Smart Car Seat

A Final Report Submitted by:

Alan Tessier (Group #11)

Florida Atlantic University

July 18, 2020

Abstract

There are many car seats available on the market, but there are quite a few that do not keep consistent track of the child safety. It is unfortunate, but there are cases where a child is left in the vehicle alone for too long, when the vehicle is too hot or when there is an unsafe driver with the child present in the vehicle. The smart car seat is a great solution to solve these types of problems. Ideally, the smart car seat will keep constant supervision on the child and make decision based on the child's safety. It will allow parents to be more aware of their actions or anyone else who is the vehicle with their child.

Optimally, the device should notify the caregiver, parent, or local authorities when the child is in danger. Using a smart device like this can be attached to any car seat, and lead to a reduce rate of carelessness and fatalities.

Contents

Abstract 2

List of Figures 5

List of Tables 6

Smart Car Seat 7

Background 7

Statement of Problem 7

Scope of Work 9

Overview 9

Literature Review 9

Alternative Solutions 11

Approach #1: 11

Approach #2: 11

Approach #3: 11

Evaluations 11

Decision 12

Implementation 13

System Specifications 13

System Requirements 13

Functional Requirements 13

System Block Diagram 14

Circuit Diagram 16

Hardware Flowcharts 17

Software 18

Database 19

Web application 20

Testing and Calibration Results 21

Lessons Learned 22

Conclusion 22

References 23

Appendices 24

Organization Chart 38

List of Figures

Figure 1: Block Diagram of Entire System..... 14

Figure 2: Logo..... 14

Figure 3: Electronic Box that contains sensors..... 14

Figure 4:System Block Diagram..... 15

Figure 5: System Circuit Diagram..... 16

Figure 6: Hardware Flowchart..... 17

Figure 7: Database Schema..... 19

Figure 8: Web App Flow Chart..... 20

Figure 9: Process of Evaluating Stability..... 20

Figure 10: Neural Network Model (Loss / Accuracy)..... 21

Figure 11 Pin Layout..... 24

Figure 12: Arduino Code..... 35

Figure 13: Tensorflow Model..... 37

Figure 14 Organization Chart..... 38

List of Tables

Table A: Approach #1 11

Table B: Approach #2 11

Table C: Approach #3 11

Table D: Parts List 16

Smart Car Seat

This final report from Group #11 focuses on smart devices for children car seats on the date July 18,2020. This group (#11) seeks a solution to reduce any fatalities and carelessness related to children's who still use car seats.

Background

The basis of this project is to find a solution or enhance methods to improve a child's safety while they're in a vehicle. It can be difficult to wrap your head around that someone can leave their child in danger by not taking simple precaution, but they do happen. For instance, there is a (Safe kids, 2014) survey where 25% of parents have operated a vehicle forgetting that their child was with them. There has been other research (University of South Florida, 2018) indicating all types of (socioeconomic and education) parents and caregivers leaving a child in a vehicle alone for a long period of time. Fortunately, today we have smart devices that can attach to the car seat, but most of them have some pros and cons. Importantly, a device can maybe keep track of the temperature in the vehicle, supervision and location, but not the driving.

Statement of Problem

The problem we have today is that most car seats are not dynamic and have no advance safety methods involve outside of the physical aspect of build the actual car seat. This tends to leave more dependency on the parents and more room for careless mistakes to happen. In order to prevent any extreme situation, the parent need to be actively reminded on their mistakes. If for some reason they cannot fix those, then someone else needs to come a fix it such as the local authorities. Overall, there needs to be a safety device that can be parents more involve and mindful of what is and isn't safe for their child.

The purpose behind this project is for parents to be more involve and aware of their actions. Even though most people feel like they do a good job at supervising and keeping their child safe, a device like this will only increase those chances. This can be the same reasoning for a seatbelt; someone can be the safest driver out their and still wear a seatbelt because they can still be put in a situation that can be deemed as life threatening. Generally, having a safety device like this can only help keep a child safe.

To continue, a regular car seat is incapable of recognizing when a child is at risk. To illustrate, imagine all of the parents that leave their child alone in the car; whether it be a 1 minute or 15, a lot of horrible things can happen in that short amount of time. Also, a regular car seat can't evaluate safe conditions for a child to stay in for a long time, whether the vehicle be too cold or too hot, these things need to be recognized. Furthermore, the driver of the vehicle needs to be held with some responsibility as well, a bad turn or break can lead to a lot of trouble. These are just a couple of things that a simple car seat can't prevent from happening.

Scope of Work

Overview

The goal is to develop a smart device that can attach to any car seat in order to mitigate and reduce risk for any harm to a child. There are a series of objectives the devices should achieve. The system should be able to detect when a child is in a seat, adult supervision, evaluate driving, notify parents, access to location, and determine unsafe situations vs safe situations.

In order to achieve these objectives, the project was separated in numerous phases. The first phase was preparation and research; this phase consists of at looking what safety devices where on the market, looking at their faults and success, and stating goals. Phase two consists of designing a way to meet those goals in the most efficient way possible. Phase three entail constructing the project by building several subsystems. Phase four consist of combing all the subsystem into one and testing the system.

Literature Review

There are a couple of companies developing SensorSafe technology ensure the safety and health of children is the first priority. The technology is embedded into child car seat's seat belt clip and is a simple plug in play. There is a transceiver inside the clip and a receiver is plugged into the vehicle; from there everything is done via Bluetooth on a mobile application. The clip mitigates the chances of the child opening the safety harness and if they by any chance they do open the clip, then an alert will be made. There are temperature alerts, GPS notifications, and adult supervisions checks all within the clip.

Clearly, this is a good example of just setup in terms of sensor because everything is embedded into a clip. The technology does a good job avoiding the chances of any child being left alone in a vehicle. Matter of fact, 1 child dies every 9 days from simply being left in a vehicle all by themselves. All in all, it's very similar to the project at hand, and the only difference would be in the clip itself and the rapid movement detection in the project at hand.

The technology gave me a good idea of how I will approach building the system. For instance, they embedded everything into a simple seat belt clip for easy setup for the consumers of this product. In my case, this seems like a great idea because the device can be taken on or off when the user wants to take it off the car seat. Having a portable device instead of having it inseparable from the car seats seems like a good idea.

Alternative Solutions

Approach #1:

Build a hybrid system that involve the sensor from the vehicle.

Table A: Approach #1

Strengths	Weaknesses
• More Accurate data related to the car • Smaller Safety Devices / Less Sensors	• Hard to integrate to older car • Coding needs to be more dynamic to retrieve sensor data from different types of vehicle

Approach #2:

Only using sensors from vehicle and camera

Table B: Approach #2

Strengths	Weaknesses
• No sensors needed • Use a camera for facial recognition	• Data sent via Bluetooth • Only useful for newer cars

Approach #3:

Put all the sensors into one device

Table C: Approach #3

Strengths	Weaknesses
• Portable • Affordable / Flexible	• Some of the data may not be as accurate

Evaluations

The goal currently is to build an affordable and efficient safety device. With this is mind, the device will be accessible to anyone who has a car seat and has simple internet access. Building a device that is more accessible and easier maintain outweigh the idea of someone more

inflexible and costly. Anyone who uses the device just needs to have access to a outlet (usb), so they do not need to worry about charging the device. The next thing needed to be dealt with is the software, the user needs to able to access information relating to the car seat, so the software needs to be flexible and not depend on one platform. Also, the user must not feel as if their privacy is being breached, so having a camera or a system that doesn't turn off seems arbitrary.

Decision

The strongest alternative solution to this problem is approach #3 considering the budget and time needed to complete this project. Opposing approach #1, the device may or may not work for the vehicles I have at now to test on. Opposing approach #2, need access to the data of the sensor of the car and have the ability to manipulate; this approach is risky and takes plenty of time.

Implementation

System Specifications

System Requirements

The goal of the whole system is to find a way to increase safety of a child car seat. In order to do so, the system must be able to be semi-independent. The system will involve external entity such as parents and caregivers to monitor the safety and concurrently doing the same thing in case of dangerous scenarios. This means that the system must have a way of involving these external entities in the system and will (the system) only make an action when someone else isn't accessing the current problem.

Functional Requirements

The functional requirement for this device:

- The system shall detect when a child is place in the car seat.
- The system shall observe the surrounding environment of the car seat
- The system shall retrieve the pinpoint location of the car seat
- The system shall determine whether a child is safe or not
- The system shall notify caregivers, parents, or local authorities when child is in danger
- The system shall evaluate driving when the child is in the vehicle.
- The system shall evaluate stability of the car seat.

System Block Diagram

Figure 2: Logo

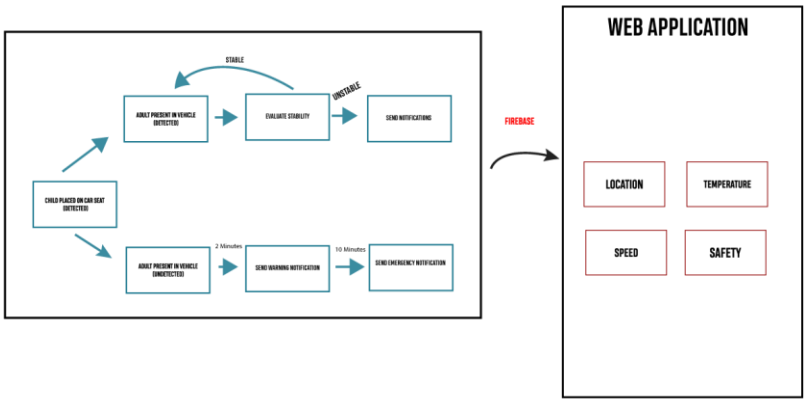


Figure 1: Block Diagram of Entire System

The system is split into a hardware and software roles. In the terms of the hardware, the sensors will be package into a portable box (figure 3) where the user can power the smart device. In terms of the software, the user will be able to log in to a web application, where they can view what is happening within the car seat. Inside of the box will contain sensors that can retrieve location, temperature, acceleration, and the Arduino microcontroller; outside the box will be the force sensor and infrared sensor. The data that is being retrieve will get sent into database and from the web application can grab that data and display it on the user ends. Figure 1 demonstrates the process of how the whole system works from start to finish

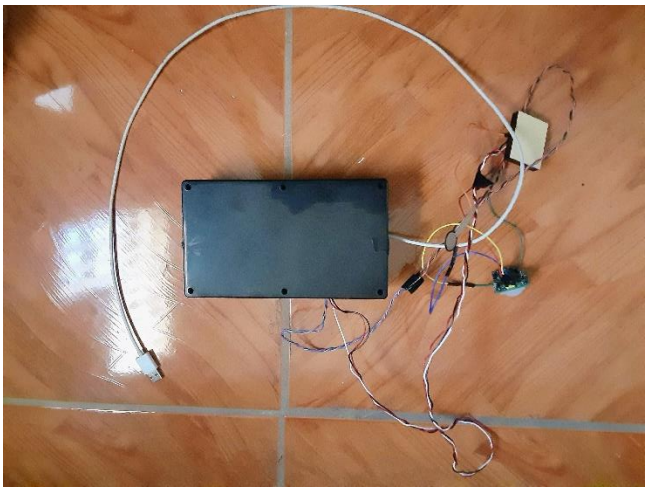


Figure 3: Electronic Box that contains sensors

To clarify, the system (figure 4) is split into mainly separated into two subsystems, the first subsystem is accountable for obtaining data from the sensor and translating that data into useable so it can be sent to the database. This subsystem is then divided into two other smaller subsystems. The managing supervision dealing with car seat; which obtaining the status of child environment. The second, overseeing any movement or location involving the car seat. The other subsystem is the user interface, which is involves retrieving data from the database and appending that data to a web application. The web application has the role of using the data it's retrieves from the database and sending it to the Google Maps API, Neural Network, and Notification. In brief, the system is split into two subsystem that communicate through the Firebase database.

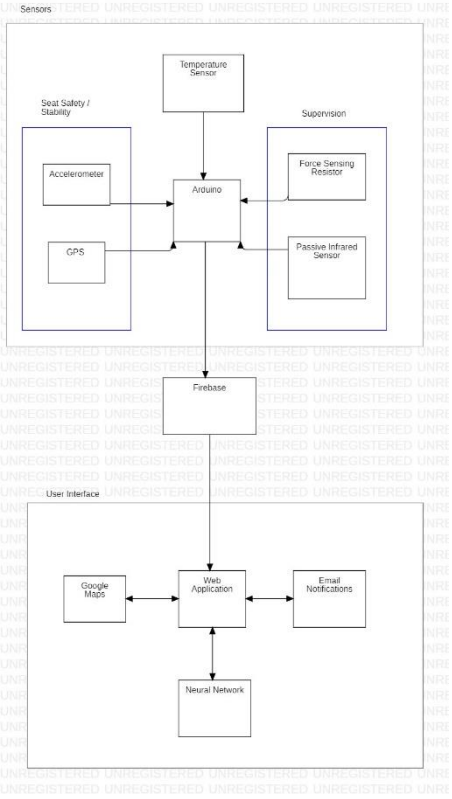


Figure 4:System Block Diagram

Circuit Diagram

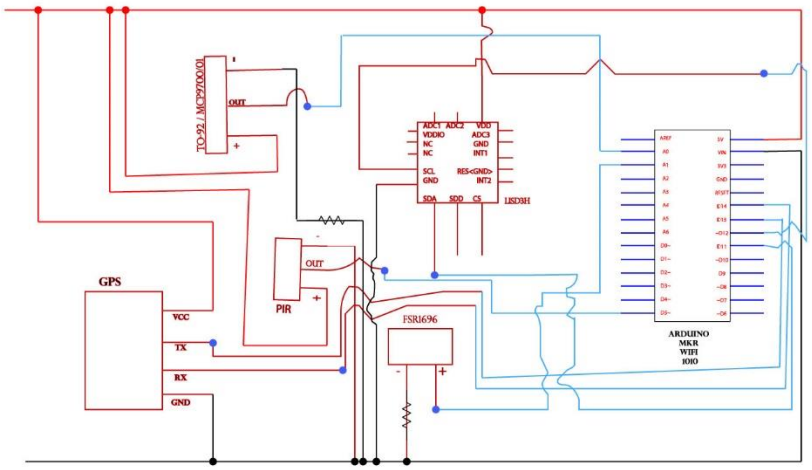


Figure 5: System Circuit Diagram

In the figure above, you can see the sensors that are being connected to the microcontroller.

Table D: Parts List

Sensors	Pinout
• Force Sensing Resistor FSR1696	• GND • Analog #1
• Passive Infrared Sensor HC-SR501	• GND • VCC • Digital Pin #5
• Accelerometer LIS3DH	• GND • VCC • SCL • SDA
• GPS Module NEO-6M MV-1	• GND • VCC • RX • TX
• Temperature Sensor MCP9700	• VCC

Sensors	Pinout
	• GND • Analog #0

Hardware Flowcharts

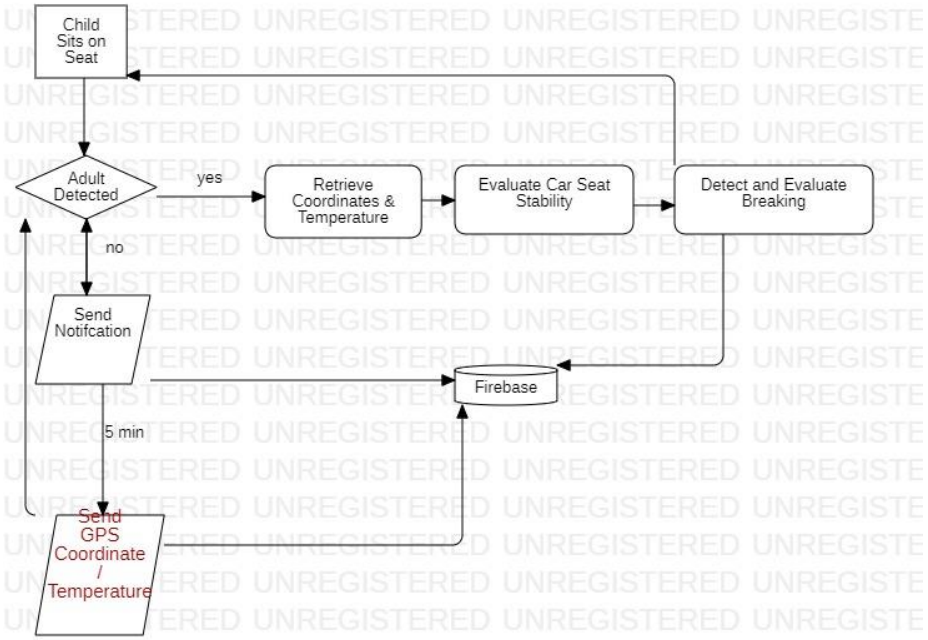


Figure 6: Hardware Flowchart

The sensor mentioned previously are behind the processes in Figure 4. The force sensing resistor is used to detect whether or not a child is place in a seat. The force sensor senses weight over 7lbs (average weight of a newborn) and then goes on to detect adult supervision. The infrared sensor is then used to complete the step; a series of processes will then happen based on whether another person is sensed to be in the vehicle along with the child. In the event that nobody is in the car with the child, then data will be sent to the database. This data will represent a warning, and if the child has been left alone for a significant amount of time, then data will be sent to the database again. At this point, the system is constantly detecting what is happening within the car seat and vehicle, so once someone apprehends the child the data will be sent to the

database with a value representing safety. The data that is being sent in this case is the safety(levels), temperature, and GPS location, so that someone knows exactly where and how to access the circumstances at hand.

On the other hand, if someone is sensed to be in the vehicle with the child, then a series of steps will occur. First location and coordinate will be retrieve using the temperature and GPS module. Next, the stability of the car seat will be evaluate using the accelerometer readings. Then, the car seat will examine the driving by obtaining the miles per hour, turns, and breaking using the GPS module, and accelerometer. Lastly, the data will then be sent to the database, and go back to step 1 of detecting whether the child is in the seat or not.

The stability is evaluating by using 10 reading within the last 10 seconds of the accelerometer. The values are calculated by averaging the X, Y, and Z plane of the accelerometer. The accelerometer is also used to get the degrees of the angle of the turn and the GPS module is used for the speed. As an example, if a driver goes 20mphs to 14mph while making a left turn within roughly a second, then the system will evaluate it as unsafe driving for the child.

Software

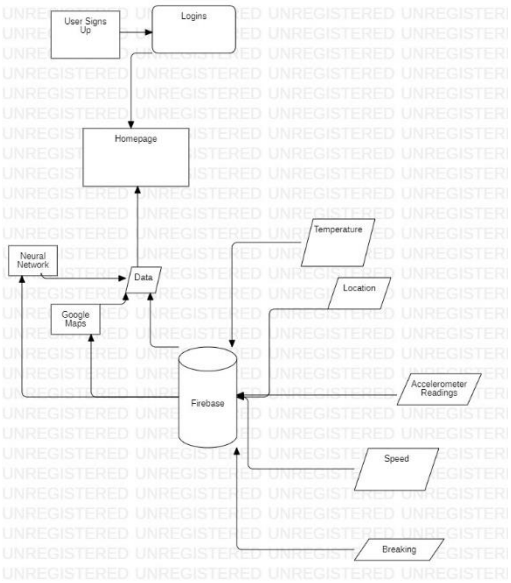
One of the objectives of this project is to consistently keep the parent / caregiver involved in the child's safety. This can be done several ways, such as email, mobile app, text, phone calls, etc. The keyword is "consistency", so user needs to have access to what is happening in real time with the car seat. Also, they need a way to actually view and understand what is happening within the seat as well. The solution to both of these factors is to use a real time database and a web application.

Database

Figure 7: Database Schema

The data needs to be collected in real time, so Firebase will be used to complete this task. The data is sent from the microcontroller and is constantly being updated in real time. The database uses two tables, one table is used for receiving non-sequential data, and the other is used for stability and driving. In the movement table, breaking is set into levels 0 – 3, each value represent the type of breaking; 0 being “safe” and anything above that is considered harsh. The accelerometer is used to get the degrees of the angle of the turn and the GPS module is used for the speed. Below that are series of data from the accelerometer measuring average acceleration.

In the other table, the temperature, speed longitude and latitude is being sent from the microcontroller. The safety levels (0 – 2) , 0 is safe, 1 is warning, 2 emergency; these values are associated with whether or not the child is left alone in the vehicle or not.



Web application

Figure 8: Web App Flow Chart

The main goal of the web application is for the user to interact and understand the data that is being sent over from the Arduino microcontroller. First off, the register for an account, the password is then hashed into a database. Once the user successfully register, they can then login and view what is happening within the system. As soon as the user logins, data is being pulled from the Firebase database, and then is being used to be displayed on the web page. Additionally, the simple data such as the speed and temperature is displayed on the user's home page. The GPS coordinates are put through the Google Maps API and displayed on the map for the user to view. Further, based on the safety level, the user can see if the child is safe or not on the website as well.

Most importantly, there needs to be a way to quickly notify the user even if they are not on the web app. So, for this instance, I will be using emails to send notifications to the user in dangerous situations. This can be done many ways, but emails seem to be the most efficient way to get this task done with the materials and budget at hand. Anyways, the user will gets emails if the child is left alone (GPS location at a certain time) and the driving (evaluating breaking).

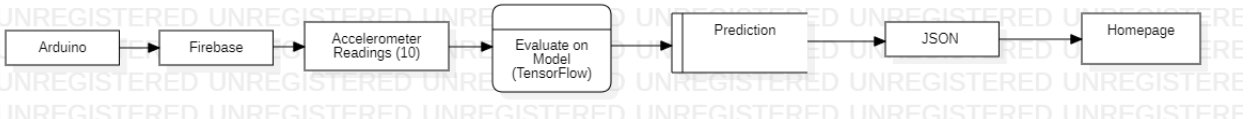


Figure 9: Process of Evaluating Stability

To illustrate, I will categorize stability into 2 categories. One being stable and the other being unstable. For instance, if the car seat isn't moving, then its stable, but if it is wobbly, then it

is unstable. The process starts with the hardware sending 10 recent readings from the accelerometer to Firebase. Those 10 readings are grabbed and are inputs for a neural network model that has been trained to evaluate. The model will then make a prediction, append the prediction to a JSON value and display it on the user homepage.

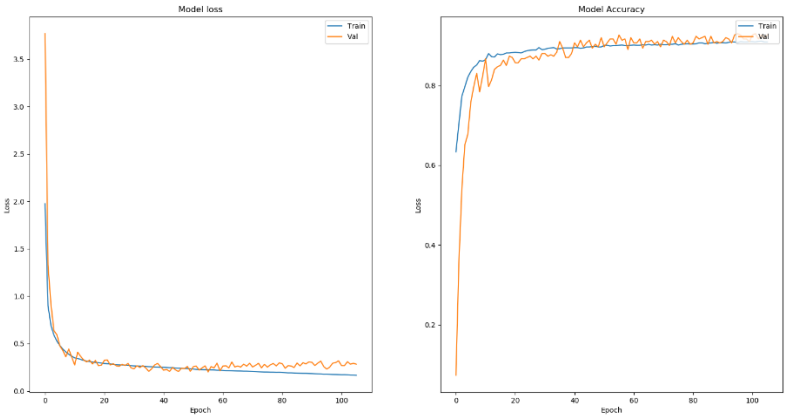


Figure 10: Neural Network Model (Loss / Accuracy)

Critically, a model had to be trained to achieve this goal. There was a series of inputs, with category, as being stable or unstable. This sets a definition for what the model can deem as acceptable. In Figure 7, the training set and validation set loss and accuracy is plotted, and you can see that they perform very well. Essentially, the model is good at making prediction based on the data that is being given.

Testing and Calibration Results

This project was tested at several different levels, each sensor was tested individually until the results were steady and accurate. At that point, the sensor was integrated with another sensor that is pertaining to its subsystem to test again before integrated into the sensor subsystem. For the most part, every sensor was tested in real life scenarios to prove the results were accurate and stable.

Lessons Learned

One of the main lessons I've learned is going through project management. In this project I did things from an agile perspective, which involve planning, designing, developing, releasing, and receiving feedback. This is how I made my decisions when I was going about this project. I've cycled through this agile method several times before I reach a point of acceptability for this project.

Conclusion

To conclude, this device isn't revolutionary, but it is indeed useful and accomplishes the goal. It can help many parents and save many lives with this safety device. It does a great job at monitoring the surrounding environment, making decision based on the environment, and protecting the child. Moreover, this device is very versatile and doesn't need to be constrained to just a car seat. It can be used for several different things such as when someone is trying to move a valuable item and doesn't want it to be damaged. The main priority of the device is ensuring safety and making it easy to use. In sum, this device can help parents monitor their child better and save many lives.

References

Safe kids. (2014, August). *NEW STUDY: 14% OF PARENTS SAY THEY HAVE LEFT A CHILD ALONE INSIDE PARKED VEHICLE DESPITE THE RISKS OF HEATSTROKE.* Retrieved from Safekids.org: https://www.safekids.org/press-release/new-study-14-parents-say-they-have-left-child-alone-inside-parked-vehicle-despite

University of South Florida. (2018, August 9). *Cognitive and Neurobiological Perspectives on Why Parents Lose.* Retrieved from Psychology USF: http://psychology.usf.edu/faculty/data/ddiamond/Research_on_Why_Parents_Forget_Children_in_Hot_Cars.pdf

Appendices

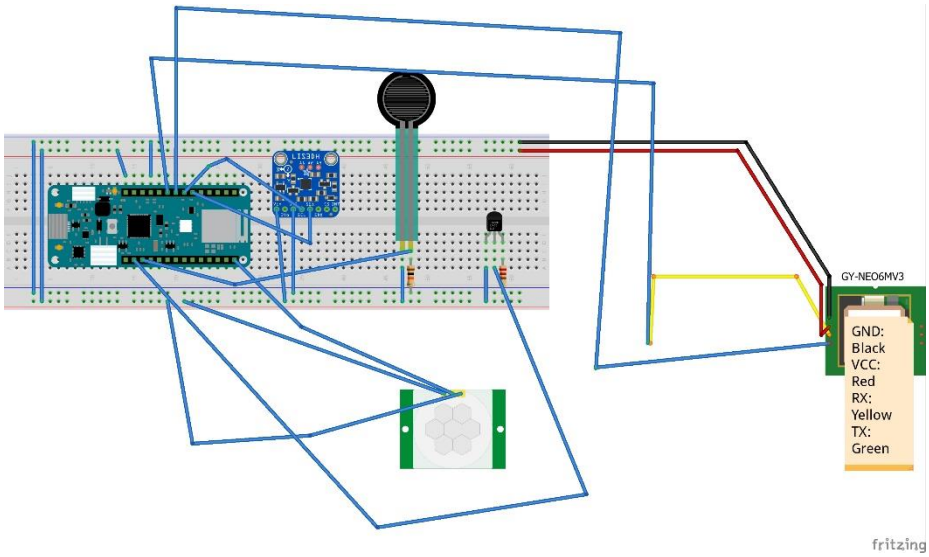


Figure 11 Pin Layout

Arduino Code (Sensors / Hardware)

```
1. Wait #include <TinyGPS++.h>
2.
3. TinyGPSPlus gps;
4.
5. #include "Firebase_Arduino_WiFiNINA.h"
6.
7. // Firebase Credentials
8. #define FIREBASE_HOST "*********”
9. #define FIREBASE_AUTH "*********"
10.     #define WIFI_SSID "*********"
11.     #define WIFI_PASSWORD "*********"
12.
13.
14.     //Define Firebase data object
15.     FirebaseData firebaseData;
16.
17.
18.     // Accelerometer readings from Adafruit LIS3DH
19.
20.     #include <Wire.h>
21.
22.     #include <SPI.h>
23.
```

```
24.     #include <Adafruit_LIS3DH.h>
25.
26.     #include <Adafruit_Sensor.h>
27.
28.     // Used for software SPI
29.     #define LIS3DH_CLK 13# define LIS3DH_MISO 12# define LIS3DH_MOSI 11
30.     // Used for hardware & software SPI
31.     # define LIS3DH_CS 10
32.
33.     Adafruit_LIS3DH lis = Adafruit_LIS3DH();
34.
35.
36.
37.     //PIR Motion Variables
38.     int pirInput = 5; // choose the input pin (for PIR sensor)
39.     int pirState = false; // we start, assuming no motion detected
40.     int pirVal = 0; // variable for reading the pin status
41.
42.     // LED Variable Declaration -- Change to Analog
43.     const int forceLED = 2;
44.     const int motionLED = 3;
45.     const int accLED = 4;
46.     void setup() {
47.         pinMode(A1, INPUT);
48.       // Controls the PIR Sensorinput
49.       pinMode(pirInput, INPUT); // declare sensor as input
50.       // LED Set as outputs
51.       pinMode(forceLED, OUTPUT);
52.       pinMode(motionLED, OUTPUT);
53.       pinMode(accLED, OUTPUT);
54.       pinMode(LED_BUILTIN, OUTPUT);
55.         Serial1.begin(9600);
56.       Serial.begin(115200);
57.          while (!Serial) delay(10); // will pause Zero, Leonardo, etc
   until serial console opens
58.
59.       Serial.println("LIS3DH test!");
60.
61.       if (!lis.begin(0x18)) { // change this to 0x19 for alternative i2c
   address
62.         Serial.println("Couldnt start");
63.         while (1) yield();
64.       }
65.       Serial.println("LIS3DH found!");
66.     digitalWrite(LED_BUILTIN,HIGH);
67.     delay(500);
68.     digitalWrite(LED_BUILTIN,LOW);
69.     delay(120);
70.     digitalWrite(LED_BUILTIN,LOW);
71.       lis.setRange(LIS3DH_RANGE_4_G); // 2, 4, 8 or 16 G!
```

```
72.
73.       Serial.print("Range = ");
74.       Serial.print(2 << lis.getRange());
75.       Serial.println("G");
76.
77.       //  WiFi and Firebase Setup
78.       int status = WL_IDLE_STATUS;
79.       while (status != WL_CONNECTED) {
80.         status = WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
81.         Serial.print(".");
82.         delay(300);
83.       }
84.       Serial.println();
85.       Serial.print("Connected with IP: ");
86.       Serial.println(WiFi.localIP());
87.       Serial.println();
88.
89.       //Provide the autntication data
90.       Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH, WIFI_SSID,
   WIFI_PASSWORD);
91.       Firebase.reconnectWiFi(true);
92.       // Controls the force sensor input
93.       pinMode(A1, INPUT);
94.       // Controls the PIR Sensorinput
95.       pinMode(pirInput, INPUT); // declare sensor as input
96.       // LED Set as outputs
97.       pinMode(forceLED, OUTPUT);
98.       pinMode(motionLED, OUTPUT);
99.       digitalWrite(accLED, HIGH);
100.      digitalWrite(LED_BUILTIN,HIGH);
101.
102.    }
103.
104.    void loop() {
105.       int count = 0;
106.     // Accelerometer Variables
107.      float accXY[] = {0,0, 0,0, 0, 0, 0,0,0, 0};
108.      float accX[] = {0,0, 0,0, 0, 0, 0,0,0, 0};
109.    // Tracks the averages of the X and Y m/s^2
110.      float averageArray = 0;
111.      float averageArrayXY = 0;
112.      float averageArray2 = 0;
113.      float averageArrayXY2 = 0;
114.      // Timer
115.      int waitTime = 0;
116.      int weight = analogRead(A1);
117.      weight /= 37;
118.      if(weight >7){
119.        Serial.println("Child placed on seat");
120.        digitalWrite(forceLED,HIGH);
```

```
121.        // Determine if there is a person in the vehicle
122.        // Need time for them to get into the car after placing the child
   on the seat
123.        delay(5000);
124.        pirVal = digitalRead(pirInput);
125.        // The sensor is active
126.       if(pirVal == HIGH){
127.          pirState= true;
128.        }
129.          while(pirState){
130.            pirVal = digitalRead(pirInput);
131.           if (pirVal == HIGH){
132.           pirState = false;
133.            }
134.           pirVal = digitalRead(pirInput);
135.           Serial.println("Data: "+ pirVal);
136.          Serial.println("DETECTION");
137.          alert(0);
138.          accData();
139.          // Accelerometer Reading when a person is in the car
140.        delay(1600);
141.
142.          }if(pirVal != HIGH){
143.            pirState = false;
144.              digitalWrite(motionLED,HIGH);
145.              Serial.println("No Detection");
146.              while(true){
147.
148.                waitTime +=1;
149.                delay(5000);
150.                Serial.println(waitTime);
151.              if(waitTime > 1 && waitTime < 5  ){
152.                //Send Active Response
153.                alert(1);
154.                int weight = analogRead(A1);
155.                weight /= 37;
156.                pirVal = digitalRead(pirInput);
157.                if(pirVal == HIGH || weight < 7){
158.
159.                    break;
160.                  }
161.                  }else if(waitTime > 5){
162.                // Send emergency response
163.                alert(2);
164.                int weight = analogRead(A1);
165.                weight /= 37;
166.                pirVal = digitalRead(pirInput);
167.                  if(pirVal == HIGH && weight < 7){
168.                        alert(0);
169.
```

```
                        break;
                        }
                    }
                }

            }
        }else{
          // System turns off
      digitalWrite(forceLED,LOW);
      digitalWrite(motionLED,LOW);
      digitalWrite(accLED,LOW);
          delay(100);
          }

    }

    void alert(int i) {
      String path = "/tempandgps";
      String jsonStr;

      float lati = 2;

      while (Serial1.available()) {

        if (gps.encode(Serial1.read())) {
          if (gps.location.isValid()) {
          digitalWrite(LED_BUILTIN, HIGH);
            Serial.println("-----------------------------------");
            Serial.println("----------Begin Set Test-----------");
            Serial.println("-----------------------------------");
            Serial.println();
     // if (gps.speed.isUpdated())
    //  {
            if (Firebase.setInt(firebaseData, path +
   "/Int/driving",gps.speed.mph() )) {
              Serial.println("----------Set result-----------");
              Serial.println("PATH: " + firebaseData.dataPath());
              Serial.println("TYPE: " + firebaseData.dataType());
              Serial.print("VALUE: ");
              if (firebaseData.dataType() == "int")
                Serial.println(firebaseData.intData());
              else if (firebaseData.dataType() == "float")
                Serial.println(firebaseData.floatData());
              else if (firebaseData.dataType() == "boolean")
                Serial.println(firebaseData.boolData() == 1 ? "true" :
   "false");
              else if (firebaseData.dataType() == "string")
                Serial.println(firebaseData.stringData());
              else if (firebaseData.dataType() == "json")
                Serial.println(firebaseData.jsonData());
```

```
218.               Serial.println("---------------------------------");
219.               Serial.println();
220.             } else {
221.               Serial.println("----------Can't set data--------");
222.               Serial.println("REASON: " + firebaseData.errorReason());
223.               Serial.println("---------------------------------");
224.               Serial.println();
225.             }
226.     // }
227.             // SEND THE LONGITUDE COORDINATES
228.
229.             if (Firebase.setInt(firebaseData, path + "/Int/safe", i)) {
230.               Serial.println("----------Set result-----------");
231.               Serial.println("PATH: " + firebaseData.dataPath());
232.               Serial.println("TYPE: " + firebaseData.dataType());
233.               Serial.print("VALUE: ");
234.               if (firebaseData.dataType() == "int")
235.                 Serial.println(firebaseData.intData());
236.               else if (firebaseData.dataType() == "float")
237.                 Serial.println(firebaseData.floatData());
238.               else if (firebaseData.dataType() == "boolean")
239.                 Serial.println(firebaseData.boolData() == 1 ? "true" :
   "false");
240.               else if (firebaseData.dataType() == "string")
241.                 Serial.println(firebaseData.stringData());
242.               else if (firebaseData.dataType() == "json")
243.                 Serial.println(firebaseData.jsonData());
244.               Serial.println("---------------------------------");
245.               Serial.println();
246.             } else {
247.               Serial.println("----------Can't set data--------");
248.               Serial.println("REASON: " + firebaseData.errorReason());
249.               Serial.println("---------------------------------");
250.               Serial.println();
251.             }
252.
253.             if (Firebase.setFloat(firebaseData, path + "/Int/LONG",
   gps.location.lng())) {
254.               Serial.println("----------Set result-----------");
255.               Serial.println("PATH: " + firebaseData.dataPath());
256.               Serial.println("TYPE: " + firebaseData.dataType());
257.               Serial.print("VALUE: ");
258.               if (firebaseData.dataType() == "int")
259.                 Serial.println(firebaseData.intData());
260.               else if (firebaseData.dataType() == "float")
261.                 Serial.println(firebaseData.floatData());
262.               else if (firebaseData.dataType() == "boolean")
263.                 Serial.println(firebaseData.boolData() == 1 ? "true" :
   "false");
264.               else if (firebaseData.dataType() == "string")
```

```
265.               Serial.println(firebaseData.stringData());
266.             else if (firebaseData.dataType() == "json")
267.               Serial.println(firebaseData.jsonData());
268.             Serial.println("------------------------------");
269.             Serial.println();
270.           } else {
271.             Serial.println("----------Can't set data--------");
272.             Serial.println("REASON: " + firebaseData.errorReason());
273.             Serial.println("------------------------------");
274.             Serial.println();
275.           }
276.
277.           // SEND THE LATITIUDE COORDINATES
278.
279.           if (Firebase.setFloat(firebaseData, path + "/Int/LAT",
   gps.location.lat())) {
280.             Serial.println("----------Set result-----------");
281.             Serial.println("PATH: " + firebaseData.dataPath());
282.             Serial.println("TYPE: " + firebaseData.dataType());
283.             Serial.print("VALUE: ");
284.             if (firebaseData.dataType() == "int")
285.               Serial.println(firebaseData.intData());
286.             else if (firebaseData.dataType() == "float")
287.               Serial.println(firebaseData.floatData());
288.             else if (firebaseData.dataType() == "boolean")
289.               Serial.println(firebaseData.boolData() == 1 ? "true" :
   "false");
290.             else if (firebaseData.dataType() == "string")
291.               Serial.println(firebaseData.stringData());
292.             else if (firebaseData.dataType() == "json")
293.               Serial.println(firebaseData.jsonData());
294.             Serial.println("------------------------------");
295.             Serial.println();
296.           } else {
297.             Serial.println("----------Can't set data--------");
298.             Serial.println("REASON: " + firebaseData.errorReason());
299.             Serial.println("------------------------------");
300.             Serial.println();
301.           }
302.
303.           break;
304.         }
305.         // Delay between updates
306.         delay(2000);
307.       }
308.     }
309.
310.     // TEMP
311.     float temperature = 0.0; // stores the calculated temperature
312.     int sample; // counts through ADC samples
```

```
313.      float ten_samples = 0.0; // stores sum of 10 samples
314.
315.      // take 10 samples from the MCP9700
316.      for (sample = 0; sample < 10; sample++) {
317.        // convert A0 value to temperature
318.        // convert A0 value to temperature
319.            temperature = ((float)analogRead(A0) * 5.0 / 1024.0) - 0.5;
320.            temperature = temperature / 0.01;
321.            // sample every 0.1 seconds
322.        // sample every 0.1 seconds
323.        delay(250);
324.        // sum of all samples
325.        ten_samples = ten_samples + temperature;
326.      }
327.      // get the average value of 10 temperatures
328.      temperature = ten_samples / 10.0;
329.      // send temperature out of serial port
330.      ten_samples = 0.0;
331.
332.      // PUSH TEMP
333.
334.      if (Firebase.setInt(firebaseData, path + "/Int/TEMP", temperature))
     {
335.        Serial.println("----------Set result-----------");
336.        Serial.println("PATH: " + firebaseData.dataPath());
337.        Serial.println("TYPE: " + firebaseData.dataType());
338.        Serial.print("VALUE: ");
339.        if (firebaseData.dataType() == "int")
340.          Serial.println(firebaseData.intData());
341.        else if (firebaseData.dataType() == "float")
342.          Serial.println(firebaseData.floatData());
343.        else if (firebaseData.dataType() == "boolean")
344.          Serial.println(firebaseData.boolData() == 1 ? "true" :
     "false");
345.        else if (firebaseData.dataType() == "string")
346.          Serial.println(firebaseData.stringData());
347.        else if (firebaseData.dataType() == "json")
348.          Serial.println(firebaseData.jsonData());
349.        Serial.println("------------------------------");
350.        Serial.println();
351.      } else {
352.        Serial.println("----------Can't set data--------");
353.        Serial.println("REASON: " + firebaseData.errorReason());
354.        Serial.println("------------------------------");
355.        Serial.println();
356.      }
357.
358.    digitalWrite(LED_BUILTIN, LOW);
359.
360.    }
```

```
361.
362.
363.    void accData(){
364.
365.      String path = "/accData";
366.      String jsonStr;
367.
368.      Serial.println("-----------------------------------");
369.      Serial.println("----------Begin Set Test-----------");
370.      Serial.println("-----------------------------------");
371.      Serial.println();
372.
373.    float x = 0;
374.    float userSpeed = 0;
375.    float breakSpeed = 0;
376.    int count = 0;
377.    int breakFlag=0;
378.    float acc[] = {0,0,0,0,0,0,0,0,0,0};
379.    float pi = 3.1415926535;
380.    float degree_val = 0;
381.     float averageVal = 0;
382.     sensors_event_t event;
383.     lis.getEvent(&event);
384.     float angleX = event.acceleration.x ;
385.     float angleY = event.acceleration.y ;
386.     float angleZ = event.acceleration.z ;
387.     float angleX2 = atan(angleX / sqrt(pow(angleY,2) + pow(angleZ,2) ));
388.     float degreeX = (angleX2 / pi)  * 180;
389.
390.    for(int i = 0; i < 10; i++){
391.     sensors_event_t event;
392.     lis.getEvent(&event);
393.      float valX = event.acceleration.x ;
394.     float valY = event.acceleration.y ;
395.     float valZ = event.acceleration.z ;
396.     averageVal = sqrt(pow(valX,2) + pow(valY,2) + pow(valZ,2));
397.     acc[i] = averageVal;
398.     delay(400);
399.      }
400.     if(x > 20){
401.     // Serial.println("REACHED OVER 20 MPH");
402.      userSpeed = x;
403.     // Serial.println(userSpeed);
404.      for(int i = 0; i < 3; i++){
405.        breakSpeed = mphRead();
406.        if(breakSpeed < userSpeed -4){
407.            count++;
408.          //  Serial.print("Count increased");
409.          //  Serial.println(breakSpeed);
410.          }
```

```
411.         delay(300);
412.         }
413.        if(count > 1){
414.         Serial.println("HARD BREAK");
415.             if(degreeX < -13){
416.           Serial.println("RIGHT TURN");
417.            breakFlag = 3;
418.           }
419.          else if(degreeX > 5){
420.           Serial.println("LEFT TURN");
421.            breakFlag = 2;
422.           }
423.          else{
424.           Serial.println("STRAIGHT");
425.            breakFlag = 1;
426.           }
427.
428.         }else{
429.         breakFlag = 0;
430.         Serial.println("SOFT BREAK");
431.           }
432.       }
433.
434.
435.         if (Firebase.setInt(firebaseData, path + "/Int/break" ,
    breakFlag))
436.         {
437.           Serial.println("----------Set result----------");
438.           Serial.println("PATH: " + firebaseData.dataPath());
439.           Serial.println("TYPE: " + firebaseData.dataType());
440.           Serial.print("VALUE: ");
441.           if (firebaseData.dataType() == "int")
442.             Serial.println(firebaseData.intData());
443.           else if (firebaseData.dataType() == "float")
444.             Serial.println(firebaseData.floatData());
445.           else if (firebaseData.dataType() == "boolean")
446.             Serial.println(firebaseData.boolData() == 1 ? "true" :
    "false");
447.           else if (firebaseData.dataType() == "string")
448.             Serial.println(firebaseData.stringData());
449.           else if (firebaseData.dataType() == "json")
450.             Serial.println(firebaseData.jsonData());
451.           Serial.println("------------------------------");
452.           Serial.println();
453.         }
454.         else
455.         {
456.           Serial.println("----------Can't set data--------");
457.           Serial.println("REASON: " + firebaseData.errorReason());
458.           Serial.println("------------------------------");
```

```
459.          Serial.println();
460.        }
461.
462.      for (uint8_t i = 0; i < 10; i++)
463.      {
464.
465.
466.
467.        if (Firebase.setFloat(firebaseData, path + "/Int/Data" + (i),
    acc[i]))
468.        {
469.          Serial.println("----------Set result-----------");
470.          Serial.println("PATH: " + firebaseData.dataPath());
471.          Serial.println("TYPE: " + firebaseData.dataType());
472.          Serial.print("VALUE: ");
473.          if (firebaseData.dataType() == "int")
474.            Serial.println(firebaseData.intData());
475.          else if (firebaseData.dataType() == "float")
476.            Serial.println(firebaseData.floatData());
477.          else if (firebaseData.dataType() == "boolean")
478.            Serial.println(firebaseData.boolData() == 1 ? "true" :
    "false");
479.          else if (firebaseData.dataType() == "string")
480.            Serial.println(firebaseData.stringData());
481.          else if (firebaseData.dataType() == "json")
482.            Serial.println(firebaseData.jsonData());
483.          Serial.println("--------------------------------");
484.          Serial.println();
485.        }
486.        else
487.        {
488.          Serial.println("----------Can't set data--------");
489.          Serial.println("REASON: " + firebaseData.errorReason());
490.          Serial.println("--------------------------------");
491.          Serial.println();
492.        }
493.      }
494.
495.      }
496.
497.
498.    float mphRead(){
499.        while (Serial1.available()) {
500.            if (gps.encode(Serial1.read())) {
501.
502.    //        if(gps.speed.isValid()){
503.             // Serial.print(gps.speed.kmph());
504.              return gps.speed.mph();
505.    //        }
```

```
506.     }  }}
```

Figure 12: Arduino Code

```
from django.test import TestCase
import functools
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.models import load_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import
Dense,Activation,Flatten,Conv2D,MaxPooling2D,Dropout
from tensorflow.keras.optimizers import SGD, Adam
from tensorflow.keras import regularizers
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
import pyrebase
import matplotlib.pyplot as plt


def seatSafety():
    Config = {
            "apiKey": "**********",
            "authDomain": "**********",
            "databaseURL": "**********",
            "projectId": "**********",
            "storageBucket": "**********",
            "messagingSenderId": "**********",
            "appId": "**********"
        }

    firebase = pyrebase.initialize_app(Config)
    db = firebase.database()
    drivingData = []
    accData = []
    for i in range(10):
        data = db.child('accData/Int/Data' + str(i)).get().val()
        #print(data)
        drivingData.append(data)


    accData.append(drivingData)
    accelerometerarr = np.asarray(accData)
    print(accelerometerarr)



    TRAIN_DATA_URL = "**********
    TEST_DATA_URL = '**********’
    csv_test_url = TEST_DATA_URL.replace('/edit#gid=', '/export?format=csv&gid=')
    csv_export_url = TRAIN_DATA_URL.replace('/edit#gid=', '/export?format=csv&gid=')
```

```
    test_df = pd.read_csv(csv_export_url,sep=",")
    test_df['Label'] = pd.Categorical(test_df['Label'])
    test_df['Label'] = test_df.Label.cat.codes
    test_x = test_df.drop(columns=['Label'])
    test_df.head()


    array = pd.DataFrame(accelerometerarr,columns =
['Data1','Data2','Data3','Data4','Data5','Data6','Data7','Data8','Data9','Data10'])
    array.to_csv('../models/data.csv')
    test_x_2 = pd.read_csv('../models/data.csv', index_col=0)
    test_x_2.head()

    df = pd.read_csv(csv_export_url,sep=",")
    df['Label'] = pd.Categorical(df['Label'])
    df['Label'] = df.Label.cat.codes
    df.head()

    train_X = df.drop(columns=['Label'])
    train_X.head()

    train_y = df[['Label']]
    train_y.head()
    test_y = test_df[['Label']]
    y_test_c = to_categorical(test_y,2)
    y_Train_c = to_categorical(train_y,2)
    n_cols = train_X.shape[1]

    #
    # model = Sequential()
    # model.add(Dense(10,activation='relu', input_shape=(n_cols,)))
    # model.add(Dense(22, activation='relu'))
    # model.add(Dense(2, activation='softmax'))
    #
model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.0001),metrics=['acc
'])
    # early_stopping_monitor = EarlyStopping(monitor='val_loss', mode='min',
verbose=1, patience=30)
    # mc = ModelCheckpoint('../models/best_model.h5', monitor='val_loss', mode='min',
save_best_only=True)
    # history = model.fit(train_X, y_Train_c, validation_split=0.2, epochs=150,
callbacks=[early_stopping_monitor,mc],batch_size=10)
    #model.save('../models/CarData2.h5')
    model = load_model('../models/best_model.h5')
    #model.summary()
    predictions = model.predict_classes(test_x_2[:])
    # print(array[:1])
    # 0 means safe and 1 means unsafe
    #print('predictions:', predictions)
    #results = model.evaluate(test_x, y_test_c, batch_size=10)
    #print('Test Loss , Test Acc: ',results)
    # plt.subplot(1,2,1)
    # plt.plot(history.history['loss'])
    # plt.plot(history.history['val_loss'])
    # plt.title('Model loss')
```

```
    # plt.ylabel('Loss')
    # plt.xlabel('Epoch')
    # plt.legend(['Train', 'Val'], loc='upper right')
    # plt.subplot(1,2,2)
    # plt.plot(history.history['acc'])
    # plt.plot(history.history['val_acc'])
    # plt.title('Model Accuracy')
    # plt.ylabel('Loss')
    # plt.xlabel('Epoch')
    # plt.legend(['Train', 'Val'], loc='upper right')
    # plt.show()
    return int(predictions)
```

Figure 13: Tensorflow Model

Organization Chart

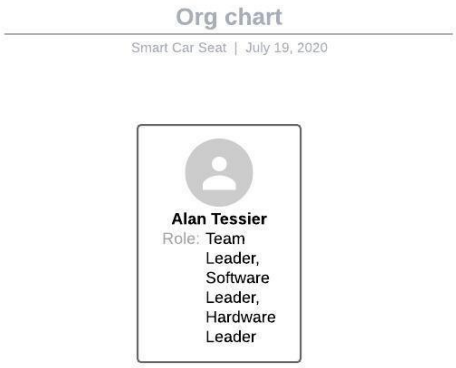


Figure 14 Organization Chart