**MINI EXPLORER PROJECT**

**Duc Nguyen, Abiodun Ojo, Aldous Mendoza**

**TABLE OF CONTENTS**

1. Introduction
2. Software Design and Specifications
3. Database Design
4. Hardware Design
5. Android Application
6. Project Overview

VII.1 Expenses

VII.2 Project Schedule

1. Contribution
2. Conclusion
3. References

**Abstract**

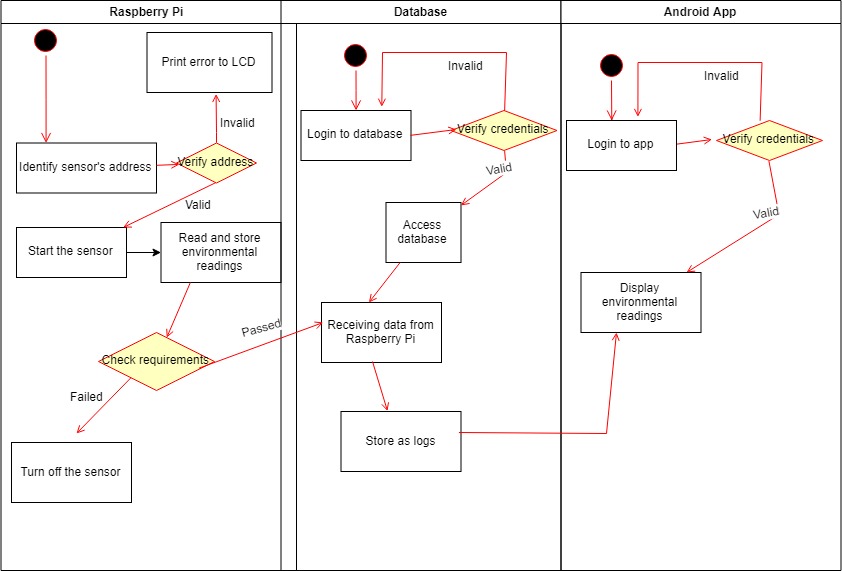
Taking advantage of technology to perform tasks that seem harmful to humans is what drives our creativity and innovation forward. Such one task is to evaluate environment conditions in certain surroundings. Some places are unreachable and thus, require small devices with enough functionalities and specifications to reach such locations. Mini Explorer Project is born to perform such activities: Approach small places (with slippery surfaces) and obtain environment readings from them. This project includes extensive research on certain hardware, software and database integration from multiple different sources. For the purpose of this report, we will only focus mainly on the software aspect.

**Declaration of Joint Authorship**

We, Duc Nguyen, Abiodun Ojo and Aldous Mendoza confirm that this work submitted for assessment is the combined work of ourselves, interpreted in our own expressions. Any external usages of any other individuals, groups or organizations, including but not limited to, programs, texts, documents, figures, explanations, are properly referenced and acknowledged.

**I. Introduction**

Mini Explorer consists of three main parts: Hardware, software, and database. The hardware backbone is the Raspberry Pi 3B+ Model, which acts as a central processing unit for every function. Along with it are three sensors: HDC1008 (Temperature and Humidity Sensor), PCA9685 (16-channel 12-bit PWM/Servo driver) and VL53L0X (Time of Flight Distance Sensor). The software is entirely written on Android Project, as the Mini Explorer is expected to be entirely controlled through the phone via Bluetooth. The database is stored on Firebase - a cloud service by Google. This is where all users’ information, environmental readings, device settings and logs are stored. All three main parts are interconnected and closely related. For visual purposes, this UML diagram will give a more thorough understanding of the whole system.



*Figure 1 - UML Diagram of Mini Explorer*

**II. Software Design**

**II.1 Raspberry Pi SD card:** Have your SD Card installed by following the instructions here: <https://github.com/six0four/StudentSenseHat/blob/master/cribpisdcard.md>

**II.2** **Raspberry Pi Car:**

1. Get source code: Download the source code by *git clone:*

*git clone --recursive* [*https://github.com/sunfounder/SunFounder\_PiCar-S.git*](https://github.com/sunfounder/SunFounder_PiCar-S.git)

2. Install python-smbus: *sudo apt-get install python-smbus -y*

3. Install PiCar module:

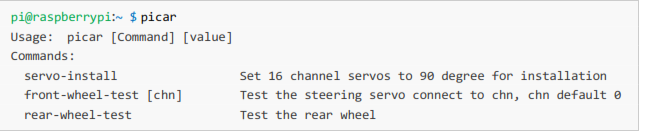
*git clone --recursive https://github.com/sunfounder/SunFounder\_PiCar.git*

*cd SunFounder\_PiCar*

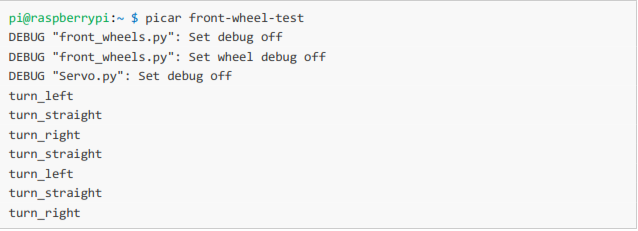
*python setup.py install*

4. Type picar:

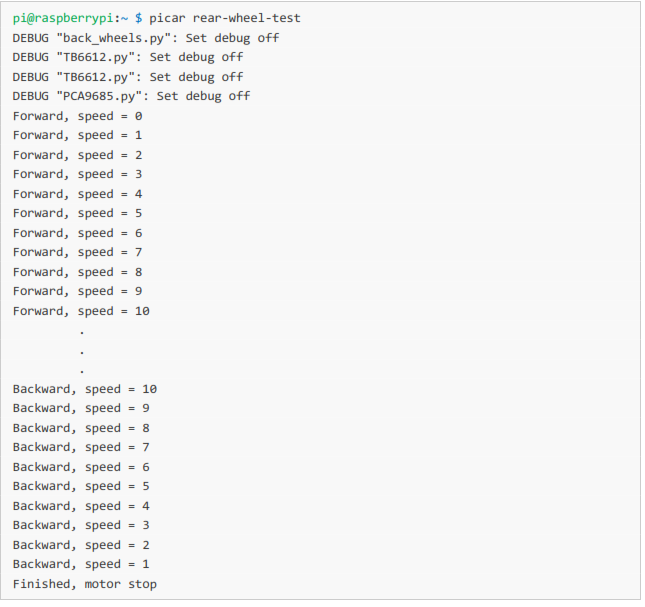
4.1 Servo Installation: The first one **servo-install** is for **servo adjustment**, which must be used after the front wheels are assembled.



4.2 Front Wheel Test: This is to test if the front wheel are able to steer left or right.



4.3 Rear Wheel Test: This is to test if the backwheels are able to rotate and make the car run



**II.3 Sensors:**

1. HDC1008: Originally retrieved from <https://github.com/ngtrangminhduc/OverheatSensor/blob/master/2018_SourceCode/hdc1008.py>

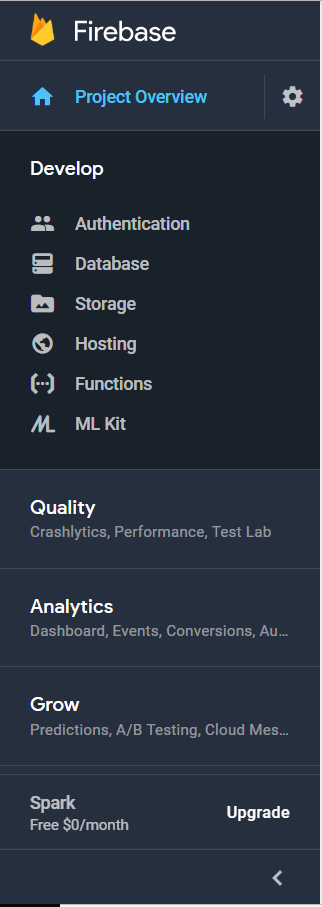
2. VL53L0X: Originally retrieved from <https://github.com/AldousMendoza/ObstructionSensor/tree/master/ProjectDocumentation/O-Sensor>

3. 16-channel PWM PCA9685: Originally retrieved from https://github.com/adafruit/Adafruit\_Python\_PCA9685/blob/master/examples/simpletest.py

**III. Database Design**

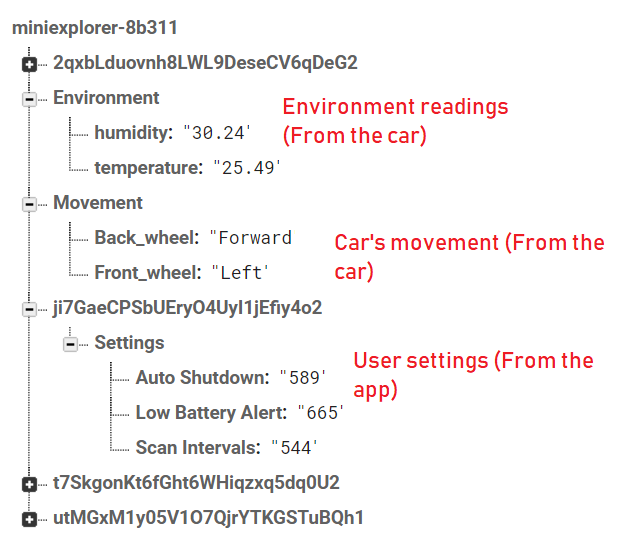
The database is designed and developed in Firebase platform. In order to use this technology, a Google Mail is required, so we decided to create an unique email solely for this project. Firebase has many utilities, but as of now, only Authentication and Database are used.

Authentication is where all the users’ email and passwords are stored. As soon as a registration is made successfully, they will be sent to Firebase and assigned with their unique user IDs. This is very beneficial to monitor individual’s activities and readings. Whenever a login attempt is made from the app, it will trigger the methods in Login Screen to scan for that specific email and password in here. Authentication also has the email address verification feature, which will be used in the future.



*Figure 2 - Snapshot of Firebase capabilities*

The Database contains different entries of each user that are sent from the App. It also contains two separate data structures: Environment (consists of real-time environmental readings) and Movement (which is retrieved from the App by the user)



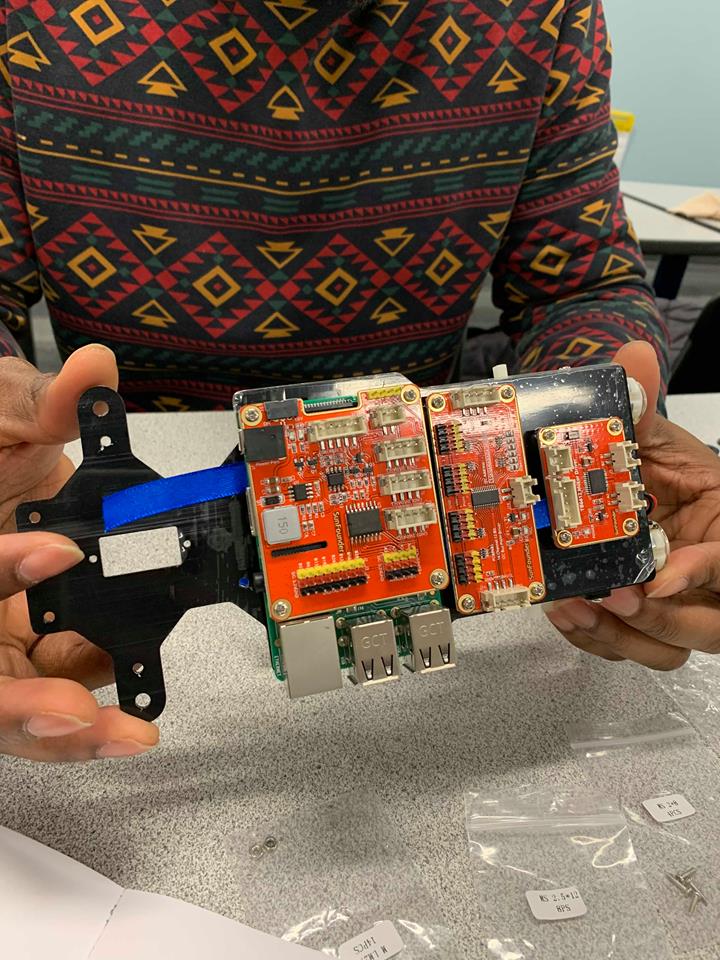
*Figure 3 – Database Structure*

**IV. Hardware Design**

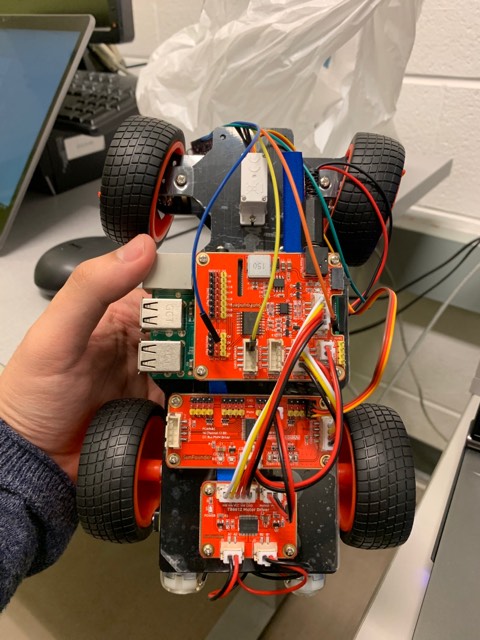
Detailed build instructions and software specifications are contained within “PiCar - User\_Manual.docx”. For convenient purposes, these are the pictures that briefly describes the process of assembling everything together.



*Figure 1 – All components*



*Figure 2 – Main components, without the tires*



*Figure 3 – The complete car*

**V. Android Application**

V.1 Android Application Design

The software is to be developed by two people with no prior experience in mobile programming / development within nearly three months. With such resources, the software design is fairly simple. It consists of two main elements: Java Front-end / Back-end development and database development. While they are not entirely dependent on each other, they have the same priority level, as both of them are crucial to the application and can be developed at the same time.

The Java Front-end determines the appearance of all elements contained within the app, and the user experience the app is expected to bring. It includes, but not limited to, creating and manipulating icons, layouts, text, buttons. As of current, the app contains 8 different screens, each with its own way of design so as to serve its functions. The app also uses 13 different, specifically-customized icons ; 41 text strings, available in both English and French.

The Java Back-end determines how the app is going to be executed by the users. It consists of, but not limited to, authentication, intents, events and is used in all screens. The screens and their functionalities are:

1. Splash screen: Acts as a launcher, will go to the login screen after 0.5 seconds of showing the app’s icon.

2. Login screen: Prompts the user to enter their email and password, and then check with the database. If the typed information is not correct, then the user cannot log in. Also has a intent to display the register screen if the user wished to.   
 3. Registration screen: Allows users to register with email and password. Users must enter a legit email format, and confirm their password to register successfully. Data is then sent to database.

4. Main menu screen: Shows all other main screens that users can go to. Clicking on the live support will not take you to a new screen, but instead, redirect you to the call activity of your phone.

5. Reading screen: Displays the temperature and humidity retrieved from database

6. Explorer screen: Shows the car’s control and power state. Users can control the car’s movement and power in this screen.

7. Setting screen: Displays the language option, and certain settings that can be applied to the hardware (low-battery alert, scan intervals, and auto-shutdown timer).

8. About us screen: Displays the developers’ names.

Listing and purposes of functions/methods being used will be discussed in part VIII.

The database development is started on the second-half of the project timeline. The database is stored on Firebase - a free Google cloud service - which is integrated in Android Studio as well. The database design will also be discussed on part V.

The requirements for the software are rather straightforward. The software should not crash or contain any bugs at all times, is expected to deliver its full functionalities, which include:

* Execute all tasks properly and flawlessly without bugs / crashes.
* Being able to register and login / logout users.
* Having access to all screens.
* Being able to retrieve environment readings from database.
* Being able to control the Mini Explorer (its power state, its movement and directions) from the maximum distance of 70 meters.
* Being able to adjust certain settings to hardware (Low battery alert, scan intervals, auto shut-down)
* Being able to call live support.

V.2 Software Specifications

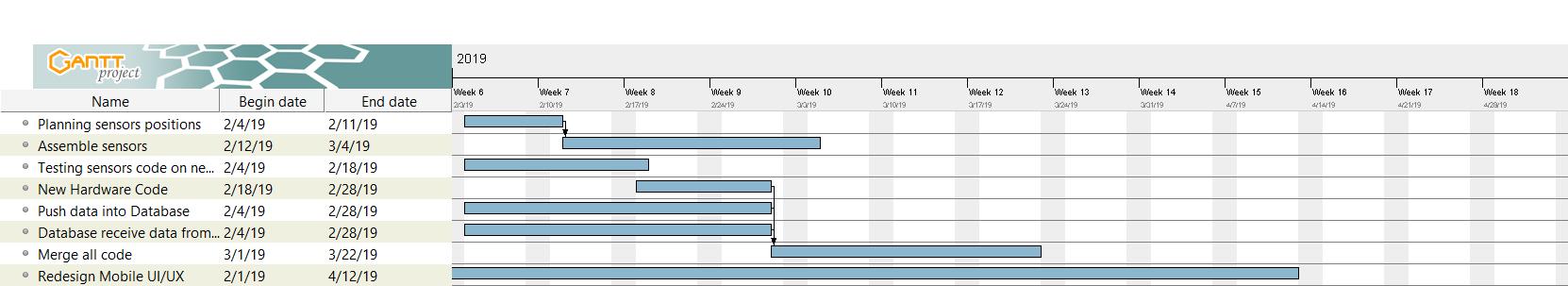
The app is intended to run on most Android devices, ranging from API 21 to API 28, which covers more than half of the Android devices today. Therefore, the app implements the Android appcompat, design and layout with 28.0.0 version. It also implements Firebase libraries with version 16.0.0. The app supports different screen sizes, so the only element to consider is the mobile’s API version.

**VI. Project Overview**

VI.1 Expenses

|  |  |  |
| --- | --- | --- |
| Name | Manufacturer / Provider | Price |
| SunFounder Raspberry Pi Car | SunFounder / Amazon | $118 |
| Raspberry Pi | Canakit / Amazon | $100 |
| Lithium Batteries + Charger | Amazon | $18.99 |
| HDC1008 Sensor | Ebay | $10 |
| 16-channel PWM/Servo Driver | Adafruit | $19.44 |
| VL53L0X Laser Distance Sensor | Adafruit | $22.4 |
|  | Total | $288.23 |

VI.2 Schedule



**VII. Contribution**

**VIII. Conclusion**

**IX. References**