

Smartphone Application to Detect Texting While Driving

Design Document

Group:

sdmay19-16
“The Face Mates”

Client:

Daji Qiao

Faculty Advisor:

Daji Qiao

Team Members:

Ryan Baker - Lead Architect
Derek Clayton - Report Manager
Lucas Golinghorst - Test Engineer
Andrew Knaack - Lead Designer
Sara Mace - Meeting Scribe
Kristina Robinson - Project Lead

Team Website:

sdmay19-16.sd.ece.iastate.edu

Version:

2.0

Version Date:

10/12/2018

Table of Contents

List of Figures	2
List of Tables	2
List of Definitions	2
1. INTRODUCTION	2
1.1 Acknowledgement	3
1.2 Problem and Project Statement	3
1.3 Operational Environment	4
1.4 Intended Users and Uses	4
1.5 Assumptions and Limitations	4
1.6 Expected End Product and Deliverables	5
2. Specifications and Analysis	6
2.1 Function and Non-Functional Requirements	6
2.2 Proposed Design	7
2.2 Design Analysis	10
3. Testing and Implementation	14
3.1 Interface Specifications	14
3.2 Hardware and software	14
3.3 Functional Testing	14
3.4 Non-Functional Testing	17
3.5 Modeling and Simulation	18
3.6 Process	20
3.7 Process Details	22
3.8 Implementation Issues And Results	22
3.9 Results	23
4. Closing Material	24
4.1 Conclusion	24
4.2 References	24

LIST OF FIGURES

- Figure 1: System Overview Diagram (page 9)
- Figure 2: Inputs and Outputs Diagram (page 10)
- Figure 3: Module Interaction Flowchart (page 21)

LIST OF TABLES

- Table 1: Model Data for Distracted Driver (page 18)
- Table 2: Model Data for Passenger (page 19)

LIST OF DEFINITIONS

- False positive: In this case, a false positive is the application incorrectly identifying a passenger as a driver and blocking their texting capabilities.
- False negative: In this case, a false positive is the application incorrectly identifying a driver as a passenger and not blocking their texting capabilities.
- TwD: An acronym which stands for Texting while Driving

1. INTRODUCTION

1.1 ACKNOWLEDGEMENT

The group wants to acknowledge our client and advisor, Daji Qiao, for thinking of this project. We also want to acknowledge the department of Electrical and Computer Engineering at Iowa State University for encouraging us to work on professional projects and providing us with extra phones for development purposes.

1.2 PROBLEM AND PROJECT STATEMENT

This project will address the issue of texting while driving. According to teensafe.com, just 3 years ago, 391,000 injuries were caused by distracted driving accidents. Even more serious than that, 9 people are killed everyday from distracted driving accidents. Answering texts while driving might seem harmless but replying to a text message takes the driver's eyes off of the road for at least a few seconds. It takes only three seconds of the driver having their eyes off of the road to be in a car crash. While texting and driving is not the only cause of distracted driving, approximately 660,000 drivers use a cell phone during the daytime which means that the chance for a TwD accident is significantly increased every day. As so many people are killed from distracted driving, and many who are driving distracted are texting, texting while driving is a major concern. In order to find a solution to this problem, an android application will be built to detect if someone is texting while driving. Accurately detecting whether someone is in the driver's seat and that they are texting while driving is the biggest challenge. In order to do that without simply locking out everyone from their phones, the solution will have to include many different measures to ensure accurate detection.

The solution to this problem will be to develop an android application to detect whether someone is TwD. Once TwD has been detected, the user will not be able to send text messages until they stop driving. The solution will ensure that TwD can be accurately detected by checking multiple different conditions. The first condition is if the driving is moving faster than 10 miles per hour. This will tell if they are in a moving vehicle. The second condition is where the user is sitting in the car. Cameras will be used to determine what is in front of the user as well as seeing which way the seatbelt is going across the user's chest. The last condition of verification is learning how the user normally uses the phone so that significant differences can be detected when they are TwD. If all conditions are satisfied, an unbreakable phone lock will occur and prevent the user from continuing to text and drive simultaneously. This solution has the potential save lives by helping people who would be driving while distracted and disabling them from doing so.

1.3 OPERATIONAL ENVIRONMENT

The operating environment will be the user's vehicle. This environment is not expected to be subject to any extreme conditions, but rather a controlled environment. If an extreme condition does occur (such as a car accident), then the app is no longer of use in that environment anyway. It is also worth noting that the application is intended to be active even when the user is not in a vehicle, since it is vital for the program to understand a user's regular texting habits when not driving. The application will be downloaded onto the user's phone, so withstanding hazards in the environment it is used in is not much of a concern.

1.4 INTENDED USERS AND USES

The intended users are anyone who drives a vehicle and has an Android smartphone. As the app is meant to detect texting while driving, if someone is not a driver then it doesn't make sense for them to have an app that prevents them from texting and driving. The user also needs to have an android smartphone as the system is an Android app, so therefore they need a phone with the Android OS to run the app.

The intended use of the app is for anyone who wants to make sure that they cannot text and drive. For example, this could be a parent who has their child install it because they want to make sure the child stays safe; or, someone who acknowledges that they have a problem with texting and driving and want to take measures against their habit.

1.5 ASSUMPTIONS AND LIMITATIONS

The list of assumptions is as follows:

1. The phone used during driving is being consistently used by the driver. They will not use someone else's phone while in the driver's seat. This assumption needs to be made since machine learning will be used to track the phone owner's tendencies. If the driver uses someone else's phone then machine learning will not help with detection.
2. The user has the app enabled consistently, even when not driving. This will allow the app to learn the owner of the cell phone's texting tendencies when they are not driving so that TwD is easier to detect.
3. The app should be relatively simple to activate. Therefore, the user does not have to go through many steps themselves to make it functional. The user will be unwilling to use the application if it is too complicated to activate.
4. The driver, if texting, will only ever be texting with one hand. This is the assumed normal behavior. Detecting texting while driving is simplified by this assumption if there is a way to determine how many hands they are using to type on their phone.

5. The application will only be used by drivers in Iowa. Only letting the application be used in Iowa will simplify the data privacy laws that need to be followed.
6. People who are running will not be texting. Since runners can reach very fast speeds that are over the speed threshold, the assumption is that the user will not be using their phone if they are running as that is difficult to do anyway.

The project limitation are as follows:

1. The application must not take up unreasonable amounts of memory or battery life to run. This could cause problems or irritation for the user if violated.
2. The application will support English and not any additional languages or translations.
3. The application will not make use of any external hardware. All of the sensors utilized will be on the phone.

1.6 EXPECTED END PRODUCT AND DELIVERABLES

The end product is a standalone Android application which detects texting while driving. This will be done with machine learning techniques and readings from the phone sensors. This application will not require any additional hardware to be run.

Since this project is split across two semesters, there will be a preliminary end product by the end of the first semester. This will be a working prototype with all intended functions of the final product in use while the application is running, but not necessarily perfected or well-developed by that point.

The application will be capable of performing multiple tests to determine the user's position in the vehicle. Such tests will include speedometer tests, vertical/centripetal acceleration tests, GPS velocity tests, texting speed/typo tests, phone handling tests, and camera tests. Using the data collected from these real-time tests, the application will determine whether the user is driving or not.

2. Specifications and Analysis

2.1 FUNCTION AND NON-FUNCTIONAL REQUIREMENTS

The following is the list of functional requirements for this project:

1. The application prevents drivers from sending texts if certain dangerous conditions are met. This is the most important functionality and the main purpose of the project.
2. The false positive rate and false negative rate should not be greater than 10%. It would be very easy to block 100% of drivers, but that would also block passengers as well. The app should block as close to 100% of drivers while blocking as few passengers as possible.
3. The application should run on Android OS 6.0 and newer. It should be available on as many phones as possible without being limited to an unnecessarily basic version of Android. Fortunately, versions of software before 6.0 are uncommon.
4. The app does not use an internet connection (it is a standalone app). This is necessary if service is poor and to avoid breaking any applicable data handling laws with outside data storage.
5. Optional: Stop the phone from viewing texts. This would of course allow the texts to be received, but would not allow the user to access them.
6. Optional: Response time is fast enough to block the reading of any incoming texts. The app should be fast enough to block these incoming texts. If it is not fast enough to do this, there is not much point in having it.

The following is the list of nonfunctional requirements for this project:

1. The app itself will be easy to set up and run. The app should be as easy as possible for the user to set up. This does not count hardware, as hardware will probably be a tedious setup and can't be helped.
2. The app must be reliable at least 95% of the time, meaning that it runs consistently without crashing. This is important since the app is active most hours of the day to monitor user habits; also, obviously it would defeat the app's purpose entirely if it crashed while the user was driving.
3. The app must comply with applicable sensitive data laws (presumably resolved by making it a standalone app). Personal information is potentially at risk, so it is important that the app does not break the law when dealing with that sensitive data.

2.2 PROPOSED DESIGN

The design incorporates six modules to determine one thing: is the phone user texting while driving. The Speedometer component determines the potentiality of danger. Texting Speed, Spell-Checking, and Phone Handling modules determine if the user is behaving abnormally while texting. The Centripetal Acceleration and Camera modules determine if the user is in the driver's seat. The combination of possible danger, distraction, and the user being in the driver's seat results in locking the phone's texting abilities for safety reasons.

Speedometer: The first and most fundamental is the speedometer, a sensor Android phones have built-in. This will determine if the system is trying to detect texting while driving at all. If the speedometer is below a safe speed, the system will switch to its learning mode. If it is above safe speed, the system will switch into detection mode and begin calculating the probability of texting while driving. The safe speed for the application is defined as at or below ten miles per hour. Its inputs are readings from the phone's built-in linear acceleration sensors. Its output is a true/false value indicating if the user is traveling at a potentially dangerous speed.

Texting Speed: When not in detection mode, key input will be captured while the texting application is in use so that the system can learn the user's typical texting speed. During detection mode, the system will take this norm and compare it to how fast the user is currently typing. If the texting speed is significantly slower, it implies the user may be distracted (possibly driving). Its first input is the danger indication from the Speedometer. Its second input is the current message the user is typing. If the danger value is false (no danger), then its output is a recalculation of the user's typical average texting speed with the addition of the analytics of the current message. If the danger value is true (possible danger), then its output is a true/false indication of distracted driving based on a comparison between the current texting speed and the recorded average.

Spell-Checking: When not in detection mode, the user's captured key inputs will be analyzed for spelling errors so that the system learns how many errors are in the average sentence the user types. During detection mode, the system will take this norm and compare to the number of errors in what the user is currently typing. If the ratio of misspelled words is significantly higher, it implies the user may be distracted (possibly driving). Its first input is the danger indication from the Speedometer. Its second input is the current message the user is typing. If the danger value is false, then its output is a recalculation of the user's typical spelling correctness percentage with the addition of the analytics of the current message. If the danger value is true, then its output is a true/false indication of distracted driving based on a comparison between the most recent spelling correctness analysis and the recorded average.

Phone Handling: While in learning mode, the system will try to become accustomed to the user's phone-handling habit, such as the user's dominant hand, how often they use the phone with one hand, etc. In detection mode, this information will be used to detect if the user is handling the phone significantly differently than normal. For example, if the user almost always texts with two hands, but is trying to text with only one hand while the system is already in detection mode, it is more likely that the other hand is being occupied steering the wheel (indicating texting while driving). Its first input is the danger value provided by the Speedometer. If false, then its output is a recalculation of the user's typical holding patterns (expressed in angles) with the addition of the analytics of the current pattern of behavior. If true, then its output is a true/false value indicating abnormal user behavior based on a comparison between the current pattern of behavior and the recorded average.

Centripetal Acceleration: Using GPS functionality, it can be determined what side of the vehicle the user is in when it makes turns. If the GPS determines the user is on the right side of the vehicle, then the user cannot be in the driver's seat and detection mode will be disabled. If the car encounters a bump in the road, GPS functionality can also determine if the user is in the front or back of the vehicle. If GPS determines the user is in the back, then detection mode will also be disabled. If the user is determined to likely be the driver, spelling and texting speed indications are much more likely to disable the phone's texting functions. Its first input is the danger value provided by the Speedometer. Its second input is the data provided by GPS. Its third input is the data provided by the phone's own acceleration sensors. If the danger input is false, this module is dormant and produces no outputs. If true, then its output is a true/false value indicating if the user is in the right (true) or left (false) side of the car.

Camera: While in detection mode, the front and back cameras of the phone can be used by the system to try to figure out where the phone is in the car. If the system can identify key features of the car like the wheel or the seatbelt, it can provide information about where the user is likely sitting (i.e. the direction of the seatbelt across the user's body can tell which side of the car the user is in). Its first input is the danger value provided by the Speedometer. Its second input is data (photos) provided by the phone's built-in camera. If the danger input is false, this module is dormant and produces no outputs. If true, then its output is a true/false value indicating if the user is in the driver's seat (true) or any passenger seat (false).

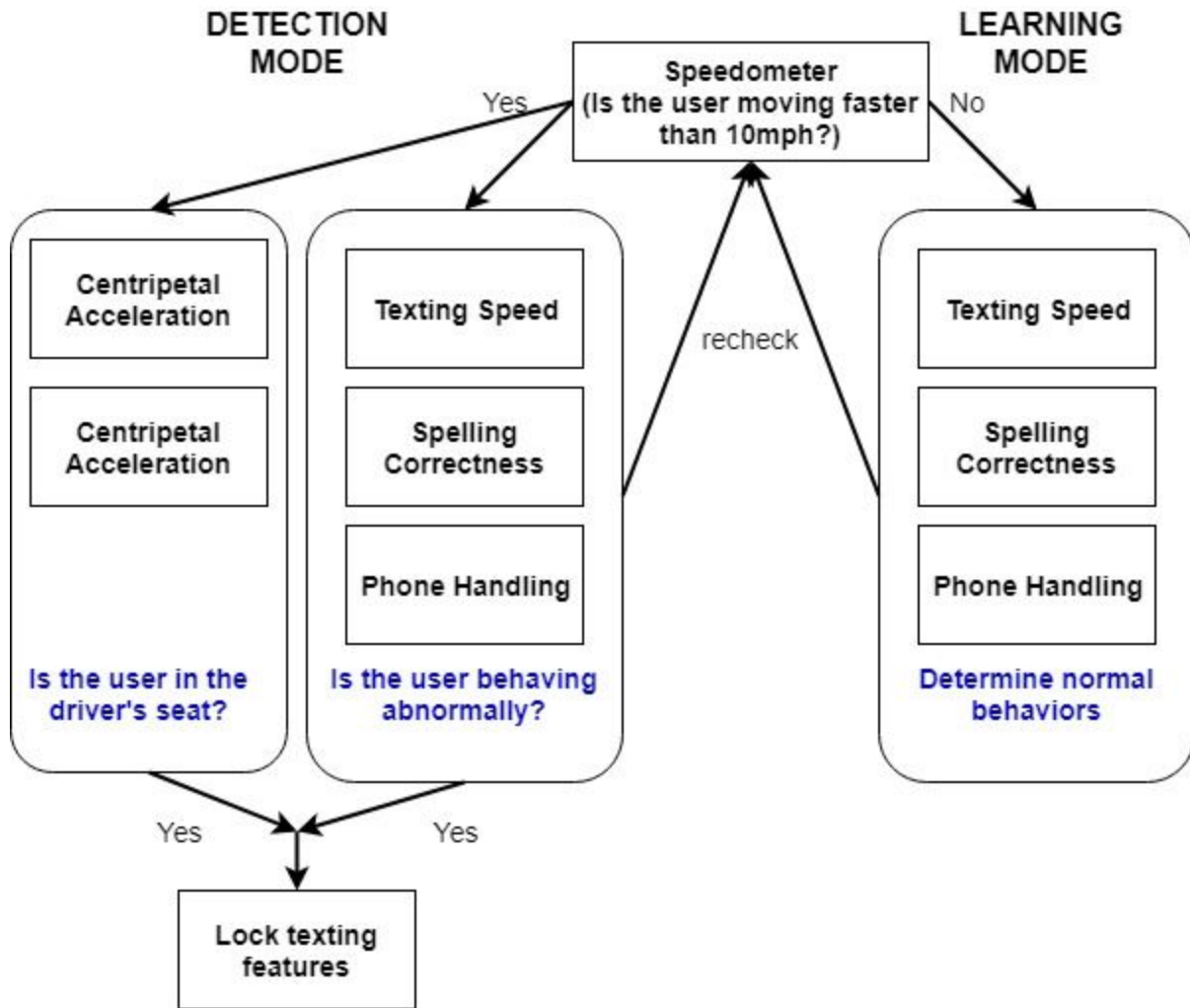


Figure 1: System Overview Diagram

Figure 1 above is given above to provide a simple visualization of how these modules work. The combination of these technologies should be able to work together to consistently learn when users are texting while driving and when they are not. It would take the combination of the danger value being true; two of the three “distraction” modules outputting true; and one of the two positional modules being true to fulfill a lockout requirement, where the phone’s texting features would be disabled until the user is back down to a safe speed for more than five minutes.

Figure 2 is given below to attempt to visualize the inputs and outputs of every module. The modules and their output arrow are color-coded to help clarify since there are so many of them. The Phone Hardware and Text Watcher boxes are already built into Android OS, and are not something that must be designed for the project.

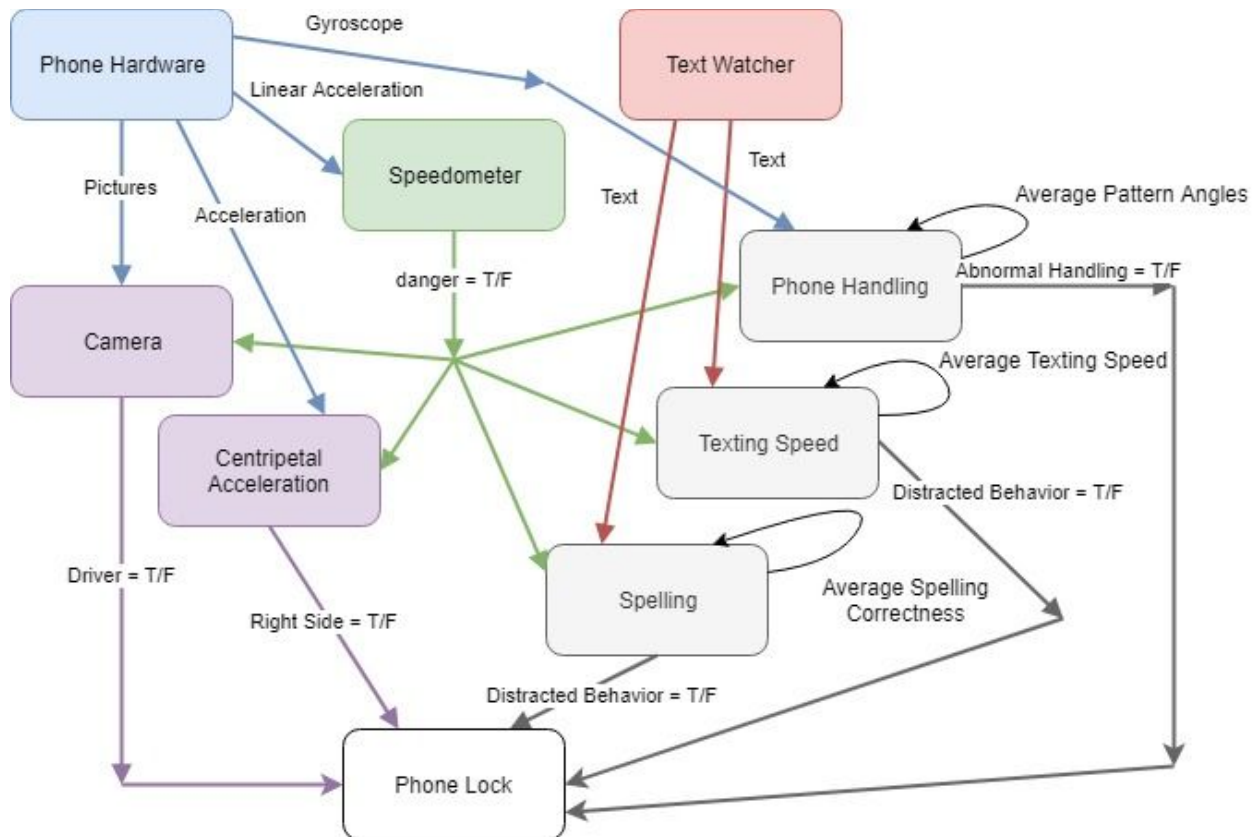


Figure 2: Inputs and Outputs Diagram

2.2 DESIGN ANALYSIS

Up to this point in the project, many weeks of research produced a solution that is unique and effective. The research done includes technologies that have been tried previously, the effectiveness of current research, and implementation of possible solutions. Throughout the research, thirteen possible technologies were found which could benefit the solution. The following are the thirteen technologies researched:

1. Analyzing texting speed when user isn't driving compared to when they are driving.
2. Analyzing grammatical errors in texts when the user isn't driving compared the when they are driving.
3. Using NFC tags in the steering wheel and drivers seat to detect the driver's smartphone. Being detected indicates that the user is definitely the driver.
4. Using the phone's built-in speedometer to determine if there is a possibility that the user is in a fast-moving vehicle.
5. Using a camera on the dashboard to track eye movement and facial positioning to see if the user has their eyes off the road.

6. Using centripetal acceleration and angular speed via multiple technologies to determine if the smartphone is on the right side or the left side of the car.
7. Using vertical acceleration when going over bumps via the same technologies to tell if the smartphone is in the front of the car or the back of the car.
8. Using the front facing camera in a smartphone to determine which side of the car the user is sitting on based on the way the seatbelt is laying across their chest.
9. Using the front facing camera in a smartphone to find predictable constants in a car.
10. Using the back camera to find either the steering wheel, the glove box, or the back of a seat to determine whether the user is in the front or the back of the car. If they are in the front, the images would be used to determine if they are in the passenger's seat or the driver's seat.
11. Monitoring leg lift with a smartphone in the pocket to tell determine which side of the car the user entered on.
12. Using the speakers in a car to send frequencies to the smartphone to determine the location of it in the car.
13. Analyzing the way a user handles their smartphone when they are not driving compared the when they are.

Once the thirteen technologies were researched, the group discussed the feasibility of each solution. The factors we took into consideration to determine if a component was feasible include: how complex the approach is, hardware requirements, resource cost, and the availability of data to test each approach. Based on these factors, the six possible components chosen to test for the solution include the six technologies discussed in the proposed solution section above: speedometer, texting speed, spell-checking, cameras, centripetal acceleration, and phone handling. Now the team is in the process of testing each technology to see if they will be effective solutions. Based on the research the team has done, these six technologies seem to be the most promising for giving useful data, as well as being able to implement them in the time frame we have to work on the project.

The first three technologies (analyzing velocity, texting speed, and spelling) are the simplest and most easily-accessible out of all six. Using the speedometer is the most feasible and the first part of the solution that has to be implemented. Determining if a phone is in a moving car will be important for deciding whether the application should be in learning mode or in detection mode. Analyzing texting speed and looking at grammatical errors will be used to learn the user's texting habits. This data will then be used to detect when their texting speed or amount of grammatical errors differs from the users normal habits significantly enough to suggest they may be TwD.

The last three technologies (analyzing cameras, centripetal acceleration, and phone handling) proposed either do not have much research to learn from or are new ideas that were invented in collaboration with the client. Using GPS to determine the centripetal acceleration on the phone can tell whether the user is on the left or right side of the car when the driver is making turns. To determine if the user is in the front of the car, GPS can be used when the car is going over bumps in the road because the upward acceleration experienced by the phone is noticeably different between parts of the car. This technology will be key to helping us determine what seat the user is in. The challenges faced with this technology are that there is very little research on this approach and it will have to be assumed that the driver is going to have to make a few turns and drive over bumps so it can be determined where they are positioned in the car. The next technology is using the front facing camera to find predictable points in a car. "Predictable points" means points in cars that would be common throughout different models and sizes of cars. An example of a predictable point would be the windows on each of the car doors. This is a new proposed approach from the client. Data will need to be collected from many different kinds of cars to see if this approach can help determine if the app can accurately and consistently locate where the user is in the car. The last technology looked into is also a new approach that would learn how a user normally holds their phone so inconsistencies can be found in the way they hold their phone when they could potentially be driving. Since this is also a new idea, we will need to figure out what kind of data will be useful to collect and how the app can use that to learn about how users handle their phone.

With these six technologies forming the basis of the modules, the design has these strengths and weaknesses:

Design Strengths:

- Modular build allows the six modules to cover each other's weaknesses and inaccuracies.
- Modular build allows for easy removal of parts of the application deemed infeasible later.
- Some modules can fail and the application will still be functional.
- Machine learning will allow for more accurate detection overtime.
- No extra hardware infrastructure is required for any of the modules to function properly.

Design Weaknesses:

- Modular nature requires work on many parts, which will be hard to implement in the limited window of time given.
- Modules require specific hardware components and are rendered completely useless if the component they require is damaged.
- Machine learning adds considerable processing time for detection
- The camera module will be working with hundreds of images per user, requiring the processing of megabytes of data which takes time and power.

3. Testing and Implementation

3.1 INTERFACE SPECIFICATIONS

A computer will be used to check the functionality of tests and components of the application. Customizable unit test cases will allow us to input values and see the result in the application. The speedometer on the car will also serve as a means of double-checking the readings on the application.

3.2 HARDWARE AND SOFTWARE

The hardware that we will be utilize to test is the phones provided by the advisor and any personal devices that we want to use to test the app, such as laptops. We will also be using a car or bus when we are testing the aspects that need to detect moving above ten miles per hour in the car. The only service that these vehicles will provide is helping the testers move fast enough to provide control for testing that is affected by the speed the person is going. In other words this will be used to control the measurement of whether a person is going fast enough where it is possible they are driving, or if they are going too slow for it to be possible for them to be driving. Even though we won't be using any physical parts of the car, the car itself is needed to test the application. We could also possibly be using some sort of hardware to distract the user to a degree that it would be similar to texting while driving. This hardware is yet to be decided on, but it is probably it will take the form of some driving simulator. With this, hopefully a similar level of distraction will be provided with none of the risk. The software we are using is very limited due to the nature of the project, but JUnit test or something equivalent will be used to verify that code works prior to testing on hardware.

3.3 FUNCTIONAL TESTING

The testing process for this project can be broken down into four testing types: unit, integration, system, and acceptance. Each of the six software modules will require coverage by each type.

Unit Testing: In our project, we have six modules that require testing on a micro and macro scale. In this case the term micro is in reference to traditional unit testing.

This will be done to ensure each smallest testable portion of code is tested for accuracy and simplicity. As the code base grows, the goal is to maintain a high level of code coverage. To give a quantitative goal, the testing suite will achieve at least 90% coverage.

This will be done with a white box strategy. Testing framework tools and mock objects will need

to be created to complete unit testing of our entire code base. The methods and constructors of each module will be unit tested heavily to ensure correct behavior. Software tools that will be used are JUnit and Mockito.

Furthermore, each module must be tested for correct behavior on a macro scale. This is intended to ensure that each module functions correctly on its own.

Each module has a unique black box testing process. Each of the modules' testing strategy is outlined in the following manner:

1. Module Name

- a. Motivation - Description of what these tests will show*
- b. Process Description - How the tests will be conducted. The number of times the process is repeated is left out at this time, for this will become clear once more data is collected.*
- c. Verification - What data will be analyzed and the error tolerance that must be met to prove module effectiveness*

1. Accelerometer

- a. The accelerometer module must be tested for accuracy and consistency. Tests will determine that the module converts the accelerometer data into a velocity value that is representative of the phone velocity. Any discrepancies in the output of this module will lead to incorrect identification of TwD.
- b. This testing process will likely require two testers: one to drive a vehicle and one to record data on a mobile phone. The driving tester will operate the vehicle at speeds ranging from 5-30 mph in 5 mph increments. The application will continually output the current velocity, which will be compared to the actual velocity of the car that is obtained from the speedometer.
- c. The verification of this test will be based on the error tolerance of ± 3 mph. This is a working number for now, and will try to be lowered as the tests are conducted. This module is an important factor in determining TwD, so outputs need to be precise.

2. Texting Speed

- a. The module will read the texting speed of the user in the unit of words per minute. These tests will determine the accuracy of the module.
- b. A user will type sample words while the module is running. Data will be manually collected of how many words they complete in a given amount of time, and the output of the module will also be recorded. This process will be repeated a number of times to ensure a sufficient amount of data is collected.

- c. To verify that the module behaves correctly an error tolerance of ± 5 words per minute will be used.
3. Spell-Checking
 - a. The spell-checking module must be tested on its accuracy of determining misspelled words.
 - b. The spell-checking module will be tested by running sample words through the module and recording the output of the module. The actual number of misspelled words in the sample data will be manually recorded as well. This process will be repeated a number of times to ensure a sufficient amount of data is collected.
 - c. The verification for these tests will require an error tolerance of 3% of words being misidentified as either correctly or incorrectly spelled will be used.
4. Centripetal Acceleration
 - a. The centripetal acceleration module determines whether the driver is in the passenger or driver seat. It will be tested for accuracy.
 - b. The user will get into the driver or passenger seat of a car while the module is running. The output of the application will be recorded and the side of the car the user is actually sitting on will be recorded. This process will be repeated until there is a sufficient amount of data.
 - c. The output of the module will be compared to the manually collected data. The verification for these tests will require a success rate of 80%. In other words, the module should correctly identify the seat that the user is in 80% of the time.
5. Camera
 - a. The image processing module will be tested on its ability to determine which seat the person in the image is sitting in.
 - b. A user will be seated in either the driver seat, passenger seat, backseat on the driver side, or backseat opposite the driver side. The module will run and output the seat that the user is sitting in. The module output will be recorded. This process will be repeated a number of times.
 - c. The output of the module will be compared to the actual seat that the user is in. The verification for these tests will require a success rate of 90%. The module should correctly identify the seat that the user is in 90% of the time.
6. Phone Handling
 - a. This module will use a machine learning algorithm to learn how the user typically holds their phone under different conditions. It will be tested on its accuracy in determining if the user may be driving at the current moment.

- b. The phone handling module will be tested by testing the effectiveness of the machine learning algorithm. The module will be fed sample data to construct a knowledge of the user's usual phone handling traits, then will be tested by having the user hold the phone in various positions. Unusual positions will be marked as positions that the module should identify as possible TwD positions.
- c. The output of the module will be compared to our expected results. The verification for these tests will require a success rate of 90%. The module should identify unusual phone handling behavior successfully in 90% of the trials.

Integration Testing: This will be necessary to incorporate new solutions to the existing code base. Each of our proposed solutions will be integration tested after they are individually tested and proven to be effective. Integration testing is going to be completed with black box strategy and using the bottom up approach. The modules that are of least complexity will be integrated through testing first. The more complex modules will be integrated as they are completed. Test drivers will not be used to simulate the higher level units because the low level units are not dependent on the high level units. This is why bottom up approach is suitable for our project.

It is important to note that only one module will be integrated into the entire system at a time, and will take place only after the white box and black box testing described above is completed.

System & Acceptance Testing: This type of testing is required to compare the performance of our software system in regards to the functional requirements. This is one of the high level requirements, so the black box strategy will be used. This testing is focused on the user, so it will primarily consist of testers using the application and recording results.

The most important functional requirement that this testing will verify is having false positive and false negative rates no larger than 10%.

3.4 NON-FUNCTIONAL TESTING

Note that our project does not require any external hardware so no test cases will be needed in that area.

The non-functional testing will be comprised of three main section: security, usability, and performance tests.

Security Testing: We will verify that the user's data is not at risk of being compromised by creating tests that attempt to breach this data. For example, a user credential will be established to ensure that no unqualified user can modify or access the data that is collected. Testing this will involve creating dummy users to verify that their capabilities are as intended. The goal for the


application is a zero tolerance for data breach, so any discrepancies that this testing exposes will be addressed.

Usability Testing: Usability testing will be focused on phone model compatibility, and battery usage. We plan on running the app on multiple versions of android phones, and evaluating its battery performance. The usability testing will be done in a black box approach. Data will be collected on battery consumed per minute of the application running.

Performance Testing: One way we plan on testing performance is recording the percentage rate that the application crashes during system testing. The software’s runtime will also be tested and recorded for each module in order to keep them as low as possible.

3.5 MODELING AND SIMULATION


We will define a model with data for each module that will be likely to be a person that is texting while driving and see if our code can correctly identify the person as a driver. The data will be as follows.

Speed	Phone is moving at 30 miles per hour.
Texting speed	Avg speed 202.77 characters per minute Current speed 123.47 characters per minutes
Spelling errors	Avg mistakes per session: 5 mistakes Current number of mistakes: 18 mistakes
Images from phone	

Centripetal acceleration	V(m/s)	Driver a(m/s²)	Middle a(m/s²)
	1.87	0.532	0.518
	2.39	0.869	0.846
	2.02	0.621	0.604
	2.48	0.935	0.911
	2	0.608	0.592
	V(m)	Driver a(m/s²)	Middle a(m/s²)
	3.17	2.54	2.661
	2.8	1.989	2.076
	4.05	4.16	4.344
	2.4	1.462	1.525
1.6	0.65	0.678	
Phone Handling	Magnetic field x value: -24 Magnetic field y value: 150 Magnetic field z value: 300		

Table 1: Model Data for Distracted Driver

We will define a model with data that will not likely be a person that is texting while driving and see if our code can correctly identify the person as a passenger. The data will be as follows.

Speed	Phone is moving at 30 miles per hour.
Texting speed	Avg speed: 202.77 characters per minute Current speed: 193.74 characters per minutes
Spelling errors	Avg mistakes per session: 5 mistakes Current number of mistakes: 7 mistakes
Images from phone	

Centripetal acceleration	V(m/s)	Driver a(m/s ²)	Middle a(m/s ²)
	1.67	0.424	0.413
	1.84	0.515	0.501
	2.09	0.664	0.647
	1.71	0.445	0.433
	1.48	0.333	0.324
	V(m)	Driver a(m/s ²)	Middle a(m/s ²)
	2.4	1.462	1.525
	1.99	1.005	1.049
	2.15	1.173	1.224
	3.2	2.598	2.712
	3.86	3.781	3.946
Phone Handling	Magnetic field x value: -120 Magnetic field y value: -46 Magnetic field z value: 0.5		

Table 2: Model Data for Passenger

Data used for these models are based off of data that all team members have collected for each module. This data would be able to classify the first model as a driver who is texting and the second model would be classified as someone who is riding in a car, not driving, and texting.

3.6 PROCESS

Speedometer: This is the simplest module that we will use. We will use multiple phones that the application is running on. We will put all these phones in the same vehicle and will see if they detect if the vehicle is going above the allowed speed. This should be a 100% detection rate so we will know we have done something wrong if this doesn't work as intended.

Texting Speed: For this module, we plan on measuring how fast the user normally texts, when they are not driving. We will then safely distract a user of the application while they are in a moving vehicle to then measure how fast the user texts when distracted.

Spell-Checking: This module will measure how many errors a user typically has when texting while not driving. We will then compare how many errors a user has when they are safely being distracted in a moving vehicle. To make sure these tests are accurate we will have multiple users do the tests and use multiple phones to see if there is a difference.

Centripetal Acceleration: This module will use centripetal acceleration to determine the location of the user in the car. It will use GPS to detect turning and combine this data with known velocities, taking data on left and right turns. It will then determine the the average of these turns combine to find the central reference point of acceleration. If the phone's acceleration is less than or greater than the acceleration of the reference point, it indicates that the user is on one side of the car. We will safely test this by having a passenger user use the application. If the application can detect if the user is a passenger, it should be able to detect if a user is in the driver's seat.

Camera: Test images will be taken for comparison of images captured while user is in the car. Tests will be done to determine accuracy of detecting when user is in the car from these image captures.

Phone Handling: This module will use a machine learning algorithm to learn how the user typically holds their phone under different conditions. From data collected, the application will discover how the user holds their phone in the event that they text and drive. If the handling that the machine learning algorithm learns, confirms a texting while driving handle on top of positives from other tests, then it will detect texting while driving.

The figure below represents the order in which these technologies must be implemented and working. The speedometer is most fundamental, since it determines which technologies are active, and if they are in learning mode or detection mode. The technologies which are active during both detection mode and learning mode must have a functioning learning mode first, since detection requires information gathered in learning mode to do anything. The technologies to the left of the Speedometer determine positioning in the car, and those on the right determine the probability of TwD.

At least one technology to detect TwD and one for user position must be functional before the system can be allowed to lock out the user. If only one half of the functionality is present, passengers will either be accidentally locked out of their phones, or a driver who *is* TwD will not be.

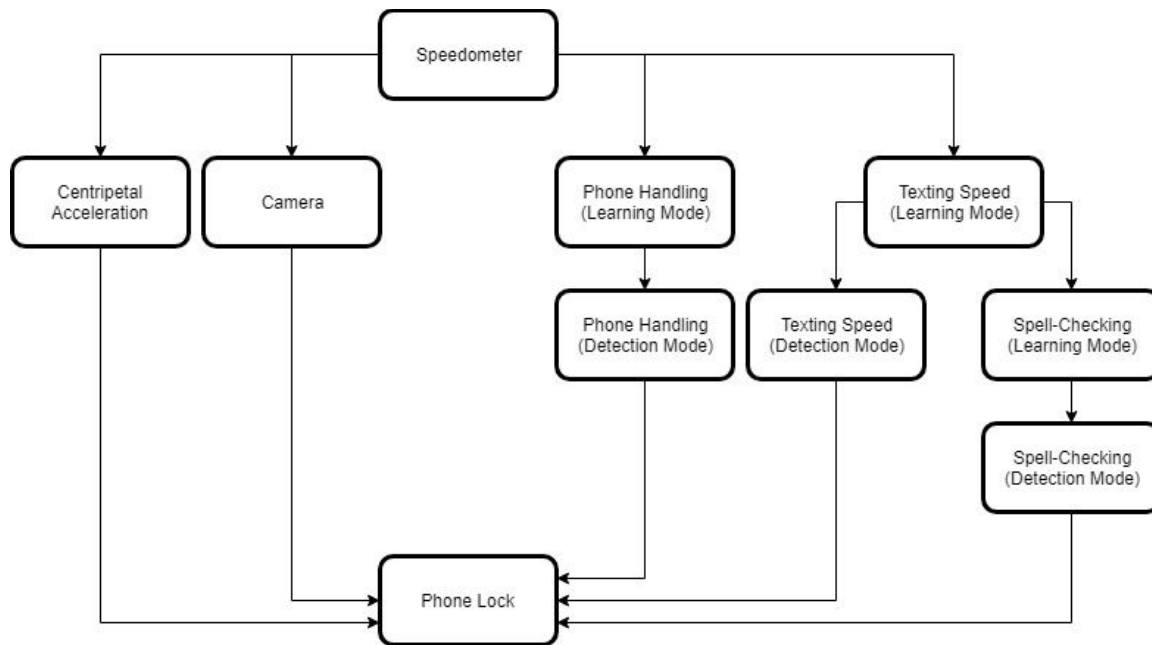


Figure 3: Module Dependency Flowchart

3.7 PROCESS DETAILS

The current design includes six modules that will work together to help determine if the user is texting while driving. These modules are going to be implemented in a specific order so we can have all dependencies satisfied for each module. The speedometer test is the first thing we have to develop because every other module depends on it. The speedometer test will be able to determine if the phone is moving at or faster than ten miles per hour or slower than ten miles per hour. Knowing the speed that the phone is traveling at will tell the application whether to be in learning mode or detection mode. Once the speedometer test is implemented then we can begin developing the other modules. The centripetal acceleration module and image detection using the camera module don't have anything else depending on them so we can develop those in the rest of the time we have left after the speedometer has been implemented. The phone handling module and texting speed module are broken down into learning mode and detection mode as well as texting speed is broken down into analyzing texting speed and analyzing spelling errors. These modules will have to be developed more carefully with respect to dependencies in the design.

3.8 IMPLEMENTATION ISSUES

The first implementation issue is that Android does not have a way to detect which button was pressed on a soft keyboard. A soft keyboard is the keyboard that pops up and pops down on the phone touch screen. Android only has a depreciated KeyListener interface. Although this

interface is deprecated and shouldn't be used it also only worked on a hardware keyboard. A hardware keyboard is a keyboard that has real physical buttons and is always on the phone. This was an implementation issue because it was the first way that was discovered to see the letters typed. The goal of monitoring what keys were typed was to check the spell check and texting speed modules of the detection program. To overcome this problem, the interface TextWatcher was used. This interface allows soft inputs to be captured, but only on specific boxes.

The spelling module has its own implementation issues. Android has a promising Spell Checker Session interface which can analyze specific words for spelling suggestions. However, for a couple years now it seems that this interface is disabled or deprecated on at least some Android phones. A new solution or workaround has not yet been found.

Another implementation issue that has been discovered is machine learning. This has been an implementation issue because the concept of machine learning itself is huge. Therefore there has had to be a large amount of research done on machine learning. The result of all the research has been that OpenCv and TensorFlow will be used. Those two sources will be used because they have a variety of functions to determine and identify the items that we are looking for. One example of the items that we are looking for is seatbelts in the pictures that the application will take.

3.9 RESULTS

So far (as of 12/3/2018), none of the modules have been finished but half of them have begun being programmed. Specifically, the three modules with dual learning/detection modes have been partially completed. The most successful thus far is the texting speed module. Using a text watcher, sufficient information about user input can be retrieved to determine a consistent texting speed in characters per second. Information can also be saved long-term between sessions of the application. This essentially completes the learning mode half of that module. The spelling correctness module is similar in implementation since it can make use of the exact same input capture method as the texting speed module. However, progress has been delayed in this area due to the unsolved implementation issues mentioned earlier. The phone handling module was first tried using the gyroscope sensor. The data that was collected using the gyroscope was not conclusive enough to be able to determine major differences between holding a phone in different positions. Therefore the decision to switch to using data from the magnetism sensor was made. The differences in the x, y, and z values for this sensor are drastically different and will make it easier to distinguish between differences in phone handling for each user.

We have not had any testing done so far, most of what we have done is research into determining what may and may not work. The next step in the project is to start collecting data for each of the six proposed technologies to determine feasibility. Once data collection is done, we can start developing and testing each component of the application.

4. Closing Material

4.1 CONCLUSION

The project aims to provide a solution that can reliably detect texting while driving. To accomplish this, we will develop a standalone android application using android developer software. Each week we will iteratively develop a more functional version of the solution and present to the client for feedback. This project will be heavy on research and prototype testing as we experiment the feasibility and accuracy of different solutions. A number of solutions of varying complexity will be explored. The ultimate goal is to create an application that will utilize the sensors embedded in the phone to successfully detect texting while driving. Once detected, a protection system will activate that makes the user unable to send text messages until they stop driving. The delivery of this solution will provide a means for safer driving through text prevention.

4.2 REFERENCES

- [1] “100 Distracted Driving Facts & Statistics for 2018.” *TeenSafe*, 5 July 2018, www.teensafe.com/distracted-driving/100-distracted-driving-facts-and-statistics-2018/.
- [2] Cheng, Bo, Jian Xuesi, Li Xiang-Yang, and et al. “You’re Driving and Texting: Detecting Drivers using Personal Smart Phones by Leveraging Inertial Sensors.” *Mobicom*. ACM, 2013.
- [3] “Drivewise® From Allstate GET REWARDED FOR SAFE DRIVING.” *Allstate*, <https://www.allstate.com/drive-wise.aspx>
- [4] “Safe Drive Zone.” *Safe Drive Zone*, www.safedrivezone.com/.
- [5] Wang, Yan, Jie Yang, Hongbo Liu, and et al. “Sensing Vehicle Dynamics for Determining Driver Phone Use.” *MobiSys*. ACM, 2013.