Another reason is to be considered is the need to discover non-rural areas like woods or desserts for civilization or searching for lost campers.

1.6 Organization

This thesis is organized into 5 chapters. Chapter Two provides information about the previous studies that already took place on this subject, and the ATV initial status at the beginning of this study. Chapter Three describes my contribution to this research. This chapter provides information on the overall system design. It discusses the methodology used to standardize and enhance the already existing modules and to build the new modules. Chapter Four illustrates the system integration, the testing plans and execution, the testing results, and the steps taken to fix bugs found in the system. Finally, Chapter Five summarizes the conclusions reached by the end of this study and states some recommendations for future work and enhancements.

CHAPTER 2: PREVIOUS STUDIES AND ALL-TERRAIN VEHICLE INITIAL STATUS

This Chapter provides information about previous studies in the topic of the autonomous self-driven ATV. Chapter 2 goes through published papers and researches to understand what may be used in this thesis that serves its purpose. As an introduction to the author's contribution, this chapter demonstrates the initial status of the ATV at the beginning of this thesis.

The first paper published by the University of North Carolina at Charlotte in this project was published in 2010 [9]. The research was initiated by the Zapata Engineering company who found the need of a Robot that can tow a trailer through non-urban roads. The Zapata Engineering company provided the University of North Carolina a Honda ATV for a senior design project in order to make it a remote-controlled device.

In the previous research, a remote-control was used to control the ATV by sending commands to an evaluation board. This evaluation board controlled a throttle controller, a braking system, and a steering controller. The ATV also had to read from stationary sensors to identify the road [6, 9].

Many papers and research were done in this domain before. Below is a representation for some of the work done previously for different modules currently implemented and used in this research thesis:

- **Throttle System:** The actual throttle in the Honda ATV is designed as a wire-driven throttle with a spring return [9] where a wire is attached to a lever positioned at the right-hand thumb. This wire is connected on the other side by the gas pump valve. When the lever is pressed, the gas valve is open allowing

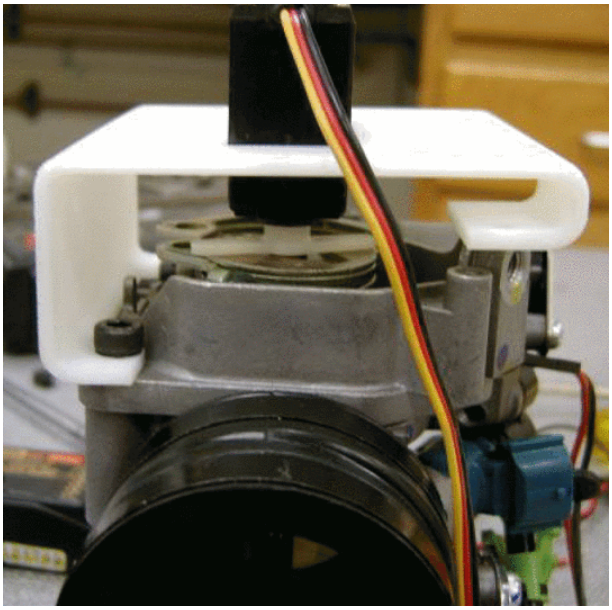

Figure 2.1: Servo attached to the Throttle assembly [9].

the gas to go to the engine. The more pressure on the lever, the higher speed the ATV will acquire. The first paper tackling this project, published in 2010, Throttle was controlled using a parallax servo motor. In order to attach the servo motor to the factory original throttle assembly, a bracket was designed and fabricated [9] as shown in image Figure 2.1. In this paper, the ATV was controlled using a remote control which communicates with a Renesas evaluation board. The board was directly controlling the throttle module.

The change to the throttle model in this study was changing the microcontroller used. At the start of this study, the throttle parallax servo motor was connected to the throttle original assemble as shown in Figure 2.1. It was not connected to a controller.

- **Steering System:** The Honda ATV is equipped by a Steering Assist module, which includes a Torque sensor. This sensor is responsible for making the Steering Assist module to either start engaging to assist the steering or not. Figure 2.2 shows the motor implemented by Honda for Steering assist module.

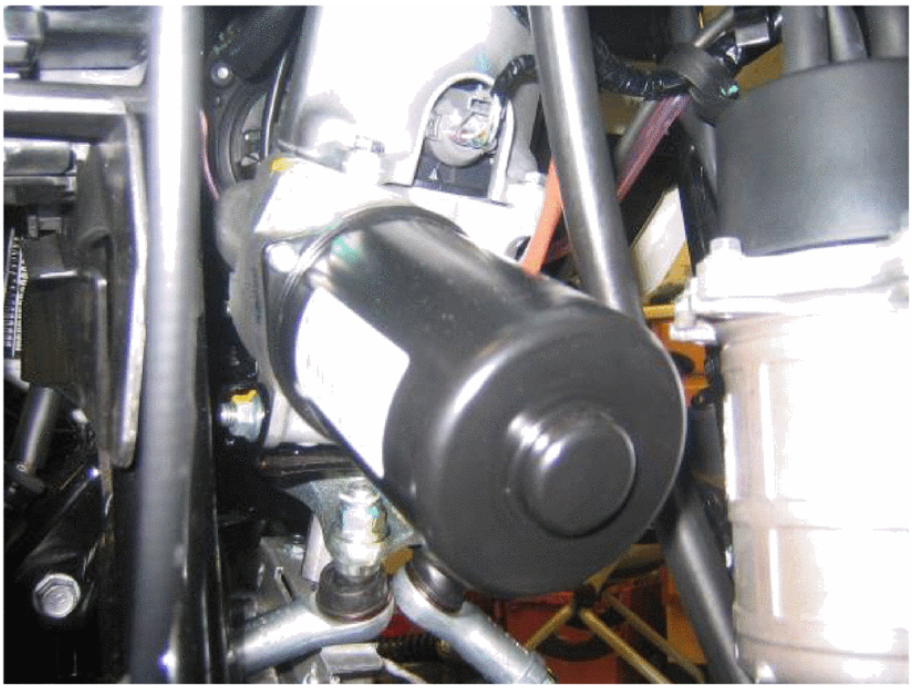

Figure 2.2: DC Power Steering Motor [9].

In the first paper published 2010 [9], It was initially assumed the system could be tricked by simulating the signal values to make it take control. This first paper had found it is not that easy to establish for some electrical issues with accurate values provided and for the steering assistant is not only depended on the torque sensor but also it is depended on the vehicle speed and other factors. In this paper, a dual H-bridge was used as a motor controller for the steering assist motor. Figure 2.3 shows the circuit implemented in this research.

Another paper for the same ATV was published two years later [8]. In this paper, a problem in the old design was addressed. The main issue for the first design was that the steering motor kept overheating. Overheating prevented the steering module from responding for all commands, and controlling the ATV was not perfect. It also may cause damage to initial ATV circuits. In the second paper, a PWM is used as input to an H-bridge instead of a continuous power signal. The PWM input signal reduces the amount of power input the motor

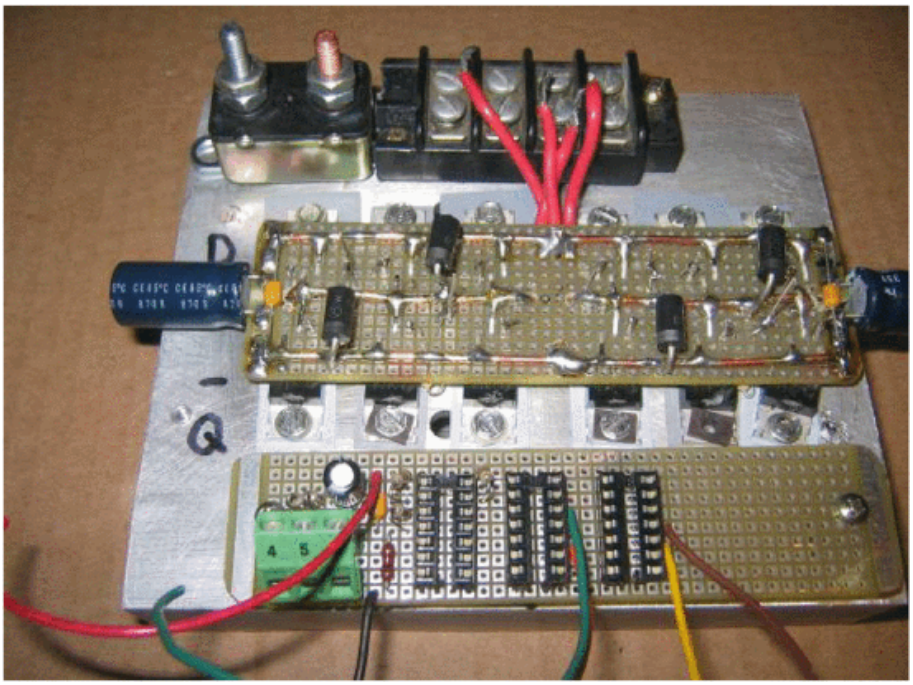

Figure 2.3: H-Bridge circuit attached to large Heat sink [9].

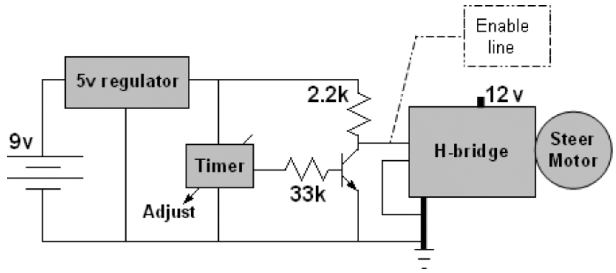

Figure 2.4: Steering PWM test circuit [8].

and reduces the overheating. A microcontroller introduces the PWM instead of having a constant voltage by a circuit. Figure 2.4 presents a diagram for testing the system using a PWM signal. Also Figure 2.5 demonstrates the circuit diagram using the micro-controller.

The steering assist module of the ATV kept overheating since it was not designed to be always energized. It was planned by Honda to work only in need according to the torque of resistance. A newer paper focused on the steering module [7] in which the torque sensor was used to control the power assist

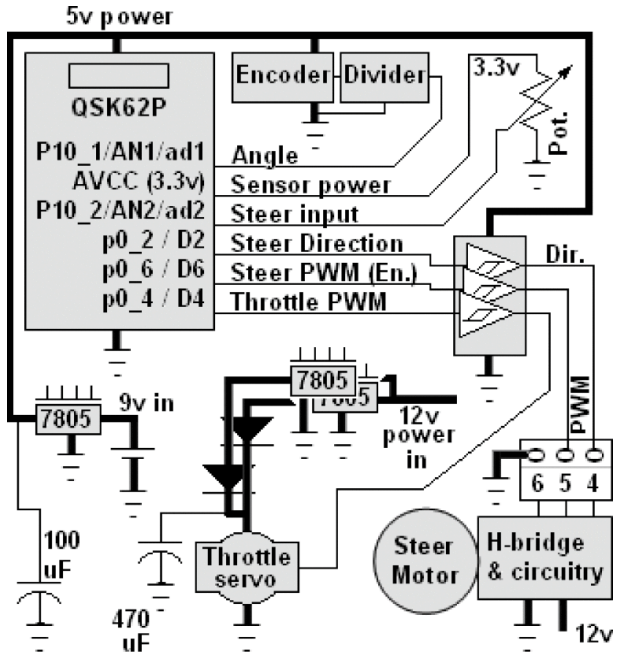

Figure 2.5: Steering circuit using micro-controller to control motor using PWM [8].

module using the OEM controller initially implemented by Honda. This design avoided overheating the motor by letting the OEMs steering motor controller handle it in the first way it was designed. According to this paper, when the steering angle changes, detected on the operator bar, a change of resistance appears on the output of the sensor. The sensor has three output wires representing two separate output resistors. One resistor is for the right side angle value and another for the left side. Simulating the sensor output was done using resistors and switches to implement this controller. Having the neutral position resistance value always connected in place of the real sensor. Using the switches to introduce parallel resistors to change the resistance values. Having this system implemented, it successfully turned the steering to the right and the left. This module was then implemented using analog switches, resistors, and a PIC micro-controller. This circuit could simulate the angle using a PWM signal to control the rate of opening and closing the switch. Therefore, the parallel resistances appear with a particular value that corresponds to a specific angle.

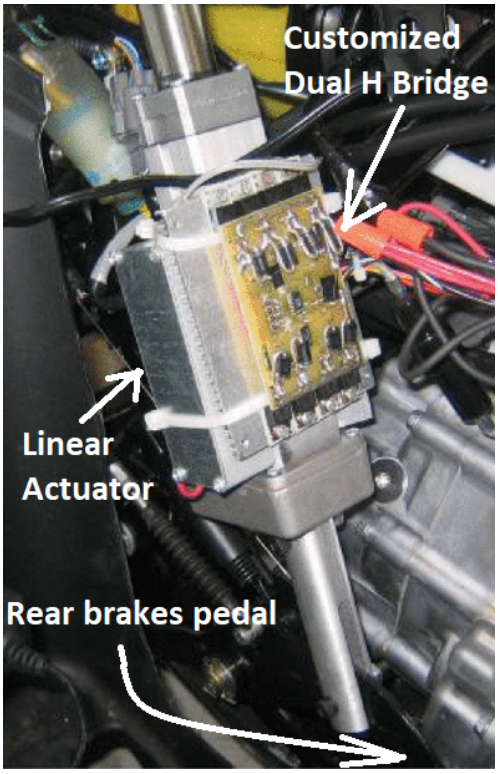

Figure 2.6: Old braking system applied in paper of 2010 [9].

The same paper [7] took it into more configuration and made the PWM signal to be configurable using CAN messages.

- **Braking System:** Two different braking systems exist in the Honda ATV. Hand lever brakes that control the front wheels brakes, and foot pedal brakes that controlled the rear brakes.

  The 2010 paper [9] provided a picture of the linear actuator used for the braking system with a customized H-Bridge. Figure 2.6 shows the braking system implemented as the linear actuator attached to the foot pedal and controlled by the customized H-Bridge.

  Another paper published 2014 [7] that improved the braking system in the ATV. In that paper, Braking System was implemented using the same linear actuator. The customized H-bridge was replaced by a Pololu motor controller to overcome overheating problems in the old one. In this paper, the brakes were controlled

Time(sec)	Stroke Length (cm)	Brake Condition
1	4.1	Full Release
1.5	5.1	1/4 Brake
2	6.1	1/2 Brake
2.2	6.4	3/4 Brake
2.5	7.2	Full Brake

Figure 2.7: Braking system different stages table in previous study [5].

using a Sakura micro-controller.

According to a recent paper in 2018 [5], a study about controlling the ATV using Robotic Operating System (ROS) was conducted. Before this study, the braking system was only either fully released or fully applied. In this study, using ROS, the braking system had three other stages, as shown in Figure 2.7.

The linear actuator and the aluminum support used to fix the linear actuator top part were used in the current study.

- **CAN Network:** The Control Area Network (CAN) Bus is a multi-master bus communication. CAN protocol is an asynchronous differential communication. It uses two signals for data: CAN-H and CAN-L for CAN High and CAN Low. The value transmitted is measured as the difference between the two signals. This communication protocol is an advantageous communication protocol in case of the ATV. As the ATV is a machine that has some motors and alternator to produce electricity, it is subject to much electromagnetic interference. This interference could affect a signal on a wire which may give false messages. However, in differential communication, the interference will introduce noise to both wires nearly the same way. While only caring about the difference between both wires, the difference values will not change as the noise bias is added to both of them.

In the automotive industry, three major communication protocols are commonly used in general among others: FlexRay, CAN, and LIN communication protocols. FlexRay is the fastest, but it is costly compared to CAN and LIN. LIN is mainly used to communicate within one module when it is a complex module with many components. LIN is a one master many slaves communication. CAN bus is widely used for multi-masters communication. It provides a reasonable speed for normal - none timely critical - communications.

The CAN Bus consists of four wires, two wires are used for CAN signals as mentioned above, and two wires are used for power. The first and last node should include a 120 Ohm resistors between the CAN-H and CAN-L wires. If these resistors do not exist, the CAN messages will fail to propagate. The CAN bus determines the end of the bus by the resistance between CAN-H and CAN-L, if the resistance exists in the middle of the bus, the rest of the bus will not be able to receive proper messages.

The CAN frame consists of start bit, control signals, destination address, data, and stop bit. The CAN ID, or the device address. There are two types of addresses, Standard ID which is an address of 11 bits, and Extended ID which is additional 18 bits to be added to the Standard ID to have an address of 29 bits total. Using only Standard or both should be specified in the control signals. The data size in each frame can vary from 1 byte up to 8 bytes. The data size should be specified in the control signals as well. Figure 2.8 shows a sample of a CAN frame using Standard ID and sending 2 bytes of data.

The CAN Network should be configured to the same baud rate and the same sampling point to avoid missing data. The baud rate and sampling point are configurable; however, it is more common in the automotive industry to use 250 kbps as baud rate, and sample at point 87.5%.

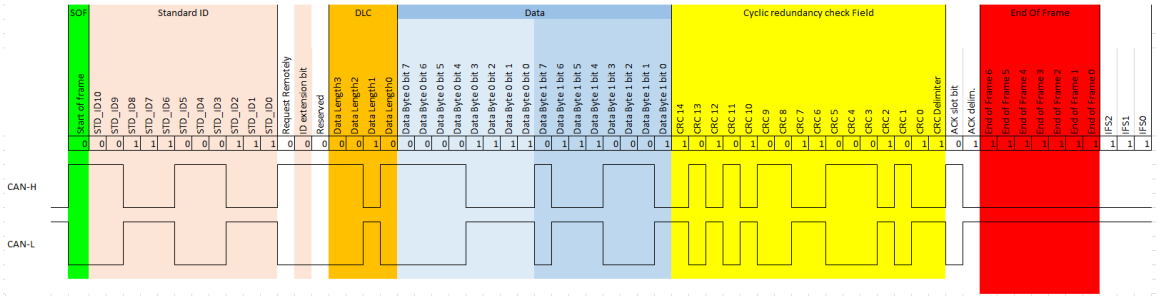

Figure 2.8: Sample of a Standard ID CAN Frame with 2 bytes of data.

In the previous research, a paper written 2014 [10] was about using a CAN bus to control the ATV. This paper used the SAE J1939 Standard for CAN communication. The CAN was used to communicate between the Brain and the steering module with extended ID of 29 bits of address. The link used 250 kbps as baud rate and 87.5% as the sample point. In this paper, a design of using Sakura micro-controller with Texas Instrument SN65HVD251 Industrial CAN Transceiver for communication between all nodes with the Brain as Renesas RX63N was implemented. However, the CAN APIs could not be used in this implementation due to compatibility issues. The paper ended by using the CAN communication only between the steering module and the Brain board.

- **Brain:** Renesas board RX63N was previously used as The Brain. It was connected to a remote-control receiver from which it gets the orders. It was used to control the Throttle Module using the three wires of the parallax servo motor. The Brain was also used to control the Braking System by providing a signal for the H-bridge for the linear actuator. Besides, the Renesas board was controlling the steering module by sending CAN messages to the steering module PCB [5, 4].

  At this point, there was no physical connection between sensors and actuators at this point except for the steering module that was getting readings from speed sensors. Also, this Brain was not used to take its own decisions to control

the ATV as it was not processing the sensors data. Instead, it was getting the commands from the remote control used by a human operator who was processing the sensors data and acting accordingly.

CHAPTER 3: MY CONTRIBUTIONS IN THE MODULES LEVEL

This Chapter illustrates the implementation of all the new modules like the CAN Network, Throttle, and braking system. It also focuses on the addition to the already existing modules and the standardization introduced to the system. The system integration is handled in the next chapter.

In this chapter, many author's publications are used [11].

3.1 System Overview

This study focused on the idea of making the ATV under study autonomous and self-driven without the human interaction to go from the initial position to the final destination.

In order to do so, an observation of a human driver is made. For a human operator to control the ATV, he needs to press the throttle to move. The operator also needs to be able to stop the ATV using brakes. As driving is not always straight forward, a need for changing direction using the steering wheel exists. All those actions are representing the human operator controlling the ATV. As the study aims at replacing the human operator, then actuators controlled by microcontrollers (controllers) are needed.

There is a need for a controller for the throttle, another for the braking system, and a third one for the steering.

The observation did not stop at this point, as the human operator should have a reason to press the throttle or the brakes, or even turn the steering. According to the observation, the human operator depends on his senses like seeing and hearing to avoid

obstacles and to know if he is going in the right direction or the wrong direction, with the knowledge of the final destination. As a conclusion, similar sensors are needed.

For the final destination, GPS is the easiest way to mark a destination for a robot, especially when it is outside a building traveling for a long distance without a human operator. Another way to detect the road and final destination, in case there is no GPS signal in the area, is to use bread crumbs sensors technique around the road toward the destination. This sensor is still under study by another study group at the Embedded Lab of UNCC.

GPS provides the current position, and this can be compared to the final destination to find the distance and understand if the ATV is moving toward or far away from the destination. The GPS provides longitude, which is the location in the East-West direction, and Latitude, which is the position in the North-South direction. As the position can easily be mapped to an X-Y plane, an angle between the north direction and the final destination can be calculated.

GPS can show the current location, distance to destination, and angle between the North direction and the final destination. In order to know the direction toward the final destination, a 3-D compass is needed. However, as the ATV is not going to run only on flat horizontal ground, the tilt will affect the accuracy of the compass. Then a tilt compensation is needed, and this introduces the need for IMU instead of a simple compass. A study is done in parallel to this study talks in more details about the IMU and GPS sensors choices, and sensors controllers' designs and implementation.

Normally detecting an obstacle in front of the ATV is done using the human operator's eyes. In the case of a robot, there are many alternatives like cameras, Light Detection and Ranging (LIDAR) units, ultrasonic sensors, and others. In order to decide which one to use, a comparison was made, and the LIDARs were chosen among the other sensors for the following reasons:

- Camera sensor is used to capture the visible light reflection. It can record

many images, but it needs a high processing power to detect and understand the contents of images. It can be done by machine learning techniques or by simpler image processing techniques. However, it still needs a lot of processing power. Another point about the camera sensor is that it is not a very good option at low light environments, and it can not work right in case of shiny surface light reflection.

- Ultrasonic sensors are sensors that emit ultrasonic waves and detect the reflection of the waves. They detect the distance by measuring the time between emitting and receiving the ultrasonic waves. They are very good at detecting obstacles at a near distance (within 10 meters range depending on the quality of the sensor and the environment), it can be used in low-speed low distance applications like in autonomous parking. However using ultrasonic sensors to detect obstacles at high speed will not be useful, when the obstacle is detected, it would be too late to take an action of stopping the vehicle or turning around it. Also, ultrasonic sensors are subject to much interference. Besides, ultrasonic waves can easily be reflected and deflected as they are propagating through air particles in circles shape which leads to different direction of propagation if hit an inclined obstacle and will lead to false information.

- LIDARs are LASER Radars. They work in a similar idea to ultrasonic sensors except the waves used are LASER not ultrasonic. As a radar, it goes to the right and to the left to scan an area. A LASER beam propagates in nearly a line without spreading. If it reflects something, it will go back straight to the source which determines the distance for the obstacle. It propagates with speed of light, and LASER can travel very long distances without signal attenuation which makes it the perfect sensor for the job. LIDARs can provide the position of obstacles by providing the angle and the distance.

However, the LIDAR is not planned to be included in this implementation. Only the choice of the sensor is made. For the number of implementations within this study timeline, the LIDAR controller and integration are not planned to be done during this thesis.

Another part of the observation was that the human eyes are not responsible for applying the brakes. Instead, it just delivers a message to the Brain through the nerves, and the Brain takes the decision and sends the commands to the muscles using other nerves. As a conclusion, a need for a Brain like a computer or a microcontroller to coordinate the messages and to handle the decisions according to sensors feedback and get the right action to the actuators is needed.

The Brain to communicate to the other controllers needs a network similar to the nervous system. The Network applied here is the CAN Bus. As mentioned earlier in Chapter 2, the CAN bus was chosen as it is a multi-master bus, and it is lower cost than the FlexRay. Its average speed is currently suitable to the speed required for the planned system with the expected data to be communicated.

As discussed in Chapter 2, some work was already done, and some modules were missing. This chapter focuses on the implementation of the missing modules beside some significant enhancements that are taking place in order to make the system more robust and reliable. Figure 3.1 illustrates the system overview for the current study.

As shown in Figure 3.1, the system consists of multiple nodes working with the idea of distributed processors to reduce the processing power needed by the central processing unit and reduce delays. The whole system is connected through a CAN Bus which is responsible for data and power distribution. The system consists of actuators represented here as Brakes, Throttle, and Steering controllers, and Sensors represented by GPS, Inertial Measurement Unit (IMU), and 2 LIDARs. A Brain is responsible for synchronizing the system by reading all sensors data and reacting

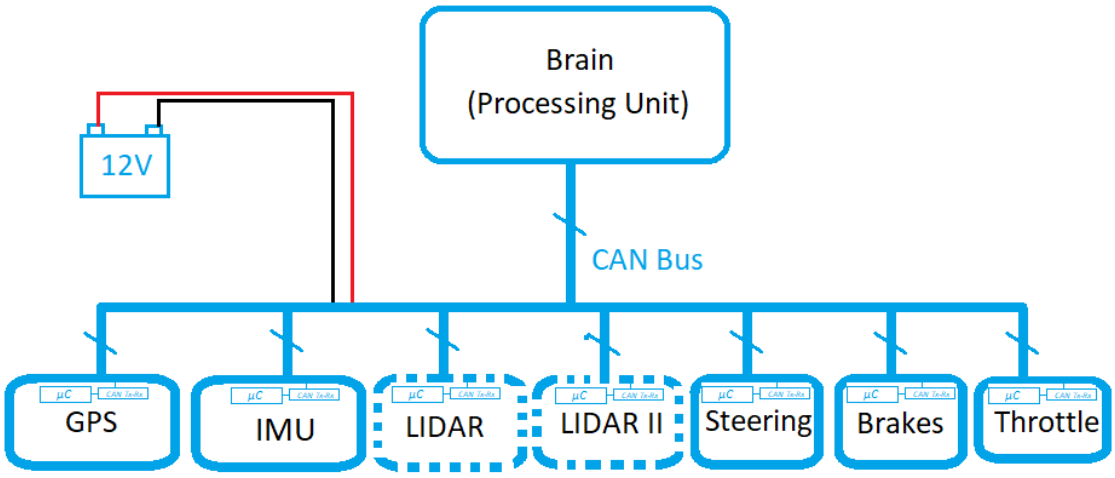

Figure 3.1: System Overview.

by sending orders to controllers through the CAN bus. Controllers are only getting the CAN messages as orders, and each controller is responsible for order execution. The system is implemented in phases, implementing standardization library first for MSP430, then implementing the CAN module and testing it, followed by the implementation of actuators in the standardized enhanced way illustrated in this thesis, after that adding the sensors one at a time. The Brain Code is implemented in agile loops by having one additional story per actuator and per sensor.

Part of standardization, the design as shown in Figure 3.1 includes in each node a microcontroller MSP430 connected to a CAN Shield with CAN transceiver MCP2515. The microcontroller was chosen first to be MSP430G2553 as it is cheap and reliable. The only problem at first was that it does not contain CAN peripheral. The option of obtaining a microcontroller with CAN peripheral was costly compared to have an MSP430 in addition to a CAN Shield. The MSP430G2553 has an SPI port that is used in the CAN Shield communication. To add other devices, like the IMU and GPS which needs I2C and UART communication. This introduced the need to have different software and PCBs for different modules interface, as the SPI can be on first

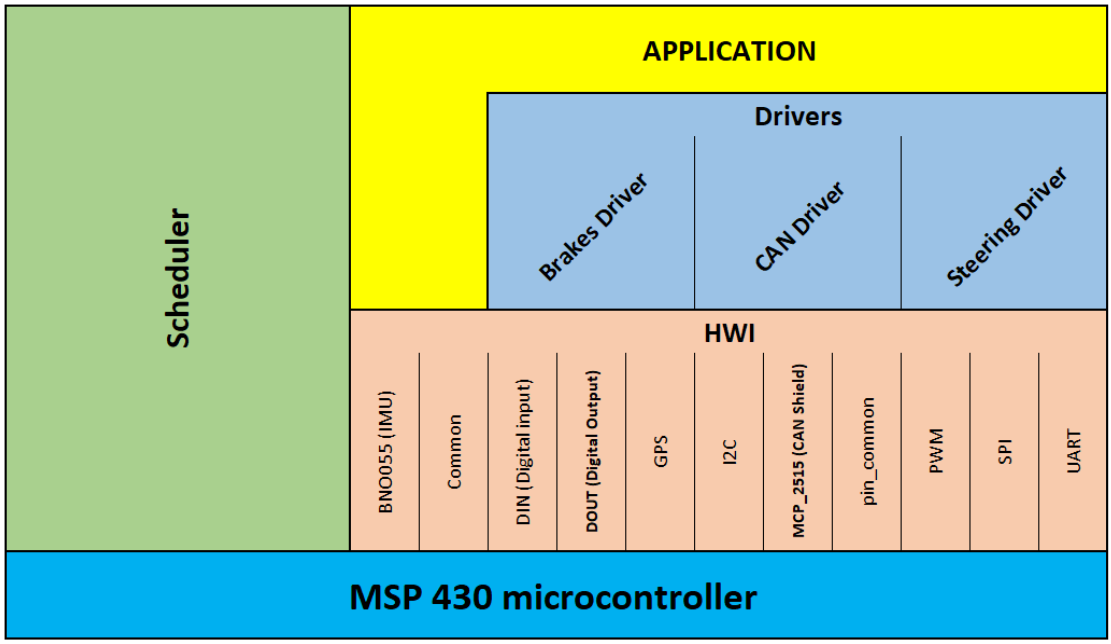

Figure 3.2: Standardized Software Architecture used.

or second port but the I2C and UART are limited to different ports. In order to have standardized modules and PCBs, an upgrade was made to MSP430F5529LP which includes eight ports instead of two ports for MSP430G2553. The F series also has higher memory size and higher processor clock with higher speed. The difference in price is not noticeable. However, it will reduce the overall cost by having standard connections and standard integration PCB as mentioned later in Chapter 4. The effort of upgrading was one day of work to change the libraries to the F-series 8 ports microcontroller.

3.2 Software Architecture

The software was implemented in a standardized architecture [2, 12]. This architecture divided the software into layers as shown in Figure 3.2.

The first layer is the lowest layer. It is called Hardware-Software Interface (HWI) layer. This layer is a library for the microcontroller used. The library is implemented in a parallel study [11]. The HWI layer is considered an abstraction layer in order to

hide the hardware (H/W) used from the rest of the software making the application independent of the H/W used. This idea is handy in case of need to change the H/W after implementing many modules. It prevents losing time and effort to rebuild the module's software in case of H/W change. Only the HWI will need adjustments, not the whole software.

The next layer is the drivers layer. This layer uses the HWI layer functions to build complex functions that are used many times in the application. The HWI and the drivers layers are the same for all modules and all applications implemented.

The highest layer is the application layer. This layer is different from a module to another as per the module's application. This layer uses the other layers' libraries to implement the module functionality.

A vertical layer that is engaged with all other layers is the Scheduler layer [13]. The Scheduler is a simple type of operating system that has multiple tasks with different timing. The Scheduler design includes a super loop (forever looping loop) in which it checks for flags. The flags are set using timers interrupts. Each task has a corresponding flag, and if this flag is set, the functions of this task are called, and the flag is reset.

The Scheduler layer is optional as per the module application. In case the application needs timing, the Scheduler is enabled in the module software architecture.

3.3 CAN Module

The CAN module implementation is divided into the physical bus wires implementation, the CAN shield connection to the MSP430, and the software configuration of the CAN Module. Below is an illustration of the different parts of the CAN module implementation:

Figure 3.3: CAN Bus Physical implementation: CAN Connectors.

3.3.1 Physical Implementation and wiring

For system implementation, it is designed to have a main processing unit, connected to separate small processing units that control and handle different nodes. All these nodes and the main processing unit are connected via a CAN bus as illustrated in Figure 3.1.

For this project which has been running for ten years, to prevent effort loss and repeating work, this implementation has a been done using CAN plugs that can be plugged-in in only one way, to prevent wrong connections. Plugs are shown in Figure 3.3. Each plug has four twisted wires [14]: CAN High (Yellow), CAN Low (Blue), High Voltage of 12 Volt (Red), and ground (Black) as shown in Figure 3.4.

The high voltage wire is chosen to be 12V not 5V. The reason behind this choice is that some nodes will need the 12V to operate like in case of the braking system. Therefore, each node that requires a 5V power supply will have an internal power supply that converts the 12V to 5V.

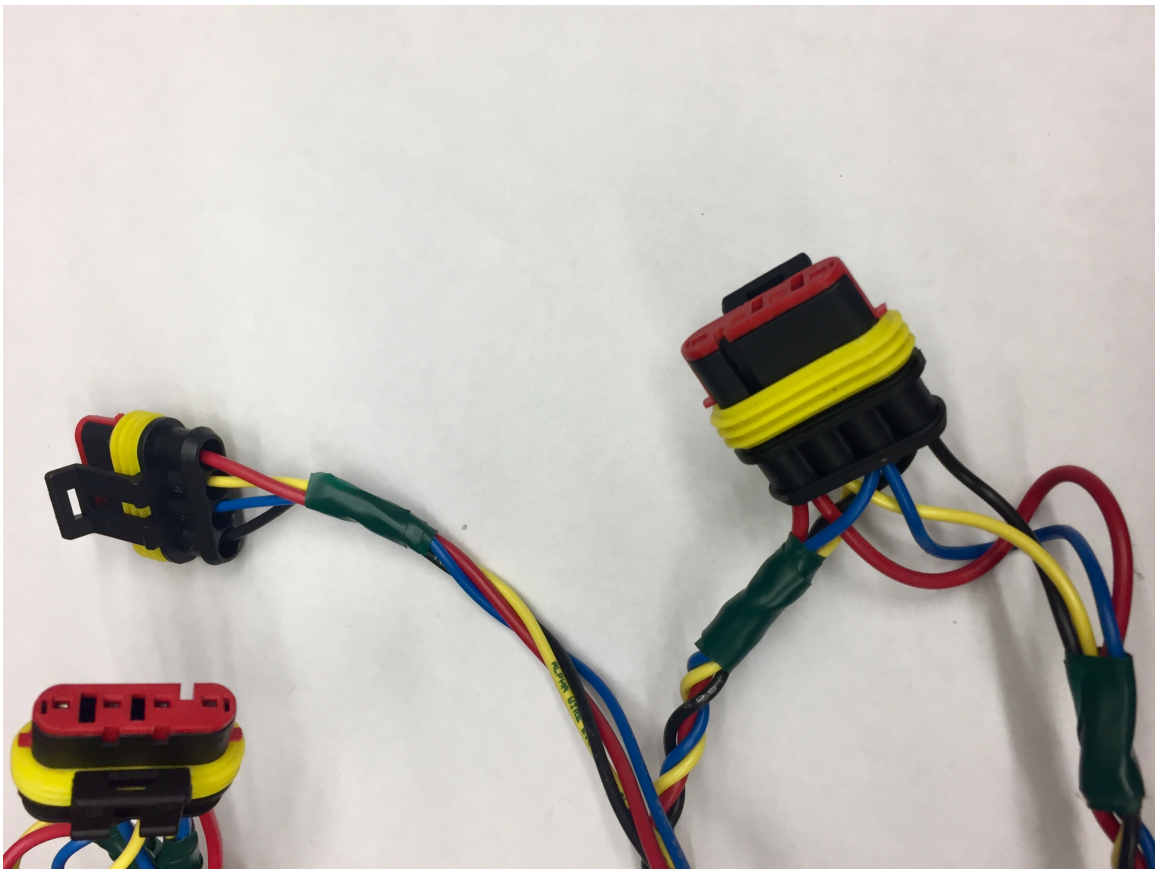

Figure 3.4: CAN Bus Physical implementation: CAN Bus after implementation.

As part of standardization, all nodes newly introduced or changed are made using MSP430F5529LP boards. In order to have the CAN enabled, and external board - CAN Shield - is used, on which a CAN transceiver chip MCP2515 exists. Figure 3.5 is a picture of the shield used. The shield was chosen for its low price and ease of connection. The MSP430 is connected to the CAN shield using SPI communications.

3.3.2 Software Configurations

Software Configuration for the CAN Shield was done using SPI write commands into the MCP2515 [10, 15, 16, 17, 18]. The software is made configurable to be able to work as Standard ID or Extended ID frames. Mainly it was made and tested for Standard ID frames. However, it is changed to be configurable as the re-used module

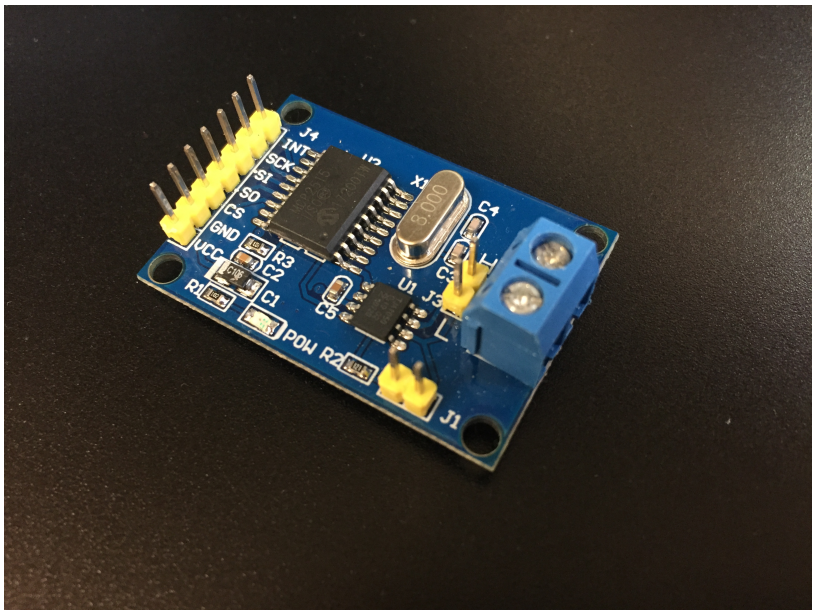

Figure 3.5: The CAN Shield with MCP2515 CAN Transceiver.

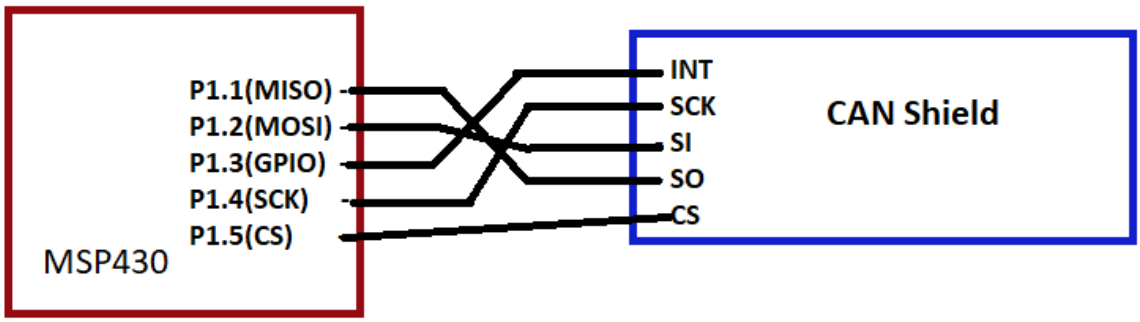

Figure 3.6: MSP430 to CAN Shield Connections.

of the steering wheel had an Extended ID [7].

In order to configure the CAN transceiver, the following steps were followed, according to the MCP2515 datasheet, and to some online examples that were modified to match this project [19, 20, 21]:

1. Initialize SPI module.

2. Reset the MCP2515.

3. Set receiving masks (the mask is used to configure which bits in the device ID

should be filtered and which one should be ignored).

4. Set CAN to configuration mode with one shot mode which means do not try to send a message more than one time.

5. Set Transmitter buffer 0 (TX0) as the highest priority.

6. Set the CAN timing (CNF registers): Baud Rate, number of samples, PS2 length. All modules on the same network should have the same settings:

   (a) Set speed to 250 kbps. At first, it was set to 125 kbps, but in order to be compatible with old modules, it is set now to 250 kbps [7].

   (b) Set Sampling to only 1 sample at the sampling time, CNF2 value is very important since if it is not set with the right value, no communication will be established. This value was one of the problems faced.

   (c) Set PS2 length to the max.

7. Configure the Receive buffer 0 (RX0) control register to use a filter to read messages for RX0, which means only read messages addressed to the device ID and configure it such that if RX0 is full, do not use RX1.

8. Configure the Control register for RX1 to use a filter to read messages.

9. Set the device address by configuring the registers: RXF0SIDH, RXF0SIDL, RXF0EID8, and RXF0EID0. The first register RXF0SIDH is for the most significant bits of the Standard ID, the register RXF0SIDL is composed of 3 parts, the least significant part of the standard ID, control pin to allow extended ID, and the most significant 2 bits of the extended ID, the last 2 registers are for the rest of the extended ID.

10. Set configure the MCP2515 to raise an interrupt to MSP430 when CAN RX0 is full.

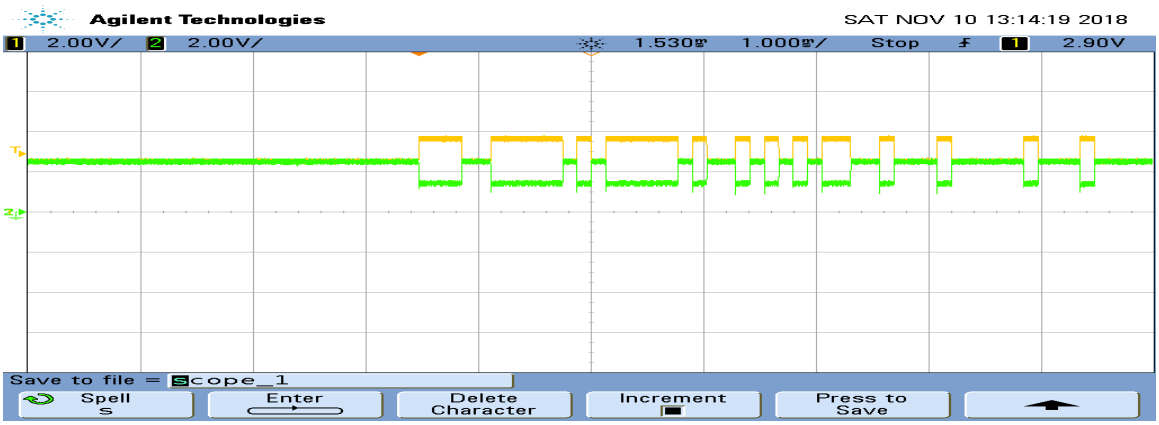

Figure 3.7: CAN Frame on Oscilloscope with Yellow = CAN-H, Green = CAN-L.

11. Configure MCP2515 back to normal mode, and then test the MCP2515 to make sure it is back to normal mode.

After configuring the CAN module, a test was made to send 0xAA from MSP430 to another, and the second board turned a LED on when it received 0xAA which means it works fine. Below is an image from the oscilloscope showing the full frame used to test the board. The image was using a standard address of 0x0181. The CAN module has many factors that are all contributing together in right functionality, and any of them may lead to system failure. It is always preferred to configure and test only one factor at a time.

First of all, the SPI module should be working and tested to be able to communicate with the CAN module. This was the first problem faced, that the MSP430 SPI is not able to send multiple messages without receiving a message. Even if no message is expected a dummy read is mandatory to clear the receive flag and allow the SPI to send a new message.

The CAN bus should be connected to a 120-ohm resistor at both ends, in the CAN Shield board used, attaching the jumper J1 on the MCP2515 board will connect the 120-ohm resistor at this end of the bus between CAN-H and CAN-L. Not having the 120-ohm connected was another problem faced at first. However, it is important to

know that when having more than two nodes on the CAN Bus, only the first and the last should be connected to the resistance. If a resistor is connected between CAN-H and CAN-L at any other node, this node will be considered as the end of the bus, as it will be discussed later in Chapter 4.

Also, the MCP2515 board should be connected to a 5V power source to be able to drive the message signal. It will not work if it takes the 3.3V VCC from the MSP430, which is the VCC voltage in case of MSP430G2553. The newer board used with the MSP430F5529LP is featured with 5V pin and 3.3V pin. The 5V pin can be used to power the CAN Shield. However, as most of the nodes include a power supplier that transform the 12V into 5V, the power is no longer an issue.

Reading the error registers in the debugging state by sending the read command and the register's address from the MSP430 to the MCP2515 using SPI module is essential and it helped in this configuration to understand some errors and malfunctions like the message was never received because of wrong CNF2 register configuration (CNF registers are responsible for rates and timing). Debugging errors this module was done by sending read SPI messages to the MCP2515 with the address of all configured registers and the address of the status registers.

Another handy tool to use for CAN testing is an oscilloscope with CAN serial communication enabled. It is beneficial to have it as it shows the address and message length as well as the message data in hexadecimal format. This format allows the tester to understand more if the address sent is right or wrong, and if the messages are right or wrong. Also by setting the oscilloscope on the required baud rate, if the message data is not the same as expected, it will imply an error in the baud rate as one possible issue.

One of the bugs that were found by the oscilloscope and fixed is that in case of extended frames, the standard ID is used as the most significant part of the address and the extended ID is the least significant part of the message address. This mistake

Figure 3.8: Throttle wire and spring at the air valve side.

is not a problem if the whole system is homemade and not communicating to any external module and if the addresses are written separately into the CAN Shield register. However, as the system is designed to comply with the CAN standard protocols.

3.4 Throttle Control

The throttle in the ATV is manufactured as a lever to be pressed by the right-hand thumb. It is connected to the air valve using a wire that opens the air valve which leads to open the gas valve and allows the gas pump to provide more fuel according to the angle the lever is moved. The more pressure on the button, the more fuel is injected, the higher speed the ATV should go. If the lever is not pressed, it is equipped with a metal spring to bounce back to the initial position. Figure 3.8 illustrate the original part equipped by the ATV. As the implementation is purely mechanical, the former team installed an electrical servo motor at the air pump terminal of the wire as shown in Figure 3.9 and Figure 3.10. In the current implementation, this motor is controlled through 3 wires, Vcc, Gnd, and a PWM signal provided by a separate MSP430. The

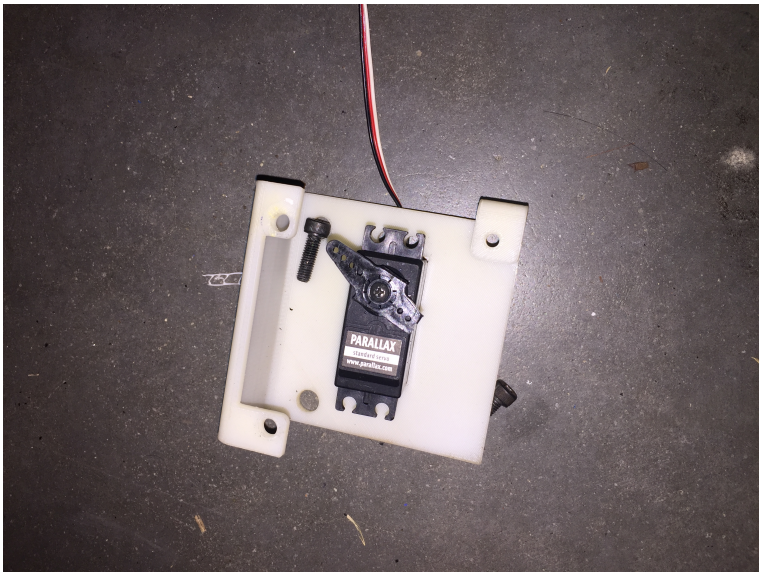

Figure 3.9: Throttle wire and spring at the air valve side.

MSP430 is connected via SPI to a CAN shield with the CAN transceiver MCP2515. The MSP430 receives CAN messages with values from 0 to 90 that represents the duty cycle for the PWM signal to control the Throttle. The higher the duty cycle, the more fuel injected to the engine. If the value exceeds 90 in the sent message, the software will put it as 90 to avoid overloading the motor. However, as it will be discussed in Chapter 4, it is preferred to run it at 62 to 65%. Below 60%, the ATV will not have enough power to move due to the ATV load and the ground friction, and more than 65%, the ATV will travel very fast. The module was tested at first by testing the servo motor alone using a function generator, and then the MSP430 was tested by using 2 MSP430s connected with CAN shields. The first MSP430 was used as the Brain, and the second as the gas pump motor controller. The first one used to send values 0x32, 0x64, and 0x00 for 50%, 90%, and 0%. Figures 3.10, 3.12, and 3.13 are images from the oscilloscope illustrating the output PWM signal. Testing the braking system on the ATV is discussed in Chapter 4.

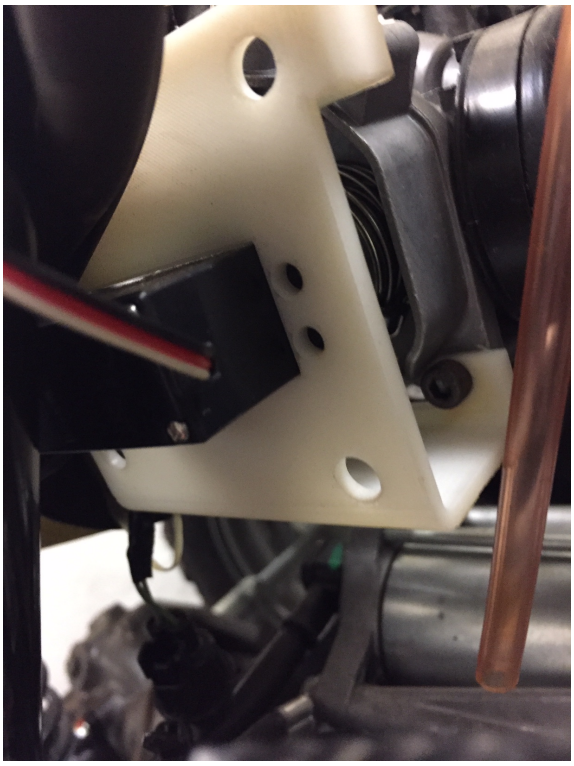

Figure 3.10: Servo Motor attached to fuel pump.

3.5 The Braking System

The ATV used is a rear wheel drive vehicle. The braking system is chosen to be used on the rear braking system controlled by the foot pedal as the rear wheels are the wheels controlled by the motor. Stopping the backward wheels will help to stop the whole ATV. If the automated braking system were using the front brakes, the front wheels would stop while the back wheels are running with the motor power, this may help the ATV to roll over to the front side. That is the main reason to choose the control to be using the rear wheels braking system. The foot brake is controlled by a 12V linear actuator, capable of 100 lbf with a 4-inch travel range that is connected to a rigid mounting bracket affixed to the chassis tube frame and a custom adapter bracket that mounts to the existing foot brake lever [11].

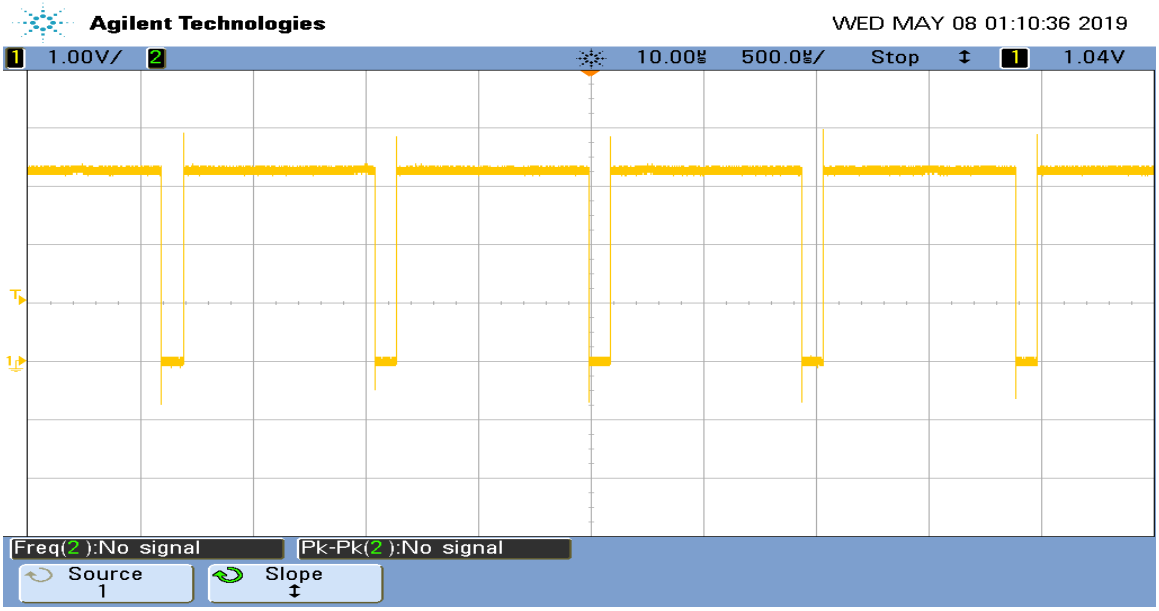

Figure 3.11: CAN messages of 90 or more.

3.5.1 Actuator

The actuator is equipped with limit switches to restrict extension and retraction to an appropriate range for the brake pedal. The placement of these switches was determined based on the position of the brake pedal with no load applied and under full braking by a human operator.

The actuator is controlled with a commercial off the shelf dual H bridge motor controller that is rated for a maximum voltage of 30V and with average current outputs of 18A and stall currents of 40A.

3.5.2 Actuator extension

The actuator extension is pinned in place with quick release pins at the actuator and the bottom mount bracket. Removing these pins allows a test operator to easily disengage the system and regain manual control over the brake system, as needed [11]. Figure 3.14 demonstrates the quick release pins holding the linear actuator in the two pieces bracket sandwich implemented around the brake pedal. Figure 3.15

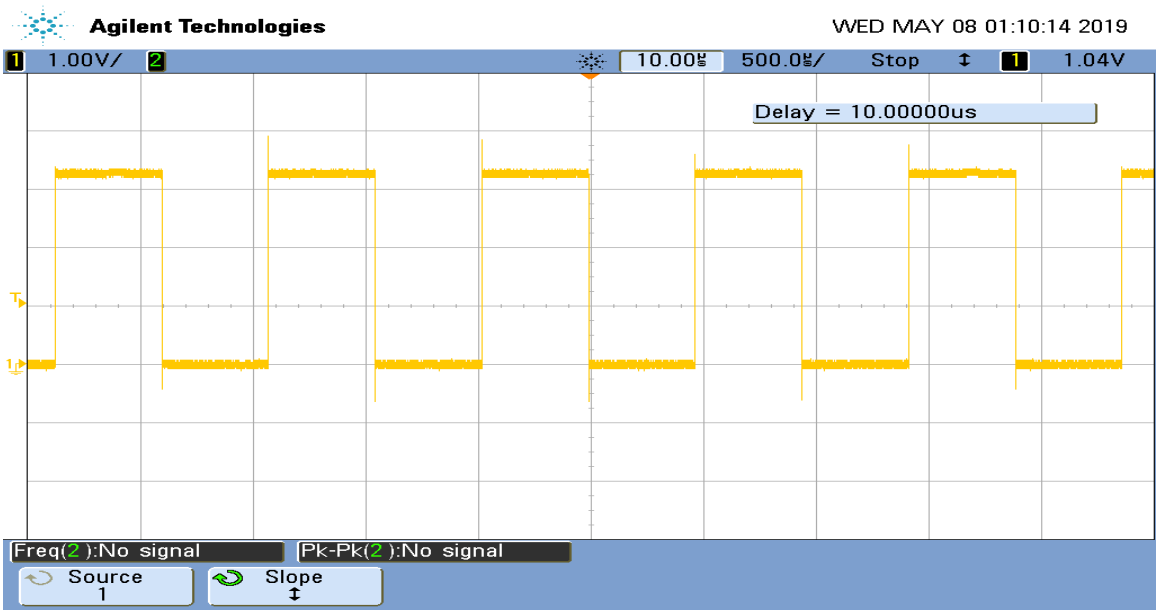

Figure 3.12: CAN message of 50.

shows the quick release pins released and the braking system unmounted.

3.5.3 Bottom mount bracket

The bottom mount bracket was developed to ensure that the force applied by the actuator was properly constrained, only influencing the rotation of the lever about its mounting axis and not allowed to excerpt force in any out-of-plane direction. The stock brake lever is a "free form" sheet metal part without constant geometry that would be suitable for easily mounting brackets too. This issue was overcome by using a two-piece bracket that sandwiched the lever using three bolts positioned in direct contact with the top and bottom of the lever to evenly distribute the force applied to the bracket through the quick release pin linkage [11] as shown in Figure 3.14.

3.5.4 Software implementation

The Braking system is controlled through an MSP430 connected to a CAN Shield which connects it to the ATV network. The MSP430 controls the dual H bridge using i2c communication. As the actuator only has one start point and one stop point, the

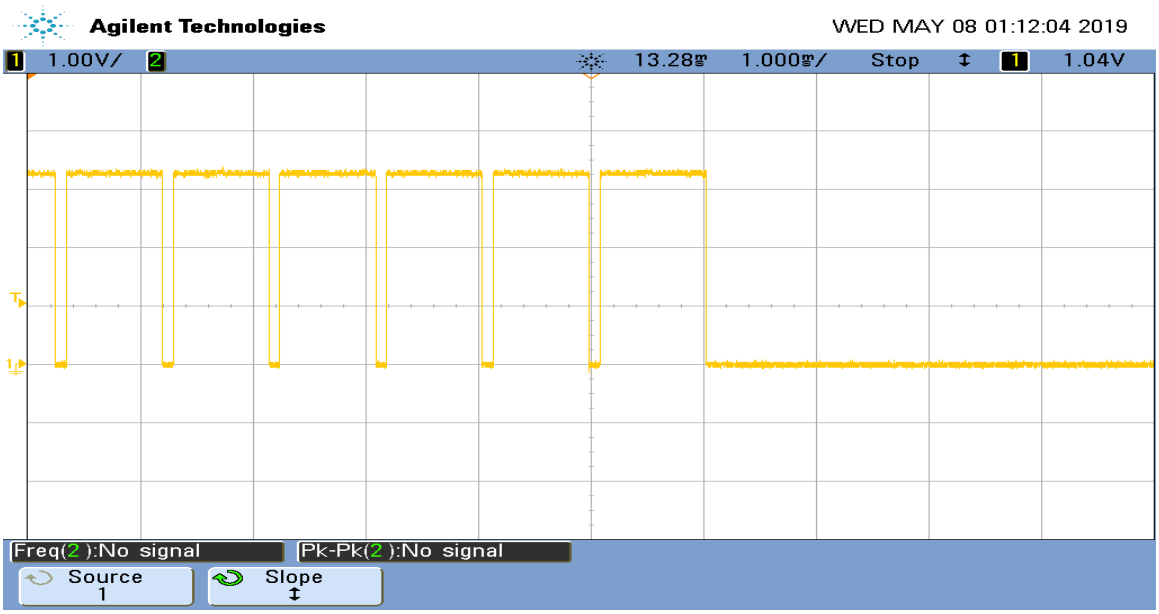

Figure 3.13: CAN message of 0 after another one of 90.

software sends a signal to apply brakes and another to release it. The motor driver used sends PWM signals to the linear actuator. The PWM is confusing the stop switches as it gives the motor a sudden start with each new cycle of the PWM. For the sake of preventing any damage to the linear actuator and the stop switches, the signal for applying the brakes is timed for 1.1 seconds and for releasing the brakes for 1.7 seconds. Those timings are taken from measuring the time needed for the brakes concerning expansion and to contraction; it was noticed that the expansion was faster for some mechanical reason.

By consequence, applying the brakes for the same time of releasing the brakes causes the start point to move a little every time the braking system is used. That is why the time the release signal is applied is longer than the time the apply signal is applied. This timing difference is to ensure that the brakes will return to the same initial point and the brakes will not always be applied after multiple uses of the braking system. After sending the release or the apply signal, a signal to stop the motor movement is sent. The signals are sent from the Brain to the MSP430 of the

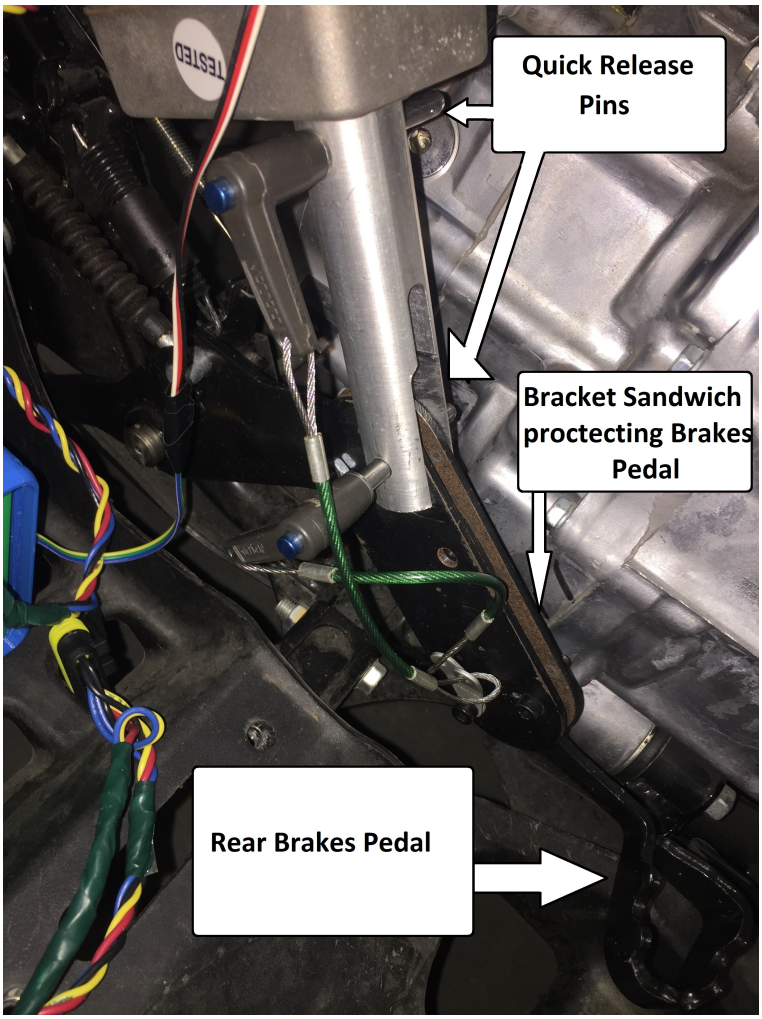

Figure 3.14: Brake Pins fixing the linear actuator to the two-piece bracket sandwiching the brake pedal.

braking system.

The braking system is set to have a flag for the last status. If the status of the braking system is the same as the command received by the Brain, the braking system will not perform the action. If the braking system status is different, it will apply the new command and change the current status. For example, if the brakes are pressed, and the Brain sends a CAN message to the braking system to press the brakes, the braking system will not perform any action. If the brakes are pressed and the Brain sends a CAN message to release the brakes, the braking system will send I2C message to the dual H bridge to release the brakes for 1.7 seconds, then it will send another

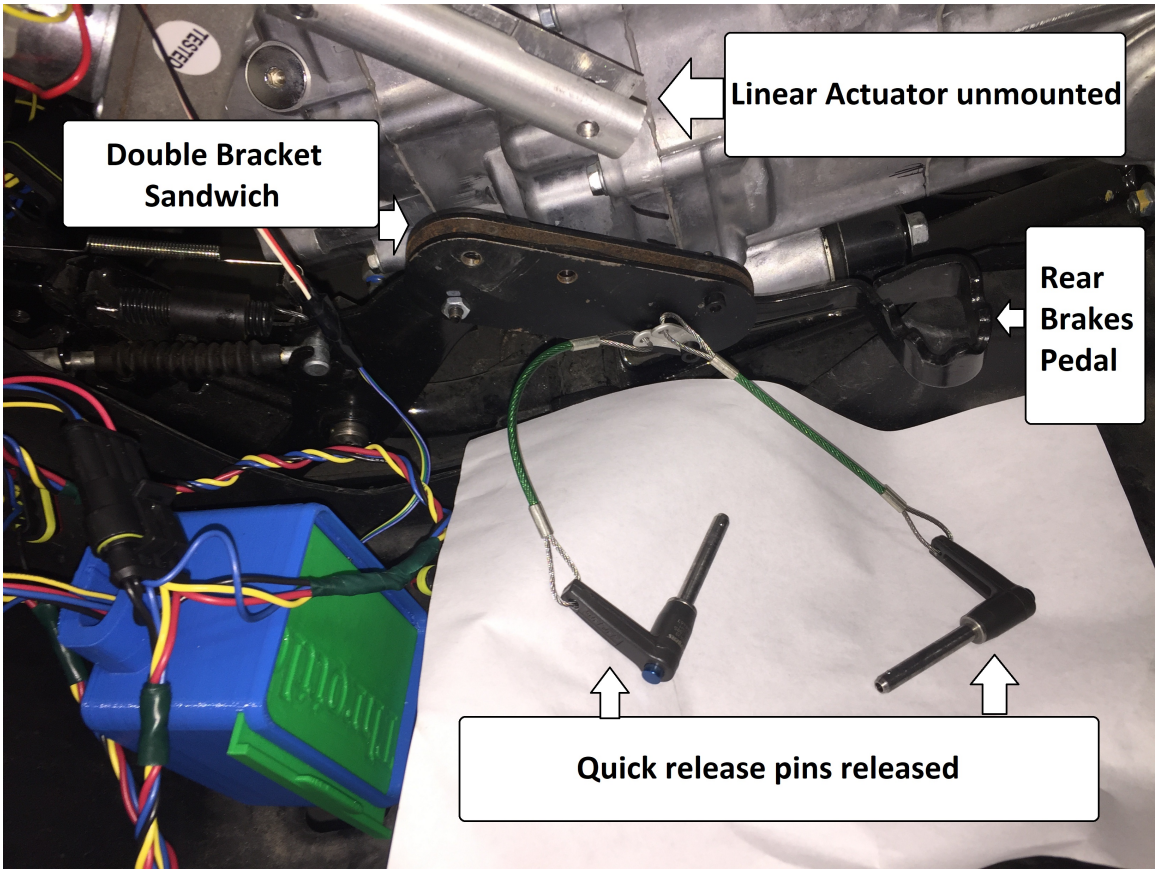

Figure 3.15: Quick release pins released and the braking system unmounted.

I2C message to stop the brakes and will change the status flag to be released.

3.6 Steering Module

This module was re-used from the old research with minor enhancements for the connector to make the whole system uses the same CAN bus connections. The already existing module is made using PIC micro-controller, that receives messages using CAN transceiver MCP2551, that communicate the angle value to the PIC using SPI signals. The PIC controls an analog switch that connects and disconnects some parallel resistors to simulate the Torque sensor values using a PWM signal [8, 7]. The main idea is to simulate the torque sensor output to fake the system pretending an attempt to turn under high resistance which will lead the steering assist motor to engage and turn the steering wheel.

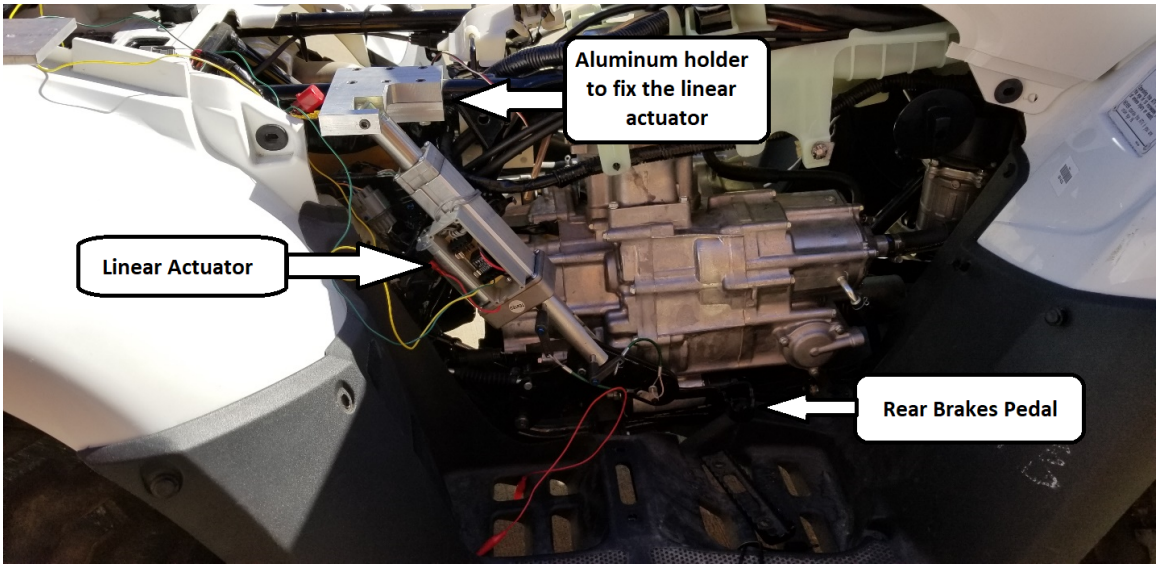

Figure 3.16: Braking System installed on the ATV

However, integrating this module was a big challenge, for the module was implemented in 2012. As part of integrating the steering module in the system, these points were taken into consideration:

- The steering module is controlled using CAN messages with 250 kbps baud rate and sample point is set at 87.5%.
- The module ID uses Extended ID, if the ID is written in the format Standard-Extended ID with Standard ID as most significant bits, the ID should be 0x18F00F00 [7].
- Steering Module is set to read CAN messages containing 8 bytes of data, with the least significant byte sent first. The center position value is set to be 721, with the maximum left direction allowed value of 566, and the maximum right direction allowed value of 876 [7]. For example, to send 721, it should be by sending this array: {0xd1, 0x02, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff}. The values from 3^{rd} to last does not matter.
- Using a CAN decoder oscilloscope is a good method for debugging this module.

Figure 3.19 shows the frames used to control the wheel position.

- To test the steering wheels at first, the front wheels are lifted off the ground, so that the steering wheels could be controlled without having the ground friction with the ATV weight. Also, while having the ATV on the ground, it is recommended to have it moving before using the steering module, to avoid overloading the steering assist motor.

- The steering module has a connection to the (PS) light in the dashboard. In this study, it was preferred not to connect it, as if it is not connected, it will show system faults and indicate if there is a diagnostics error message (DTC) in the system or not. As per the Honda service manual [22], when the steering module has an error, and a DTC message saved in the Honda diagnostics system, the (PS) LED will turn ON, and the steering assist module will be stopped till the DTC message is cleared.

- In case the (PS) LED is turned ON, a DTC reset is needed. There are multiple ways to read the error message and reset the error message. The first option is to buy the expensive diagnostics machine from Honda to read the message on a screen and see the recommended fix and apply it. The second option is to use a simple tool from Honda to switch the ATV to diagnostic mode. The tool is used to connect two wires in the DTC cable. Once the two wires are connected, the car key (K15) should be turned to ON position. The (PS) LED will blink in a sequence to demonstrate a number. This number should be checked in the Honda service manual to find what is the equivalent error message.

  The (PS) LED can be turned ON in case:

  1. neither the steering module nor the original factory torque sensor is connected to the electrical power steering (EPS) system.

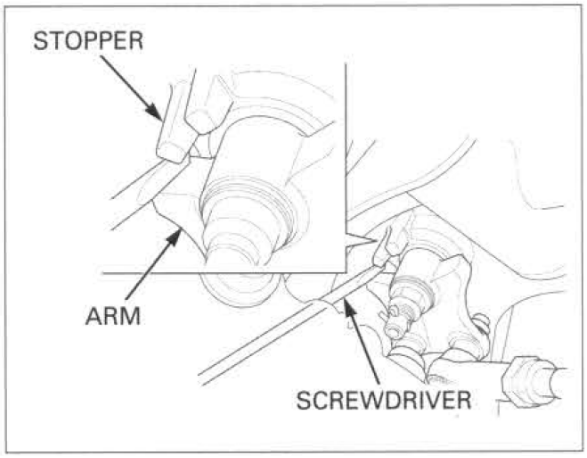

Figure 3.17: Screwdriver position for EPS DTC reset [22].

2. bad wiring, loose connection
3. Overheated EPS motor

Instead of buying this tool, simply a wire was used to do the short circuit and enter the diagnostics mode. In order to have the EPS working again, after the problem is fixed, the following sequence to reset the DTC message should be followed [22]:

1. Put K15 on OFF position
2. Connect the torque sensor back to the EPS
3. Connect the diagnostics tool to switch into the diagnostics mode
4. Place a 6 mm screwdriver between the steering stopper and the steering arm as shown in Figure 3.17
5. Turn the steering wheel to the maximum left position
6. Make sure the engine stop switch is not stopping the engine
7. Turn K15 to ON position
8. Once the (PS) LED is turned off, turn the steering wheel back into a straight position

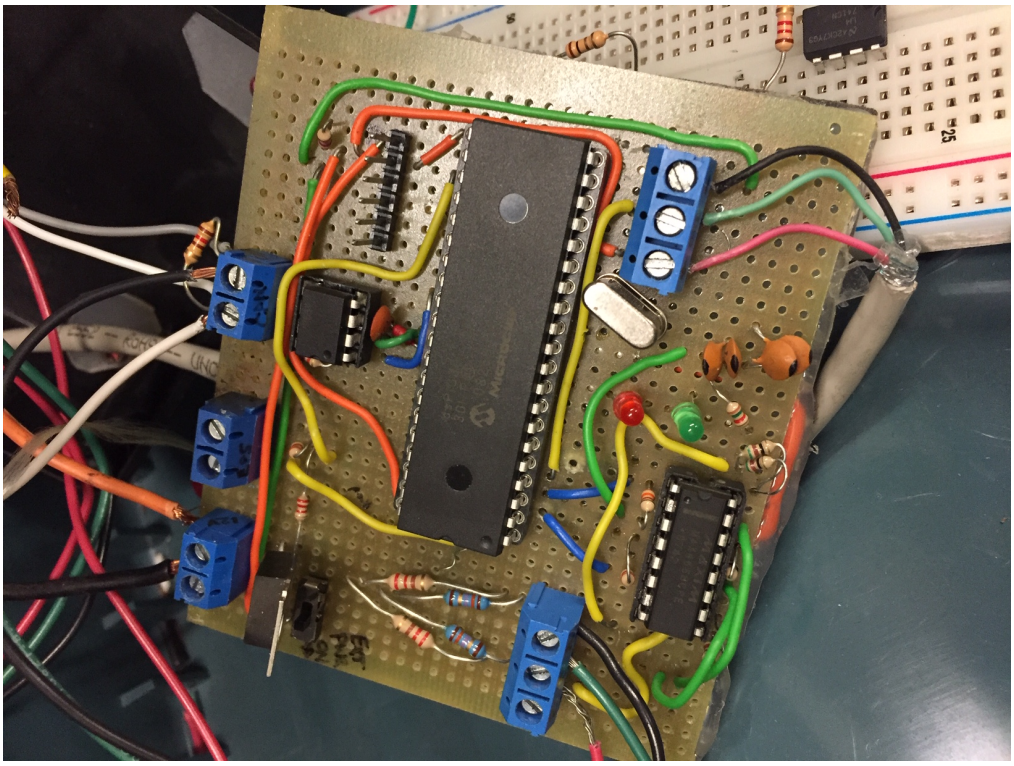

Figure 3.18: Steering Module

9. Once the (PS) LED is back ON, turn the steering wheel to the maximum left position
10. Once the (PS) LED is turned off again, turn the steering wheel back into a straight position
11. The (PS) LED will blink twice, this means the reset is done successfully.
12. Turn K15 back to off, and remove the tool/wire used to set the ATV into the diagnostics mode.

After connecting the system to the vehicle, the controller over CAN bus is able to successfully move to the right and stop at a certain angle, move to the left and stop at a certain angle, and back to the center position and stop.

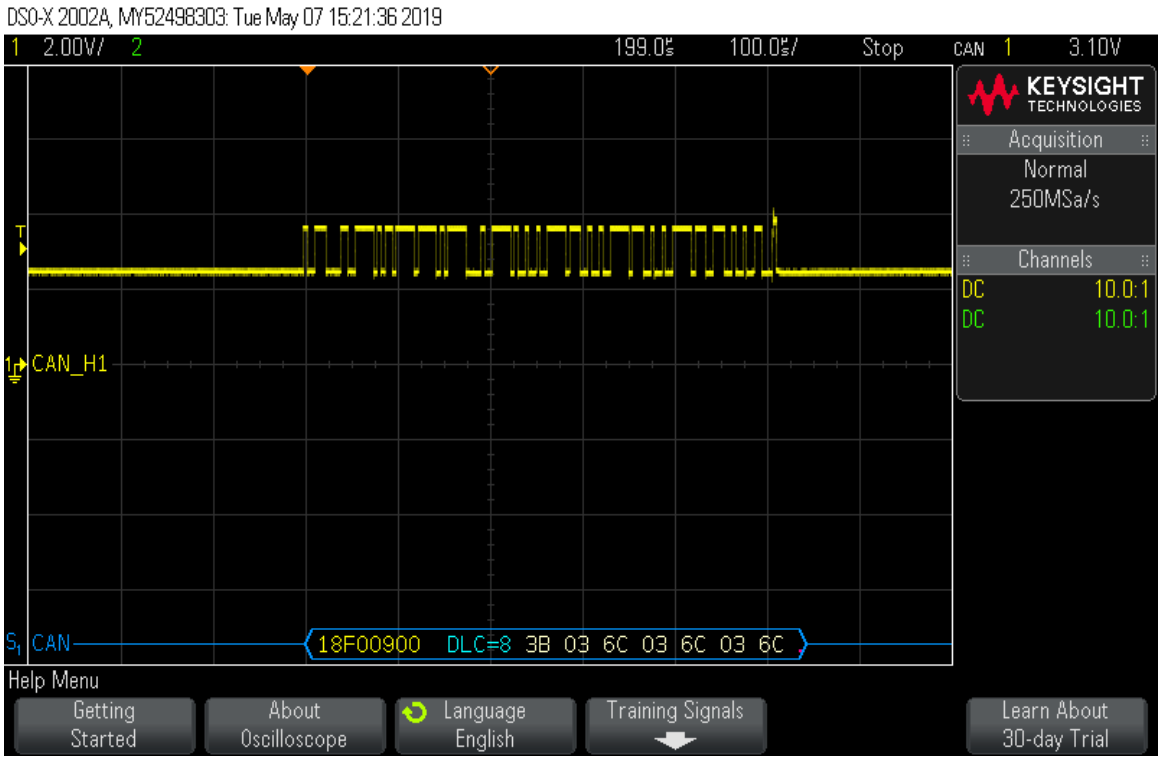

Figure 3.19: CAN message to turn steering right

3.7 The Main Processing Unit (Brain)

This module is another MSP430 connected to a CAN shield to connect to the ATV CAN network. At first a comparison was made to decide if this module should be an MSP430 or a laptop. A laptop is better for processing power and speed. The MSP430 is cheaper and more practical in an ATV for the size and power consumption. In exchange for the processing power of the laptop, the processing is distributed among the whole network. This reduced the role of the Brain to control the schedule, read the sensors, and send the commands to the actuators. The Brain does not analyze the sensors data, as each sensor is connected to another MSP430 that is responsible for processing and filtering the sensor row data and sending the processed data to the Brain. Also for the actuators, each actuator is connected to its own controller. The Brain does not control the actuators, it just sends high level commands to the

controllers responsible for the actuators. The controllers are responsible for executing the Brain commands using the actuators. A distributed processing reduces the load on the central processing unit.

The Brain implementation was done in agile technique. Every time a module is done, the module is tested, then it is adjusted to be controlled by CAN messages in case of controllers, and to send the processed data over CAN messages in case of sensors.

Chapter 4 focuses more on the Brain building in many phases as Chapter 4 talks about integration and testing. The Brain implementation depends more on the system integration. However, the first steps of building the Brain where the Brain is only controlling a controller was done as follows:

3.7.1 Brain communication

First step was to test the Brain ability to illuminate a LED on another MSP430 using the CAN bus. This step is as simple as it looks. It is very important to assure that the CAN communication works fine. A code was written to send a command to illuminate the led, and another code was written on the other MSP430 to check the received message, and if it matches the expected one, the LED is turned ON. This step was done successfully.

3.7.2 Brain communication to the Throttle

This phase was done to check the ability of the Brain to send commands to the throttle controller. In this case, the Brain was sending values from 0 to 100 on CAN bus to the throttle module address. On the other side, the throttle is programmed to check the CAN messages, it filters the messages coming to the throttle. If the message is of a size of 1 byte of data, it checks the value, and if it is more than 90, it clips it to 90 and use the value to set the duty cycle of a PWM signal. The PWM signals were displayed on the Oscilloscope to make sure the values are represented in

the right way. Figures 3.10, 3.12, and 3.13 show the results of this phase.

3.7.3 Brain controlling the Throttle

Implementing the Brain part responsible for the steering module was a bit challenging because the steering module was treated as a black box. The Brain code was simply sending data to the steering address. The steering did not respond at first. The possible bugs were many:

- The data size in the message frame according to the documentation is 8 bytes to be sent as 8 unsigned characters variables. The value to control the steering according to the documentation [7] should be between 566 and 876. All the values need 2 bytes to be represented. The first test was to change the sequence, instead of sending the low significant part first, it was sent second. This did not solve the problem, so it was reversed to the normal sequence as per the documentation [7].

- The connections were revised intensively. Debugging the PCB shown in figure 3.18 was necessary, as this board was implemented in 2014. Debugging the board did not clarify the problem.

- Debugging the SPI messages from the Brain's MSP430 to the CAN shield and make sure CAN messages are available on the bus was also done. This phase did not fix the problem neither.

- Using Oscilloscope with CAN communication analysis feature was used to check the CAN frames sent by the Brain was the next step to do. This step helped a lot as it covered 3 bugs:

    1. The address used was not the right one. The CAN communication was working fine with all homemade modules as they were all done in the same method with the CAN address sent with Extended part of the ID as the

most significant part of the address. This was a mistake as the standard part should be the most significant part of the message and it was fixed by some calculations.

2. The CAN message only included 1 data byte, not 8 bytes. This was another bug in the code that was fixed simply.

3. The CAN message values were wrong, and this was due to a wrong data type used for the steering angle value, which clipped the values, and used only 8 bits instead of the 16. This bug was also fixed easily.

- After fixing the bugs in the previous point, the Brain module was connected again to the steering module which is connected to the vehicle, and started to test it. The module did not work. Therefore, debugging the software again was needed, so the microcontroller was connected to the laptop to start debugging, and before starting the debugging, the module started to work. Further investigation was done for this phenomenon, and it appeared that for this microcontroller (it was still using the old one MSP430G2250), to connect the lanchpad evaluation board to another power supply, so the power supply jumper had to be disconnected first. The new evaluation board used for the MSP430 F series does not have this problem.

Now having the module working, the steering module control movement was showing an obvious overshoot reaction when the Brain send a command to go to the maximum left or right position. In order to avoid sudden movement, the Brain code was adjusted to send the message in steps by increasing 1 degree in each message till the final angle required is reached. This testing was done on the ATV while it was suspended and the front wheels are not touching the ground.

3.7.4 The Brain implementation for the braking system

A test drive for the ATV showed the ATV goes into stationary state once the throttle lever is released. This make the idea of no need to partial press the brakes to slow down. As whenever a slow down is required, the duty cycle value of PWM controlling the throttle is capable to slow it down. The only use for the braking system is to get the ATV to complete stop. The Brain was configured to send a message with value of '8' to apply brakes, and another message with a value of 127 to release the brakes. The values were chosen to have only one '1' value in applying the brakes, and only one '0' in releasing the brakes. The code on the braking system was getting the messages and act accordingly as described in the braking system section above.

3.7.5 The Brain algorithm for reaching the final destination using the GPS and IMU

The job for the Brain is to guide the ATV toward the final destination by comparing the readings of the GPS at the current position and comparing it to the final destination while using the IMU to make sure the ATV is moving in the right direction.

The software takes the longitude and latitude from the GPS. This information is the location in North-South and East-West. As the study is performed in the USA, it will always be North and West. Comparing the values in the North direction, if the value of the latitude is less than the destination value, the ATV must head north. Otherwise, if it is more than the destination value, then it must head south. Same idea for longitude readings in the West direction. According to GPS and IMU, the Brain will order the steering wheel controller to go to the smaller angle of the circle to face the destination, after starting the throttle at medium PWM duty cycle. The steering wheel should not get any angles if the throttle duty cycle is less than 60% to

avoid overheating the power assist motor.

If the distance is more than 500 feet, the Brain will give the order to the throttle to run on 75% duty cycle, if between 100 feet and 500 feet, it will be running at 65% duty cycles, less than that, at 62%. When the destination matches the value, it shall apply 5% duty cycle to the throttle and apply brakes.

This description is without LIDARs sensor. In case LIDARs are fixed, in addition to the above software, an obstacle avoidance mechanism will apply. According to LIDARs and ATV motion, if the obstacle is moving in the same speed or higher away from the ATV (in the same direction), then no action is required. If an obstacle is fixed or moving toward the ATV, the Brain should apply an angle to the steering controller according to the obstacle dimension. The direction of the angle is planned as per the shortest way to avoid the obstacle. After obstacle avoidance is successful, a back on track mechanism should start again using the IMU and GPS one more time.

Having the ability to reach a GPS position, in case of mountains and the need to turn around, the ATV may have multiple destination points in a sequence. Once the ATV reaches the first destination, the next point is the new destination. Using this method helps to have a route to follow.

CHAPTER 4: OVERALL SYSTEM INTEGRATION, TESTING, AND TESTING RESULTS

This chapter is about overall system integration and testing. The idea was to add a module at a time, adjust the Brain to handle this module and test it, then to test two modules with the Brain, then three and so on. Some modules were in need to be adjusted when the results showed some failure until the system was finally working as required.

The process consists of the physical integration and installation of all modules to the ATV followed by ten phases that were called Brain phases. Most of the phases were tested twice. The first time when the ATV was suspended in the air, above the ground, using four Jackson's metallic pyramids. The second time is testing it on the ground. The main idea is to see the reaction of the ATV to the actuators actions without any accidents. Also, check the reaction of the actuators to the commands received by the Brain. Besides, in the previous chapter, some simple tests were done, but it was a one on one communication; between the Brain and the actuator. Attaching an additional component may have an unexpected reaction on the other actuators. That is why it is safer to test the ATV while suspending it in the air before putting it back on the ground. Figure 4.1 shows the ATV suspended on Jackson's stands.

For more safety, a "kill switch" was installed in the ATV that kills the engine in emergency cases. This switch is designed to be pressed in case the ATV testing goes out of control. The switch shown in Figure 4.2 was connected to the engine ON/OFF switch. In order to have this switch, the engine ON/OFF button should be switched to OFF.

Figure 4.1: The ATV suspended on Jackson's stands.

Figure 4.2: The ATV safety kill switch.

4.1 Physical installation of the autonomous ATV components

While testing the modules at first, each module has its MSP430 connected to the CAN Shield using some wires. All of them are powered using the lab power supply which is not the best setting to use on the ATV. First of all, the lab power supply is not a practical power supply. It is just for testing. On the other side, the idea of connecting the CAN Shield to the MSP430 using wires while having all ATV vibration is going to lead to loose wires and bad connections. These bad connections can lead to another malfunctioning of certain modules. As a solution to this problem, a PCB was designed to have all components to be connected. This PCB was designed in a generic method. It is connected to the CAN Bus power supply and has its power supply to convert the 12V into 5V. It also has a place to mount the MSP430, another place for the CAN Shield, and a place for the IMU in case the IMU module is used, and a place for the GPS in case of using the GPS module. Figure 4.3 shows the PCB design Schematic. A PCB layout shown in Figure 4.4 is then generated, and all Gerber files were exported, and a sample was printed in the University Lab. After that, the Gerber files were sent to a fabrication facility in China to produce ten boards for the ATV modules. Figure 4.5 shows the manufactured board while Figure 4.6 demonstrates the MSP430 connected to the CAN Shield using the fabricated board. A simple test is done after mounting the boards to make sure the connections are working fine.

Then, in order to fix the different modules on the ATV, ten boxes were designed and printed using a 3-D printer. The idea behind those boxes was to contain and protect the boards from different environmental situations. Boxes were designed to keep the boards safe while the ATV is moving. They were also useful in case of rain to protect the boards from the water. Figure 4.7 shows the 3D model, while Figure 4.8 shows the 3D printed boxes connected using the CAN bus. The box was designed as follows:

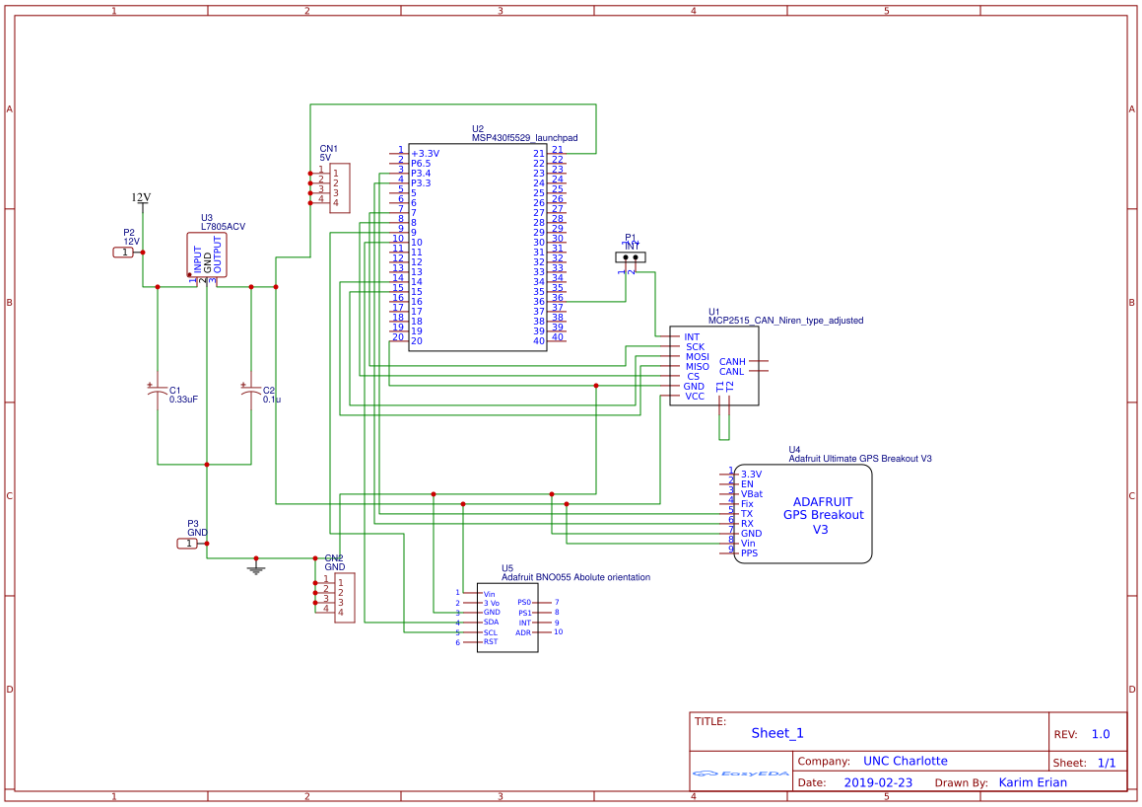

Figure 4.3: PCB Design Schematic

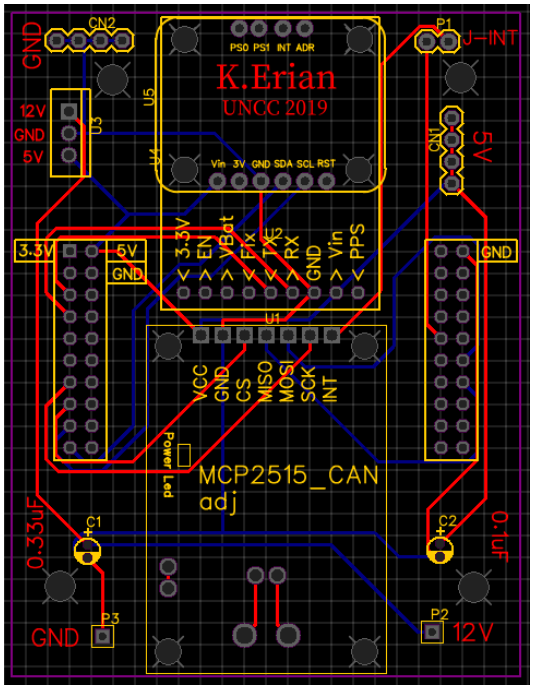

Figure 4.4: PCB Layout

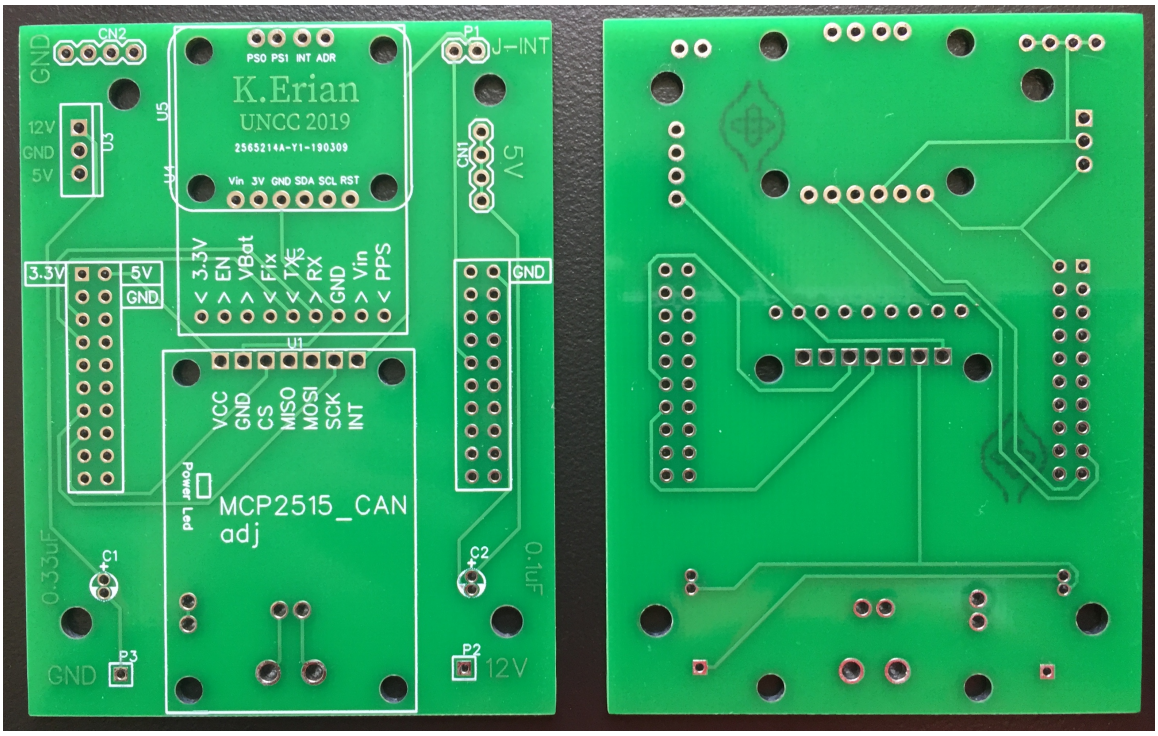

Figure 4.5: Fabricated PCB

- The box has a place at the front to include the CAN bus connector.
- It has four cylinders at the bottom to hold the board using screws. In the first trials, the cylinders were fragile; with this height, the cylinders broke very fast. That is why the cylinders are made with bigger thickness.
- The box also has two holes on the side, one of them is above the middle in case wires are needed to connect from the MSP430 to the actuator like the wires needed to control the servo motor of the throttle. Another hole is below the middle to allow the wire for the GPS antenna.
- The cover of those boxes are made as sliding covers to go in and out from the side.
- For the braking system, another box design was made to contain the Dual H Bridge motor that controls the linear actuator. The difference was the size of

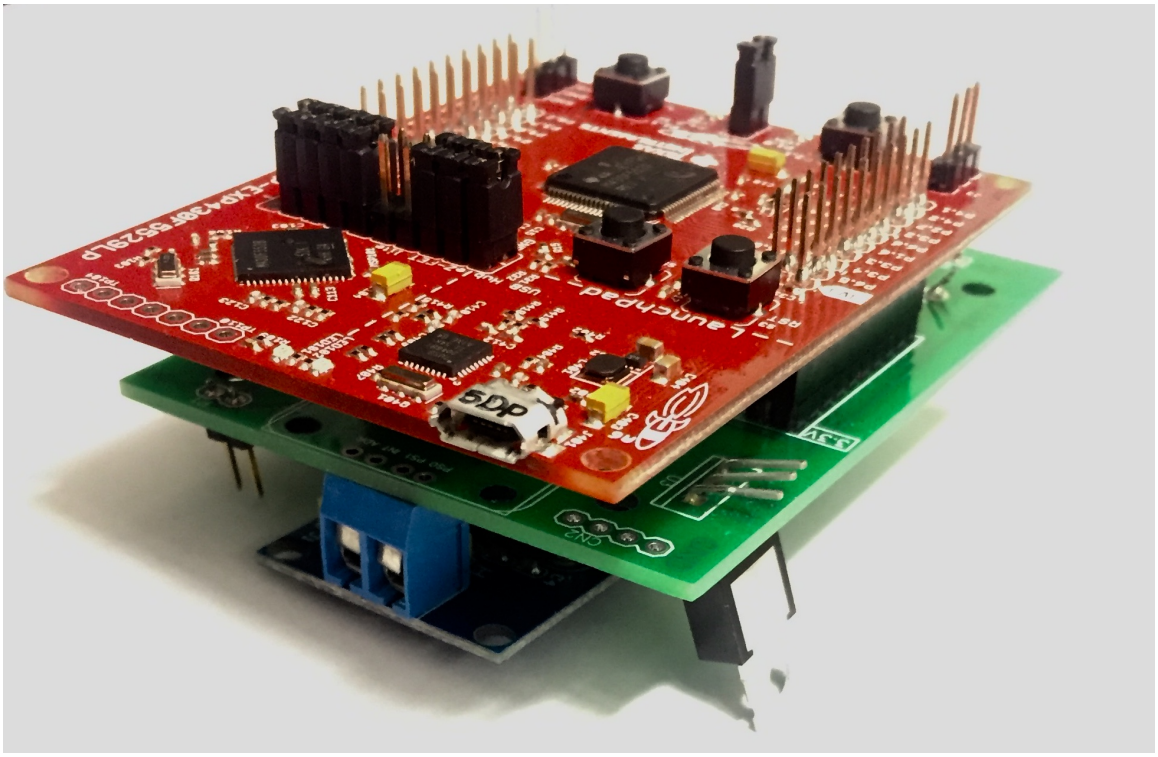

Figure 4.6: MSP430 connected to CAN Shield through the fabricated PCB

the box as the board has a bigger dimension than the other modules. However, the controller for this board was still using the same small design. The reason for having two boxes for the braking system is to isolate the Dual H bridge connections from the MSP430 pins to prevent short circuit.

- The boxes are fixed in the ATV using double-sided tape.

Now having the boxes made, the PCB fabricated, and the modules assembled, it is time to put the CAN bus in the ATV. In order to mount the bus on the ATV, the Red wire is connected to the 12V connector of the ATV battery, and the black wire of the bus to the ATV battery's ground connector.

4.2 Brain Phase 1: Brain-Throttle integration

In this phase, a test for the Brain-throttle interaction was done. Test objective was to make sure of the Brain ability to control the throttle. The throttle module was

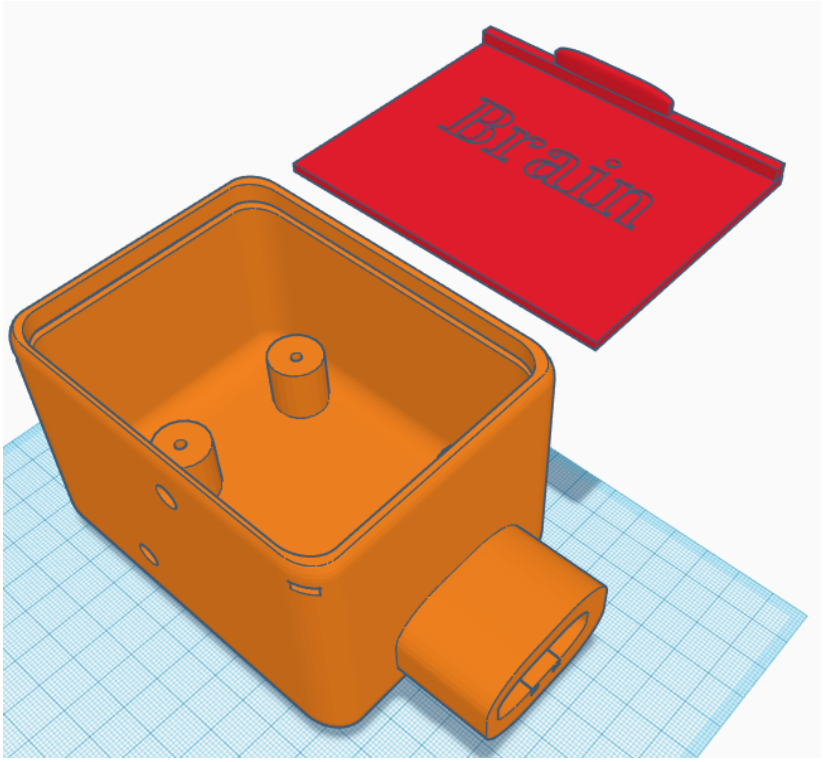

Figure 4.7: The 3D model design for the box used to contain most of the modules

connected to the servo motor and the CAN bus. The Brain was connected to the CAN bus. Its software sent in sequence values 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 60, 30, and 0 separated by 1 second delay each. The wheels started to move, and the speed was constant for the first 6 seconds, then it started to accelerate for 5 seconds, then slowed down for 3 seconds and moved slowly afterward.

The ATV was put on the ground where it did not move during the first 6 seconds; then it accelerated and reached a higher speed for the value of 90 before it stopped.

Next, the ATV was mounted on the Jackson's, and another series was sent to the throttle controller: 0, 40, 0, 50, 60, 0, 70, 0, 90, 0, 60, 0. The throttle actuator servo motor was under observation. It was noted that the throttle does not go back to zero position each time zero was sent. It was also noted that the servo does not move the ATV throttle factory assembly for values less than 60. Another observation was that the servo does not go from 50 to 60 to move the throttle assembly. The servo motor

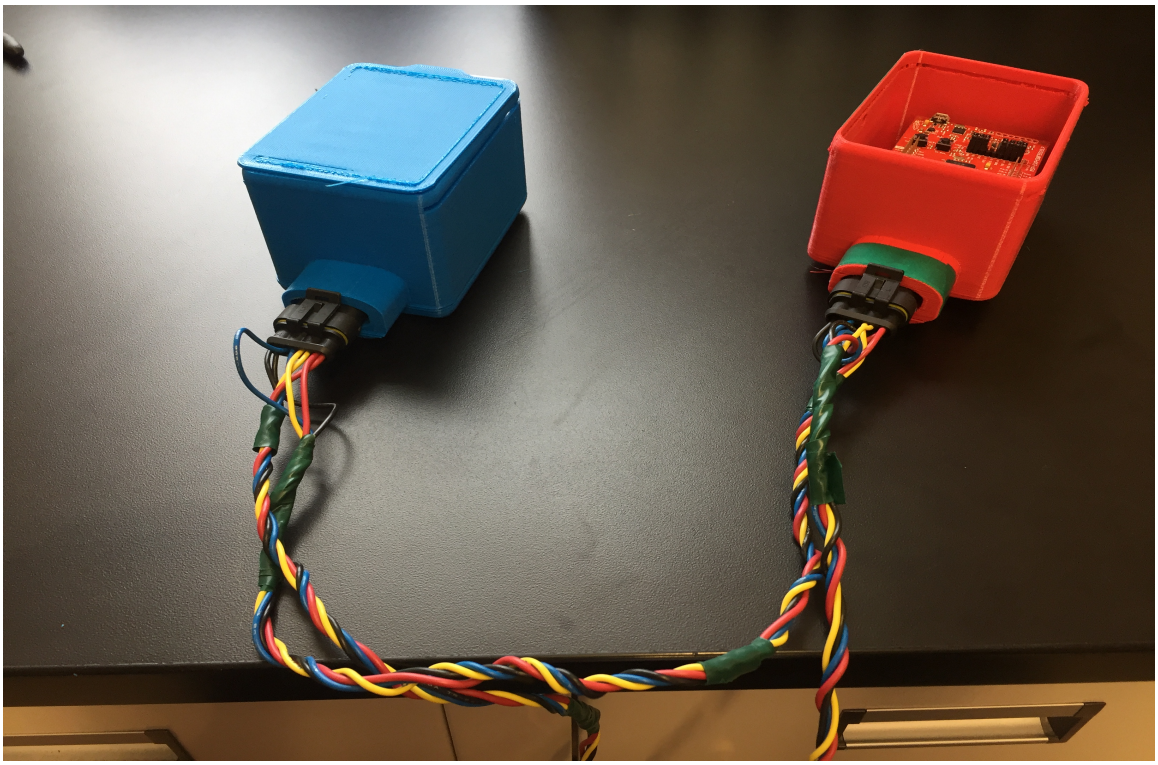

Figure 4.8: The 3D printed boxes connected using CAN bus

seems not powerful enough to move it. However, it can move it if it is applied from 0 to 60 directly.

The following conclusions are reached:

- In order to move the ATV, at least a value of 60 should be sent to the throttle controller.
- As all values below 60 do not affect the ATV, any value above zero is used in case zero is needed. As for some reason, sending zero was not getting the value. It might be the CAN messages with only zeros are neglected. The value of 0x05 is used when there is a need to stop supplying the motor by gas from the throttle.
- For the sake of the servo motor power, in order to move the throttle assembly, it is preferred to move it for a high angle, like from 0x05 duty cycle to 70 duty

cycle.

- The ATV speed while suspended was calculated at 60% duty cycle to be 5.6 mph or approximately 9.6 km/h. The calculation was done by using a camera to monitor the tire revolution per minute for 30 seconds. It reached 80 RPM. Having the tire diameter to be 24" (60 cm), and the RPM, the ATV speed was calculated. This speed value neglects the ATV weight and ground friction which have considerable speed reduction effect on the ATV.

4.3 Brain Phase 2: Brain-Brakes integration

In this phase, a test for the braking system with the Brain is in place. The Brain was connected to the CAN bus. The braking system controller was connected to the CAN bus, and the dual H bridge. The Dual H Bridge was connected to the linear actuator, and the linear actuator was mounted to the ATV.

The Brain software sent to the braking system to apply the brakes command after 5 seconds, and then after 10 seconds another command was sent to release the brakes.

As a result, the braking system did not work first when the system was connected to the power source. When the Brain reset button was pressed, the braking system worked.

As a conclusion, the braking system needs 14 seconds to start working. A delay was introduced in the Brain code to wait for the braking system boards initialization to be done before sending any messages to the system. This delay fixed the issue, and the braking system works with the Brain as designed.

4.4 Brain Phase 3: Brain-Throttle-Brakes integration

In this phase, the Brain was tested for controlling the throttle and the braking system together. The Brain, the throttle, and the braking system were connected to the CAN bus and mounted on the ATV. After finishing the introduced delay, the Brain software sent 0x05 to the throttle to reset it, then 1 second later it sent 60 to

the throttle. Then after 2 seconds, it sent to the throttle 0x05 to stop the air supply to the engine. After 20 ms, the Brain sent to the brakes the apply command then after 5 seconds the brakes release command.

As a result for this experience, the engine sound went up when it was set to 60, then down when 5, then up again when the brakes were released. However, the brakes did not work.

It was noticed that when the throttle was removed, the brakes were working fine, but when the throttle module was attached to the CAN bus, the braking system did not work. The throttle module was located on the bus between both the Brain and the braking system. A further investigation took place, and it was noticed that the CAN Shield 120-ohm resistor was connected in all nodes. These resistors made the Brain able only to communicate with one node after and one node before it. As the existence of the 120-ohm resistor terminates the bus, the Brain was not able anymore to communicate to the braking system.

As a fix, all resistors in any CAN Shield were removed except for the first and last nodes on the bus.

The test was repeated again. The braking system worked fine. However, the throttle servo motor was moving whenever a command was sent to the braking system. When the Brain sent release command to the braking system, the engine sound goes up, as the release command is a message with the value 127. This loud sound meant that the throttle controller was listening to the messages addressed to the braking system. The system was not supposed to act this way. A further investigation was done, and it was noticed that in the CAN shield initialization, the mask register was cleared, and the filter was set to accept any messages.

As a fix, the mask was set to 0xffffff which means to apply a filter on all bits of the address, and the filter was set to accept only the messages addressed to the specified module. This solution was applied to all modules. The test was rerun, and it worked

as expected. Then, the same test was done on the ground, and the ATV moved forward for 2 seconds then it stopped, then after 5 seconds, the brakes were released. This phase was finished successfully.

4.5 Brain Phase 4: Brain-Throttle-Brakes-Steering integration with a turn right

This phase added one more device, the steering wheel controller. In this case, the Brain sent the throttle a reset to 0x05 command then another command to run on 65, then sent the steering wheel a command to turn right with a value of 100 points to the right (value of 821). As discussed earlier, the steering value for the center position is 721, and the maximum right position is 876. Then after 2 seconds, returned to the center position, and then put the throttle on 0x05 and applied brakes for 5 seconds and then released them. This test was done while the ATV was suspended in the air. The wheels started to move then the steering jumped to the right with an angle of 33^o for the right tire and 34^o for the left tire. Then after 2 seconds, the steering wheel did a sudden jump to the center position, then the brakes were applied for 5 seconds and then released. The following conclusions were made:

- The movement of the steering wheel was too fast. This sudden change may cause the ATV to be unstable and turn over. As a result, a function was implemented to smooth the turns. This function increments the angle within a period by adding 5 points at a time and wait 250 ms between each increment.

- Some calculations were made to understand the time for the steering to be maintained to the right in order to have a circle or to make a 90 degrees right turn.

- This phase is completed as expected. However, it is not tested on the ground, as on the ground turning was done in further phases with specific angles.

4.6 Brain Phase 5: Brain-Throttle-Brakes-Steering integration with turning left

This phase is similar to the previous phase but testing the turns to the left. In this case, the Brain sent the throttle a reset to 0x05 command and after 1 second, a command to run on 65, then sent the steering wheel a command to turn left with a value of 100 points to the left (value of 621). As discussed before, the steering value for the center position is 721, and the maximum left position is 566. Then after 2 seconds, returned to the center position, and then put the throttle on 0x05 and applied brakes for 5 seconds and then released them. This test was done while the ATV was suspended above the ground. The wheels started to move then the steering moved in phases to the left as the smoothing function from the last phase was applied. It went to the left with an angle of 31^o for the right tire and 29^o for the left tire. Then after 2 seconds, the steering wheel did went back to the center position smoothly, then brakes were applied for 5 seconds and then released. As a conclusion, this phase met the expectations.

4.7 Brain Phase 6: Turning ATV in one complete circle

In order to make the ATV turn in one complete circle, some calculations had to be done. These calculations used the steering dynamics theories with the Ackerman steering condition (having a different angle for the inner wheel different than the outer wheel to all a turn with a slip-free) [23]. Figure 4.9 demonstrates a turning vehicle. In the Figure 4.9, L is the distance between the center of the front wheel and back wheel, in this case, $L = 50$". θ is also known as per the angle used in previous phases to be $30°$. To get the time needed to complete one circle, the following calculations

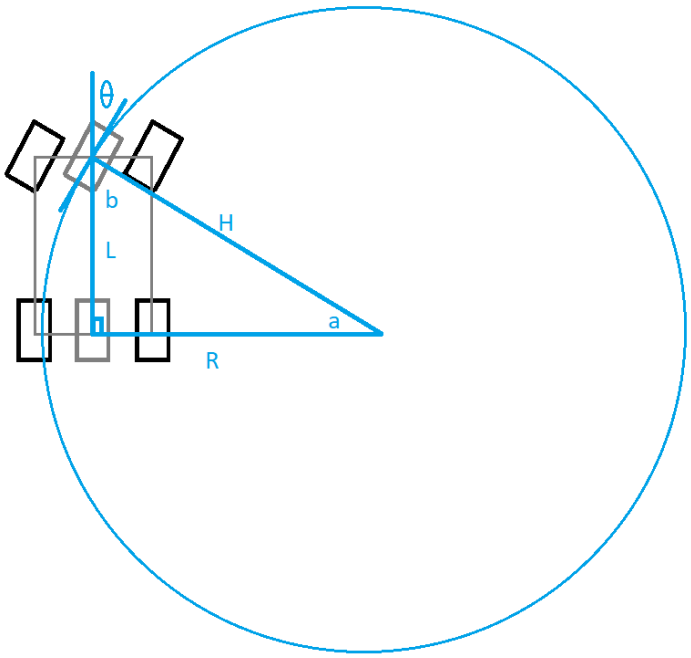

Figure 4.9: A turning vehicle

are used:

$$
\begin{aligned}
b &= 90 - \theta \\
b &= 60^\circ \\
\cos b &= \frac{L}{H} \\
H &= \frac{L}{\cos b} \\
&= \frac{50''}{1/2} \\
&= 100'' = 2.54\text{m}
\end{aligned}
$$

$$
\begin{aligned}
CirclePerimeter &= 2\pi \times \mathrm{H} \\
&= 628.318'' = 15.96\mathrm{m} \\
Speed &= RPM \times TirePerimeter \\
&= \frac{80 \times 0.6 \times \pi}{60 \text{ seconds}} = 2.51\mathrm{m/sec} \\
v &= \frac{\Delta d}{\Delta t} \\
\Delta t &= \frac{\Delta d}{v} = \frac{15.96}{2.51} = 6.35\mathrm{sec}
\end{aligned}
$$

With the previous results, the Brain software was coded to move the ATV forward and then start turning left for 6.4 seconds delay then stop. The ATV response while suspended in the air was good.

However, to make sure it works, it was the time to test it on the ground. On the ground testing, the ATV moved forward, but the steering wheel did not turn in any direction.

This problem was investigated, and a conclusion was made. The EPS motor is relatively weak to move the steering wheel by itself for a small angle with 5 points at a time. The steering function was modified to increase the degrees from 5 points to 10 points in a shorter interval of 100 ms instead of 250 ms. Still, this did not fix the issue.

The test case was modified and rerun. After 1 second, the throttle was stopped, and the brakes were applied. Then the Brain code was changed to do a sudden change of 100 points, by directly sending a message to the steering module with the value of 821. As an observation, the steering wheel moved but only 7°, not 30°.

In order to fix this problem, the maximum right value of 876 was sent directly. As a result of this test case, the ATV front tires moved to the right with only 10°. This angle made the circle be 15 meters of diameter, and it would take more time and

distance to have the circle done.

The same test was done again while putting the ATV on the reverse gear with the steering value at 876. The result was astonishing. The ATV steering angle was 30°.

Further investigations were done to understand the reason behind having a higher angle while going backward than going forward. As a conclusion, there are two main reasons:

The first one is that the ATV is equipped with a power assist module that is meant to help a human operator to turn the steering wheel easier when the ground resistance is high. The EPS needs human operator force to work. It is not meant to move the steering wheel alone, so the motor power is not strong enough to make it work by itself.

The second reason, which makes it turn easier when the ATV is going backward, is the Caster Angle [24]. The ATV front wheels axis are not perpendicular to the ground. The axis was inclined with an angle called Caster Angel. This angle is intended to make it easier for the ATV to maintain a straight line position by making the stable mechanical position for the front wheels in the center position while going forward. As shown in Figure 4.10, the projection of the axis is in front of the tangent point of the wheels on the ground, which makes the wheels follow the axis projection. Whenever the steering wheel moves to the left or the right, the ATV weight will act as a force that pushes the steering wheel back to the center position.

This phenomenon is not the case while going backward. When going backward, the projection of the axis on the ground is behind the wheels tangent point to the ground. This position is not a stable mechanical position for the wheels and leads the wheels to try to go behind the axis' projection. As a consequence, when the steering wheel moves a little bit away from the center position, the ATV weight will help the steering wheel to move more. Figure 4.11 demonstrate the Negative Caster Angle.

Another test was made to prove The Caster Angle reason theory. The test was

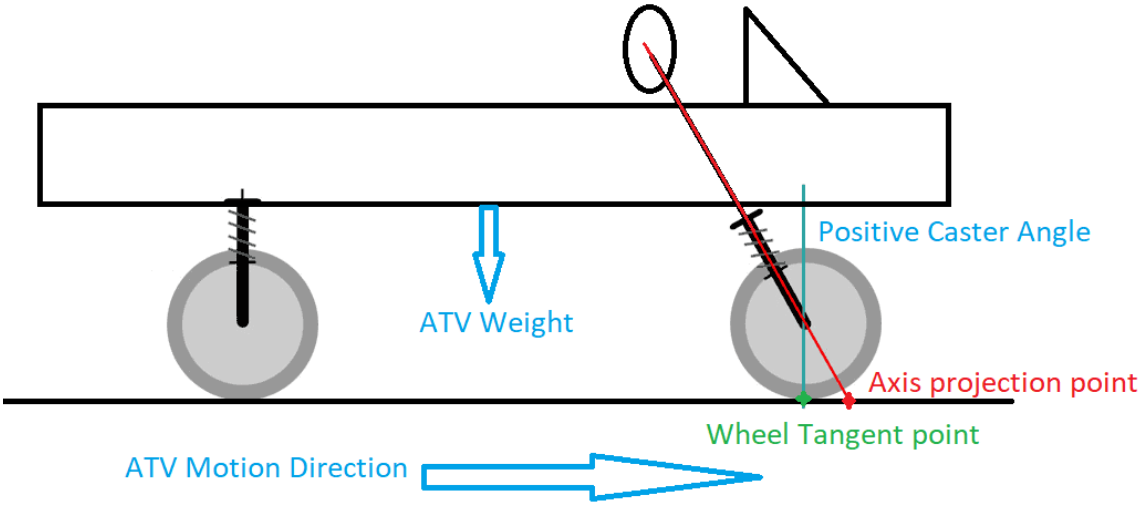

Figure 4.10: Positive Caster Angle for the vehicle direction.

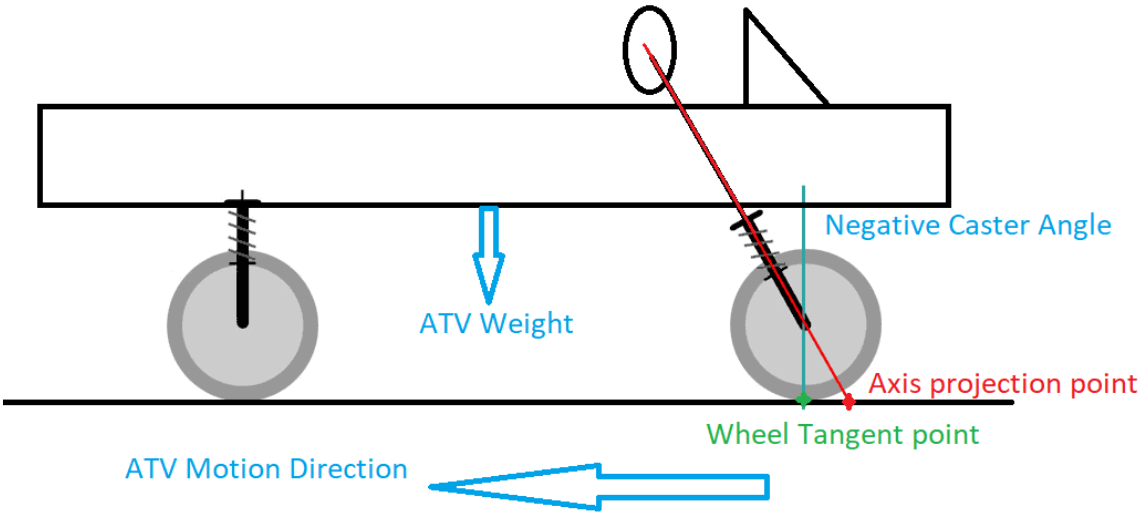

Figure 4.11: Negative Caster Angle for the vehicle direction.

to manually move the steering wheel slightly to the right when the ATV is running backward while disconnecting the steering module controller. As a result, the steering wheel went to the maximum right position. This result proved that the stable point for the wheels while going backward is behind the projection point of the Axis on the ground.

As a conclusion to this phase, for the sake of the steering angle, the ATV should be used backward, or the Caster Angle should be adjusted to be neutral Caster Angle, or the EPS motor should be replaced with a more powerful motor. At this state, a decision was taken to use the ATV backward; after all, this is an autonomous ATV and not designed for passengers.

4.8 Brain Phase 7: Left backward turning angle in a continuous circle

In this phase, the ATV was tested to go backward in a circle. The test case is much similar to the previous test case of the last phase. The engine was started, and the ATV was set on the reverse gear. The Brain had reset the throttle to 0x05, and the steering wheel to center position. Then the Brain had set the throttle to 63, after 1 second, the Brain sent a message to the steering wheel with the maximum left value of 566, and keep it running till the kill switch is pressed.

As a result of the first run of this test, the ATV started to run backward in a circle, but after 7 seconds, the steering wheel turned back to the center position.

A conclusion was made, that the steering module has a timeout after which it resets itself to the center position. A fix was done by periodically sending the steering value over the CAN bus.

The test was repeated, the steering kept its direction, but the throttle went to zero after a timeout. Another fix was made by having the Brain reset the throttle to 0x05 then back to 63 periodically. Keeping the throttle on 0x05 for 100ms does not affect the ATV speed. However, this method overcomes the timeout for the throttle motor.

The test was repeated a third time. The ATV started to run backward in a circle

continuously. Measurements were taken, and the circle radius was 2.54 meters. The average of the steering angle of both wheels was 30°. The average time per complete circle was 10 seconds. This duration means the ATV speed at a horizontal surface with the throttle set at 63% duty cycle is 1.59 m/sec which is nearly 3.56 mph. Concluding, the ATV weight and the ground friction reduced the speed from 5.6 mph to 3.56 mph with a percentage of 37%.

4.9 Brain Phase 8: Moving the ATV to drive figure '8' backward and stop it at the starting point

In this phase, the previous measurements were used to prove the ability to control the ATV autonomously. The Brain software started with a throttle reset to 0x05 value, and the steering wheel to center position. The ATV was on the reverse gear. Then the Brain set the throttle on 63% for 1 second. The Brain then set the steering wheel to the maximum left and kept it running for four seconds. Following, the Brain then reset the steering wheel to the center position. After two seconds, the Brain sent the command to the steering module to go to the maximum right position. Eight seconds later, the Brain reset the steering module to the center position. Two seconds after, it set the steering module to the maximum left position. After four more seconds, the Brain reset the throttle to 0x05 and apply the brakes.

This phase showed astonishing results by drawing the '8' figure on the ground and stopping at the start point.

4.10 Brain Phase 9: making a 90 degree turn to the right while running backward

In this phase, another application of the measurements done in Phase 7. This test was to check the Brain ability to control the ATV to make it turn for 90° to the right. The Brain started by resetting the throttle to 0x05, and the steering to the center position. The ATV is set on reverse gear. The Brain sent to the throttle to start at 63% duty cycle, and after 1 second, it sent to the steering module to turn

right at maximum value. After 2.5 seconds, the Brain sent another CAN message to the steering module to reset the steering module to the center position. One second later, the Brain reset the throttle to 0x05 and applied the brakes for 10 seconds then released it.

As a result, the ATV was able to make a 90 degrees turn to the right by having an arc of a circle with a radius of 2.5 meters approximately.

4.11 Brain Phase 10: making a 90 degree turn to the left while running backward

In this phase, another application of the measurements done in Phase 7. This test was to check the Brain ability to control the ATV to make it turn for 90° to the left. The Brain started by resetting the throttle to 0x05, and the steering to the center position. The ATV is set on reverse gear. The Brain sent to the throttle to start at 63% duty cycle, and after 1 second, it sent to the steering module to turn left at maximum value. After 2.5 seconds, the Brain sent another CAN message to the steering module to reset it to the center position. One second later, the Brain reset the throttle to 0x05 and applied the brakes for 10 seconds then released it.

As a result, the ATV was able to make a 90 degrees turn to the left by having an arc with a radius of nearly 2.5 meters.

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

5.1 Conclusion

As far as this study reached, it is obvious that transforming an existing ATV to an Autonomous ATV at low cost is feasible. The integration is the most challenging phase but is an essential one for a properly working system. Following one standard for implementing and integrating all the parts, makes integration much easier, as all parts speak the same language at the end. The CAN network is successful in communicating all modules together. The distributed processing effort reduces the CPU load of the Brain, makes it faster, and allows to use a lower grade, cheaper, and more reliable processor.

In this study, I was able to implement a standardized platform for autonomous self-driven ATV at a relatively low-cost ready for extra modules with low effort for integration. This system works without any human interaction. The ATV is capable of driving (backward) and execute the turns of a previously programmed path. The ATV is able to perform a figure '8' and stop exactly at the same start point. The ATV is capable of making sharp backward turns with a radius less than 3 meters.

The code for all modules follows the same hierarchy by having an HWI layer as the abstraction layer between the Hardware and the Software layer. A driver layer is implemented above the HWI layer using the HWI library to perform specific functions for different modules like the CAN communication, the braking system, the steering control, and others. An application layer is built using all the previous layers, the HWI layer, and the drivers layer. This Application layer changes from one module to another according to the module application. This hierarchy is the standard one for the software among the entire system. The HWI and Drivers layers are the same for

all devices.

The hardware used in most of the modules is MSP430 F-series connected to the CAN shield with the MCP2515 CAN transceiver through a custom fabricated PCB which includes the power supply. This system is enclosed in a 3-D printed box that includes a uniform CAN bus connector. All boxes are connected to the same CAN bus with the same configuration of 250 kbps, and sample point at 87.5%.

This hardware is similar in all modules with minor addition as needed of extra wires or extra boards according to the hardware functionality.

The standardized system is controlled using a simple, low-cost Brain made by MSP430 and a CAN shield. The Brain is capable of controlling the throttle, the steering wheel, and the brakes and synchronize between all of them in a real-time environment.

The system is ready for GPS and IMU integration. The GPS and IMU modules are implemented in a parallel study, and the libraries for integrating the GPS and IMU are designed, implemented and ready for testing.

This platform is standardized in a way which allows an easy new modules integration without huge effort. All what is needed is either to use another MSP430 with CAN Shield with the same PCB and connect it to the new sensor or actuator, and use the libraries to communicate with the Brain, or to use a different hardware but configure the CAN configuration to use same configurations with 250 kbps and 87.5% sample point. Then add the new module code to the Brain MSP430.

5.2 Future Work

This study proved that there are many chances of improvements and it is technologically feasible to make the ATV more advanced. Ideas for enhancements are listed below:

- Testing the GPS and IMU integration, and make the ATV follow a final destina-

tion by comparing the final destination with the GPS reading. The GPS library implemented provides the distance between two GPS locations and the angle between those locations with respect to the North direction. Using the IMU which provides the angle between the ATV direction and the North direction, it is easy to control the ATV toward the final destination.

- Adding to the previous point, multiple destination points may be used in case the final destination is hard to be reached in a straight line like in case of rivers or mountains or lakes.

- Add a speed feedback system by integrating the wheels encoders installed on the ATV. As when the ATV was tested in different environments, with different slopes, the ATV speed is significantly different. This speed variation will impact the turning angles, at a higher speed to do 90° turns will need less time. This feedback should be used to detect the error in the throttle duty cycle and adjust it accordingly.

- Implementing obstacle detection module, it is preferred to use two LIDARs, one to detect the obstacles in front of the ATV, and the other to detect cliff or holes in the road.

- Add machine learning and AI to control the ATV. Right now the ATV is controlled using algorithms. Those algorithms may seem simple in small roads with no obstacles and no movable objects. However, with real environments, it is tough to cover all situations using algorithms. It is much easier to have the ATV learn from a human operator by recording feedback from all actuators and compare the values with the corresponding sensors values. Using the recorded data to train a machine learning model and then use the trained model to control the ATV will be an excellent idea to achieve [2].

REFERENCES

[1] "Electric Cars, Solar Panels & Clean Energy Storage | Tesla."

[2] "Valeo - Smart technology for smarter cars."

[3] "Future of mobility overview | Deloitte Insights.."

[4] B. B. Rhoades and J. M. Conrad, "A survey of alternate methods and implementations of an intelligent transportation system," in *SoutheastCon 2017*, pp. 1–8, IEEE, 2017.

[5] B. B. Rhoades, D. Srivastava, and J. M. Conrad, "Design and Development of a ROS Enabled CAN Based All-Terrain Vehicle Platform," in *SoutheastCon 2018*, pp. 1–6, IEEE, 2018.

[6] B. B. Rhoades and J. M. Conrad, "A Novel Terrain Topology Classification and Navigation for an Autonomous CAN Based All-Terrain Vehicle," in *SoutheastCon 2018*, pp. 1–6, IEEE, 2018.

[7] J. R. Henderson, J. M. Conrad, and C. Pavlich, "Using a CAN bus for control of an All-terrain Vehicle," in *IEEE SoutheastCon 2014*, pp. 1–5, IEEE, 2014.

[8] A. Cortner, J. M. Conrad, and N. A. BouSaba, "Autonomous all-terrain vehicle steering," in *2012 Proceedings of IEEE SoutheastCon*, pp. 1–5, IEEE, 2012.

[9] R. A. McKinney, M. J. Zapata, J. M. Conrad, T. W. Meiswinkel, and S. Ahuja, "Components of an autonomous all-terrain vehicle," in *Proceedings of the IEEE SoutheastCon 2010*, pp. 416–419, IEEE, 2010.

[10] J. R. Henderson, J. M. Conrad, and C. Pavlich, "Using a CAN bus for control of an All-terrain Vehicle," in *IEEE SoutheastCon 2014*, pp. 1–5, IEEE, 2014.

[11] K. H. Erian, S. Mhapankar, S. Gambill, and J. M. Conrad, "System Integration over a CAN Bus for a Self-Controlled, Low-Cost Autonomous All-terrain Vehicle," in *IEEE SoutheastCon 2019*, pp. 1–8, IEEE, 2019.

[12] "AutoSAR - Layered Software Architecture," tech. rep.

[13] "BMW.com | The international BMW Website."

[14] R. Cuffley, "Amp Tyco Superseal waterproof connector assembly guide for wiring boats cars, well anything - YouTube."

[15] S. K. Gurram and J. M. Conrad, "Implementation of CAN bus in an autonomous all-terrain vehicle," in *2011 Proceedings of IEEE SoutheastCon*, pp. 250–254, IEEE, 2011.

[16] "MSP430 CAN interface - electrodummies."

[17] “Compiler/diagnostic messages/MSP430/1528 - Texas Instruments Wiki.”

[18] “MSP430 SPI Peripheral | Argenox.”

[19] “CAN Bus Shield.”

[20] “MCP2515 NOTES,” tech. rep.

[21] “Build an Arduino CAN Bus Network | Henry’s Bench.”

[22] HONDA, “Honda Trx420 Rancher 420 Service Manual Repair 2007-2010.”

[23] “Steering Theory Chapter 7: Steering Dynamics,” tech. rep.

[24] “Learn About Positive and Negative Camber, Caster, and Toe.”