

AMERICAN UNIVERSITY OF BEIRUT

AUTOMOTIVE SAFETY INTEGRATION USING DRUNK
DRIVING BEHAVIOR DETECTION AND MULTI-FACTOR
AUTHENTICATION

by
AHMAD EL BASIOUNI EL MASRI

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Electrical and Computer Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
December 2017

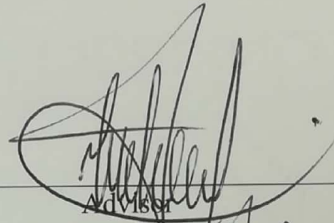
AMERICAN UNIVERSITY OF BEIRUT

AUTOMOTIVE SAFETY INTEGRATION USING DRUNK
DRIVING BEHAVIOR DETECTION AND MULTI-FACTOR
AUTHENTICATION

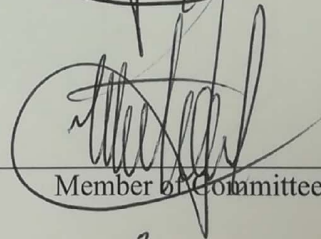
by
AHMAD EL BASIOUNI EL MASRI

Approved by:

Dr. Haitham Akkary, Associate Professor
Electrical and Computer Engineering



Dr. Hassan Artail, Professor
Electrical and Computer Engineering



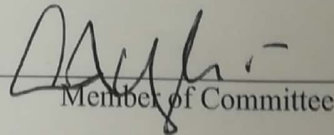
Member of Committee

Dr. Hazem Hajj, Associate Professor
Electrical and Computer Engineering



Member of Committee

Dr. Mazen Saghir, Associate Professor
Electrical and Computer Engineering



Member of Committee

Date of thesis defense: Dec 18, 2017

AMERICAN UNIVERSITY OF BEIRUT

THESIS RELEASE FORM

Student Name: El Basiouni El Masri
Last

Ahmad
First

Salim
Middle

Master's Thesis

Master's Project

Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes
after : **One** — year from the date of submission of my thesis, dissertation, or project.
Two — years from the date of submission of my thesis, dissertation, or project.
Three — years from the date of submission of my thesis, dissertation, or project.



Signature

13 Feb 2018

Date

ACKNOWLEDGMENTS

I would like to thank Prof. Haitham Akkary, my advisor, for his continuous support and faith in me. Thank you for being patient and helping me gain valuable knowledge through my studies. I would also like to thank Prof Hassan Artail, Prof. Hazem Hajj, and Prof. Mazen Saghir my thesis committee members, for their valuable support along my journey.

Last but not least, I would like to thank my parents who gave me valuable support taking care of my son during the exam periods and heavy workload weeks. I would also like to thank my wife for her patience and support during my study.

AN ABSTRACT OF THE THESIS OF

Ahmad El Basiouni El Masri for Master of Engineering
Major: Electrical and Computer Engineering

Title: Automotive safety integration using Drunk driving Behavior Detection and multi-factor authentication

With the emergence of the self-driving cars, and the expansion of the electronic controls of the modern vehicle, there exist a need for better security measures and self-policing capabilities. Considering the emerging use of 3rd party wireless connected car dongles, we explored building a prototype to expand upon the existing solution's safety and security features.

First, we investigated several approaches to connect our prototype to the car's CAN Bus interface. Using messages collected from the car's OBD-II port, it detects the signals: Ignition, location, trip start, trip end, and car shutdown. It also collects several can bus signals like speed, rpm, engine load, throttle and other OBD-II PIDs (Parameter IDs) . Using these signals and telematics, we can have second factor authentication/notification systems for the important events. These events may include unauthorized car start, unauthorized route, abnormal driving behavior like drunk and intoxicated driving, out of bounds, and any other odd behavior. The acknowledgement or decline of these events may be handled differently according to severity. For example, an un-authorized ignition may not allow the car to start, while a minor intoxicated behavior detection may send a notification to the loved ones, and a major detection of an intoxicated driver may lead the car to gradually stop and notify the authorities. To achieve this, the prototype utilizes an existing solution, 'Acceptto', that offers second-factor authentication capabilities.

For the rest of the thesis, we propose a method to detect drunk driving patterns using only basic car sensors, available through off the shelf OBD-II dongles. The sensor data include standard On-Board Diagnostic sensor information along with an accelerometer sensor and GPS coordinates which are provided by the dongle. We collect the information through drive tests of normal driving behavior and controlled drunk driving behavior. The controlled driving emulation reflects the most common cues relating to drunk driving. After datasets are collected, a window-based approach was used for data smoothing and feature extraction. Finally, our approach makes use of a machine learning algorithm (Logistic Regression) for classification to achieve an accuracy of 82%.

CONTENTS

ACKNOWLEDGEMENTS	v
ABSTRACT	vi
LIST OF ILLUSTRATIONS.....	xi
LIST OF TABLES.....	x
1. INTRODUCTION	1
1.1 Motivation	11
1.2 Background.....	12
1.2.1 Connected Cars.....	13
1.2.2 OBD-II and CAN Bus	14
1.2.3 Acceptto Cognitive Authentication	16
1.2.4 Drunk Driving Statistics	19
1.3 Thesis Contribution and Organization.....	20
2. BUILDING THE PROTOTYPE	22
2.1 Literature Review	22
2.2 Method.....	24
2.2.1 Big Picture	24
2.2.2 Components	26
2.2.3 Acceptto Integration	28
2.3 Demo.....	30
3. DRUNK DRIVING DETECTION.....	32
3.1 Literature Review	33
3.2 Method.....	34
3.2.1 Data Collection	36
3.2.2 Drunk Driving Cues.....	38
3.2.3 Data Cleaning and Feature Extraction.....	41

3.3 Results	42
4. CONCLUSION AND FUTURE WORK	42
REFERENCES	47

LIST OF ILLUSTRATIONS

ILLUSTRATION	Page
1. Udoq Quad Board	13
2. Acceptto concept [25].....	17
3. Acceptto Authentication process	18
4. The high level Design.....	25
5. Overview of the System collecting OBD-II telematics	27
6. Demo application for Acceptto Multifactor Authentication with Car's Ignition	31
7. Drunk driving detection methodology	35
8. Sample data output for the engine load parameter	40
9. Common drunk driving trajectory scenarios [11].....	41

TABLES

Table	Page
1. OBD-II PIDs List.....	16
2. Sensors used to classify drivers	37
3. Normalized Confusion matrix without GPS data	43
4. Precision, Recall and F1 value.....	44

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

In a recent study by Gartner in [20], it is estimated that by 2020, there will be around a quarter billion connected vehicles on the road, introducing the IoV (Internet of Vehicles) as a major player in the world of IoT. Gartner predicts that the Internet of connecting things will reach about 25 billion by 2020. This rapid increase in the market of connected vehicles requires more research in the field of driving telematics. As IoT is connecting our daily used objects and getting valuable telematics, it is time for automotive applications to get into the field. Although there is already many driving analytics solutions ranging from predictive maintenance, driving assistance, safety oriented behavioral analysis, and energy efficiency applications, most of these solutions are proprietary. Furthermore, few solutions take full advantage of the vehicle's internal sensor network and the richness of telematics offered by the CAN bus. Nevertheless, we can utilize driver behavior models with the data inferred from the car's OBD-II to open the way for more complex intelligent automotive applications.

With the emergence of many solutions for connected cars, much less attention is given for safety and security. While there exist many products such as Vinli, Zubie, automatic and HUM, they don't provide solid security options to prevent car theft or hackers access to the car's internal CAN bus system. Furthermore, these solutions provide access to several telematics such as car's speed, location, throttle, engine load and other sensors. However, there are only few proprietary products that investigate the wealth of data collected from the Car's OBD-II port. This thesis presents a security measure using Acceptto's patented solution for cognitive authentication for the car's different events like ignition, out of area, theft possibility, intoxication, etc. We also propose a method to detect drunk driving patterns using only basic car sensors, available through off the shelf OBD-II dongles.

The rest of the chapter includes an overview of OBD-II PIDs (Parameter IDs), an introduction to Acceptto Technology, and an overview to the drunk driving behavior we can capture using our method.

1.2 BACKGROUND

The project we are going to present involves three parts

- Prototype
- Acceptto Integration
- Drunk Driving Analytics

The prototype utilizes an Udoo board [21]. Udoo is an open source, open hardware Arduino-powered Android / Linux Mini PC. It composes an Arduino functionality to connect to the car's CAN bus, and a powerful ARM processor for heavy analytics processing capabilities. The board is shown in figure 1, and will be used for collecting the telematics and events from the car's OBD-II port, and make minor preprocessing of the data, and send these telematics to a back-end server. The server will run an anomaly detection algorithm (in our case Drunk driving behavior detection), and it will communicate with the board for further actions/notifications.

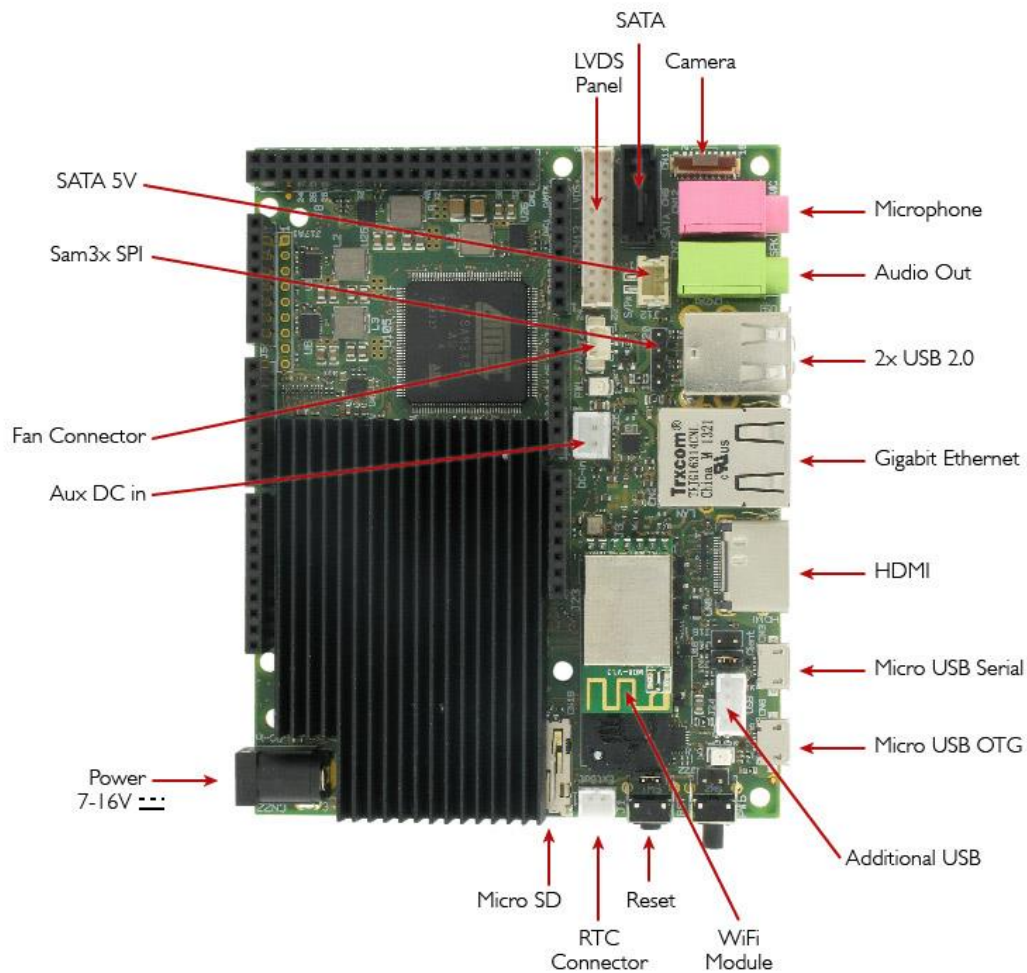


Figure 1: Udo0 Quad Board

1.2.1 Connected Cars

It looks extremely exciting to imagine a world where we can control almost everything from a smartphone, and almost everything feeds you back status details regarding its surroundings. Imagine a world where you can connect everything around you and make it take smart decisions as a cooperative ecosystem. This is what we are emerging into with the evolution of wireless sensor networks, embedded systems and actuators that can be manifested into numerous smart applications. The IoT creates an intelligent interconnected network that can be controlled and programmed. IoT-enabled devices utilize embedded technology that allows them to collect and exchange data, and communicate, directly or indirectly, with each other or the Internet [22].

These advances in the internet of things (IoT) and cloud computing as well have provided a great opportunity to resolve the challenges in a variety of applications. One of these applications is the automotive industry, where the concept of connected cars is being introduced.

There are several projects ongoing in an attempt to standardize the connected car and the interconnection with the IoT. On January 6, 2014, Google announced the formation of the Open Automotive Alliance (OAA) a global alliance of technology and auto industry leaders devoted to bringing the Android platform to cars [24]. The Alliance includes now more than 40 automotive manufacturers. Also, there have been recent works on standards released by ETSI within the Technical Committee on Intelligent Transportation Systems, especially ones related to CAM (Cooperative Awareness Messages). As an example of the awareness from the Automotive manufacturers to the business value of automotive IoT, Volvo has built an Apple Watch version of its On Call connected car platform, which provides safety and location services along with other intelligent services [23]. The Volvo app became available at the end of June 2015, needless to say, that other automotive manufacturers will follow soon with similar products and apps.

1.2.2 OBD-II and CAN Bus

OBD-II (on Board diagnostics) refers to the car's self-diagnostic capabilities. Through the OBD-II port, we can gain access to the status of the various car's subsystems. OBD-II is an improvement over OBD-I. The standard specifies the type of the port connector and pinout, the signaling protocols associated, and the messaging format. We can obtain sensor data and diagnostic information from the electronic control unit (ECE) through the OBD-II port. Thus, we can consider the OBD as a sensory network that provides standard interfaces to collect and analyze the internal engine sensors.

To obtain these sensor information from the OBD-II port, we had several options to explore:

- A CAN transceiver module. The CAN module will only be used as a transceiver to connect to the OBD-II Port. The actual communication with the

OBD-II port and collection/storage of data will be performed through a process on the Linux Board.

- 3rd party OBD-II modules: There exist a lot of 3rd party OBD modules, in which most communicate through Bluetooth. We have to support the connection to the most common OBD-II modules individually. These modules include a module called OpenXC, which is an open source hardware/software solution supported by Ford Motors to provide additional OBD-II PIDs specific to Ford cars.
- Vinli, or similar products like Zubie or Automatic: Vinli has an LTE connection, at a sampling speed of 1 sec. If there is an internet connection problems, messages are buffered, and then retransmitted to the server.

Below in Table 1 we have a list of some of the OBD PIDs (parameter IDs) that can help us with our driving behavior analysis.

Key	Description
S	Standard PID
OXC	Open XC Standard for Ford Cars
OBD-II PIDs	On-board diagnostics Parameter IDs

Official Signals	PID Type	Possible use in Drunk driving detection	Possible use in Aggressive driving detection	Possible use in Sleepy driving detection
steering_wheel_angle	OXC	Y	Y	Y
torque_at_transmission	S	Y	Y	Minor
engine_speed	S	Y	Y	Minor
vehicle_speed	S	Y	Y	Minor
accelerator_pedal_position	S	Y	Y	No
parking_brake_status	OXC	Minor	No	Minor
brake_pedal_status	OXC	Y	Y	Minor
transmission_gear_position	OXC	Minor	No	No
gear_lever_position	OXC	Minor	No	No
odometer	OXC	Minor	No	Minor
ignition_status	OXC	No	No	No
fuel_level	S	Minor	No	Minor
fuel_consumed_since_restart	OXC	No	Minor	No
door_status	OXC	Minor	No	Minor

headlamp_status	OXC	Minor	No	No
high_beam_status	OXC	Minor	Not Sure	Minor
windshield_wiper_status	OXC	Minor	Minor	Minor
latitude	OXC	Biased	No	No
longitude	OXC	Biased	No	No
Absolute load value	S	No	Y	No
Absolute throttle position B,C,D,E,F	S	No	Minor	No
Calculated engine load value	S	No	Minor	No
Commanded throttle actuator	S	No	Minor	No
Engine fuel rate	S	No	Minor	No
Relative throttle position	S	No	Minor	No
Throttle position	S	No	Minor	No
Engine run time	S	No	No	Minor
Ambient air temperature	S	No	No	No
Vehicle identification number (VIN)	S	No	No	No
Sum	0	18	16	12

Table 1: OBD-II PIDs List

1.2.3 Acceptto Cognitive Authentication

Two-factor or multi-factor authentication ensures that entities at the other end of the communication channel are indeed what it claims to be; i.e. provide two or more independent pieces of information as means of authentication such as

- Something they know (e.g. password, PIN, pattern).
- Something they have (e.g. smart card, key fob, mobile phone)
- Something they are (e.g. biometric such as fingerprint, facial or voice recognition)
- Some unique contextual data associated with the user (e.g. location, known device token, known network, etc.)

It is important to note that not only the number of factors involved affects reliability, but also independence of these factors. The more correctly implemented independent factors we use, the higher probabilities that the entity claiming the identity is indeed the owner of the

identity. However, convenience is also a factor to be considered, and a security solution has to consider a tradeoff between security and an acceptable level of usability.

The strength of Multi-Factor Authentication lies on the assumption that if an entity has several authentication factors, then compromising all these independent factors seem far-fetched and much harder for the attacker to penetrate. Nowadays, many companies, including Google, Facebook, Twitter and Apple are now offering their users optional two-factor authentication mechanisms based on OTPs (one-time passwords that are valid for only one login session or transaction).

The solution we are going to use for our project is from a startup called Acceptto. Their product eGuardian is a two-factor authentication solution provider [25]. As seen in figure 2 below, Acceptto Adds a second factor authentication based on personal smartphone pairing. The smartphone is paired with your account and identity; when you Connect to a system and proceed with the initial authentication, the system will attempt to authenticate with the Acceptto's API. Acceptto server will then authenticate your information, and send a notification to the smartphone paired with your account. After you receive the notification, you can give the authorization, deny it or ignore it, as ignoring the notification will render the request unauthorized [25].

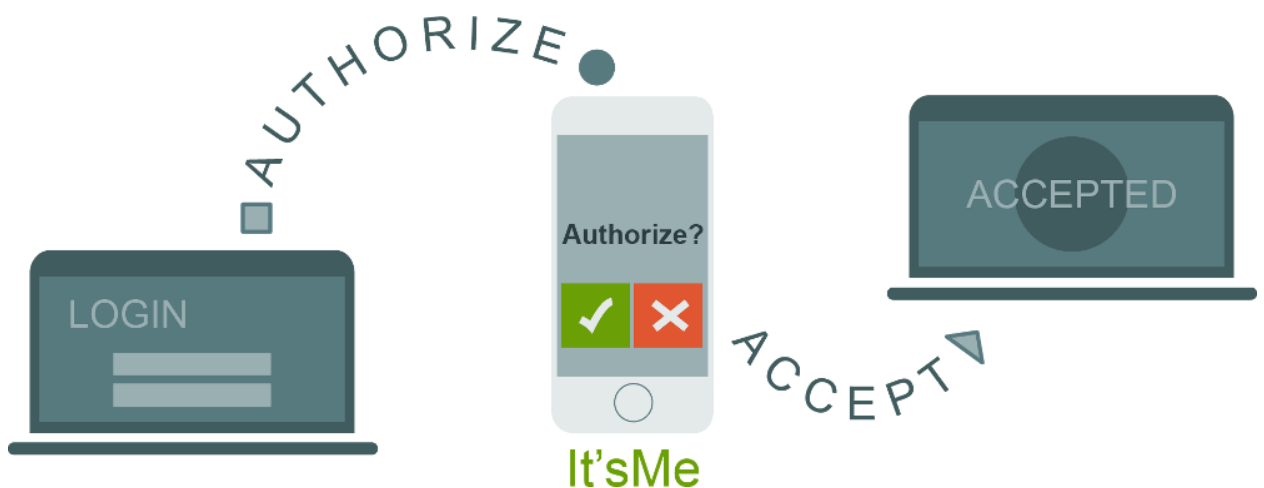


Figure 2: Acceptto concept

Below is a simple illustration of a login transaction using Acceptto's second-factor Authentication:

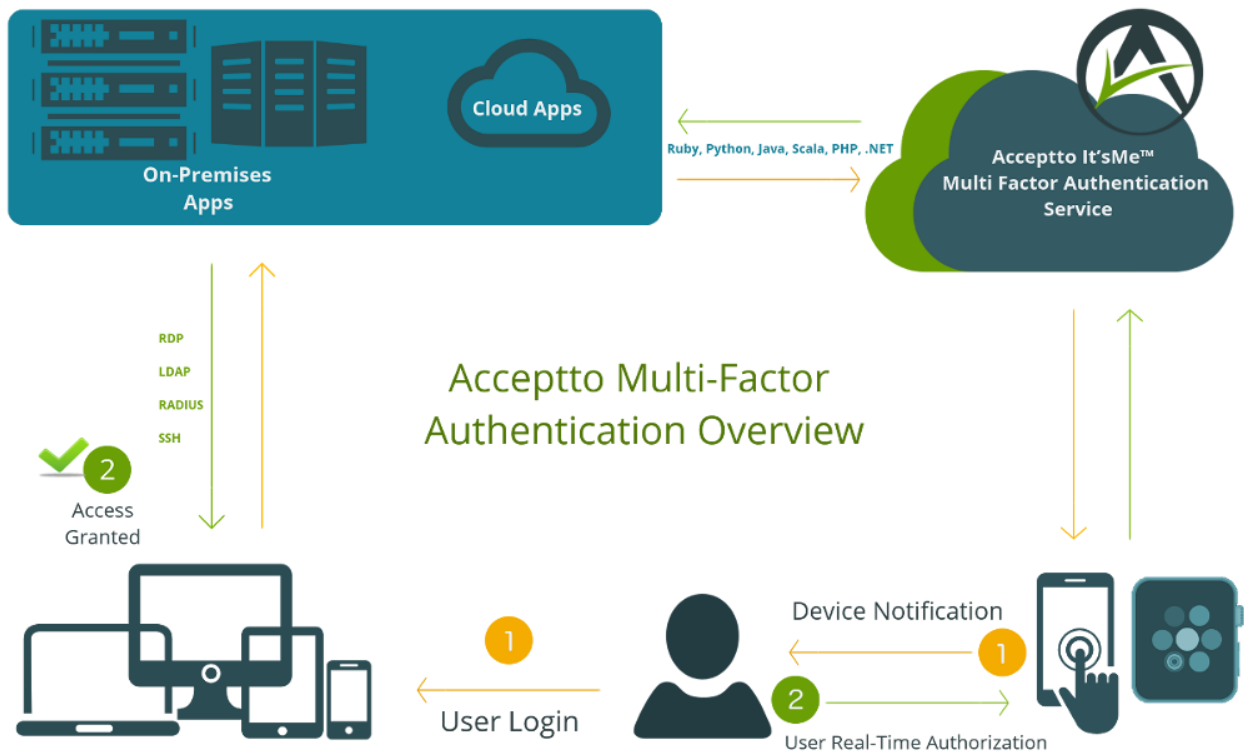


Figure 3 Acceptto Authentication process

After a user login/performs a transaction that needs authentication, the application tries to authenticate as usual. The backend application that has Acceptto integration, will send a request to Acceptto servers indicating an authentication request. Then Acceptto will send a request to the smartphone associated with the device via a mobile App notification. After the user confirms the transaction by accepting or rejecting the notification, Acceptto servers are notified and inform the backend server of the successful authentication status. Afterwards, the backend either authorizes the user or denies access according to the reply from Acceptto. By default, Acceptto rejects authorization after a preset time limit indicated by the backend server where there is no reply from the smartphone user.

1.2.4 Drunk Driving Statistics

For the last 2 decades, the percentage of automotive crashes fatalities resulting from drunk driving remained relatively unchanged, averaging around 29 to 32 percent of total car accidents fatalities [1]. Around 10,000 to 13,000 drunk driving related deaths

occurred per year during this period. These numbers indicate the need for better methods of drunk driving behavior detection. The main detection method used so far rely on visual observations made by police officers. Alcohol-impaired drivers tend to have certain behavior while driving that can be visually detected. Accordingly, the United States National Highway Traffic Safety Administration (NHTSA) has conducted several research studies to provide the law enforcement officers with scientifically validated information that help predict the behavior of impaired or intoxicated drivers. Their study [11] has identified the main drunk driving behavior cues, which are divided into four categories:

1. Problems Maintaining Proper Lane Position
 - Weaving
 - Straddling a lane line
 - Swerving
 - Turning with a wide radius
2. Speed and Braking Problems
 - Stopping problems
 - Accelerating or decelerating for no apparent reason
 - Varying speed
 - Slow speed
3. Vigilance Problems
 - Driving in opposing lanes or wrong way on one-way
 - Slow response to traffic signals
 - Slow or failure to respond to officer's signals
 - Stopping in lane for no apparent reason
 - Driving without headlights at night
 - Failure to signal or signal inconsistent with action
4. Judgment Problems p=.35-.90
 - Following too closely
 - Improper or unsafe lane change

- Illegal or improper turn
- Driving on other than the designated roadway
- Stopping inappropriately in response to officer
- Inappropriate or unusual behavior (throwing, arguing, etc.)
- Appearing to be impaired

We will use some of these cues (Mainly Categories 1 and 2) as our main patterns for drunk driving detection. According to [11], the more cues detected simultaneously, the more probability of drunk driving. For example, if we detect the driver to be weaving, the probability of the driver being intoxicated is around 50%. If we detect a combination of weaving, and acceleration problems, the probability of intoxication rises to around to-70%. For testing purposes, we emulate behaviors falling mainly in the first two categories in order to generate corresponding cues that we measure through the various sensors in the car using the OBD (onboard diagnostic) interface.

1.3 THESIS CONTRIBUTION AND ORGANIZATION

This opens the door for the opportunity to develop a low cost prototype utilizing off the shelf open source hardware and software, to connect existing Cars to the IoT. Furthermore, we are going to harness the diversity of telematics offered by the Car's CAN bus to provide advanced driving behavior telematics. We are going in particular, to explore utilizing car's analytics for increasing energy and fuel efficiency, while also detecting and mitigating any driver's or driving risky behavior. If time allows, we are going to investigate combining telematics from the CAN bus and from smartphones allowing for more complex behavioral analysis for potential driver's risky behavior. From the security perspective, although we are targeting mainly non safety-critical applications, our solution will also incorporate a security function offered by a startup called 'Acceptto' for multi-factor authentication using smartphones.

The thesis is organized as follows:

Chapter Two: Methods to Determine SRAM Stability

- Various stability measures used to determine SRAM stability are presented with the available graphical, analytical and statistical methods of retrieving them.

Chapter Three: The DRV Computing Circuit

- The proposed test circuit, the DRV Computing Circuit, is presented.
- Each of its building blocks is described with the theory behind it and the different approaches to implement it.

Chapter Four: Conclusion and Future Work

- This chapter concludes my work and presents the opportunity to expand on this work in the future.

CHAPTER 2

BUILDING THE PROTOTYPE

In this chapter, an overview of the measures used to determine SRAM stability is presented along with the techniques – graphical, analytical and statistical – introduced in the literature to derive these measures. The first section introduces the stability measures, their significance and the graphical approaches used to derive them. In the second and third sections, the analytical and statistical techniques are discussed along with their advantages and disadvantages. The fourth section concludes the chapter by motivating the introduction of the DRV Computing Circuit.

2.1 LITERATURE REVIEW

There was extensive research regarding Car's sensor network and the on-board vehicle computer system. Nowadays, OBD-II and CANbus protocol are the norms for the access to the various vehicle subsystems. In a recent paper [27], they investigated the use of smartphones sensors for automotive analytics vs the sensors from the OBD sensory network.

They categorized the development in the products relating to driving analytics into two distinct classes based on how the data is sourced, first based on using the on-board diagnostic(OBD) devices, and second relying only on smartphones. The main questions they asked [27] was the ability of the smartphone to be an effective substitute for the OBD. They explored the ability of the smartphones to accurately measure the vehicle's speed, more specifically the instantaneous speed and whether it can come close to the OBD understanding of the vehicle's motion patterns such as turns, sudden stops, and a crowdsourced trend of car driving behavior (start/stop/decelerate/max speed, etc). Also, they claimed that smartphones are superior to the OBD in sensing the driver's personal behavior.

They found that smartphones can offer very reliable sensor data such as speed, user location, and others that came very similar to OBD data in terms of accuracy. The claim that with further sophistication and advancement in sensor fusion and machine learning technologies, a new range of applications would arise like enable detection of risky driving

behavior, such as hard braking, aggressive acceleration, road accidents and many others. Even though no single smartphone sensor can directly estimate speed like the OBD, they can together collect information from various sources to achieve a 96% similarity to OBD. This was achieved through understanding when a car stops and turns from accelerometers and compasses, utilizing crowd-sourced data from the roads, analyzing the driving pattern logs, etc. At the end of the paper, they didn't get to a conclusion to whether one system (OBD or smartphones) is best for driving analytics. However. They showed that smartphones, through many data that can be collected, can provide accurate readings.

From our side, we believe that to a wider range of applications and scenarios than what was explored in the paper, a hybrid solution that gets benefits from both smartphones and OBD can be orchestrated.

Another interesting finding we found during the research, that although driver behavior analytics range to diverse applications and possible directions, the current concentration in research and products tends to target road safety and accident avoidance. Driver distraction represents an increasingly prime contributor to crashes and fatalities. Many research papers investigate the Technology that can detect and assuage distraction by alerting the distracted drivers [28]. They showed that even for a week eye-steering relationship, their algorithm was able to produce metrics that can indicate distraction.

In this work [26], a system integrating the in-vehicle CAN/OBD network and an IoT network of wireless devices with an Intelligent Transport System deployed following the standards released by ETSI (European Telecommunications Standards Institute) within the Technical Committee on intelligent transportation system was presented. They made use of the Cooperative Awareness Message (CAM) to implement periodic transmission for vehicular communication. They have implemented the full stack from Physical and MAC layers (i.e. IEEE802. 11p standard) to the Networking & Transport Layer supporting Geonetworking and IP communications. They have also advised a model for the sensor IoT network for which they implemented an IEEE802. 15. 4 network with IPv6 interoperability. The target of the research was to find a standardized prototype bridge to enable VANETs

(Vehicular ad hoc Network) to communicate with the IoT network they advised in the Car's added sensor network.

They discussed the current lack of standardization in the automotive IoT and the efforts for considering IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) as a platform for the IoT of an automotive sensor network in non safety-critical applications.

They used a Beaglebone Black board commonly used as a DIY development board. They have demonstrated that starting with off the shelf, low cost hardware and software, a bridge between the IoT to the CAN and VANET network was orchestrated, and they believe that it will compete with proprietary applications that use silo-style solutions.

Our project will build on this effort but with a different approach. We will not use M2M messages and VANET networks, as we will build a system that connects directly to the cloud for any intelligent decision or communication.

2.2 METHOD

The target of this research is to develop a product which can compete in the "Connected Car" market. This product will have to be modular. It should easily integrates to already available solutions to offer extra functions and features.

2.2.1 Big Picture

We have designed and tested two cases, in which we will explain later in more details. However, the overall architecture remains the same and only few changes are applied. Figure 4 displays the high level design of the solution.

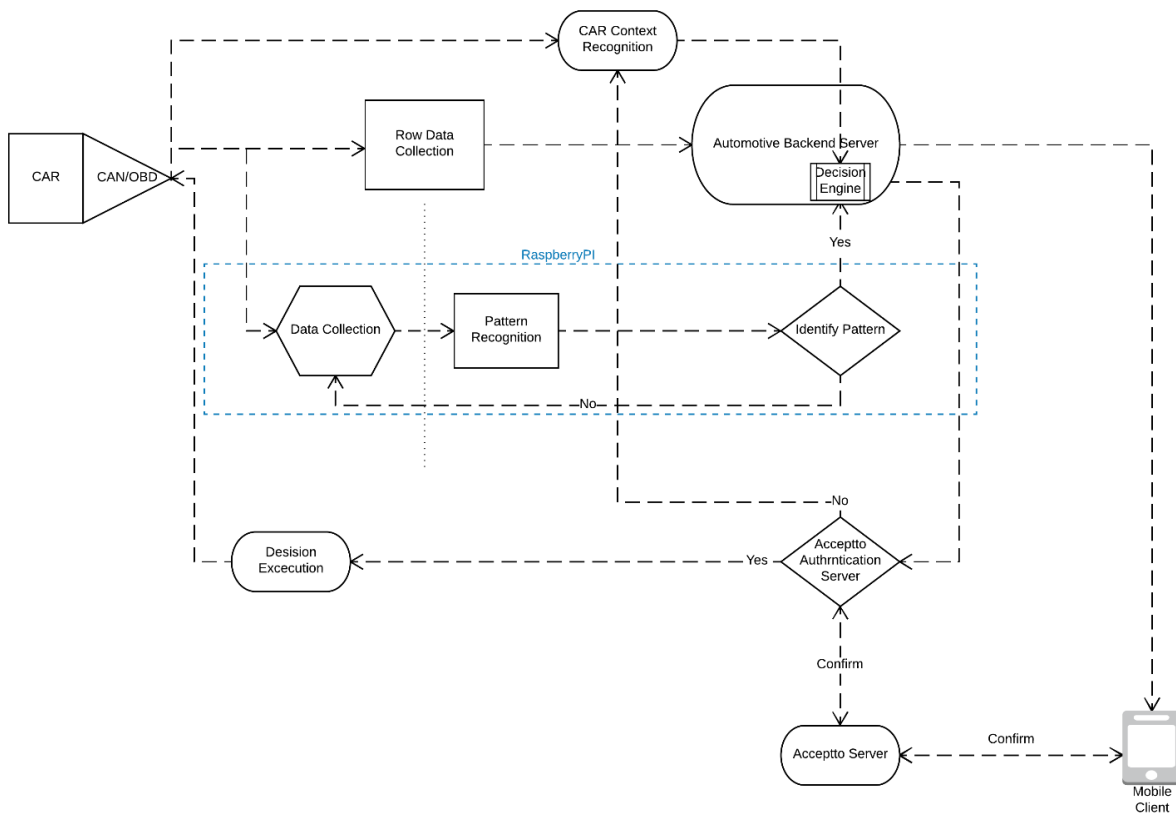


Figure 4: The high level Design

For our first prototype, we didn't have a 3rd party OBD Dongle. Instead, we used a CAN-Bus transceiver module. In this case, Arduino part of the Udoo board will connect to the transceiver module which will only be used as a transceiver to connect to the OBD-II Port. The actual communication with the OBD-II port and collection/storage of data will be performed through a process on the Udoo board/Raspberry PI. Initial Processing of the CAN data will collect only the desired info and will be sent to the server every preset duration (1 sec). The data will be buffered if there is a problem with the internet connection. The server will collect and display live data as simple analytics. Advanced analytics will be further performed on the data in the backend. Decision Engine will analyze the results and will communicate with Acceptto server or raspberry pi accordingly. Raspberry PI/Udoo Module will decide if the pattern recognized require a local decision without the need of a notification to the backend server. The Car Context will be determined as a collection of OBD data and other external sensors.

For our analytics collection, we used Vinli. Vinli has an LTE connection, at a sampling speed of 1 sec. If there is an internet connection problems, messages are buffered, and then retransmitted to the server. We are currently connecting to the Vinli server to retrieve the info via their rest API, and we have successfully created a demo that detects that the car's ignition, speed and stop. We are detecting some delay in receiving the info (around 10-20 secs) from Vinli server due to LTE connection issues.

The Raspberry PI/Udoo will act as a central processor. The OBD-II module and other external sensors will connect to the central module through different interfaces. Our CAN Bus module will connect directly through a cable. As for Vinli, we have retrieved the data through a REST API on Vinli server to our back-end server.

2.2.2 Components

We are going to design a prototype to bridge the connected car to the Cloud of IoT network. The product would engage simple CAN data handling before sending the useful data to the cloud for smart analysis. We have used a very well-known open source prototyping board called UDOO. This board combines the power of several Raspberry Pi boards and an Arduino board. It already has wifi connectivity, and we added Bluetooth and 3G connectivity via external modules. The below figure represents an overview of our system structure.

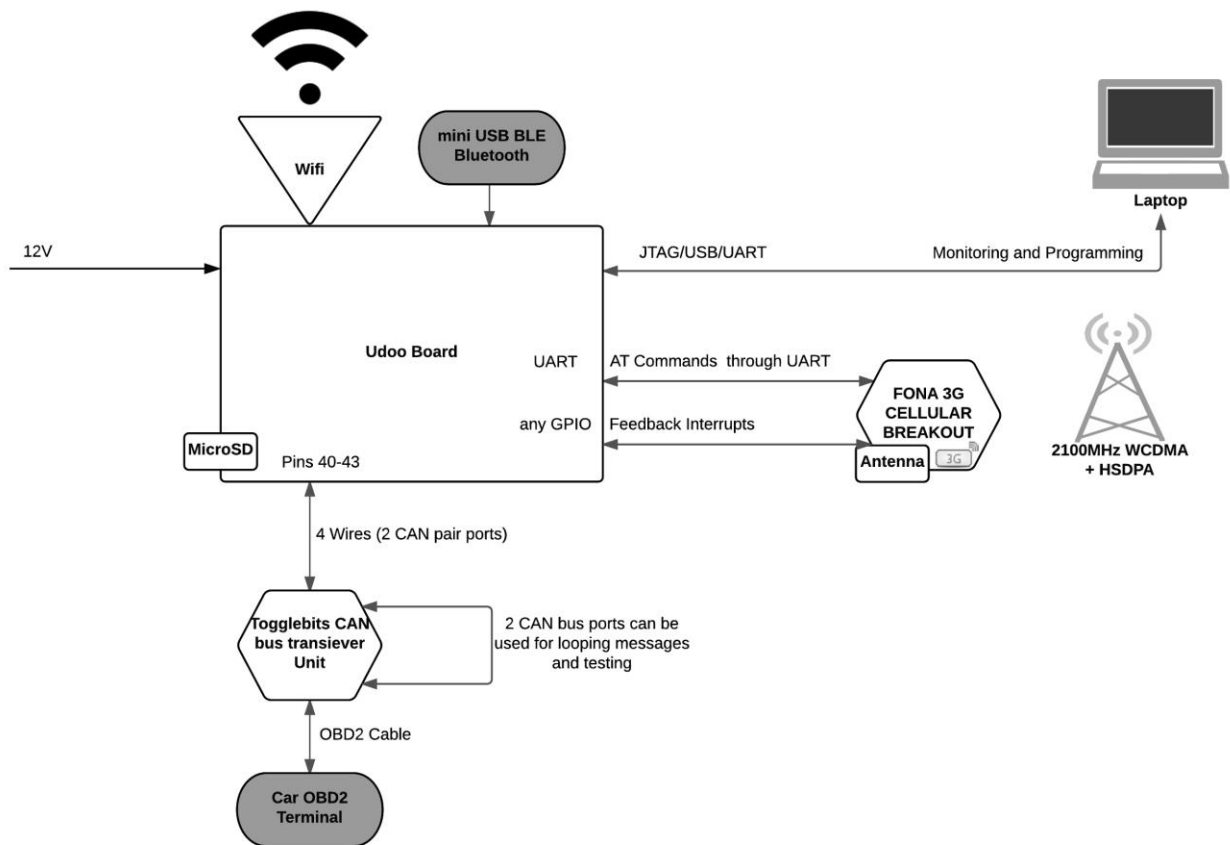


Figure 5: Overview of the System collecting OBD-II telematics

Our system comprises three main parts:

- Main Processing board: This board will be connected to the CAN bus through car's OBD port and be responsible for
 - Collection of car's sensor data
 - Sending control commands to the Car (through relays)
 - Performing minor driving analytics if needed
 - Connecting to the Backend-server
- Backend Server
 - Communication with Acceptto APIs
 - Performing intoxicated driving behavior analytics

Below is a List of components used in our main demo:

Part Number	Description	Usage
2691	FONA 3G CELLULAR BREAKOUT	3G Module provides 3G connectivity to the Udoo or Arduino Board
UDOO QUAD	UDOO QUAD CORE SBC OPENSOURCE	Main Processing board
UDOO-SK-EU	KIT STARTER EU	Cables and power supply kit for UDOO
1991	SLIM STICKERTYPE GSM/CELLULAR Q	Antenna for 3G module
BLED112V1	RF TXRX MOD BLUETOOTH TRACE ANT	Bluetooth BLE mini USB
CAB10087	OBDII TO DB9 CABLE 10087	Cable to connect to the CANbus
ToggleBits	Arduino DUE CANShield	Provides CAN bus transceiver shield for the Arduino DUE board

2.2.3 *Acceptto Integration*

For the Second factor/Cognitive authentication, we used Acceptto solution. Acceptto provides APIs to use its MFA solutions. The process is as follows:

- An Authentication request is issued from the Udoo Board to the Acceptto Server carrying
 - application ID
 - application secret
 - Message to me displayed: For example: “Car Just started, Is it you?”
 - user email
 - Authentication mode: e.g. login, car movement, boundary application, etc.
 - timeout: (Optional) How much time to wait while there is no response from the user before rejecting the authentication request
 - Authentication type (Optional): push notification, text or voice call. Push is default.

- Risk Profile (Optional): From low, medium and high. For example, high risk profile only accepts biometric authentication.
- The application (Udoo or back-end server) receives the request ID and keep polling for the status of the request.
- When the authentication type is push, the user either accepts or rejects the message received. Acceptto server then changes the status of the authentication request.
- The application (Udoo board or back-end server) polls for the status change for the duration of timeout-period. If status is changed, then according to the status, it performs actions accordingly.

For the integration, there exist a connection with Acceptto at both the Udoo/Raspberry PI board, and the back-end server. For the simple commands/action requests like car start/stop, speed limit, etc the Linux board directly send an authentication request to the Acceptto server. For any action related to advanced analytics like detecting an intoxicated driving behavior cue, the back-end server initiates the authentication request. Below is a sample code from the Udoo board describing the process.

```

AuthStatus=false;
subscription= Observable.interval(3000, TimeUnit.MILLISECONDS)
    .takeUntil(onAuthStatusReturned())
    .flatMap(new Func1<Long, Observable<AccepttoChannel>>() {
        Int counter=0;
        @Override
        public Observable<AccepttoChannel> call(Long aLong) {
            if (counter ==0) {
                counter++;
                mAPI.getAccepttoChannel().getAuthenticationChannel(Constants.HTTP.PARAMS).takeUntil(stopPredicate);
                return mAPI.getAccepttoChannel().getAuthenticationChannel(Constants.HTTP.PARAMS);
            }
            else {
                counter++;
                return mAPI.getAccepttoChannel().getAuthenticationStatus(Constants.HTTP.PARAMSAUTH);
            }
        }
    })
    .subscribeOn(Schedulers.io())
    .observeOn(AndroidSchedulers.mainThread())
    .subscribe(new Subscriber<AccepttoChannel>() {
        @Override
        public void onCompleted() {
            Log.d("api", "rx completed");
        }
    })

    @Override
    public void onError(Throwable e) {}

```

```

@Override
public void onNext(AccepttoChannel accepttoChannel) {
    if(accepttoChannel.getChannel()!=null){
        Constants.HTTP.PARAMSAUTH.put("channel",accepttoChannel.getChannel());
        accepttoChannel.setChannel(accepttoChannel.getChannel());
    }
    if(accepttoChannel.getChannelStatus()!=null) {
        accepttoChannel.setChannelStatus(accepttoChannel.getChannelStatus());
        TextView authStatus = (TextView) findViewById(R.id.authStatus);
        authStatus.setText(accepttoChannel.getChannelStatus());
    }
    if("approved".equals(accepttoChannel.getChannelStatus())){
        Log.d("Approved", "onNext: test");

        if (test>0){
            mAdkManager.writeSerial("1");
        }

    }
    else if ("rejected".equals(accepttoChannel.getChannelStatus())){
        mAdkManager.writeSerial("0");
    }
    Log.d("api", "response: " + accepttoChannel.getChannel() + "-" + accepttoChannel.getChannelStatus());
}});

```

2.3 DEMO

Our prototype was connected to the car's diagnostic OBD-II port. As seen in the below figure, when our system detects a signal from the car diagnostics requiring a certain security check (We have tested this with car's ignition, out of area, and over a speed limit check), it communicates directly with the backend server, which in turn send a request to the acceptto server requiring to authenticate the user. Acceptto server will then send an authorization request holding information to the detailed security action requested by the car to the user's paired smartphone as a mobile notification. If the user gives the authorization, acceptto servers will confirm the identity to the backend server and our embedded in-car system will consider the access granted. Otherwise, if the user ignores the notification or sends a reject message, our system will be notified and some precautionary actions will be taken as in sounding the alarm, or turning off the car if it is not yet in the drive mode.

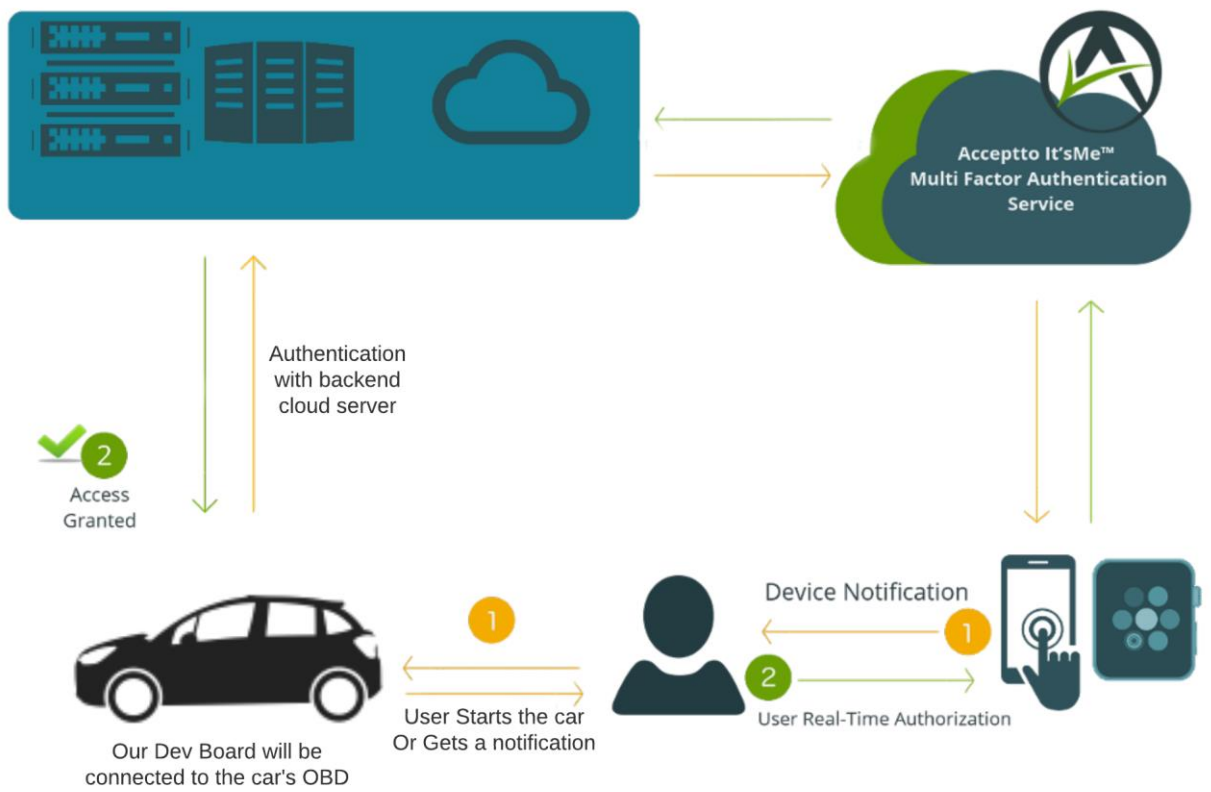


Figure 6: Demo application for Acceptto Multifactor Authentication with Car's Ignition

CHAPTER 3

DRUNK DRIVING DETECTION

Our main focus will be the detection of drunk driving cues using any off the shelf OBD-II module. OBD-II is the second generation standard of the On-Board Diagnostics protocol (OBD). OBD-II was made mandatory for all cars manufactured in the United States since 1996 [17]. For our proof of concept, the device which we used in this work is called Vinli [16]. It connects to the diagnostics port of the car and provides access to main OBD-II PIDs (Parameter IDs) through an LTE connection to the backend server. The company provides APIs and simple SDKs to help developers build applications on top of their platform. The sampling rate is 1 Hz which should be adequate for our study, as in a fully integrated cloud solution, it will be hard to collect information at a higher frequency. The obtained data accuracy will serve our purpose as verified by a recent publication [2], which has thoroughly investigated the accuracy of the OBD-II interface parameters versus the raw Controller Area Network (CAN) bus values that can be extracted through a direct connection to the ECU (Engine Control Unit). A CAN bus is a standard for communications between different car's components (ECUs), and it is one of the protocols used by the OBD-II interface. The authors of [2] collected the data at a 5 Hz frequency and sent it to the cloud. The results showed that even though the OBD-II information is not as accurate as the CAN data, it could still be used for data analytics, as it provides a reasonable precision, and is cost-effective.

The variety of the off-the-shelf products currently available on the market opens a great opportunity for collection of very valuable vehicular data and for performing advanced analytics at the backend server. These products can be classified into two categories: Bluetooth OBD-II Dongles and Proprietary 3G and LTE enabled OBD-II Dongles.

Simple Bluetooth dongles can be obtained starting as low as 10\$ a piece. There are several mobile apps created to interface with these dongles and provide limited services like zone notifications, tracking, and simple car health monitors like fuel level and trouble codes.

Proprietary dongles, on the other hand, provide cellular connectivity to extend the use of the connected car. These solutions include Vinli, Mojio, Verizon Hum, Automatic, Zubie, and others. While some dongles like Vinli provide APIs to allow 3rd party applications, some solutions like Hum and Automatic have their own apps and features.

In this paper, we explore the possibility of detecting drunk driving patterns from data collected through a third party OBD-II module, namely Vinli.

The rest of the paper is organized as follows. Section II, next, surveys the related literature and works that concern drunk driving detection and retrieving OBD data then analyzing it to draw a conclusion that concern driving. Next, in Section III, we describe the detection methodology that we employ, followed by explaining the data collection and preprocessing process. Section IV explains the machine learning part of our work, used to classify drunk driving behaviors, and then presents the obtained results. We use Section V to discuss how the detection probability can be improved, and how our work can be extended. Finally, Section VI concludes the paper.

3.1 LITERATURE REVIEW

The most recent works that try to detect drunk driving behavior used external sensors. As an example, in [3] and [4], a system was developed for drowsiness detection using a breath sensor. Other works and projects used eye movement detection through a camera, like in [5]. In a recent paper [15], a deep convolutional neural network was used to analyze and detect facial expressions and gestures that would imply drowsiness or distractions.

In the most part, OBD-II sensors were used to detect fuel efficiency. In [6], for example, the authors used a multi-sensor fusion method using Bayesian Networks for the estimation of the car's performance. The experiments used OBD-II data and a combination of different motor types for a precise performance evaluation. On the other hand, in [7], OBD-II sensors were used for fuel economy estimation using statistical data regressions. A more relevant study to our research employed methods that use accelerometers and OBD-II data to detect driving behavior. Only Phone sensors were used in [8] to detect drunk driving behavior, where the authors identified the main driving cues that could be detected through the phone's gyroscope and accelerometer. In particular, they used longitudinal and lateral

acceleration pattern matching for the classification process, based on the detection of minimum and maximum variations. Also, using the accelerometer and gyroscope of the smartphone, the work in [9] introduced an algorithm for detecting dangerous driving behaviors. Although it is an interesting approach, the approach faces the problem of the willingness of the driver to install the desired app on his phone, and therefore may not be practical.

A paper that appeared recently [10] uses OBD-II data, combined with phone sensors, to make driver classification based on driving behaviors. They used three cars and 14 drivers to collect the dataset. The reported results showed an accuracy of about 85% for classification based on OBD-II data, 75% based on the phone sensors, and 86% when using combined data. However, the accuracy fell down drastically with the classification of more than 3 drivers.

From the above research, we can observe that the drunk driving behavior was being predicted using external sensors such as phone sensors [8] or cameras/breath analyzers [3] [4] [5] [15]. And while OBD-II information was mainly used for performance reports [6] [7], a recent paper [10] used both in-vehicle and phone sensors for the analytics, for the purpose of classification of drivers based on the driving behavior. Thus, we will explore the detection of drunk driving behavior using an OBD-II dongle providing both OBD data and GPS and accelerometer telematics.

3.2 METHOD

The Methodology is explained in Fig.1. The first step is data collection. We connect the Vinli dongle to the car's OBD-II port and the device will send the data to the server for storage. After the data is collected and annotated, it needs to be cleaned to remove any outliers. The next step is feature extraction. We extracted some manual features from the sensors (min, max, mean, etc.) as well as 100s of automatically collected features (Entropy, # of peaks, etc.). This generated around 8000 features which were filtered to around 1200 features. Lastly, for classification, we used a linear logistic regression classifier. We split the

data randomly at 80/20 to training and testing sets. Lastly, we validated our results using 10-fold cross-validation.

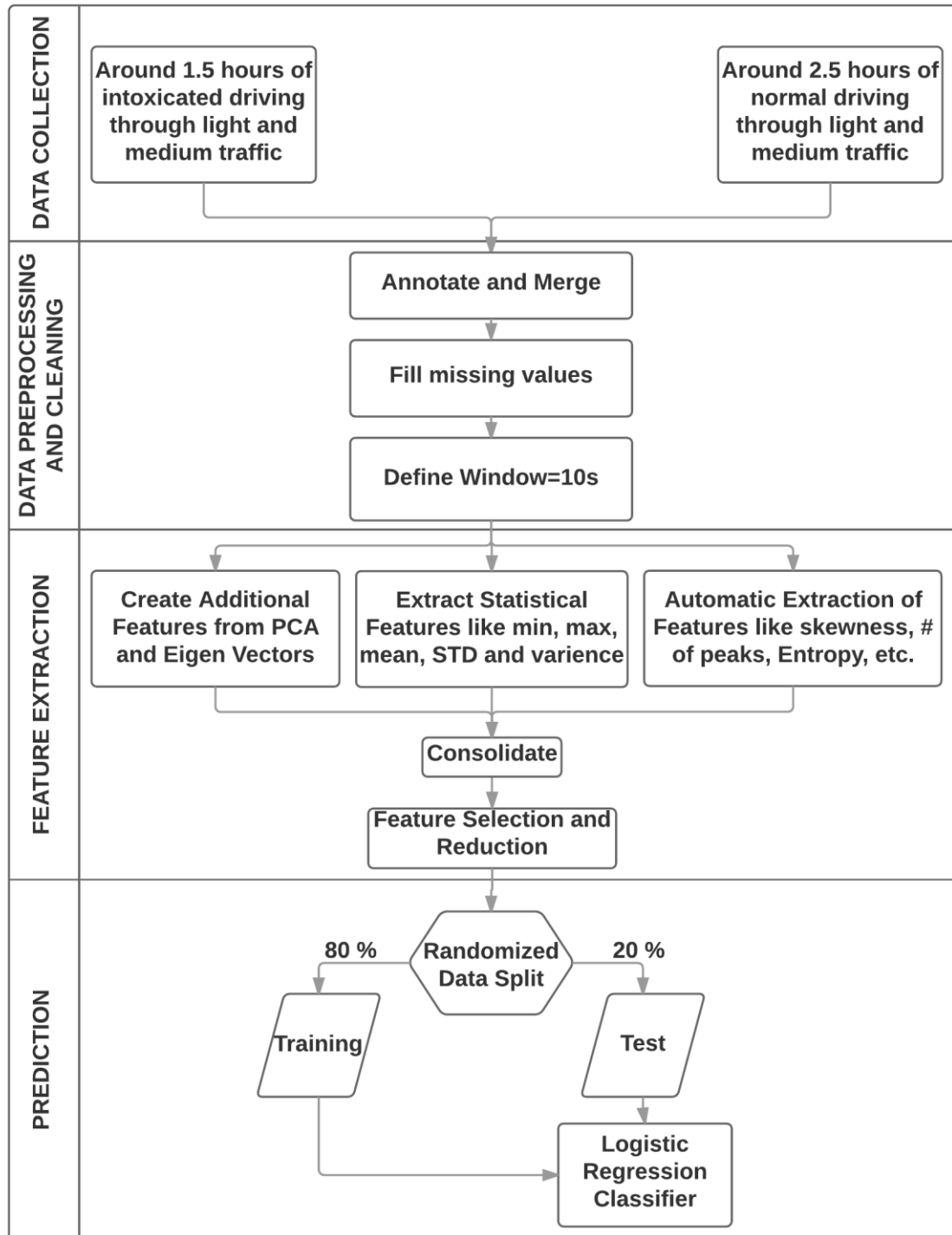


Figure 7. Drunk driving detection methodology

3.2.1 Data Collection

To emulate drunk driving, we acquired Drunk Busters goggles [13]. The Goggles can simulate the effects of intoxication, including reduced alertness, visual distortion, delayed reaction time, depth and distance perception problems, and reduction of peripheral vision associated with BAC levels (Blood Alcohol Concentration) between 0.04 and 0.35 [13]. What motivated this approach to emulate drunk driving behavior is the fact that in [12] the authors used vision impairment goggles in addition to a car simulator to simulate the intoxicated driving patterns.

The acquired goggles simulate impairment of 0.15 to 0.25 BAC levels and was planned to be used while driving the Carsim simulator [14], which was preferable (more convenient) over driving a real car, given such levels of intoxication and the danger in driving in that situation. The CarSim simulator is an internationally recognized mechanical and software tool for analyzing vehicle dynamics, developing active controllers, and calculating a car's performance characteristics [14]. However, due to setup issues with the simulator, we used the goggles while driving a real car on empty side road sections, after midnight, to avoid collisions with passing by cars. Moreover, the car was driving by the driver wearing the goggles with an assistant to minimize the likelihood of causing hazards while driving.

Two data sets were collected, where the first set contained around one hour of emulated drunk driving behavior, while the second set contained more than three hours of normal driving behavior by four drivers in various driving conditions of open roads and medium traffic. The data was carefully collected and annotated, where data obtained during startup and turn off of the car was removed from the data for better classification and avoid bias. We annotated each sample of our data as either Drunk or normal driving. The collected data included the below parameters provided by the OBD-II device at different frequencies. For example, torque, RPM, speed, and others were collected every 1 second, whereas the relative throttle position and spark timing advance were collected every 1 minute.

As depicted in Table 2, some parameters could only be accessed at a rate that is less than 1 Hz, thus making them inappropriate for our detection algorithm which needs data that is available at 1 Hz or higher frequency. Almost all of the current OBD-II dongle wireless

solutions, like Vinli and Zubie, collect OBD-II data at a frequency of 1 Hz or less. The collection of certain sensors at a lower frequency stems from the fact that these sensors' values, like spark time advance or intake manifold temperature, change at relatively slower rates as compared to sensors like speed or rpm. This is mainly done to relieve the load on the car's CAN bus, and also on the 3G or LTE network.

Parameter	Frequency
Torque	1 Hz
GPS	1 Hz
Accelerometer X,Y,Z Max and Min	1 Hz
Intake Manifold Pressure	1 Hz
Engine RPM	1 Hz
Vehicle Speed	1 Hz
Acc. pedal position D and E	0.016 Hz
Relative Throttle Position	0.016 Hz
Timing Advance	0.016 Hz

Table 2. Sensors used to classify drivers

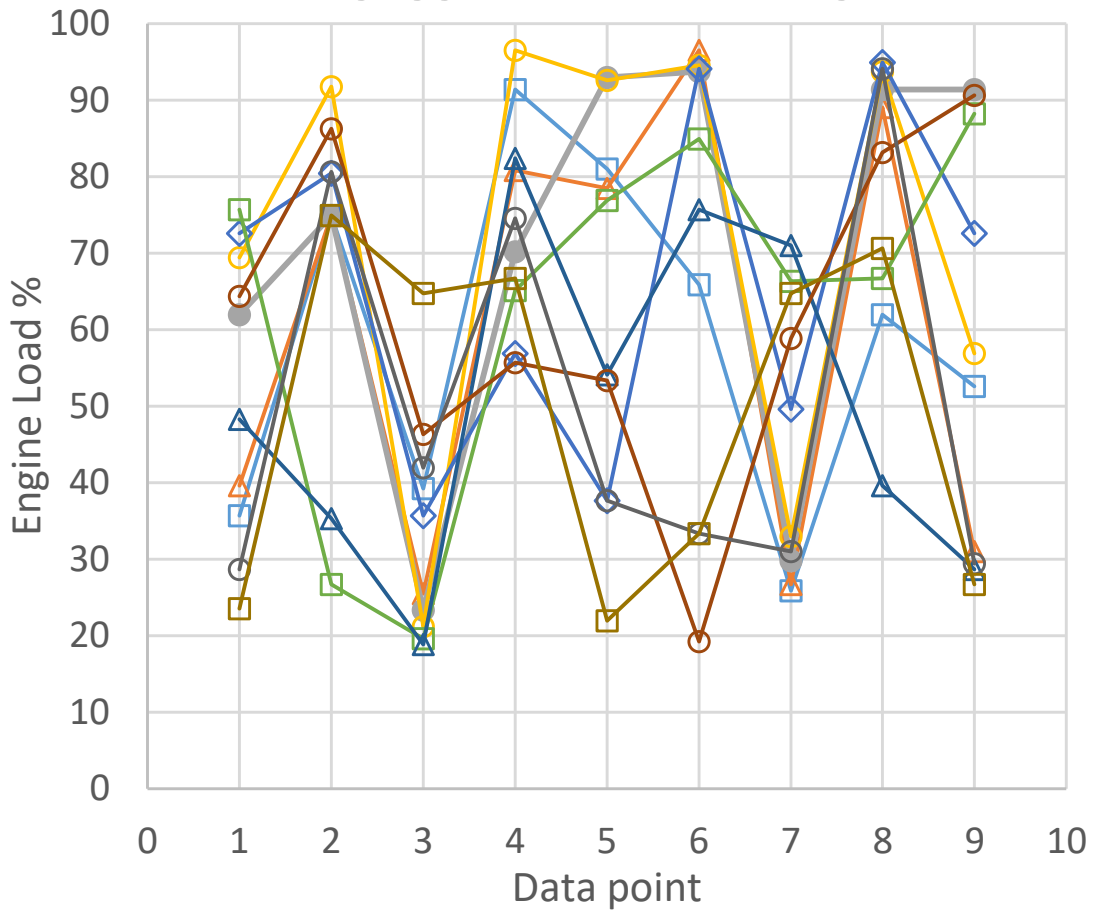
We plotted snapshots of the dataset at random windows to study the visual discrepancies. Figure 8 illustrates an example of driving drunk and normal, inferred from the sensor Engine Load. It is clear that in general, drunk driving has more variations and peaks in the data. This is true for several sensors like RPM, Engine Load, and accelerometer values. This gives an indication that a relatively good classification accuracy can be attained.

3.2.2 Drunk Driving Cues

There are some drunk driving detection cues that are defined by the US NHTSA, as explained in [11]. The cues were identified as strong indicators of possible drunk driving scenarios. In this work, we have focused on cues that relate to speed measurements and lane position variations. According to NHTSA, the above parameters are two of the main driving cues that can be visually detected [11], and were confirmed during our drunk driving emulation scenarios. Figure 9 shows some of the problems that are related to maintaining proper lane position caused by drunk driving. These are, as indicated on the four illustrations: 1) drifting, 2) weaving, 3) swerving, and 4) turning with a wide radius. ON the other hand, indications that may be derived from speed and braking measurement data are 1) jerky or abrupt stops, and widely carrying speeds.

Some drives consisted of one repeating driving cue, while other drives had a combination of several cues that can be detected in a specific timeframe. Although during the drunk driving emulation through using the goggles, not all the drunk driving cues could be visually detected, many of the cues defined by NHTSA were observed. These observed cues included weaving, severing, turning with a wide angle, abrupt corrections of the trajectory, and varying speeds and breaking patterns.

With goggles (drunk driving)



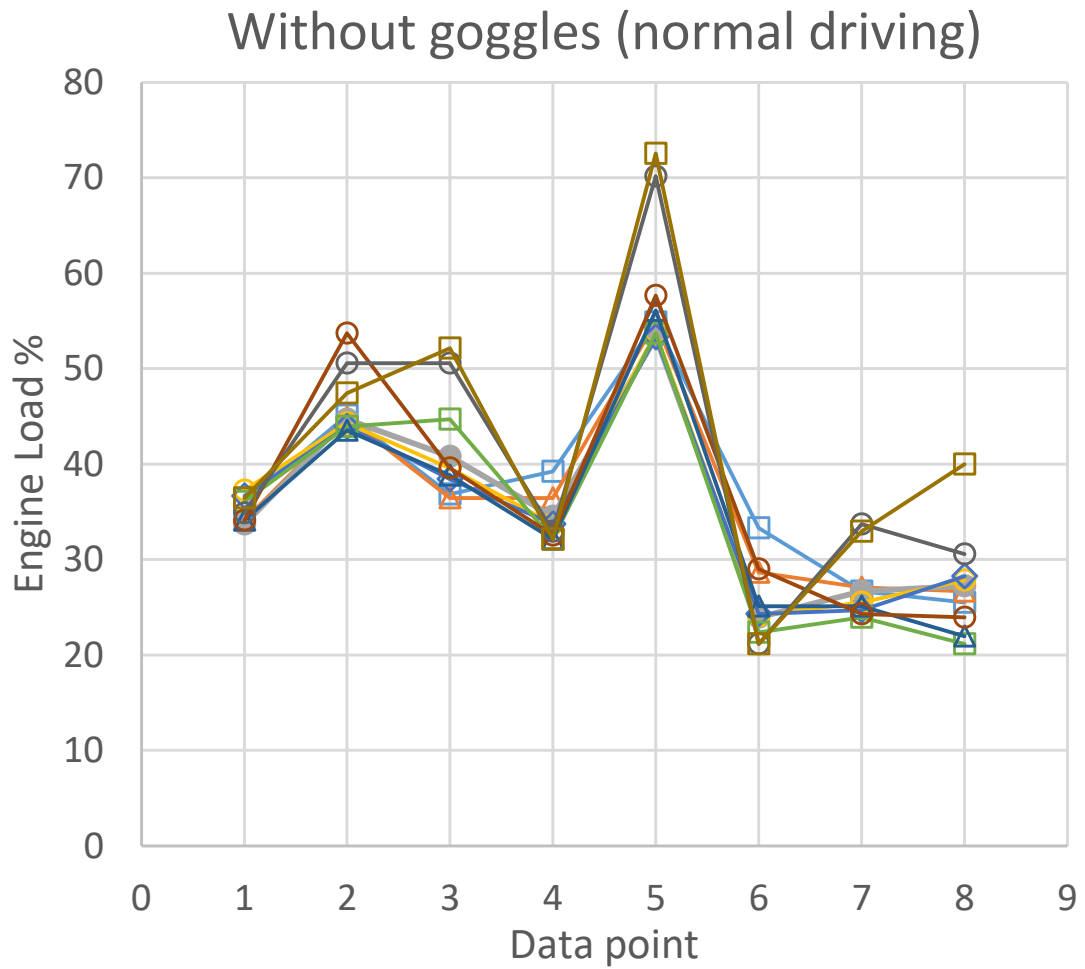


Figure 8. Sample data output for the engine load parameter during drunk driving (top), and normal driving (bottom).

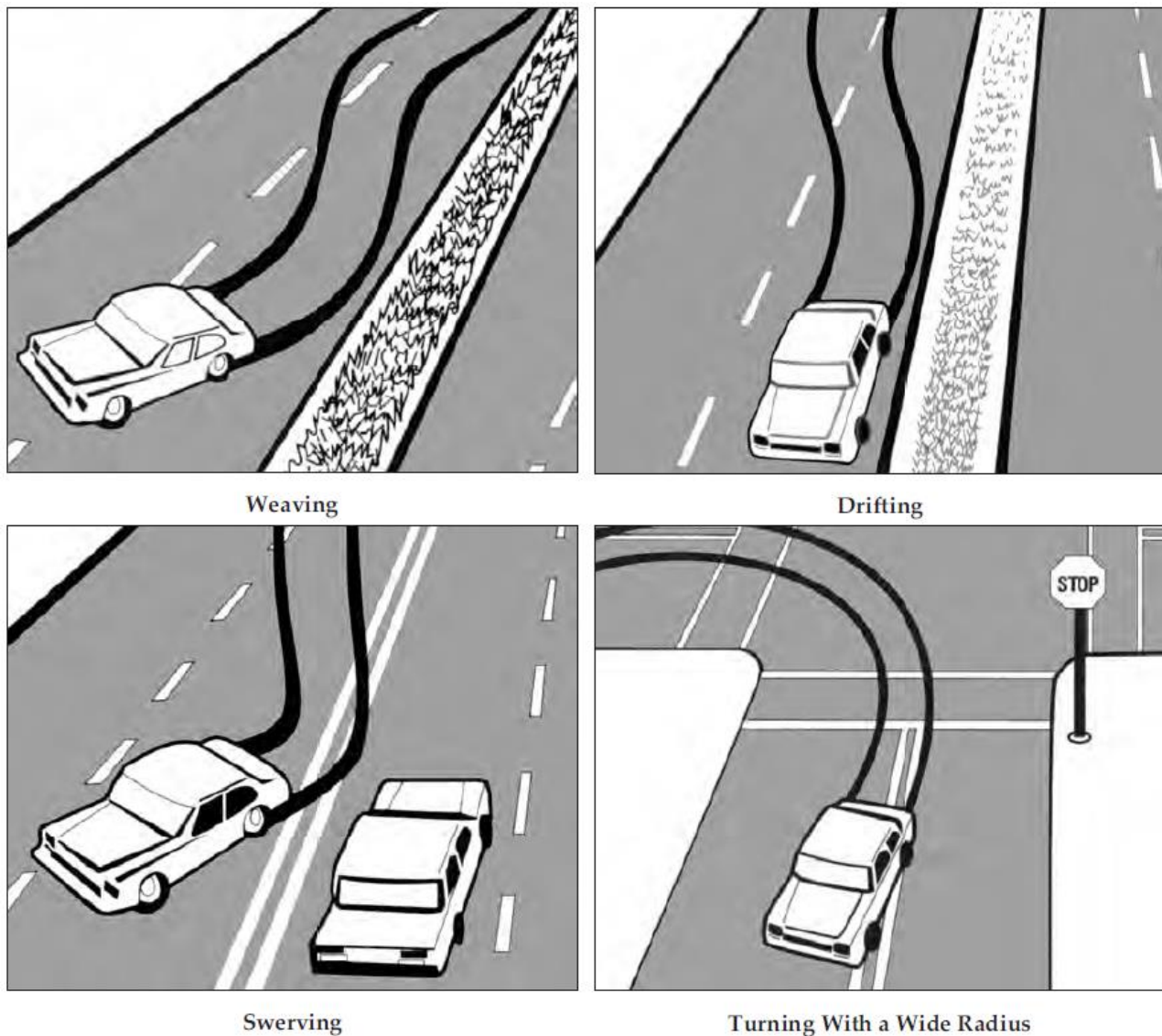


Figure 9. Common drunk driving trajectory scenarios [11]

3.2.3 Data Cleaning and Feature Extraction

We used python for preprocessing of the collected data from the driven cars. The dataset contained intermittently missing data, which were mostly GPS locations or OBD-II parameters. To counter-effect this, the data sets were smoothed using a rolling mean window on the 1 Hz parameters, whereas the rest of the features were dropped due to their lower data acquisition rates. Moreover, we supplemented the acquired data with computed data, like speed differentials (current speed – prev. speed) and distance covered during each measurement interval using GPS longitude and latitude data.

Afterwards statistical features were extracted from the time series using a 10-second window. These included 1) minimum, maximum, mean, median and variance per sensor; 2) PCA (Principal Component Analysis) and Eigenvectors; and 3) hundreds of other features, such as absolute energy, sum of changes, entropy, Fast Fourier Transform, number of peaks, skewness, and many others. In total, we extracted around 10,244 features, which were reduced to 1204 based on the information gain and the impact on their effect on accuracy. As we have mentioned earlier, we used the TFRESH python library to achieve this. During the feature reduction process, each feature vector was individually evaluated based on its effect on the target results [18] (drunk or normal). With this evaluation, ranking the significance of each feature was then used to filter the features using the Benjamini-Yekutieli procedure, which is explained in [19].

3.3 RESULTS

The dataset was randomly split into two sets: a training set and a testing set that comprised 20% of the total data. We trained a linear Logistic Regression model to predict drunk driving patterns through feeding the classifier (logistic regression) with vectors of extracted features that are actually windows of 10 seconds of data points. That is, every 10 points of data form a window to be classified as drunk or normal driving. We used L1 regularization since the data we have is sparse and produced a higher prediction score. The L1 regularization technique is used in the Lasso Regression model (Least Absolute Shrinkage and Selection Operator), which adds the “absolute value of magnitude” of coefficient as a penalty term to the loss function. The Lasso model shrinks the less important feature’s coefficient to zero thus, removing some features altogether.

We trained the model for different C and stopping criteria (tolerance). We tested for C values between 1.0 and 150, in increments of 25. The parameter C refers to the inverse of regularization strength, meaning that the lower the C parameter’s value, the stronger the regularization is. With more regularization, the algorithm tries to generalize the model, whereas with less regularization, the model may become overfitted. The model stops training when the score is no improving by at least the tolerance for 2 consecutive iterations. The

model used a tolerance of 0.01, C=1.0, and the Liblinear solver (a python library for large linear classification). The Liblinear solver was better suited for our classification model as it has a binary target and a relatively small dataset. The other solvers, mainly ‘sag’ and ‘saga’, didn't perform well for our prediction. This combination generated the best F1-value at around 83%. Our goal was to improve the F1-value as it is a measure of both precision and recall. It is the harmonic mean of precision and recall which are functions of TP (True Positives), FP (False Positives), and FN (False Negatives):

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1} = \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The results showed an average F1 value of 81%, which is very reasonable concerning the size of the dataset and the missing values associated with a commercial product.

Normalized Confusion Matrix		
	Normal (Predicted)	Drunk (Predicted)
Normal (Actual)	0.86	0.14
Drunk (Actual)	0.25	0.75

Table 3. Normalized Confusion matrix without GPS data

The normalized confusion matrix in Table 3 shows that we achieved very low values of False Positives (FPs) (14%), and relatively low values of False Negatives (FNs). FPs occur when our model predicts a drunk driving behavior while the behavior is actually normal, while FNs occur when we fail to predict a behavior of intoxicated driving. Our system thus achieves relatively good performance for minimizing FPs and FNs.

For the window size, we tried to see the results with different window sizes of 5, 10 and 15 seconds. We got the best results when the size was set to 10 seconds. The smallest window size gave a relatively low F1 value of 47%, which is reasonable as a 5 seconds

window does not capture the drunk driving cues within the timeframe. The F1 values were 47%, 83% and 79% for the given 5 sec, 10 sec and 15 sec widows, respectively.

Table 4 gives the results using a 10 seconds window without GPS data. We validated this data using 10-fold cross-validation, where the validation score ranged between 74 and 87%, thus verifying the model. The scores were 0.828, 0.8268, 0.865, 0.766, 0.796, 0.873, 0.747, 0.864, 0.796, and 0.844 respectively for the 10 iterations, thus producing a mean of 82%.

The results in tables 2 and 3 are without the use of the GPS data as part of the original dataset. When the GPS coordinates are included in the dataset before the automated feature extraction, we get results of around 73% accuracy. This is possibly due to the nature of the GPS coordinates and its weak correlation to the driving style.

	Precision	Recall	F1-Score
Normal	77%	86%	81%
Drunk	85%	75%	80%
Average	81%	81%	81%

Table 4. Precision, Recall and F1 value

CHAPTER 4

CONCLUSION AND FUTURE WORK

This work can be built upon further in several directions. First, since our solution utilizes third-party OBD-II dongles for CAN bus data collection, we can develop a supporting system for collecting, storing, and analyzing available car's sensors data. Hence, the car system can react in real time when inferring a drunk driving pattern (i.e., as produced by the classifier). The system's response can be in the form of 1) producing an alarm to alert the driver if the probability of drunk driving is low (e.g., when the percentage of data segments producing drunk driving classification is low); 2) transmitting a message over the network to the driver's designated contact persons, when the probability is rated medium; or 3) in the extreme case, when the probability is high, to control the car to drive increasingly slower until it comes to a complete stop by cutting the fuel pump supply. These are only possible outcomes that can also benefit from additional drunk driving statistics that were not available at the time to be included in the training datasets.

The probability of drunk driving can be ascertained using information that is readily available to the car. Such information includes the current time, as most drunk driving incidents occur near or after midnight and on weekends [1]. Furthermore, GPS data can be used to detect driving near locations containing pubs and nightclubs. These are example data that can be used to increase the probability of correct drunk driving classification. Another direction that could be pursued concerns the datasets of measurements. Datasets of certain car sensors could be hard to obtain using a standard OBD interface (e.g., steering wheel angle), moreover, they take time and effort to annotate properly. Increasing the number of types (i.e., sensors) of datasets and their sizes can improve the reliability of the model, and can help in distinguish the various patterns of drunk driving (see Figure 8). This will in turn require annotation of the dataset with the types of drunk driving behavior cues, like weaving and swerving, which in turn can be used to decide on the most appropriate action to take to prevent an accident. We also should note that if the car can identify its position accurately on the road, it can force the driver to coast toward the side of the road.

Finally, we note the other driving behaviors that are equally dangerous, like sleepy driving. Indeed, such driving scenarios share with drunk driving behaviors several characteristics, like some of those depicted in Figure 9. Hence, our work can also be used, although with some alterations, to detecting sleepy driving conditions and then taking suitable actions.

We investigated the ability to detect drunk driving behavior from the car's CAN bus that was collected using a third-party standard OBD-II module. Our work was based on driving an actual car using Drunk Busters goggles to actually emulate drunk driving. Using machine learning, we were able to classify drunk driving versus normal driving with an accuracy that is upward of 80%. This opens the door for implementing on-car control mechanisms to prevent related accidents from occurring.

REFERENCES

- [1] TRAFFIC SAFETY FACTS 2015 by National Highway Traffic Safety Administration
- [2] Z. Szalay et al., "ICT in road vehicles — Reliable vehicle sensor information from OBD-II versus CAN," Int'l. Conf. Models and Technologies for Intelligent Transportation Systems (MT-ITS), Budapest, 2015, pp. 469-476.
- [3] M. Sakairi, "Water-Cluster-Detecting Breath Sensor and Applications in Cars for Detecting Drunk or Drowsy Driving," in IEEE Sensors Journal, vol. 12, no. 5, pp. 1078-1083, May 2012.
- [4] H. A. Attia, M. Takturi and H. Y. Ali, "Electronic monitoring and protection system for drunk driver based on breath sample testing," 5th Int'l. Conf. Electronic Devices, Systems and Applications (ICEDSA), Ras Al Khaimah, 2016, pp. 1-4.
- [5] Y. Katyal, S. Alur and S. Dwivedi, "Safe driving by detecting lane discipline and driver drowsiness," Int'l. Conf. Advanced Communications, Control and Computing Technologies, Ramanathapuram, 2014, pp. 1008-1012.
- [6] Multi-sensor Fusion Method Using Bayesian Network for Precise Vehicle Performance Evaluation
- [7] Lee M. G., Park Y. K., Jung K. K., and Yoo J. J., "Estimation of fuel consumption using in-vehicle parameters," Int'l. Journal of u- and e-Service, Science and Technology, vol. 4, no. 4, pp. 37-46, 2011.
- [8] J. Dai, J. Teng, X. Bai, Z. Shen and D. Xuan, "Mobile phone based drunk driving detection," 4th Int'l. Conf. Pervasive Computing Technologies for Healthcare, Munich, 2010, pp. 1-8.
- [9] F. Li, H. Zhang, H. Che, X. Qiu, "Dangerous driving behavior detection using smartphone sensors," 19th Int'l. Conf. Intelligent Transportation Systems (ITSC), Rio de Janeiro, 2016, pp. 1902-1907.
- [10] C. Zhang, M. Patel, S. Buthpitiya, K. Lyons, B. Harrison, G. Abowd, "Driver classification based on driving behaviors," in Proc. 21st Int. Conf. Intell. User Interfaces, Sonoma, CA, Mar. 2016, pp. 80-84.
- [11] U.S. NHTSA, "Traffic Safety", <http://www-nrd.nhtsa.dot.gov/PPubs/811172.pdf>
- [12] M. Shirazi and A. Rad, "Modeling the steering behavior of intoxicated drivers," 2012 15th Int'l. Conf. Intelligent Transportation Systems, Anchorage, AK, 2012, pp. 648-653.
- [13] Drunk Busters of America, LLC, [available online], viewed on 10/4/2017, <http://drunkbusters.com/>
- [14] CarSim Simulator, [available online], viewed on 10/4/2017, <https://www.carsim.com/products/carsim/>
- [15] K. Yuen, S. Martin, M. Trivedi, "Looking at faces in a vehicle: A deep CNN based approach and evaluation," 19th Int'l. Conf. Intelligent Transportation Systems (ITSC), Rio de Janeiro, 2016, pp. 649-654.
- [16] Vinli. (2017). Vinli - The easiest, most advanced connected-car system in the world.. [online] Available at: <https://www.vin.li/>
- [17] En.wikipedia.org. (2017). On-board diagnostics. [online] Available at: https://en.wikipedia.org/wiki/On-board_diagnostics.

- [18] Tsfresh.readthedocs.io. (2017). Feature filtering [online] Available at: http://tsfresh.readthedocs.io/en/latest/text/feature_filtering.html.
- [19] Benjamini, Y. and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, 1165–118.
- [20] Gartner report "Predicts 2015: The Internet of Things." a world's leading information technology research and advisory company. <http://www.gartner.com/document/2952822>.
- [21] <https://www.udoo.org/udoo-dual-and-quad/>
- [22] Chase, J. (2013). The Evolution of the Internet of Things. Texas Instrument, 1(February), 7.
- [23] Kirk, R. (2015). Cars of the future: The Internet of Things in the automotive industry. *Network Security*, 2015(9), 16–18.
- [24] <https://www.openautoalliance.net/>
- [25] It'sMe™ Multi-Factor Authentication. Retrieved from <https://www.acceptto.com/>
- [26] Carignani, M., Ferrini, S., Petracca, M., Falcitelli, M., & Pagano, P. (2016). A prototype bridge between automotive and the IoT. *IEEE World Forum on Internet of Things, WF-IoT 2015 - Proceedings*, (April 2014), 12–17.
- [27] He, W., Yan, G., & Xu, L. Da. (2014). Developing vehicular data cloud services in the IoT environment. *IEEE Transactions on Industrial Informatics*, 10(2), 1587–1595.
- [28] L. Yekhshatyan and J. D. Lee, "Changes in the Correlation Between Eye and Steering Movements Indicate Driver Distraction, " in *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 136-145, March 2013.