1. **Proposed solution**

The possible solutions for our system are listed with their properties. Then our solution is presented with the justification.

* 1. **Possible solutions and tradeoffs**

The system has a software and hardware part. Each of these parts has its own possible solutions.

**Hardware possible solutions**

* Comparison between different sensors

There are many different sensors that can be used for detecting the presence of vehicles which are suitable for our smart parking system. One of them is the RFID (Radio Frequency Identification) method/technique/approach which comes with a reader and a tag. The idea is to use radio frequencies for detection and identification of each vehicle by installing an RFID tag on it. Another approach is to use an Infrared sensor which can detect a vehicle. This is done by detecting the reflected infrared light waves that the sensor emits. A third method is to use an Optical sensor that uses light rays that are converted into electrical signals for detection. Additionally, an Ultrasonic sensor can be used which emits and detects any reflected sound waves. Finally, a camera with image processing applied can be used for detecting vehicles.

In [18], a study was conducted which compared the different types of sensors used in existing approaches. A short version of their table is provided in Table 4-1 below.

**Table** Error! No text of specified style in document.**‑1: Comparison between different sensors**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Flexible | Small Size | Installation | Accuracy | Cost |
| Infrared |  |  | \* | \*\* | \* |
| Ultrasonic |  |  | \* | \*\*\* | \* |
| Camera |  |  | \* | \*\* | \*\*\* |
| Optical |  |  | \* | \*\* | \* |
| RFID |  |  | \*\* | \*\*\* | \*\* |

* Comparison between different development boards

Arduino and Raspberry Pi development boards can both be used as part of our solution. In [19], a comparative table was made between the two. A short version of their table is provided in Table 4.2 below.

|  |  |  |
| --- | --- | --- |
| SL | Raspberry Pi | Arduino |
| 1 | It is a mini computer with Raspbian OS. It can run multiple programs at a time. | Arduino is a microcontroller, which is a part of the computer. It runs only one program again and again. |
| 2 | It is difficult to power using a battery pack. | Arduino can be powered using a battery pack. |
| 3 | It requires complex tasks like installing libraries and software for interfacing sensors and other components. | It is very simple to interface sensors and other electronic components to Arduino. |
| 4 | It is expensive. | It is available at a low cost. |
| 5 | The processor used is from ARM family. | Processor used in Arduino is from AVR family Atmega328P. |

**Table** Error! No text of specified style in document.**‑2: Comparison between Raspberry Pi and Arduino**

**Possible software solutions**

* Comparison between services

Availability and Reservations are the two main services that can be provided to the user. Availability is mainly about showing the user a map that shows if a parking spot is available. Reservation allows the user to reserve a parking spot before coming to the parking area.

The two main criteria that we are going to use to compare between the two services are flexibility, and guarantee of a parking spot.

Regarding flexibility, the availability service is more flexible than the reservation because whenever the user parks, he can stay in the parking for as long as he wants. However, in the case of reservation, the user should leave whenever his reserved time is over. Users of the Reservation service guarantee the availability of parking spots once they successfully place a reservation. However, in the Availability service, users may not find the parking spot available as they can be occupied at the time of arrival.

|  |  |  |
| --- | --- | --- |
| Services | Flexibility | Guarantee of available parking spot |
| Availability |  | X |
| Reservation | X |  |

**Table** Error! No text of specified style in document.**‑3: Comparison between services**

Regarding the reservation period/duration, there are several options that can be taken into consideration, but they vary in realism and defects. As mentioned in Table 4-4, it is possible to extend with hour-based reservation whilst it is not with day-based reservation. Using the day-based reservation approach, the parking spots will not be utilized to the maximum, as opposed to the hour-based reservation. For instance, if a user wanted to reserve a parking spot for one hour in a day-based reservation system, he will have to reserve the whole day which will waste a parking spot for a day. This also means the system cannot serve the maximum number of users possible.

**Table** Error! No text of specified style in document.**‑4: Comparison between reservation duration options**

|  |  |  |  |
| --- | --- | --- | --- |
| Reservation Duration | Extension possibility | Serve maximum number of users | Resources  utilization |
| Per hour | Possible |  |  |
| Per day | Not Possible | X | X |

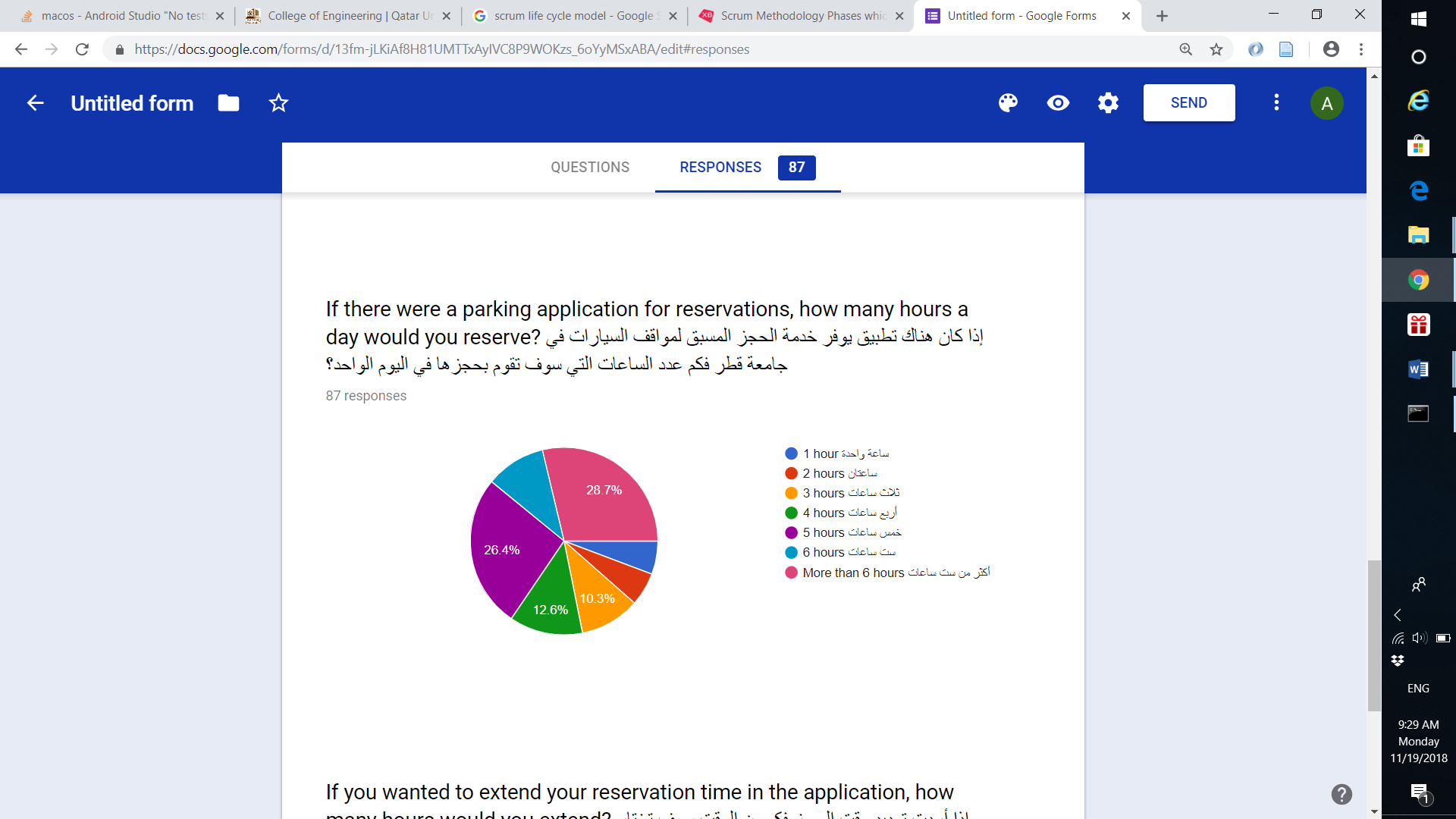
* 1. **Proposed solution**

**Overview**

After surveying the above options regarding the sensors, microcontrollers and services, we combined several components to give the best solution we think will satisfy the goals of the project. The proposed solution is a hybrid solution consisting of two main modules: A Reserved Parking module and a Reservation Free Parking module. The reasoning behind our design decisions will be provided throughout this section.

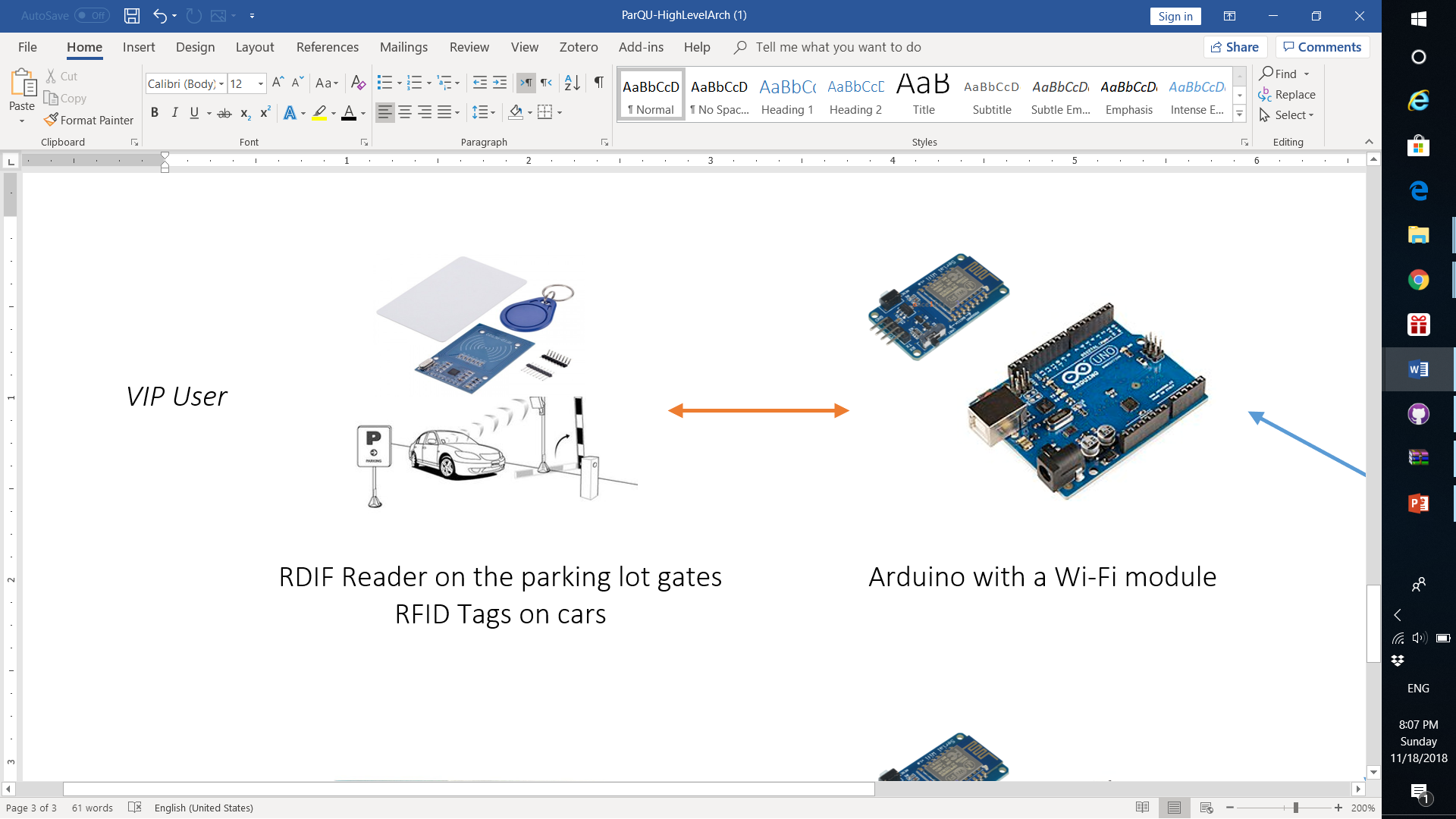
The modules will provide the user with the two main services discussed above: reservation and availability. However, each of these services along with some additional services will be given according to the type of user using our system. The system acknowledges two types of users: VIP users and Normal users. On one hand, Normal users do not require registration nor any type of information from the user. On the other hand, VIP users require subscription registration and installation of an ID tag to be able to enter the parking area. Normal users will be given the availability service whereas VIP users will be given the availability, reservation and request car care services. The reservation includes additional services such as extension and cancellation. In contrast, the Availability service gives Normal and VIP users the ability to view the current parking area status by displaying the parking area map with a status shown for each parking spot.

To help us make some design decisions, we collected information about the user needs in our system by conducting a survey for QU students. To establish a suitable number of hours to reserve per day, we found from the survey, as shown in Figure 4-1, that on average the most appropriate number of hours to reserve per day is 6 hours. To account for exceptions, our system also provides reservation extension service in case there is a vacant parking spot at the time of the extension. Adding to the flexibility of our system, a VIP user is also allowed to cancel a reservation as late as one hour before the start of the reservation time. ((صغري الصور الواضحه نفس مالت السيرفي

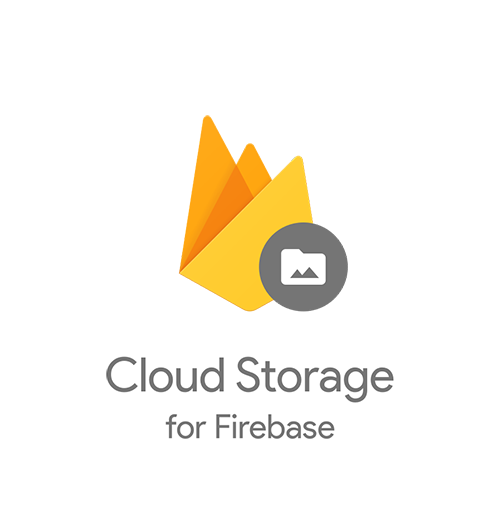


**Figure** Error! No text of specified style in document.**‑1: Statistics from Survey**

**High level architecture (NEW high level Arch must be added)**

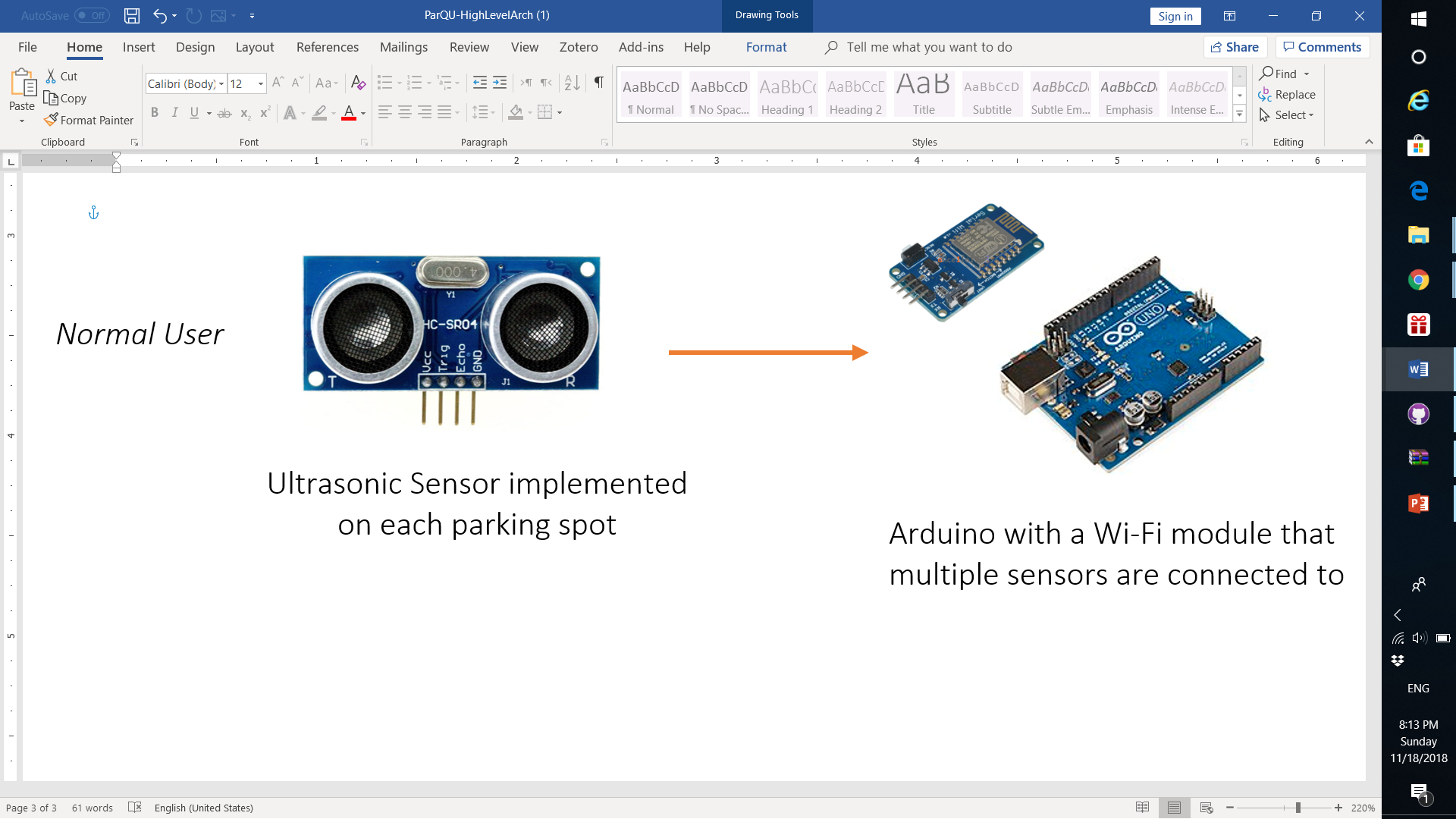


Reserved Parking Module





Reservation Free Parking Module



Internet

Wi-Fi

Serial Communication

* Reserved Parking Module

In this module the user is required to reserve through the mobile application or website. Whenever the user arrives at the gate of the parking area, the RFID reader will read the RFID tag and send the information to Arduino. The Arduino will send the acquired data via Wi-Fi to the (Firebase) cloud database service. The application running on the Arduino module will check the response received from Firebase and if there is a reservation associated with this tag at this date and time. If the response is verified, then the Arduino will order the Servo motor to open or close accordingly. The connection between RFID sensor and Arduino is serial.

* Reservation Free Parking Module

On the other hand, the main purpose of this module is to let the users know if there are available parking spots in a certain zone through a mobile application or website. The application will display the current parking area status in real time by showing a map. Whenever a car is parked in an available space, the Ultrasonic sensor will detect it and inform the Arduino which will then update the Firebase. The connection between Ultrasonic sensor and Arduino is serial as well.

Both modules have a Wi-Fi connection with Firebase whilst the application or website is connected to the Firebase through the internet.

* 1. **Hardware/software to be used**

**Hardware components**

**Table** Error! No text of specified style in document.**‑5: Hardware components with descriptions**

|  |  |  |  |
| --- | --- | --- | --- |
| Component Name | Description | Price | Image |
| Arduino Uno Rev 3 | Arduino Uno is a microcontroller board. Arduino Uno can be programmed to perform tasks. In the system , it is used to process the information given from the sensors and RFID reader and pass it on to NodeMCU. | 115 QR | rduino Uno Rev3  **Figure** Error! No text of specified style in document.**‑2: Arduino Uno** |
| NodeMCU V2 Model ESP8266 | NodeMCU V2 is an open source IoT platform which is able to connect to Wi-Fi. The chip is also able to perform tasks with a certain program. NodeMCU is used to connect and update to Firebase database. | 44 QR | https://res.cloudinary.com/www-arduiely-com/image/upload/v1477989507/ejc2rdha8cphsxxyrulr.jpg  **Figure** Error! No text of specified style in document.**‑3: NodeMCU model ESP8266** |
| Ultrasonic Sensor - HC-SR04 | Ultrasonic sensor to detect whether a car is parked or not | 17 QR | https://res.cloudinary.com/www-arduiely-com/image/upload/v1477387012/hbo10xrcxwqwcqwgfl6e.jpg  Figure ‎4‑4: Ultrasonic sensor - HC-SR04 |
| RFID Kit - MFRC522 | RFID Kit that includes an RFID Reader module, round tag, and then RFID card. The RFID tags are put on the cars as an identification, then the RFID reader is used to read the tags and identify cars at the gates of the parking lot. | 35 QR | FID kit  Figure ‎4‑6: RFID kit |
| DPDT Self Locking Button | Self-locking button to switch between zones (As it will be explained in section 5) | 5 QR | ../../../../../Desktop/Senior%20Project/6_pin_pushButton.jp |
| LEDs (Red and green) | LEDs to indicate the state of the parking lot gate. Red as closed and green as opened. | 2 QR | ../../../../../Desktop/Senior%20Project/Hardware%20Arch%20-%20Senior/51eLYXtROlL._SX5 |
| 9V to Barrel Jack Adapter | This cable is used to battery-power the Arduino that needs 9V and has an on-board barrel jack | 4 QR | V to Barrel Jack Adapter  Figure ‎4‑7: 9V to barrel jack adapter |
| Full-Size Breadboard and Jumper Wires | For the connections between the different parts and Arduino, a full-size breadboard and jumper wires are used. | 20+20 QR | https://res.cloudinary.com/www-arduiely-com/image/upload/v1477354013/zpip78vzw7yclkwfzbwq.jpg  Figure ‎4‑8: Full-size breadboard and jumper wires |

**Software components**

**Table** Error! No text of specified style in document.**‑6: Software components and descriptions**

|  |  |  |
| --- | --- | --- |
| Software | Image | Description |
| Firebase | C:\Users\Alaa\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\CEBC0621.tmp  **Figure** Error! No text of specified style in document.**‑10: Firebase** | Firebase is a complete backend solution that can manage authentication, real time database and hosting as well. It can be used in iOS apps, android apps and even in web apps. |
| Android Studio v.3 | https://lh6.googleusercontent.com/jTvi6UxK1jbVlVwhfCD0vq2tQoF305WO86a6fDMDvc9Q-t-6C9acI10janx4Sq3pdawF8nmT8UIf3lYLRIhe-387lXfUGVKqZpkfU4Oln3CtelwOJ8_t5qEk1LPIhE5H0cO3gXDh  **Figure** Error! No text of specified style in document.**‑11: Android Studio** | Android Studio is the official IDE for Android development. It can be used to design user-friendly app as it has a variety of built-in functionality. |
| Arduino IDE | C:\Users\Alaa\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\973B0CFD.tmp  **Figure** Error! No text of specified style in document.**‑12: Arduino IDE** | Arduino software supports C/C++ based programming language. Used to write program to control the Arduino and NodeMCU microprocessor to perform tasks. |
| Fritzing |  | Fritzing is an open-source that is used to develop the design of the electronics hardware. The software supports different kinds of hardware circuits diagrams. |
| WebStorm | C:\Users\Alaa\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\2BF120DE.tmp  **Figure** Error! No text of specified style in document.**‑13: WebStorm** | WebStorm is JavaScript IDE that is used to build website applications. |
| Postman |  | A Google Chrome application interacting with HTTP, used as a tool for prototyping API’s, in addition to several testing features. |
| Adobe Photoshop | A close up of a sign  Description generated with very high confidence | Graphics editor published by Adobe Inc, used to create, edit and enhance images. |

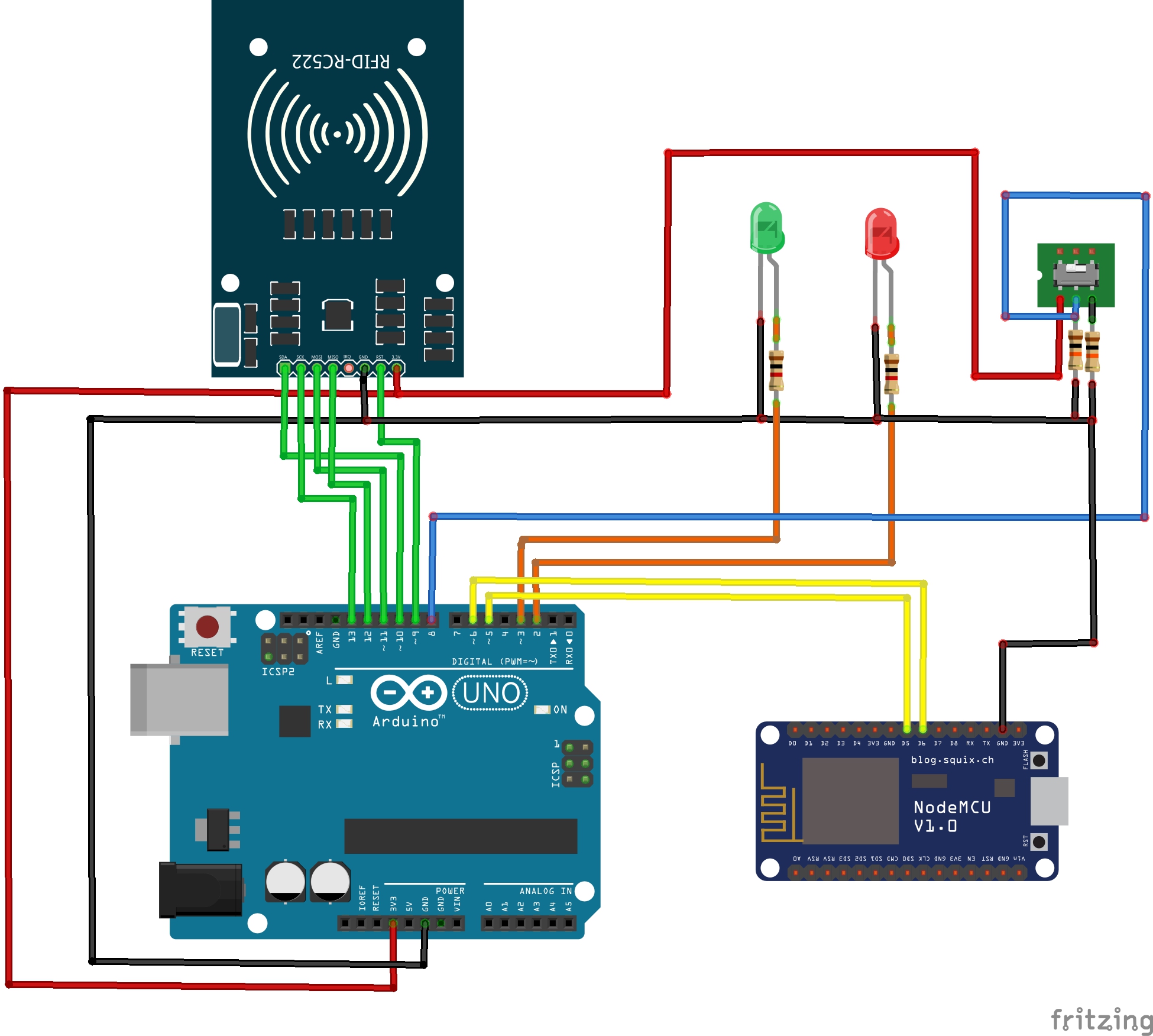
* 1. **Hardware design**

The proposed hardware solution is divided according to the module. The Reservation Free Parking module’s hardware solution consists of an Ultrasonic sensor to detect if there is a car, Arduino board to collect information from the sensors and NodeMCU board to allow the Arduino to send and receive data to the Firebase. The Reservation Free Parking module needs a sensor implemented on each parking spot and so in total a huge number of sensors are needed depending on the number of parking spots. The Ultrasonic sensor has a high accuracy and a low cost. Hence, it was chosen among the other sensors in Table 4-2. Also, the Arduino was chosen over the Raspberry Pi because it is suitable for interfacing sensors with it and has a low cost which makes it more scalable. The connectivity circuit diagram for the module is shown in Figure 4.x.

A circuit board

Description automatically generated

In addition, the hardware solution for the Reserved Parking module consists of an RFID reader to read the RFID tag on the car, Arduino board to collect information from the RFID reader, LEDs (green and red) to indicate the status of parking area gates and NodeMCU to allow the Arduino to send and receive data to the Firebase. The RFID was chosen because it provides an important functionality over the other sensors in Table 4-2. Unlike the other methods, RFID is able to identify exactly the users entering or leaving the parking area. Hence, the RFID sensor can be implemented at the gates of the parking area and the system only allows those who have registered and reserved a parking spot to enter. The Arduino was also chosen in this module because it is more suitable for the RFID and the LEDs that are interfaced with. To power the system in our testing prototype, a 9V battery will be used. However, a power grid should be used in real life implementation. The connectivity circuit diagram for the module is shown in Figure 4.x.



**Hardware components**

The components of the system besides the microcontrollers Arduino and NodeMCU are: Ultrasonic sensor, RFID sensor and the DPDT switch.

* *Ultrasonic Sensor HC-SR04*

HC-SR04 is a distance measurement sensor which is able to detect object and their distance. The sensor is able to give 2cm to 400cm non-contact measurement with an accuracy ranging up to 3mm or less. The sensor includes an ultrasonic transmitter, a receiver, and a control circuit. The four pins to connect the sensor with the Arduino are:

* VCC: 5v power
* Trig: Trigger (Transmitter) pulse
* Echo: Echo (Receiver) pulse
* GND: 0v ground

The working principle of the sensor is as follows:

* Transmitter (Trig pin) sends a signal (a high frequency sound).
* Signal is reflected when it detects an object.
* Receiver (Echo pin) receives the reflected signal.
* Time is measured by how long it took the signal from to reach the object and then go back to receiver.
* Object distance is calculated from measured time.
* *RFID Reader MFRC522*

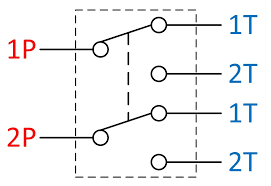
Radio-Frequency Identification (RFID) uses electromagnetic fields to transfer data over short distances. The RFID system consists of two components:

* RFID tags: The tags hold data known as unique identification (UID). The tag used in our project is the keychain tag (Figure 4.x) and should be attached to the object to be identified.
* RFID reader: A two-way radio transmitter-receiver. The reader (Figure 4.x) sends a signal to the tag to power it up and reads its response which is its UID. The reader is powered with 3.3v, its read distance is 0-60mm. MFRC522 supports SPI, I2C and UART serial communication links. The reader communicates with Arduino using SPI interface.

There are 8 pins on the RFID reader (Figure 4.x) to connect with Arduino. Four of them connect SPI interface of reader (SDA, SCK, MOSI, MISO) and the other four are for power connections (IRQ – interrupt, RST – reset, GND – ground, 3.3v – power)

* *DPDT Switch*

Double Pole Double Throw (DPDT) Switch is a two switch units in one. Figure 4.x shows the schematic diagram of the switch.



Each switch has three pins, the center pin which is the Common pin (or Pole), the other two pins are the Throw pins. Hence, the name Double Pole (two Poles as in two switch units) and Double Throw (two Throw pins per switch unit). Each Throw pin is a state that the Pole pin can switch to.

Our system only uses one switch unit to switch between zones. High (T1 supplied with 5V) being one zone, Low (T2 supplied with 0V) being the other zone and the switch (P1) switches between them.

* 1. **Software design**

The software solution consists of a mobile application, website and cloud service. Android studio is the software that was used to build the mobile application. For the cloud service we selected Firebase as it has many important features such as having a real time database. Moreover, Firebase provides APIs that supports Android and web development and that makes it easier to integrate them. In addition to that, the FirebaseArduino library is a library that allows the Arduino to communicate with the Firebase database directly without the need to have a server in between. For front-end web development, Vue.js, HTML, CSS and JavaScript are used and for the back-end development a Firebase real time database is used.

* + 1. **Structural model**

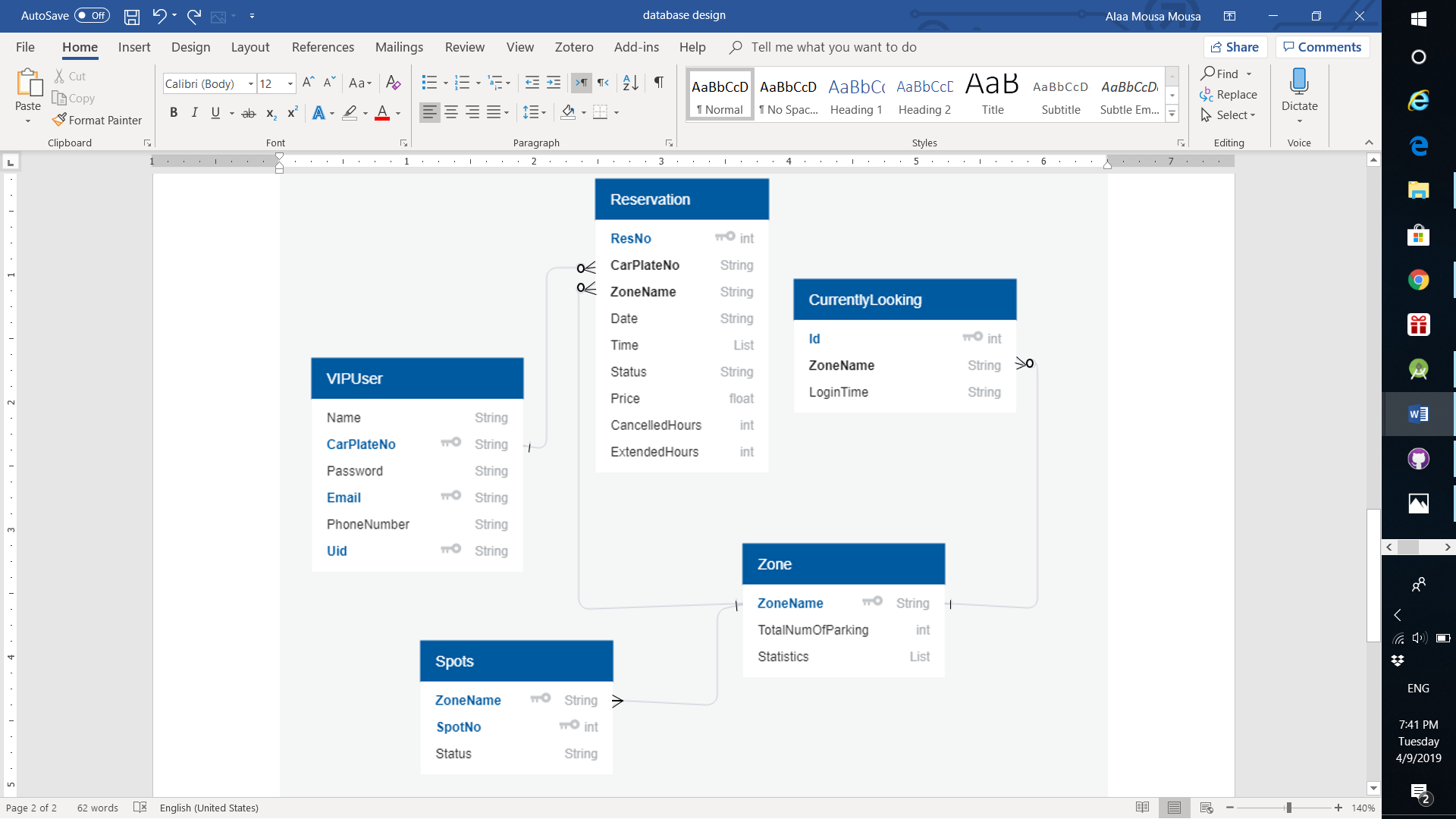
**(Class diagram)**

* + 1. **Behavioral model**

**(Activity diagram) + (Sequence diagram)**

* + 1. **Database design**

In the database design, the system stores information about 5 main components which are VIP User, Reservation, Parking area (Zone), Spot and Currently looking. The following figure illustrates the overall Entity Relationship diagram for the system:



From the above ER diagram, we can figure out the relationships that exist between each two entities:

* VIPUser has zero or many Reservation.
* Each Reservation belongs to one VIPUser.
* Each Reservation is associated with one Zone.
* Zone is associated with zero or many Reservation.
* Zone has zero or many CurrentlyLooking.
* Zone has many Spot.
* Each CurrentlyLooking belongs to one Zone.
* Each Spot is assigned to one zone.

In Figure 4.x, an ER diagram is used to help visualize the relations between entities. However, Firebase Realtime Database that the system uses does not follow the ER relations. Instead, the data is stored in JSON format. Basically, the entire database is a big JSON tree with multiple nodes.

JSON syntax is derived from JavaScript object notation syntax:

* + Data is in name/value pairs
  + Data is separated by commas
  + Curly braces hold objects
  + Square brackets hold arrays (lists)

The following example illustrates how a reservation object is stored in Firebase Realtime Database

{

"cancelledHours" : 0,

"carPlateNo" : "123",

"date" : "2019-04-13",

"extendedHours" : 0,

"price" : 5,

"resNo" : 841,

"status" : "ended",

"time" : [ 17, 18 ],

"uid" : "97 D1 34 83",

"zoneName" : "CBAE Female & Male Zone"

}

* + 1. **User interface design**

**(Mock ups)**

* + 1. **Design pattern**

The design pattern our system follows and suits our implementation the most is the Model View Controller (MVC). The MVC pattern divides our software part into three main components:

* **Model**: Includes the classes that represents stored data. The Model can only see and interact with the Controller.
* **View**: Includes the interface that the user can view and interact with
* **Controller**: Includes the system logic and interacts with both the View and Model components by retrieving inputted data from user and passing it to the model after processing any required logic with it.

The MVC structure explains the main steps our system follows. The user interacts with the user interface of our system (application or website) and whenever the user requests to input or output a certain data from the view, the view sends that request to the controller along with any inputted data. Afterwards, the controller processes the request and retrieves the needed data from the model, and then sends the appropriate output (results) back to the view to be shown to the user.

View

Controller

Model

Advantage of using this pattern:

* Separation of Concerns:

The MVC pattern separated our system into components that are independent from each other, this is called “separation of concerns”. A failure can easily be found because of the components independency, and an edit can be done on any of the components without disturbing other components.

* Re-usability and Flexibility:

The MVC pattern can increase the scalability of our system, this is because a function can easily be added or changed in a component without changing other components. Hence, our system is easily flexible to scalability. Additionally, different component of our system can be reused in other systems to give the same service.

* High Cohesion and Low Coupling:

The MVC model automatically makes the system have high cohesion and low coupling. This is because each component in the system must be related in functionality and serve the system with a specific job, which makes the functions in one component highly cohesive. Also, the different components in the system are lowly coupled, meaning changing one component, whether a major or minor change, does not interrupt the work of other components.