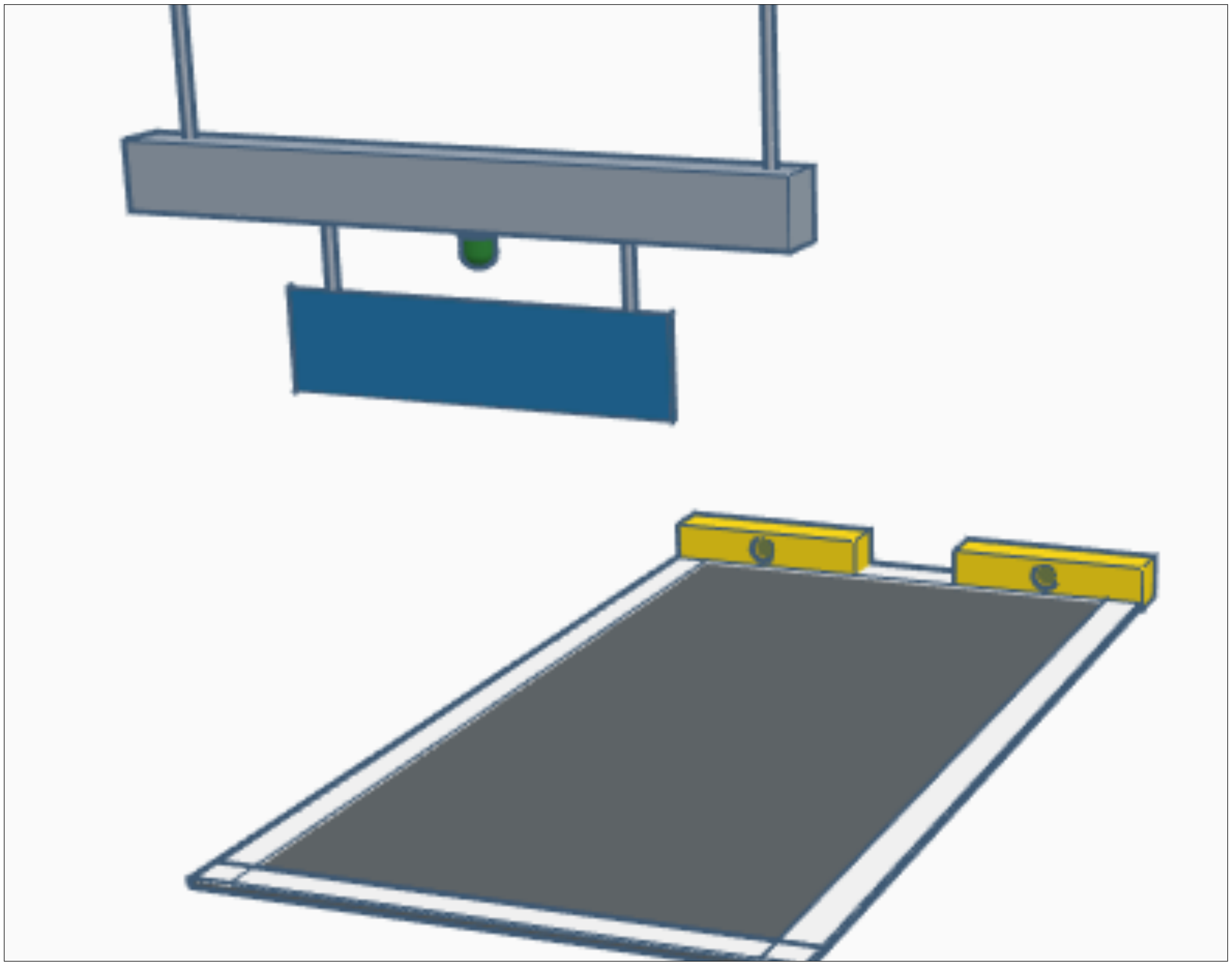


Project Report

Prepared for: IEEE QP meeting

Prepared by: Mingkai Li, Ameya Mandale, Arjun Tapde, Mrinmoy Dutta

November 29, 2019

Team Number: 103

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Objective

Major cities such as New York, Los Angeles, Milan, London and many other highly populated cities are always booming with the amount of traffic that occurs throughout the years and will only continue to grow as time goes on by. With so many cars, there are way too many cases in which trying to find a place to park your car, an action that should only take a few minutes to complete, can turn into a nightmare and consume up to 30-45+ minutes. Our product allows for those inconveniences to be solved with just what is in our pocket.

- The phone.
- Tap,
- park,
- pay, and go all done with a mobile device so you can just park and go without having to wait for a huge line.

Functions

QweickPark is an interactive web app that expands based on modern Smart Parking using Express App and Arduino. Driving can be a very anxious activity especially when the traffics aren't great. Anxiety level adds on as the driver driving around an unfamiliar town trying to find a parking spot. QweickPark provides the solution by tracking users' real time location and provide suggestions on parking structures those are close by. Users are able to reserve for parking spots, and upon entrance, a simple scan on the QR code will allow you to enter the parking lot, and a timer will start as the vehicle parks in the assigned location. Where the entire process will be fully online.

In an average urban city, finding a parking spot can take up to 30+ minutes, the amount of CO2 produced by that period of time can be saved by converting this entire process into a 1-5 mins process using the idea of online smart parking. "Tap, park, pay, go" these actions not only saves the frustration, but also helps us to make our environment a better place.

Solution

A system and method that assists drivers' actions of searching, reserving, parking and paying for a parking spot online hence the online smart parking system. The online smart parking system assigns an encrypted id for each user as well as their vehicles, utilizing a real time location tracking feature to give suggestions on closest available parking spots. The system allows users to reserve parking spots at a desired parking structure, and scans the user as well as the vehicle id upon entry by QR code(or other means). Each spots will be guarded with a microcontroller with sensors that verifies the users' identity upon parking and provides a timing feature to calculate elapsed parking time as well as amount due for payment. The online smart parking system stores and manages available parking lots, reservations, vehicle ids, and elapsed parking time under the program.

MATERIALS

Building Budget

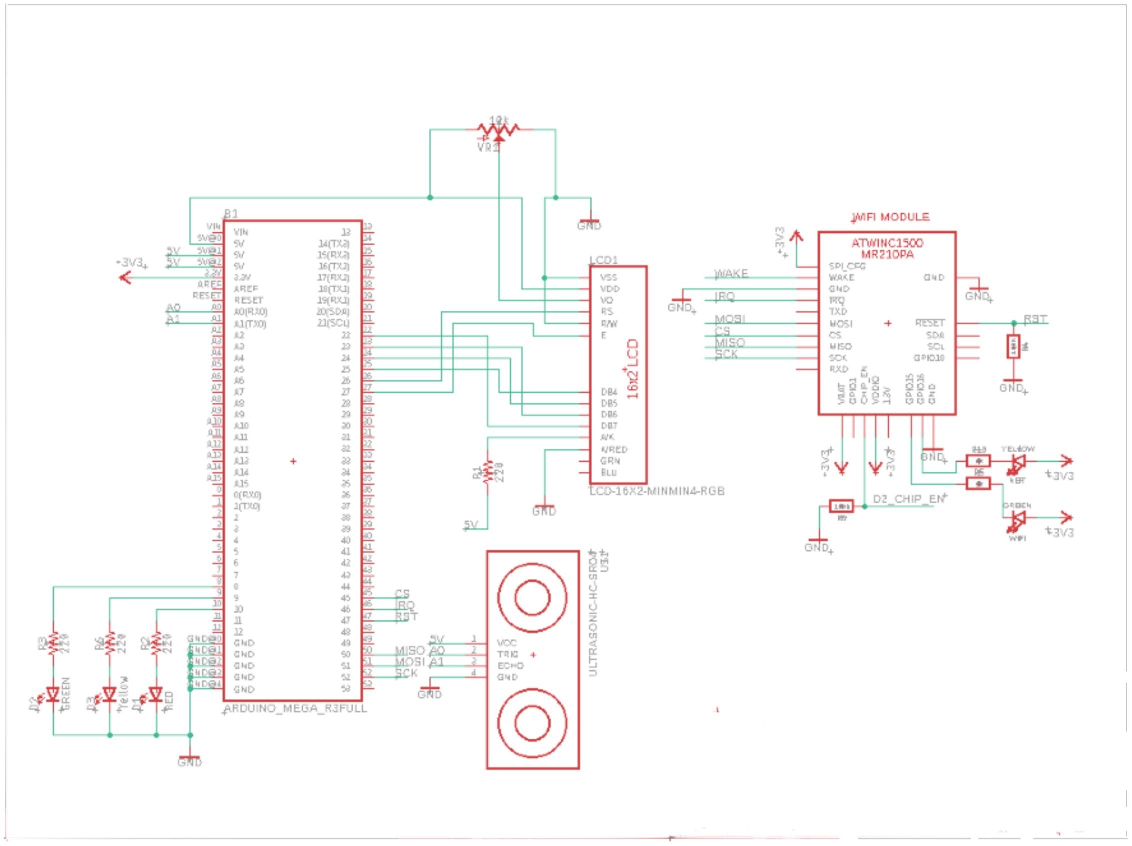
All the materials for hardware design is included here in the table.

Description	Quantity	Unit Price	Cost
Arduino Mega	1	$ 15	$ 15
Adafruit Winc1500c wifi Module	1	$ 15	$ 15
LCD Display	1	$ 9	$ 9
Ultrasonic Distance Sensor	1	$ 2	$ 2
Total			**$ 41**

Budget pre unit is calculated roughly as 41 dollars. Different sensors can be used on implementing the project like cameras, and different cloud base receiver/transmitter can also be used.

BUILD INSTRUCTIONS

Hardware



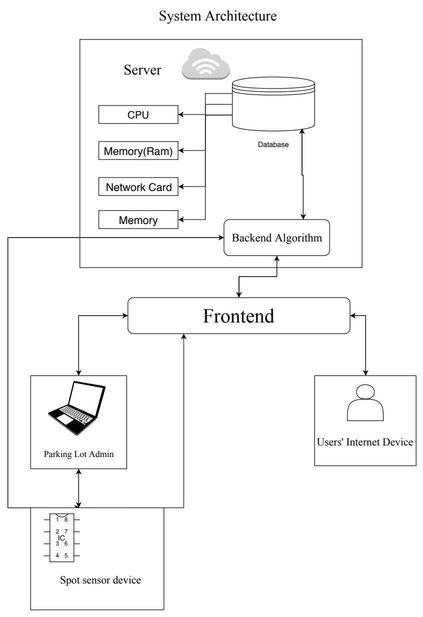
Implementation:

An ultrasonic distance sensor with LCD, and ATWINC1500 wifi module was used along with Arduino Mega to build this project.

- When the Arduino serves as a server as the users reserve a parking spot, the server will access the Arduino, hence communicate to let Arduino update its spot to the corresponding user profile.
- When the Arduino serves as a client is when the car have left the parking spot, the Arduino will visit a hidden link in the server to let it know that the car has left the parking structure.

BUILD INSTRUCTIONS #2

Software



System Architecture:

The software of the project consist of using Mysql database to store customers' information, Vehicle information, and update real time parking situations for each parking structures. Users sign up in the interface, and register their vehicle, Express server will update the users' information into the cloud database.

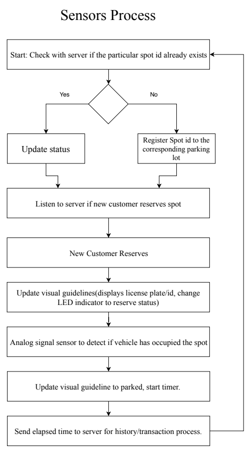
When users needed to make reservations for parking, the system will logon to the particular lot Arduino to make updates which allows users to reserve their parking spots by displaying visual guidelines that contains users' license info on the display.

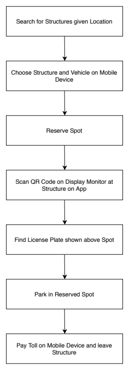
Furthermore: when the users arrive the parking lot, they will use their phones to scan unique QR code that indicates users' arrival in the parking structure.

Sensor&Customers Logic:

The sensor starts scanning for parking as soon as the QR code of the parking log was scanned, the Ultrasonic distance sensor will be enabled.

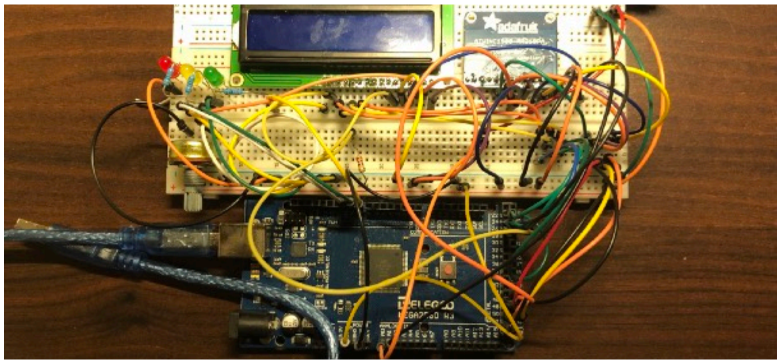
A timer will also start as the car has parked in the parking structure. When the customer left the parking lot, there will be no more timing on the particular parking event, and it will update the server with the elapsed time and potentially ask users to pay their parking event through online parking APIs(Apple Pay, Google Pay, etc).


Sensors Process
Start: Check with server if the particular spot id already exists
Yes
No
Update status
Register Spot id to the corresponding parking lot
Listen to server if new customer reserves spot
New Customer Reserves
Update visual guidelines(displays license plate/id, change LED indicator to reserve status)
Analog signal sensor to detect if vehicle has occupied the spot
Update visual guideline to parked, start timer.
Send elapsed time to server for history/transaction process.


Search for Structures given Location
Choose Structure and Vehicle on Mobile Device
Reserve Spot
Scan QR Code on Display Monitor at Structure on App
Find License Plate shown above Spot
Park in Reserved Spot
Pay Toll on Mobile Device and leave Structure

STEP BY STEP INSTRUCTION

Setup Circuit according to the schematic



Parking detection circuit

Install the needed API: nodejs, Mysql

https://nodejs.org/en/
https://console.developers.google.com/?pli=1
https://www.mysql.com/

Install the source code

- run the migration file on mysql by running: db-migrate up
- More instructions can be found here: https://db-migrate.readthedocs.io/en/latest/Getting%20Started/usage/

Run the Server

```
npm install       //install all needed packages

cd bin          //move into folder Called bin

node www.       //running the www file for starting the server.
```

SSL certificate on localhost(optional)

- **Generate a RSA-2048 key and save it to a file named rootCA.key**
  ```
  openssl genrsa -des3 -out rootCA.key 2048
  ```
- **This certificate will have 1024 days, but you can change the days to anything you want.**
  ```
  openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem
  ```
- **Next, open Keychain Access on your Mac and go to the Certificates category to change rootCA.pem into Always Trust.**
- **Create a new OpenSSL configuration named server.csr.cnf**
  ```
  [req]
  default_bits = 2048
  prompt = no
  default_md = sha256
  distinguished_name = dn

  [dn]
  C=US
  ST=RandomState
  L=RandomCity
  O=RandomOrganization
  OU=RandomOrganization
  emailAddress=yourEmail@email.com
  CN = localhost
  ```
- **Create a v3.ext file**
  ```
  authorityKeyIdentifier=keyid,issuer
  basicConstraints=CA:FALSE
  keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
  subjectAltName = @alt_names

  [alt_names]
  DNS.1 = localhost
  ```

- `openssl req -new -sha256 -nodes -out server.csr -newkey rsa:2048 -keyout server.key -config <( cat server.csr.cnf)`

- `openssl x509 -req -in server.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out server.crt -days 500 -sha256 -extfile v3.ext`

CONCLUSION

Challenges

A major challenge we ran into is ensuring that real time tracking would accurately display the amount of elapsed time a car was parked. Due to technical limitations of the ATmega328P's, simulating actual real-time update will lag by a 1-2 milliseconds which was able to be neglected due to creating a basic clock system updating seconds, minutes and hours. The Arduino's internal clock cannot be reset to zero by practical means (we may not press the reset button) which gave us the issue of resetting our timer counter for when a new car was parked. This was solved using assembly language techniques to manually reset the internal clock to 0 (a different solution would be to buy an external RTC but that adds to cost).

The alignment of the run time between Arduino and the server is an issue due to different processing capabilities. A microcontroller will not be able to process all the instructions to simulate simultaneous run time with a server that is processed with a computer.

Accomplishments

We are very proud that the entire team maintained the same attitude towards this project from the point that this started to the very end. The team collaborated with each other when we met challenges, and are able to pull through with a project that had a much more tremendous scope than initially planned.

Next Step

Almost a decade ago, some of the big companies like Apple, Google at silicon valley started a big financing revolution by introducing online payment methods like apple pay and google pay. This huge jump signifies the start of a new era "electronic financing"(paperless). However, after almost 10 years, people are still using cash and credit cards. We think it is because the little things in our lives are not yet supporting these advanced payment methods. The only way of promoting this revolution is to start from these little things just like parking. We hope to add more features to QweickPark such as supporting digital transactions to ensure the whole process goes paperless and more efficient transaction process to help with the growing demand in California and other locations in the world to design products that will benefit the environment.

Other possible implementations could be tracking frequent visitors that use the product for both possible discount incentives for long time users and average time frequent users park to give queue times to other users when parking may not always be available.

REFERENCES

Software

Nodejs, ExpressJS, Arduino, Google Map API, Direction API, Geolocation API, MYSQL database

https://nodejs.org/en/
https://console.developers.google.com/?pli=1
https://www.mysql.com/

Arduino

LiquidCrystal, Timer, SPI, WIFI101

https://www.arduino.cc/