Pizza Hot Digital Order and Delivery System

A Design Project Report

Presented to the School of Electrical and Computer Engineering of Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted by

Taihang Ye (ty322), Junyin Chen (jc2954), Baoyue Wang (bw476)

MEng Field Advisor: Joseph Skovira

Degree Date: May (t322), Dec(jc2954, bw476), 2016

Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title: Pizza Hot Digital Order and Delivery System

Author: Taihang Ye, Junyin Chen, Baoyue Wang

Abstract:

This project is aimed to provide a customized solution of quicker and more satisfying food order and delivery system under the digital business trend. For the food order part, a "Pizza Hot" website and mobile applications are implemented for customers to view dishes and place orders. Meanwhile, restaurant managers are able to manage dishes, orders, comments by logging in the website management dashboard. For the food delivery part, an intelligent delivering system is built, including a mobile application for delivery staff and a smart delivery box integrated with a microcontroller temperature sensor, Wi-Fi and GPS modules, which can show temperature of pizza, time, and current location on the screen as well as navigate delivery staff to next delivery by audio output.

Distribution of Work

For the report, individual contribution is shown below.

	Introduction	System Design	Implementation	Tests & Results	Issues	Future Work	Conclusion
Taihang		2.1 2.3.2 2.3.3	3.2.2 3.2.3	4.2 4.3	5.3		7
Junyin		2.2	3.1	4.1 4.3	5.1		
Baoyue	1	2.3.1	3.2.1	4.2	5.2	6	

Table 0-1. Individual contribution sections

For the project, individual contribution is shown below

	Webpage Front End	Webpage Back End	Mobile App for Customers	Mobile App for Driver	PIC32 Development	Amplifier Design & Implementation
Taihang		✓	✓	✓		
Junyin				✓	✓	✓
Baoyue	✓		✓			

Table 0-2. Individual contribution for the project

Executive summary

Pizza Hot is aimed to provide

1) an order system on multi-platforms for customers to select dishes and place orders

2) a convenient management dashboard for restaurant manager to easily manage the whole system

3) a smart delivering system for helping delivery staff improve the quality of delivery service.

The Order system contains two parts: a responsive website and a mobile application launched both on Android and iOS platforms.

- The website is implemented using Ruby on Rails as the backend with functionalities including viewing dishes, adding dishes into shopping cart, placing orders and writing feedback for the restaurants.
- The mobile application is implemented in a hybrid way which can be deployed on both Android and iOS platforms using Apache Cordova, AngularJS and Ionic Framework. It includes several modules such as viewing dishes, adding comments, adding dishes to users' favorite list, reserving a table in the restaurant, registration and login, which are really similar to the website mentioned in the first bullet.

The management dashboard is part of website for the restaurant manager, which realized functionalities such as managing order status, view customers' feedback and inviting new administrators.

The Smart delivery system contains a customized mobile application for delivery staff and a smart delivery box which can detect the temperature inside the box in real time. Besides that, the system can load the routing data of an order so it can navigate for the driver, the turning notification is on audio basis. Everything a delivery man wants is integrated into this hand-free device.

Table of Contents

1. Introduction

Nowadays, digital business platforms are very popular and save us much effort and time in our daily life. E-commerce companies such as Amazon and Ebay could deliver goods to customers very efficiently. On the one hand, customers could select goods and place orders online without visiting the shop, which is usually time-consuming. Besides, they do not need to carry the goods to home. Instead, the shop would deliver the goods and save customers' efforts. On the other hand, using digital business platforms could make it more convenient for shop owners to manage orders, collect and analyse data and provide better service. In the catering industry, the demand of combining the convenience of digital business with their traditional delivery service is increasingly growing. Unlike common e-commerce companies, the restaurants usually could deliver food in less than half an hour and actually saves customers' time when compared to visiting the restaurants.

In order to stand out in the digital business trend of catering industry and provide more satisfying service, we designed this Pizza Hot project. Customers usually expect fast delivery and food in good condition to eat. So in order to adapt to customers' expectation and earn more profits for restaurant owners, we improved the traditional digital business platform. In addition to the general functionalities of e-commerce platform, we developed a hardware system which is intended to be put in the delivery box. After the restaurant receives customers' orders and the dishes are ready to deliver, the delivery man put the portable system in the delivery box. The system could record the temperature in the box and the time of delivery. After the dishes arrives, it will calculate a suggested tip for customers based on temperature, time and distance the delivery man covered. If the food's temperature is suitable, the tips could be higher and otherwise the system gives a discount. In this way, we could realize a win-win relationship between customer and restaurants. Customers can expect fast delivery and hot food. Restaurants can make more money by providing good service, which also helps them stands out among competitors. Delivery staffs will be provided information for more efficient operations such as turn by turn navigation.

1.1 Design Alternatives

We want to implement a responsive website for customers to view dishes, place orders and send feedback to restaurants. We also want to implement functionalities such as managing orders, inviting new admins and viewing feedback for admin. There are several options for the website implementation. For the frontend, the first one is not using any framework and developing the CSS from draft. This approach requires great ability of developing responsive website and design experience. The second one is using Bootstrap framework. The framework is easy to use and has many default styles for website elements such as forms and buttons. Therefore, we chose to implement the website using Bootstrap. For the backend, there are many options. With three of the most popular options, PHP, Java and Ruby on Rails. By using Ruby on Rails, we could add new features easily and do agile development. So we chose Ruby as our backend language.

We want to design mobile applications for customers to view dishes, leave their comments and make reservations online. There are several ways to finish this task. The first one is sticking on the web site we developed and let customers to visit our website on the mobile devices because of responsive web design we used in our website, which is the slowest and fully hosted in the mobile browser. The second approach we can use is develop fully native mobile applications for both Android, iOS device and even windows phone which would acquire highest performance. The third approach which we selected is packaging mobile application development using css, javascript and html into a native wrapper and delivering it as a native application--Hybrid mobile application, which is slow but comparable to native applications based on functionality and has some access to device capabilities.

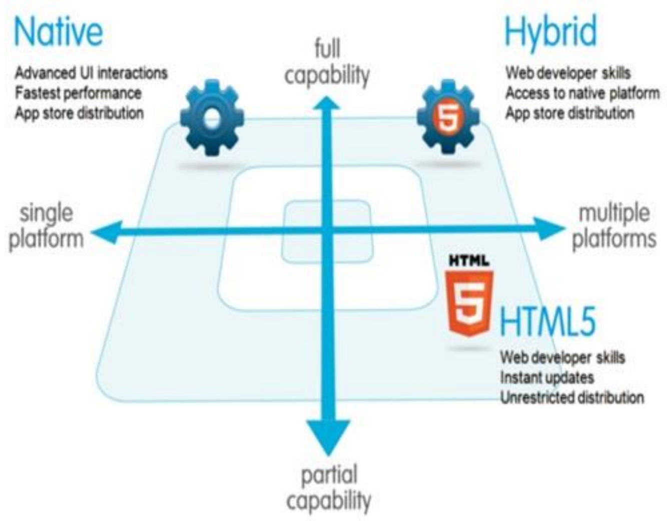


Fig 1-1 design choice

Hardware design choice would be covered in part 2.1.

2. System Design

2.1 High-level Design

The whole system works as following steps:

1) The customers will visit our website or mobile applications to select the pizza they like and place the order.
2) The order will send to the server side and restaurant manager is able to manage detail information and status of orders.
3) Delivery staff will driver's mobile application and smart delivery box to help them navigate to the right locations with optimized path from Google.
4) The customers will receive their "hot" meal with temperature showing on the smart delivery box.

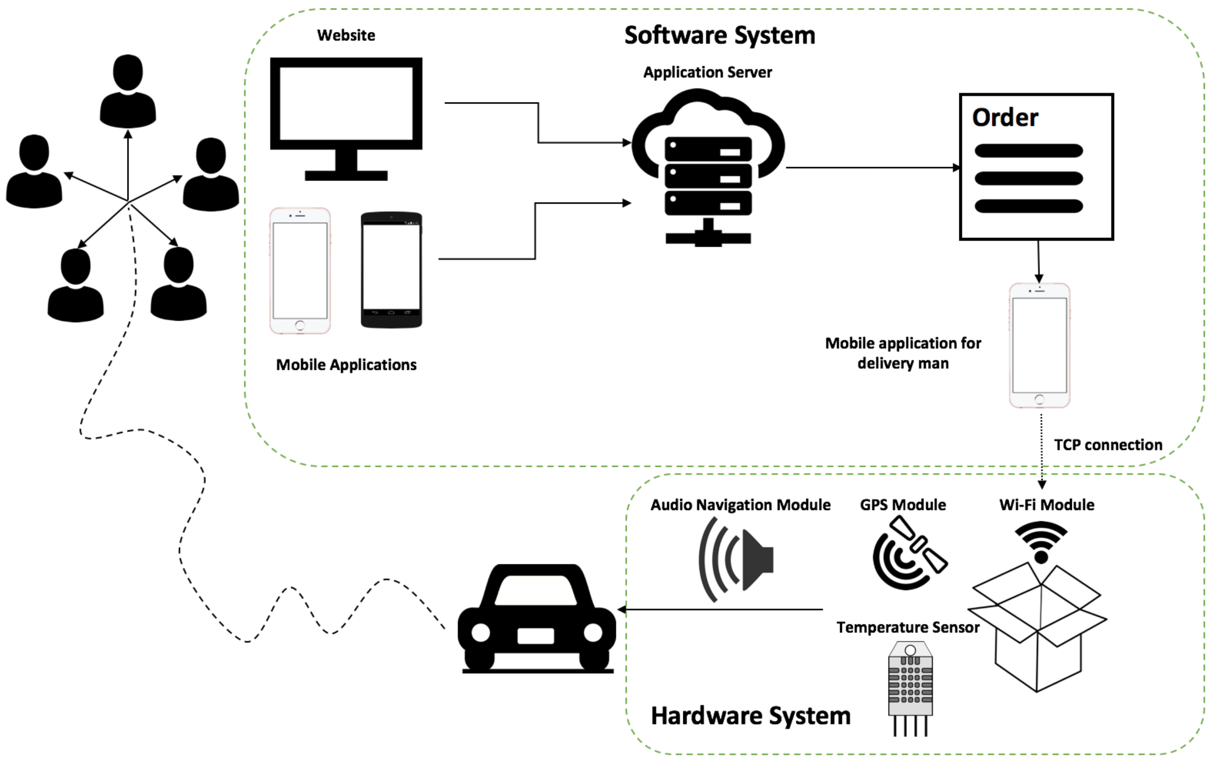


Fig 2-1 High Level Design of System

2.2 Hardware System Design

High-level hardware design and software to drive these hardware are discussed in this section. For pure software part, website and mobile apps, please refer to 2.3 Software System Design. More details about implementation would be covered in chapter 3.1 hardware implementation.

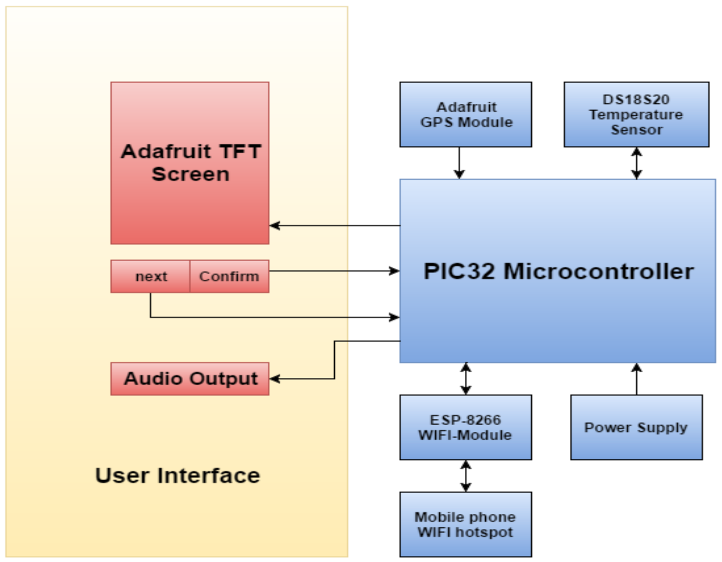


Fig 2-2 Hardware System Block Diagram

Above is the block diagram for the entire hardware system. DS18S20 temperature sensor is used to measure the temperature inside the delivery box. Adafruit GPS module provides raw data from satellite to microcontroller for locating driver's position. ESP8266 WIFI Module is running as a server. Basically, it connects to an access point, known as a WIFI hotspot which is created by the mobile phone, then reads routing data from Google maps and sends it back to the microcontroller.

PIC32 Microcontroller is used as the central control unit, it collects data from GPS module, sensor, and WIFI-Module, processes them, then sends processed data to the user interface. The user interface consists of screen, two buttons, and audio output. Specifically, TFT screen displays processed data from microcontroller so users can actually see them. Two buttons are used to select working modes of microcontroller, for example, navigation mode. Audio output is used in the navigation mode, so the user would be notified to turn left or turn right at specific positions without getting into the trouble of taking look at the screen all the time.

2.2.1 PIC-32 Microcontroller

Microcontroller is the central control unit for this project. Among various kinds of microcontrollers, PIC32 microcontroller was picked mainly because it's a 32 bit microcontroller, able to process data faster, and its tiny volume supports a more compacted project.

More specifically, PIC32MX250F128B was used. One good feature of this microcontroller is, pins could be used flexibly, one specific pin could be used as general purpose I/O pin, or UART transmission, or even analog output. Some features and parameters of this microcontroller are listed in the table below [1].

Device	PIC32MX250F128B
Max Operation Frequency	50 MHz
Flash	128KB
SRAM	32KB
Operation Temperature	- 40 °C ~ 105 °C
Operation Voltage Range	2.3 V ~ 3.6 V
Pins	28

Table 2-1 PIC32MX250F128B Specification Summary

We are using Microstick II as supporting hardware for this microcontroller chip. It has MCP1727 voltage regulator, a mini USB interface for programming/debugging/power supply, one reset button, and peripheral resistors/capacitors. The main purpose of regulator is to interface 5V USB power supply and 3.3V microcontroller chip. It has 1.5A output current capability, multiple standard fixed output voltages support [2]. Programming and supplying power becomes much easier with the support of Microstick. Testing/debugging becomes less complicated as well because the package of this small board is breadboard friendly, and reset button onboard provides one-click trouble-free reset solution for user. Below is the picture of Microstick.

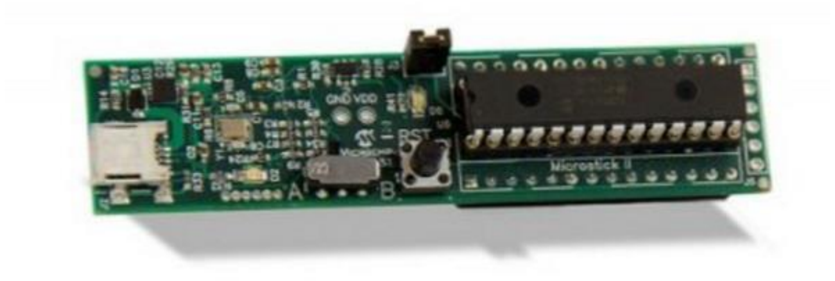


Fig 2-3 Microstick II

2.2.2 GPS Module

Adafruit Ultimate GPS Breakout module is used for our project. It's the core module to find the location of user. We are using this module because it's 3.3V - 5V compatible, it's breadboard friendly, it starts up fast, typically within 2 minutes, and it has an LED output to indicate fix status [3]. Fix status, in easy words, essentially shows how you are tracked by the satellites. Typical fix status contains no fix, 2D fix, and 3D fix.

The protocol this device uses is NMEA 0183 standard. NMEA 0183 is a combined electrical and data specification for communication between marine electronics such as echo sounder, sonars, anemometer, gyrocompass, autopilot, GPS receivers and many other types of instruments. It has been defined by, and is controlled by, the National Marine Electronics Association. It replaces the earlier NMEA 0180 and NMEA 0182 standards [4].The communication interface of this module with peripheral circuits is UART. UART is the abbreviation for Universal Asynchronous Receiver/Transmitter, which is commonly used in serial communication between computer systems.

Once the power is supplied for the GPS module, it begins sending out NMEA stream, regardless of the status of the receiving device. Even if no satellite has been assigned for finding the location of user, transmission of NMEA stream would not be stopped, instead, the NMEA stream being sent at this stage contains information showing no fix type. Among different sentences of NMEA stream, we pick sentences begin with $GPGGA to get information. Below is an example translation of $GPGGA NMEA sentence [5]:

$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

GPGGA	Global Positioning System Fix Data
123519	12:35:19 UTC
4807.038,N	Latitude 48 deg 07.038' N
01131.000,E	Longitude 11 deg 31.000' E
1	Fix quality: 0 = invalid, 1 = GPS fix (SPS)
08	Number of satellites being tracked
545.4,M	Altitude, Meters, above mean sea level
46.9,M	Height of geoid (mean sea level) above WGS84

Table 2-2 Example translation of GPGGA NMEA sentence

Above example demonstrates parsing GPGGA sentence is adequate to collect data we need for navigation, especially longitude and latitude.

2.2.3 WIFI Module

The WIFI module we are using is Adafruit HUZZAH ESP8266 breakout. It has a reset button, a user button to boot load the module, UART interface, and 3.3 V out, 500mA regulator. We used Arduino IDE to program it so it runs as a server for our specific application.

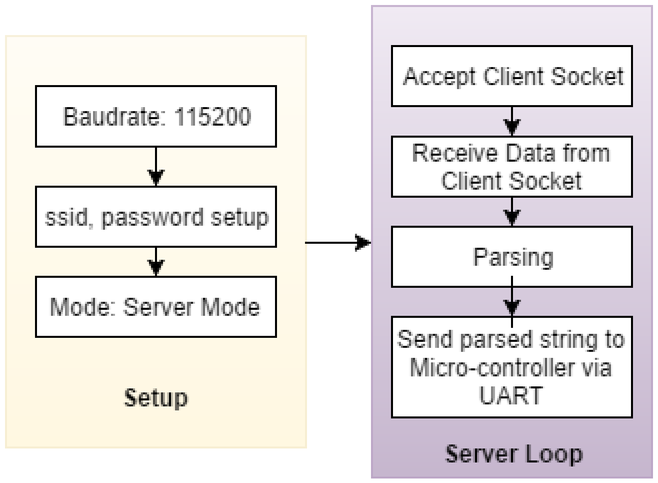


Fig 2-4 WIFI Module Program Flow Chart

First, in the setup stage, we set the baudrate to be 115200 to improve the throughput of data. Then the WIFI module needs to connect an access point, ssid and password needs to be specified. In our case, we used personal mobile phone to create a hotspot, ssid is the hotspot name the user creates, and password is the password for the hotspot. After setting the module run in server mode, it goes into server loop. In the server loop, the specific task this module completes in our project is, accept client socket created by mobile-phone application first, then receive data from client socket, process the data, then send processed data to microcontroller via UART connection.

2.2.4 Temperature Sensor

The temperature sensor we are using is DS18S20 temperature sensor. Compared to traditional analog sensor, this sensor does ADC inside itself and represents information about temperature in digital form, so it's much more robust and much less sensitive to noise. This sensor has ±0.5°C accuracy from -10°C to +85°C. This sensor is using one-wire protocol to communicate with other device. Though, it is just a sensor, it also has inner ROM and it has its own command system, in other words, it is programmable.

There is no built-in one-wire protocol for PIC device, so we picked several instructions below:

1) Reset
2) Write 0xCC - Skip the ROM
3) Write 0xBE - Start ADC Conversion
4) Read - Each time 1 byte

Details for the timing of this temperature sensor would be covered in the implementation detail chapter.

2.2.5 Audio Output

Since we are implementing navigation function on turn-by-turn basis, it makes little sense if the user has to look down at the screen all the time. Hence, we designed an audio output block for our design. The schematic of this block is shown at the next page.

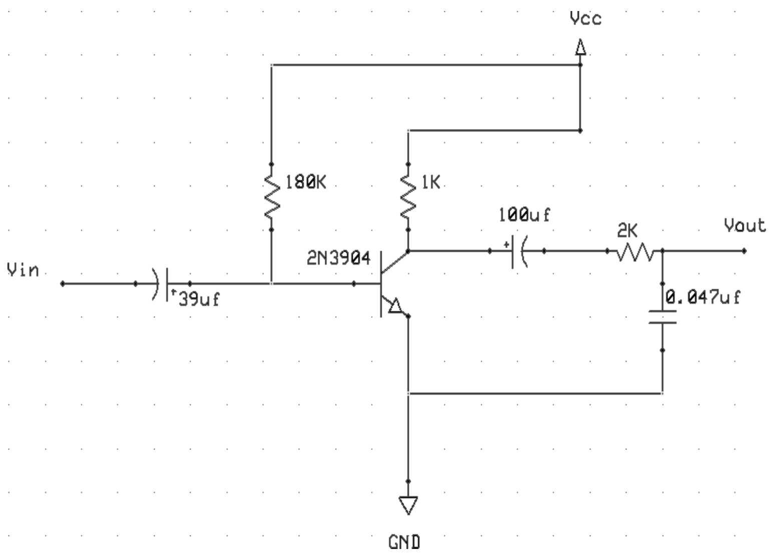


Fig 2-5 Schematic for amplifying circuit

The 39μf and 100μf capacitor are used to remove DC components of the audio. Since the audio we are generating is a voltage signal from an analog output pin, ranging from 0 ~ 1.2V, we need an amplifier with good performance both in current and voltage. Hence, common emitter amplifier is chosen. Based on our experiment and trade-off between the gain and distortion, 180K and 1K resistor are picked for the amplifier. 2K resistor and 0.047μf capacitor makes up a low-pass filter, which is used for suppressing high frequency noise.

The output is soldered with an audio jack. The reason for not choosing a speaker is:
1) Audio jack is more flexible, some speakers support audio jack, but directly soldering onto a speaker gives no chance to go back to earphone.
2) The DAC for audio output is only 4 bits. In other words, output voltage only has 16 different levels, therefore it's not very consistent, in other words, it has much harmonic and it affects audio quality. Earphones typically suppress noise, while speakers work in the opposite way. Earphone definitely performs better in suppressing harmonics.

2.2.6 User-Interface: TFT-Screen & Buttons

Blocks covered in section 2.2.1 - 2.2.5 are doing background jobs. The user needs somehow a way to know temperature, position, and navigation data in a straightforward manner. Also, they need the interface, as simple as possible, to switch back and forth in different pages of screen. We use Adafruit TFT screen to display information, and two push buttons to control the state machine. We use the Adafruit TFT library ported to PIC32, written by tahmid, as the driver of screen [6].

2.3 Software System Design

The software system involves a website, Android and iOS mobile applications for customers and mobile application to help navigation.

2.3.1 Website Design

2.3.1.1 Website workflow

The website is designed for both users and restaurant owners. For users, they could view menu and select dishes. After that, they could complete a form of personal information and place order. The transaction workflow of users is as described in Figure 1. For restaurant owners, we require a login for their access to the management panel. On the management panel, they can view and manage the status of orders. In addition, they could add more administrators to help them with management. The workflow of managers is as described in Figure 2-5.

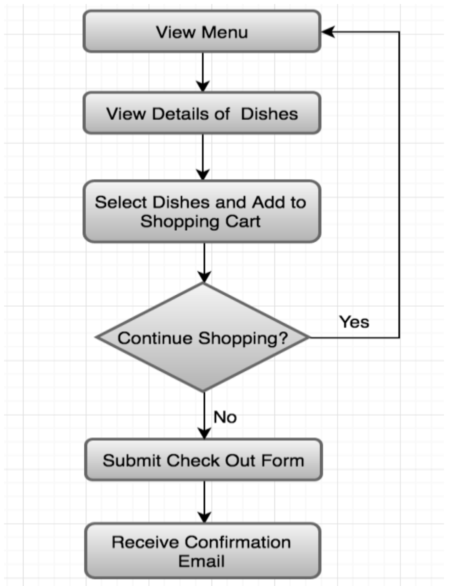


Fig 2-6: Webpage Flowchart of User

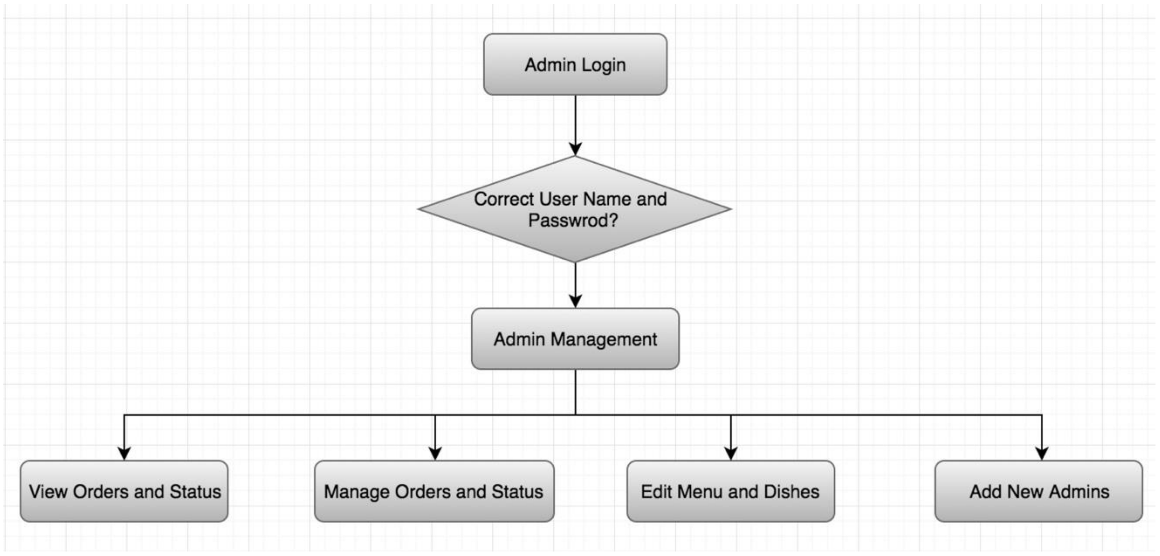


Fig 2-7: Webpage Flowchart of Admin

2.3.2 Hybrid Mobile Application for Customers

2.3.2.1 Software stack

The Software stack we used for building the mobile applications targeting different platforms is as follows

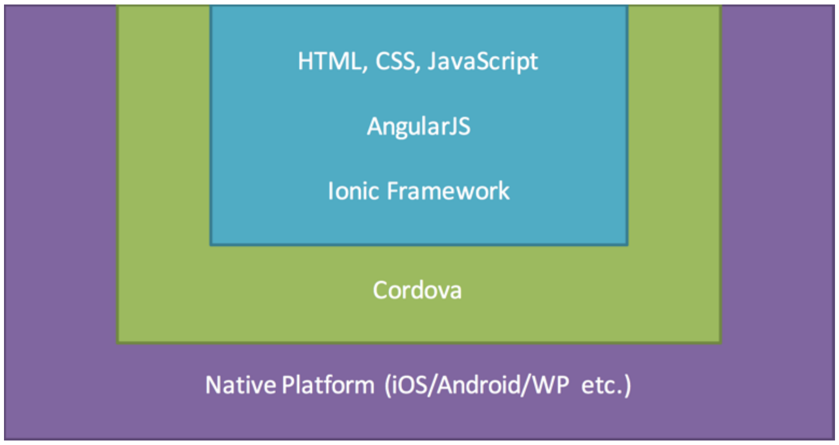


Fig 2-8 Software Stack

The ionic framework is the HTML SDK to help build apps with native look and feel, which uses AngularJS and mainly focuses on the front-end. The ionic framework uses Angular directives, native styled elements and Sass to build a mobile-optimized HTML5 and CSS3 based UI elements.
Apache Cordova [7] allows for building native mobile applications using HTML, CSS and JavaScript. This tool helps with management of multi-platform Cordova applications as well as Cordova plugin integration.

2.3.2.2 Mobile Application MVC framework

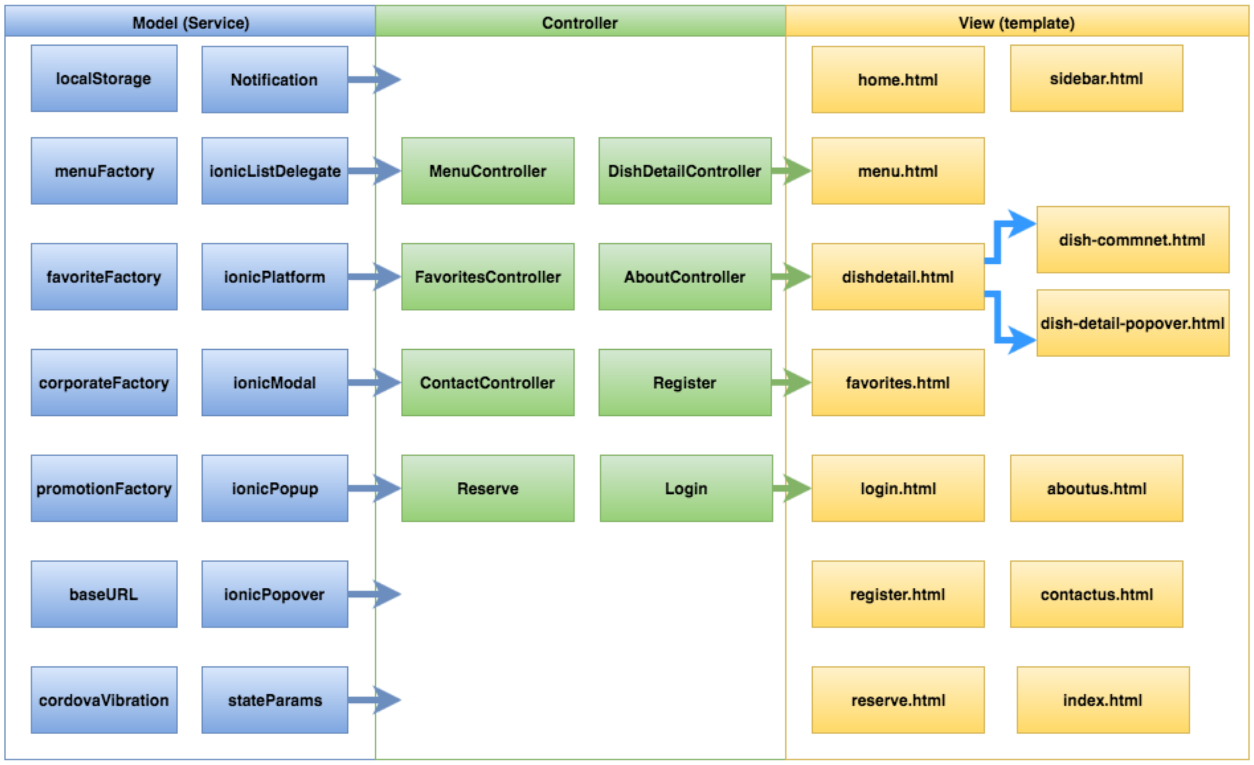


Fig 2-9 Mobile Application MVC framework

2.3.3 Mobile application for drivers

This mobile application also makes use of ionic framework, which references an example project on Evothings Studio [8] tutorial. This application is mainly for delivery staff to use which will help them navigate to the right delivery location.

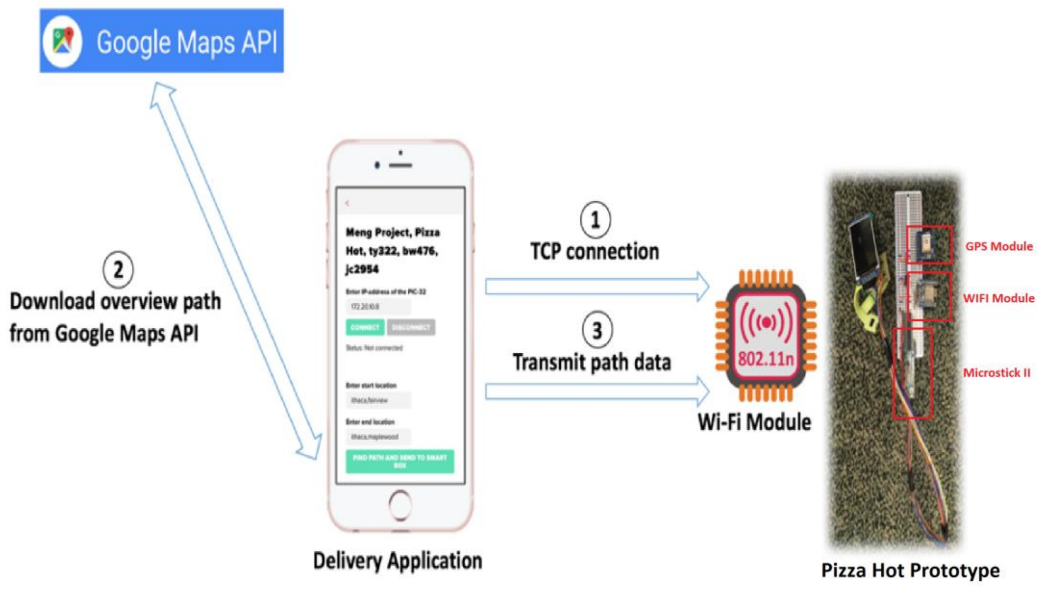


Fig 2-10 Mobile app for drivers

3. Implementation

3.1 Hardware System Implementation

3.1.1 Interfacing PIC32 microcontroller, GPS Module, and WIFI Module

Communication among PIC32 microcontroller, GPS Module, and WIFI module is implemented through UART. UART channel 1 of PIC was linked to WIFI module, and channel 2 was linked to GPS Module. Note that two channels are different. Usually, UART2 was mapped to pin 21 and pin 22, they are 5V tolerant, but UART1 pins are not 5V tolerant. Though the WIFI module outputs 3.3V voltage, one 3K resistor must be connected in serial to limit the current, otherwise pins might be burnt. Below is the schematic for interfacing PIC, GPS Module, and WIFI Module.

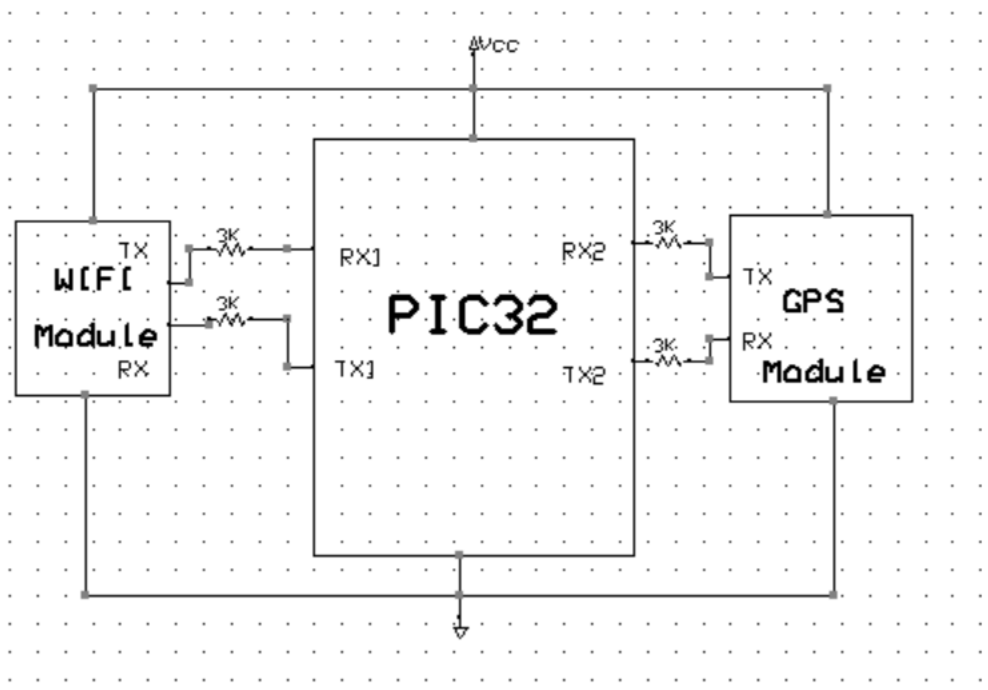


Fig 3-1 schematic for interfacing PIC, GPS, and WIFI Module

As mentioned in chapter 2, GPS module actually keeps sending NMEA Streams out in spite of what is happening, WIFI module just receives data from mobile app, then sends it back to PIC. PIC only has 32KB SRAM, apparently it does not have enough memory to store all of data GPS and WIFI module send to it. However, the stream interpretation part of code has to be executed only if complete message has been sent. To solve this concurrency problem, we are porting protothreads, originally written by Adam Dunkels [9], with some specific hardware support (timers, etc) for PIC32 added by Professor Bruce Land [10], to our project.

There are 3 threads in the program, one for GPS UART, one for WIFI UART, and one for GPS Navigation. GPS UART just spawns a child thread and yields. The child thread spawned by GPS UART is for reading the stream from GPS UART to the GPS buffer, one character by one character, until it reads the '\n' sign, indicating next sentence, then clears the buffer, restarts the procedure above. For the time one character arrives but new character is not, the child thread yields, so the transmission is non-blocking. A GPS buffer of 256 bytes is safe enough to accommodate one complete GPS sentence being parsed by GPS interpreter sub-block, and the protothread yields in the time gap between characters to guarantee the concurrency, in this way the contradiction mentioned in the paragraph above is solved.

The idea for solving the contradiction of memory limitation and concurrency is similar for WIFI thread, except the data structure to implement this is different. Since the objective of WIFI thread is to provide route data for navigation, the mobile app is sending the data on a waypoint basis, and all of waypoints must be properly stored to guarantee correct function of navigation. We are using linked list to store waypoints data. When the route is being load, it dynamically allocates memory for each waypoint, and when the navigation has finished, routing data would be freed.

Navigation thread starts running when the state machine jumps into navigation state. It's comparing the user's position with the next waypoint, when it's closed enough, it outputs an audio to notify the user to turn. The background math for computing the turning angle is shown below.

$$\vec{a} = currentwaypoint - prevwaypoint$$

$$= (\mathrm{currentlon} - prevlon, \mathrm{currentlat} - \mathrm{prevlat})$$

$$= (a_x, a_y)$$

$$\vec{b} = nextwaypoint - currentwaypoint$$

$$= (\mathrm{nextlon} - currentlon, \mathrm{nextlat} - \mathrm{currentlat})$$

$$= (b_x, b_y)$$

$$Turnleftorright = \vec{a} \times \vec{b}$$

If $\mathbf{Turnleftorright} > 0$, the user should turn left at a waypoint, otherwise he should turn right.

$$turningangle = \cos^{-1} \frac{\vec{a} \cdot \vec{b}}{abs(\vec{a}) \cdot abs(\vec{b})}$$

Fig 3-2 Background math for navigation

3.1.2 Porting DS18S20 one-wire sensor to PIC32

The DS18S20 temperature sensor is using one-wire protocol to communicate with peripheral devices. The bus remains high-level when the bus status is idle. However, there is no existing library for one-wire protocol. As mentioned in chapter 2, the sequence to get DS18S20 working is:

1) Reset
2) Write 0xCC – Skip the ROM
3) Write 0xBE – Start ADC Conversion
4) Read – Each time 1 byte

Below we expand on this set up functions.

Reset

The function of reset is handshaking of two device. According to DS18S20 datasheet, the timing goes like the following [11]:

1) Host (PIC32) pulls down the bus for at least 480 µs.
2) Host releases the bus for at least 15 – 60 µs.
3) If client (Temperature sensor) is present, the client pulls down bus for 60 – 240 µs.
4) The host samples the bus.
5) Client releases the bus.
6) An interval of at least 480 µs for the recovery of bus.

Write

Writing or reading a specific value to or from the device is based on writing 0/1 or reading operation. Expanding writing or reading operation is of trivial work once the most basic writing or reading operation behaves correctly.

The write operation of 1 bit must stay for at least 70 µs. Based on DS18S20 datasheet and our experiment, the timing of writing which works in our implementation goes like following [11]:

Writing 0:

1) Host pulls down the bus for at least 65 µs.
2) Host releases the bus.
3) Client samples the bus.
4) Delay 5 µs for recovery of bus.

Writing 1:

1) Host pulls down the bus for at least 5 µs.
2) Host releases the bus.
3) Client samples the bus.
4) Delay 65 µs for recovery of bus.

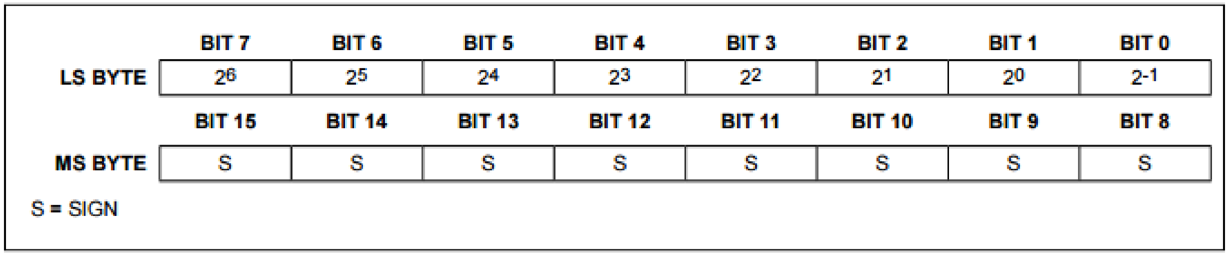
It is worth pointing out that writing an instruction is on byte-by-byte basis. So group every 8 write operation as one byte write is a good idea.

Read

Read operation has the following timing [11]:

1) Host pulls down the bus for 5 µs.
2) Hold 2 µs until the bus state becomes stable.
3) Host samples the data.
4) Host wait for 65 µs for next read cycle.

First 8 bits from sensor are low 8 bits. Second 8 bits are high 8 bits. The raw data is formatted in the following way [11]:



	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	S	S	S

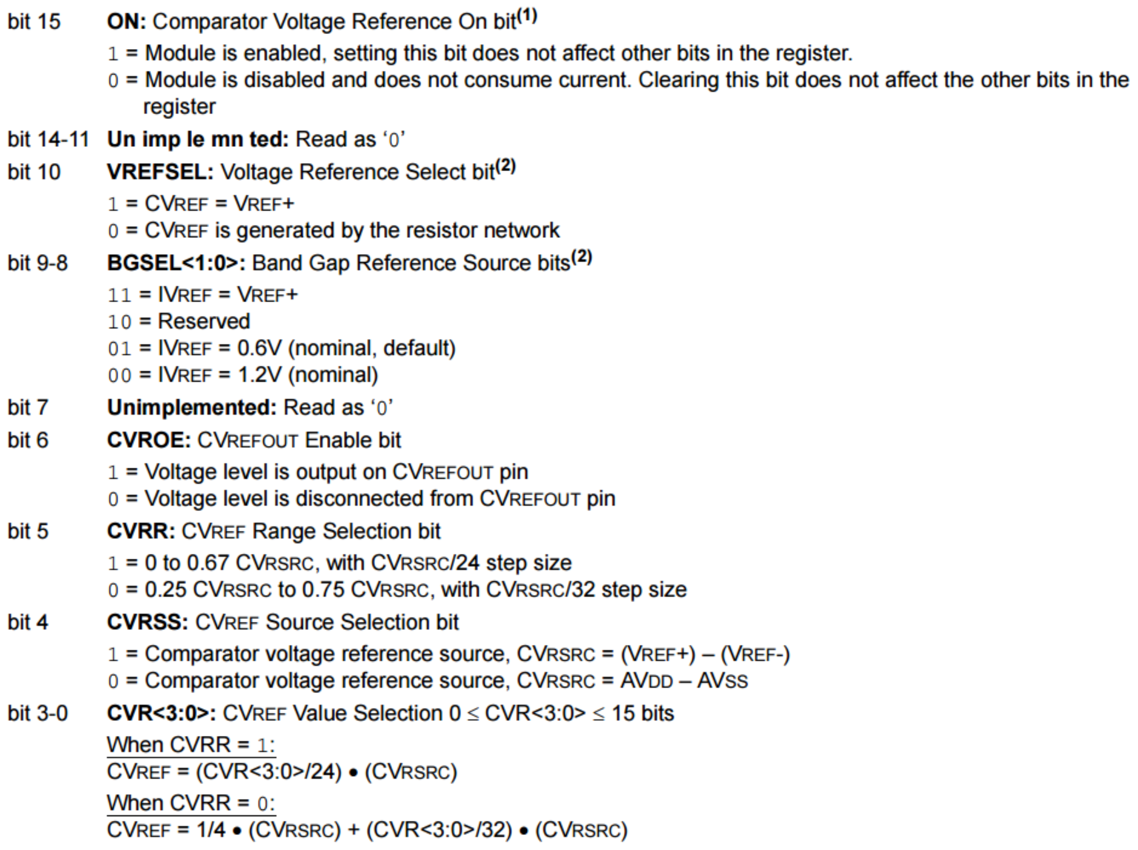
S = SIGN

Fig 3-3 Raw Data Representation

To interpret this raw data, shifting low 8 bits by 1 bit gives decimal part of the temperature, while least significant bit in low 8 bits makes fractional part. If the high 8 bits is 0, it means the temperature is under 0 degree in Celsius. To get more robust result, we added a software median filter which picks the median value of temperature every 3 cycles.

3.1.3 Audio Output Implementation

We discussed about the amplifying circuit in the previous chapter. However, the soundwave needs to be generated in the first place. We are using pin 25 of the PIC to output analog signal. The register that controls this pin is CVRCON. The following table explains the functionality of each bit of this register [12].



bit 15	**ON:** Comparator Voltage Reference On bit(1) 1 = Module is enabled, setting this bit does not affect other bits in the register. 0 = Module is disabled and does not consume current. Clearing this bit does not affect the other bits in the register
bit 14-11	**Un imp le mn ted:** Read as '0'
bit 10	**VREFSEL:** Voltage Reference Select bit(2) 1 = CVREF = VREF+ 0 = CVREF is generated by the resistor network
bit 9-8	**BGSEL<1:0>:** Band Gap Reference Source bits(2) 11 = IVREF = VREF+ 10 = Reserved 01 = IVREF = 0.6V (nominal, default) 00 = IVREF = 1.2V (nominal)
bit 7	**Unimplemented:** Read as '0'
bit 6	**CVROE:** CVREFOUT Enable bit 1 = Voltage level is output on CVREFOUT pin 0 = Voltage level is disconnected from CVREFOUT pin
bit 5	**CVRR:** CVREF Range Selection bit 1 = 0 to 0.67 CVRSRC, with CVRSRC/24 step size 0 = 0.25 CVRSRC to 0.75 CVRSRC, with CVRSRC/32 step size
bit 4	**CVRSS:** CVREF Source Selection bit 1 = Comparator voltage reference source, CVRSRC = (VREF+) – (VREF-) 0 = Comparator voltage reference source, CVRSRC = AVDD – AVSS
bit 3-0	**CVR<3:0>:** CVREF Value Selection 0 ≤ CVR<3:0> ≤ 15 bits When CVRR = 1: CVREF = (CVR<3:0>/24) • (CVRSRC) When CVRR = 0: CVREF = 1/4 • (CVRSRC) + (CVR<3:0>/32) • (CVRSRC)

Fig 3-4 Register Representation

And we set the register bits as:

Bit 15: 0, disable the module.
Bit 14-11: 000
Bit 10: 0, using inner reference voltage.
Bit 9-8: 00, using 0~1.2V bandgap reference.
Bit 7: 0
Bit 6: 1, enable the output of pin 25.
Bit 5: 1: 24 steps.
Bit 4: 0, use full range comparator.
Bit 3-0: keeps changing to output different levels of analog voltage.

So, in our implementation, bit15-bit4 is fixed at the value of 060 in hexadecimal, or 96 in decimal. Lowest 4 bits are playing a role as a 4-bit DAC. By changing the value of Lowest 4 bits properly, the expected sound wave could be generated.

However, direct digital synthesis for human voice is extremely hard, and our way of deriving data to send to CVRCON register is:

1) Pre-recorded human voice
2) Convert the format to wav file
3) Read wav file in MATLAB
4) Cut one audio rail data
5) Scale and round the data 0 - 15 in decimal
6) Add the converted value with 96 to coordinate with Bits 15-4
7) Exporting the data as an array that is usable by microcontroller
8) Use DMA channel to send data in the array to CVRCON register

For doing step 3) - 7), we revised the example code [13] in ECE 4760 as listed in the reference.

Based on our experiment, the best way to generate empty sound, instead of disable the DMA, is sending a constant value (in our case it's 107) to CVRCON. The idea is to add DC offset so little noise is raised to higher voltage, so the noise together with DC offset could be filtered all at once. Otherwise, noise at low voltage is hard to filter. This increases the power consumption but effectively suppresses the noise when no audio should be generated.

3.2 Software Implementation

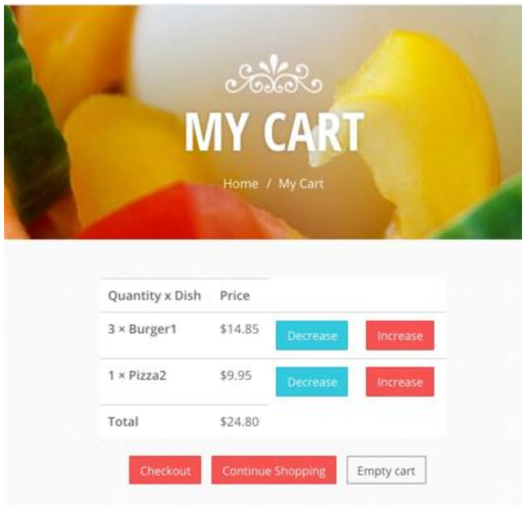
3.2.1 Website implementation

The layout of the website is implemented by Bootstrap, HTML, CSS and JavaScript. We implemented the functionalities by using Ruby on Rails [14]. The functionalities of the website are included as follows:

3.2.1.1 Add dishes functionality
After the user selects a dish, the system pop up the shopping cart window. Users can view their selected dishes here. The dish and the quantity the user set will be add to the cart.

3.2.1.2 Edit cart functionality

The line-item folder implements the functionality of editing the products in the shopping cart. By clicking the shopping cart button, users can change quantity of the products they put in the cart or delete products.

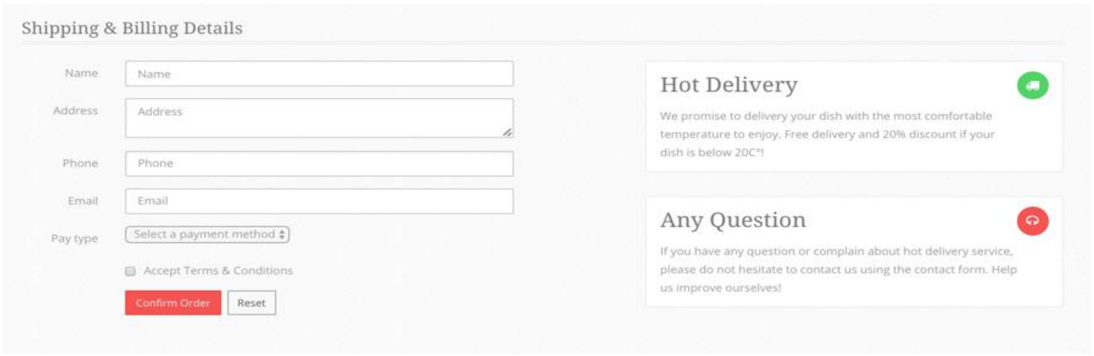


Quantity x Dish	Price		
3 × Burger1	$14.85	Decrease	Increase
1 × Pizza2	$9.95	Decrease	Increase
Total	$24.80		

Fig 3-5 Shopping cart page-User function

3.2.1.3 Place orders and manage orders functionality

The orders folder realizes the functionality of checkout for users and order management for admins. For the users, when they decide to checkout they must fill in a form with details of the order. It includes displaying the name, email, phone number, address. For the admins, they can view the list of orders and details and status of each order. The admin can delete orders or mark the order as shipped using the buttons on each record of the order list.


Shipping & Billing Details

Name	Name
Address	Address
Phone	Phone
Email	Email
Pay type	Select a payment method

☐ Accept Terms & Conditions

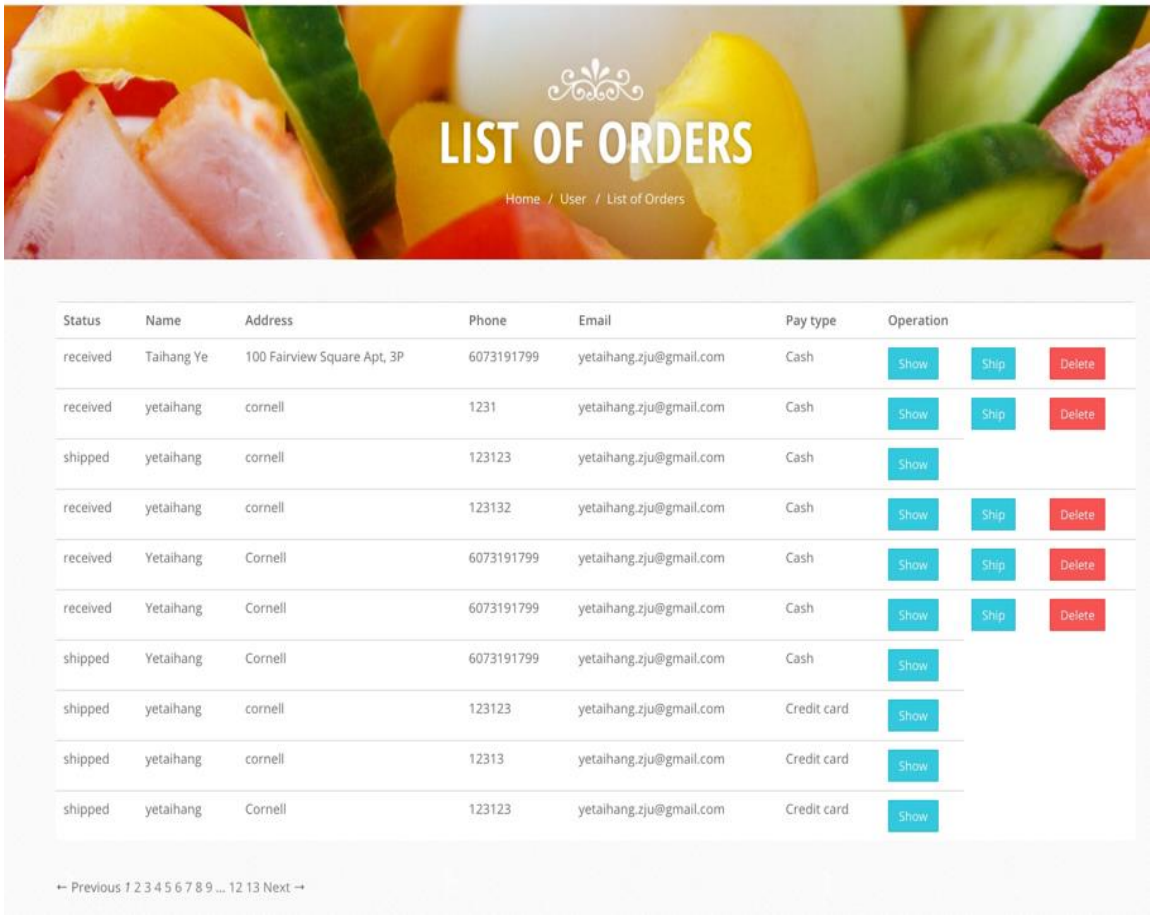
Hot Delivery

We promise to delivery your dish with the most comfortable temperature to enjoy. Free delivery and 20% discount if your dish is below 20C°!

Any Question

If you have any question or complain about hot delivery service, please do not hesitate to contact us using the contact form. Help us improve ourselves!

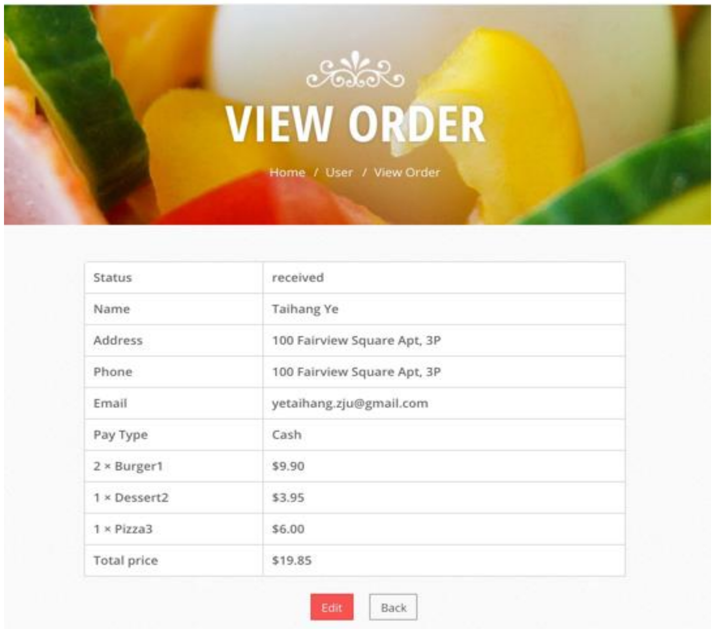
Fig 3-6 Place an order-User function



Status	Name	Address	Phone	Email	Pay type	Operation		
received	Taihang Ye	100 Fairview Square Apt, 3P	6073191799	yetaihang.zju@gmail.com	Cash	Show	Ship	Delete
received	yetaihang	cornell	1231	yetaihang.zju@gmail.com	Cash	Show	Ship	Delete
shipped	yetaihang	cornell	123123	yetaihang.zju@gmail.com	Cash	Show		
received	yetaihang	cornell	123132	yetaihang.zju@gmail.com	Cash	Show	Ship	Delete
received	Yetaihang	Cornell	6073191799	yetaihang.zju@gmail.com	Cash	Show	Ship	Delete
received	Yetaihang	Cornell	6073191799	yetaihang.zju@gmail.com	Cash	Show	Ship	Delete
shipped	Yetaihang	Cornell	6073191799	yetaihang.zju@gmail.com	Cash	Show		
shipped	yetaihang	cornell	123123	yetaihang.zju@gmail.com	Credit card	Show		
shipped	yetaihang	cornell	12313	yetaihang.zju@gmail.com	Credit card	Show		
shipped	yetaihang	Cornell	123123	yetaihang.zju@gmail.com	Credit card	Show		

← Previous 1 2 3 4 5 6 7 8 9 ... 12 13 Next →

Fig 3-7 List of Orders-Admin function

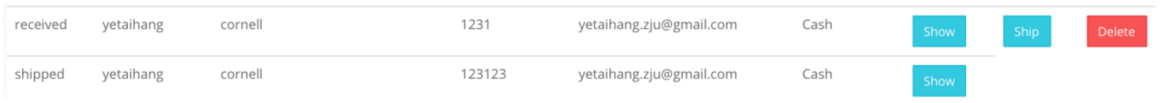


Status	received
Name	Taihang Ye
Address	100 Fairview Square Apt, 3P
Phone	100 Fairview Square Apt, 3P
Email	yetaihang.zju@gmail.com
Pay Type	Cash
2 × Burger1	$9.90
1 × Dessert2	$3.95
1 × Pizza3	$6.00
Total price	$19.85

Fig 3-8 View Order-Admin function

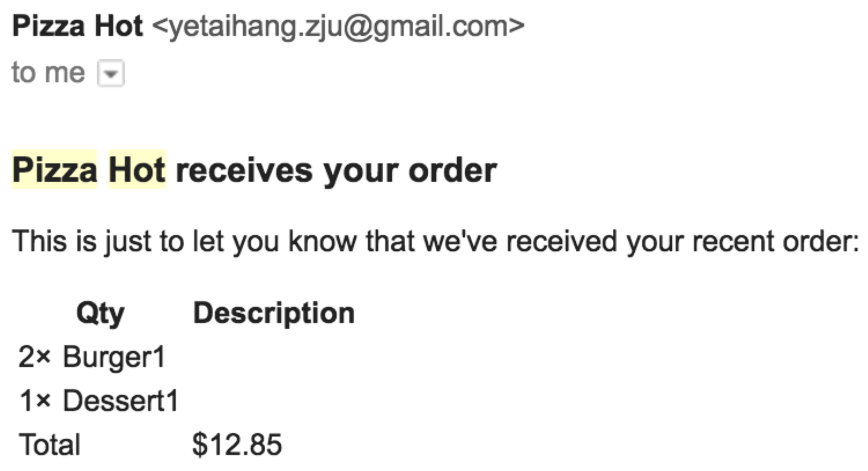
3.2.1.4 Order status notification functionality

After users place their orders successfully, users will receive a confirmation email, which notifies them their orders have been received. After the admins change the order status to shipped, the system also sends a notification email to notify them about status. The functionality is implemented in order_notifier folder.



received	yetaihang	cornell	1231	yetaihang.zju@gmail.com	Cash	Show	Ship	Delete
shipped	yetaihang	cornell	123123	yetaihang.zju@gmail.com	Cash	Show		

Figure 3-9 Order status



Pizza Hot <yetaihang.zju@gmail.com>
to me

Pizza Hot receives your order

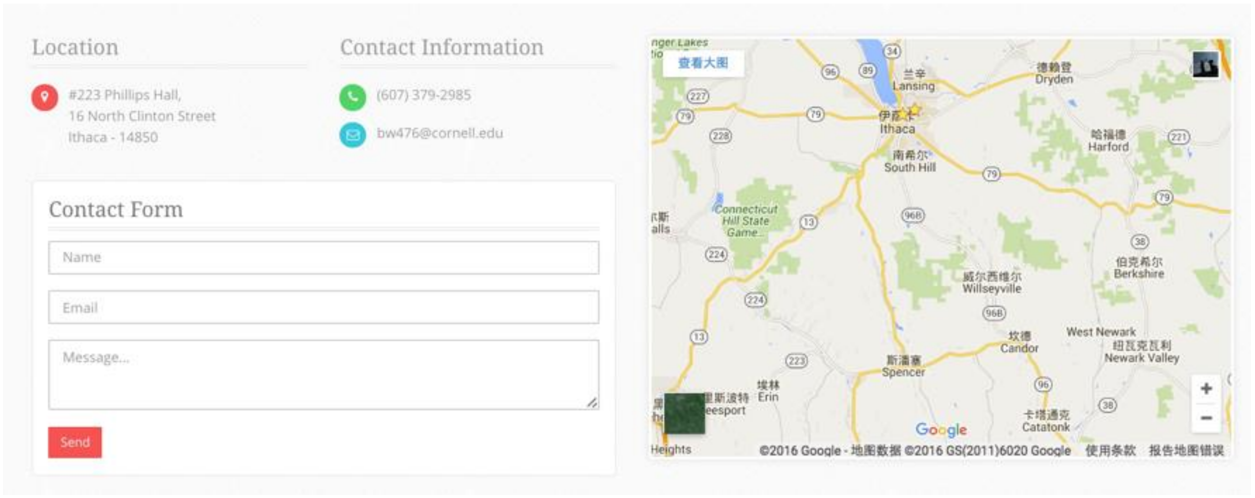
This is just to let you know that we've received your recent order:

Qty	Description
2× Burger1	
1× Dessert1	
Total	$12.85

Figure 3-10 Email notification

3.2.1.5 Contact restaurant functionality

The contact folder realizes the functionalities of submitting feedback to the restaurant for users and view feedback for the admins. If users successfully submit their feedback forms, the admin can view a list of feedback. They can choose to respond to the feedback then.



Location

#223 Phillips Hall,
16 North Clinton Street
Ithaca - 14850

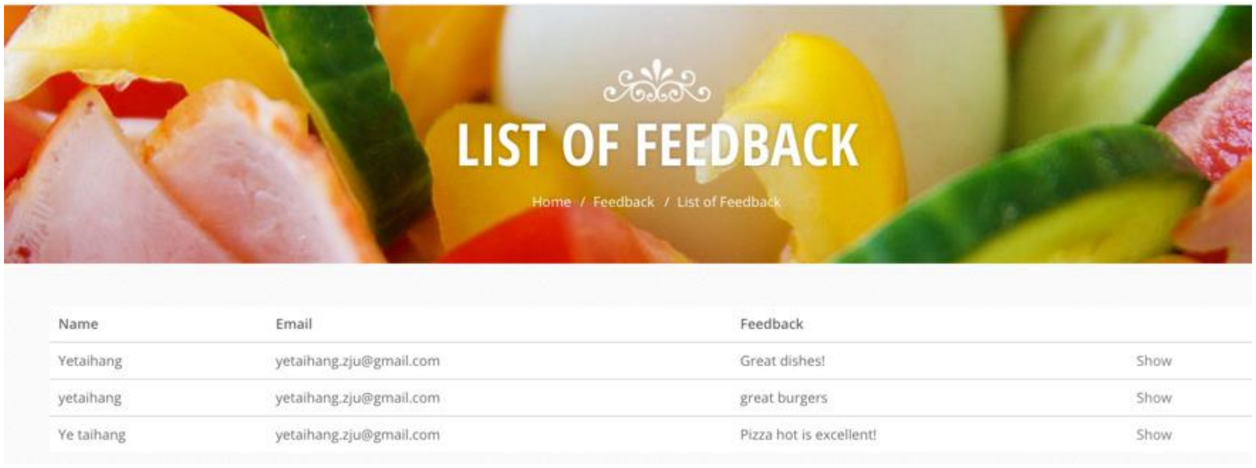
Contact Information

(607) 379-2985
bw476@cornell.edu

Contact Form

Name
Email
Message...
Send

Figure 3-11a Feedback page

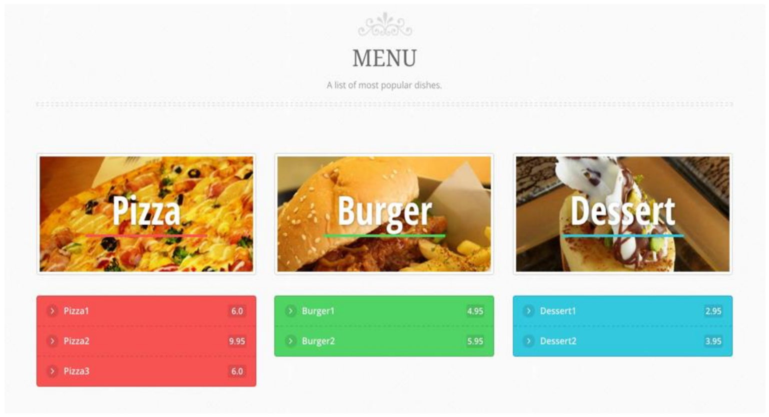


Name	Email	Feedback	
Yetaihang	yetaihang.zju@gmail.com	Great dishes!	Show
yetaihang	yetaihang.zju@gmail.com	great burgers	Show
Ye taihang	yetaihang.zju@gmail.com	Pizza hot is excellent!	Show

Fig 3-11b List of Feedback

3.2.1.6 Display dishes functionality

The products folder realizes the functionality of displaying menu and dishes dynamically. The admin can put new dishes into different categories when they add new dishes. Then dishes will be displayed under the categories they belong to dynamically.

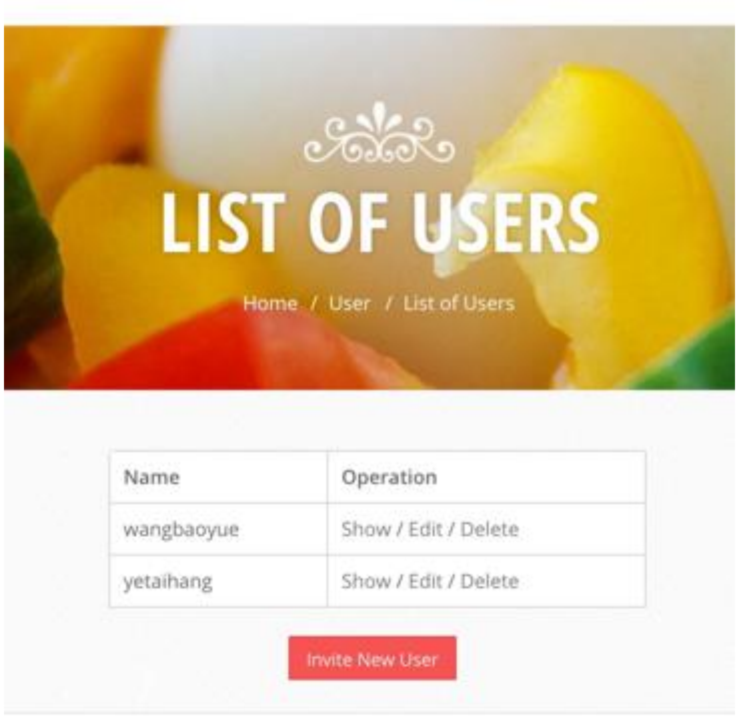


Pizza1	6.0	Burger1	4.95	Dessert1	2.95
Pizza2	9.95	Burger2	5.95	Dessert2	3.95
Pizza3	6.0				

Fig 3-12 List of Products

3.2.1.7 Manage admin functionality

The admin functionalities include three parts: invite users, show users list and delete users. The functionalities of admin are in users' folder, which includes edit.html.erb, index.html.erb, new.html.erb and show.html.erb


LIST OF USERS

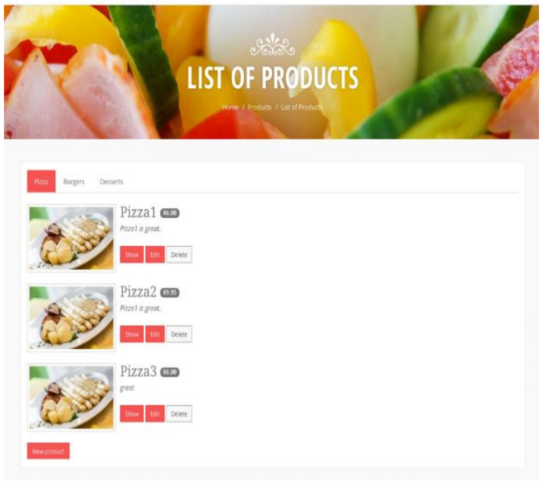
Home / User / List of Users

Name	Operation
wangbaoyue	Show / Edit / Delete
yetaihang	Show / Edit / Delete

Fig 3-13 Manage Admin

3.2.1.8 Add dishes functionality

Restaurant managers can add new dishes, edit or delete existing dishes.

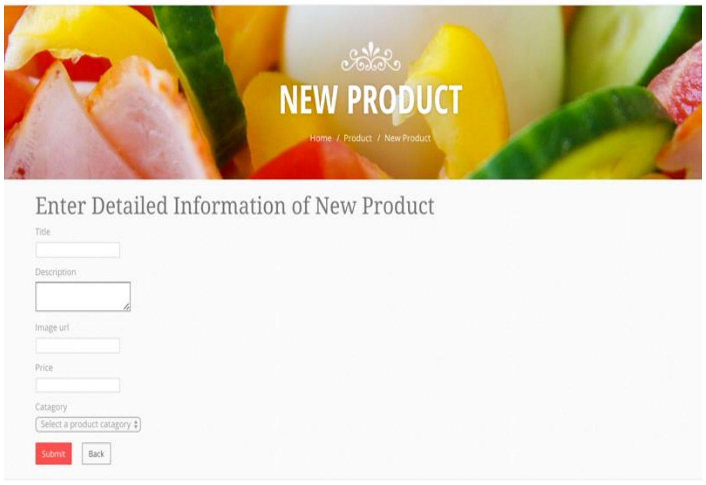

LIST OF PRODUCTS

Home / Products / List of Products

Pizza1 $6.00
Pizza1 is great.

Pizza2 $9.95
Pizza1 is great.

Pizza3 $6.00
great


NEW PRODUCT

Home / Product / New Product

Enter Detailed Information of New Product

Title

Description

Image url

Price

Catagory

Select a product catagory

Figure 3-14a Manage existing dishes

Figure 3-14b Add new dishes

3.2.2 Mobile application implementation

3.2.2.1 app.js functionality

The app.js plays an important role in the whole application as a manager. All the services are injected here. Meanwhile, it configures ionic platform and prepares cordova framework in order to start the application.

The app.js includes the following states and each of these states has corresponding templates in the template folder and controllers in controller.js.

1) “home”
2) “aboutus”
3) “contactus”
4) “menu”
5) “favorites”
6) “dishdetails”

3.2.2.2 service.js functionality

1) The service.js is used to fetch the data from server. It has six services as follows “localStorage”, “menuFactory”, “promotionFactory”, “corporateFactory”, “feedbackFactory”, “favoriteFactory”.

2) The menuFacroty service is aimed to get specific dish using dish id, and promotionFactory, corporateFactory, feedbackFactory are doing the similar things, using its id to request the information from remote server.

3) The favoritesFacoty has a little difference, it will use localStoage service to get the favorites customer has already selected and provides three functionalities “addToFavorites” which is store the new favorite dish into customer’s favorite list, “getFavorites” which just returns the favorite list and “deleteFromFavorites” which would delete the favorites from the customer’s favorite list.

3.2.2.3 controller.js functionality

1) IndexController: it will provide baseURL, leader information, promotion information and a loading message for the template, which will show those information on the template.

2) MenuController: it contains select, isSelected, addFavorite functionalities. Select is used to select menu category in the menu template. IsSelected is a function taking checkTab as parameter and checking which category is selected and return a boolean value (true or false). AddFavorite will add dishes to the favorite list and show the notification both on the screen and notification area using “cordovaLocalNotification” and “cordovaToast”.

3) DishDetailController: it has three parts. The first part is to provide a functionality showing or hiding the dish-detail-popover.html, which uses “ionicPopover” service. The second part is adding favorites to customer’s favorite list. The third part is showing or hiding dish-comment.html, which provides customers a form to submit their comment.

4) AboutusController: it provides the leaders information of the restaurant.

5) FavoritesController & favoriteFilter: it provides customer’s favorite list and the functionality that can delete the favorites by using “ionicPopup” service. For the favoriteFilter part, a filter is implemented to filter out favorites from all the dishes.

6) Login functionality: this functionality is implemented in three parts. The first part uses “ionicModal” to call a login template. The second part includes “closeLogin” and “login” is used to close or show the login template. The third part includes “doLogin” function is used to check username and password.

7) Reserve functionality: this functionality is almost the same as the login functionality described above.

8) Register functionality: this functionality is almost the same as the login functionality described above except that it provides additional functionalities to access device’s camera and gallery to take a picture or choose a exist picture in user’s gallery. It make use of “cordovaCamera” and “cordovaImagePicker” plug-ins to achieve these functionalities.

3.2.2.4 Templates

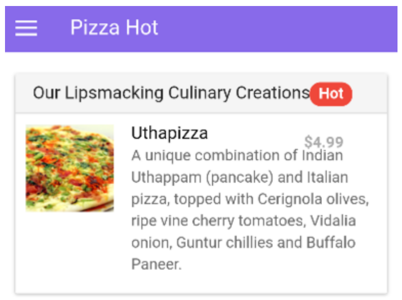

Pizza Hot
Our Lipsmacking Culinary Creations
Hot
Uthapizza
$4.99
A unique combination of Indian Uthappam (pancake) and Italian pizza, topped with Cerignola olives, ripe vine cherry tomatoes, Vidalia onion, Guntur chillies and Buffalo Paneer.

Fig 3-15 home.html

In the template, we will use “cards” from Google design guidelines [15] to display the pizza’s information. We design the card element with the headline on the top part, image as thumbnail on the left part, dish name, price and description on the right part. Also, we show a badge beside the headline.

Menu.html
In menu.html, it contains two parts. The first part is a list of menu items, which has a similar view with the card view in home.html. The second part is a tab region with four different categories. When choosing different category, select() function will be called and isSelected() function will change its boolean value. As the result, template will show the different items in list part.

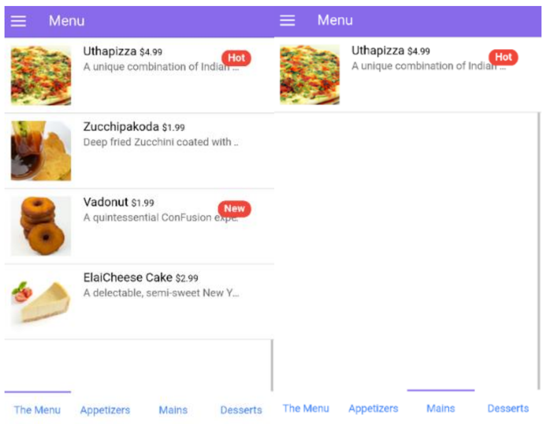

Menu
Uthapizza $4.99
A unique combination of Indian ...
Hot
Zucchipakoda $1.99
Deep fried Zucchini coated with ..
Vadonut $1.99
A quintessential ConFusion expe...
New
ElaiCheese Cake $2.99
A delectable, semi-sweet New Y...
The Menu
Appetizers
Mains
Desserts

Fig 3-16 Menu in Mobile App

When choosing one dish in the menu list, it can swipe left to show a add button so that customer can easily select this dish into favorites list. After adding the dish, it will notify the user with a short message on the main screen saying "Add FavoriteXXX" as well as on the notification area in the phone.

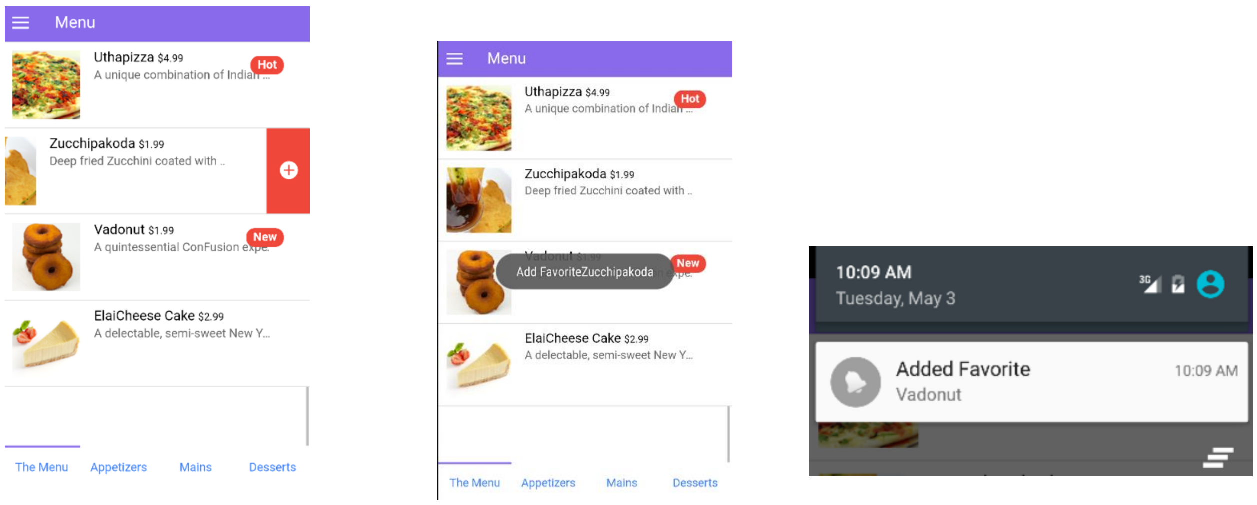
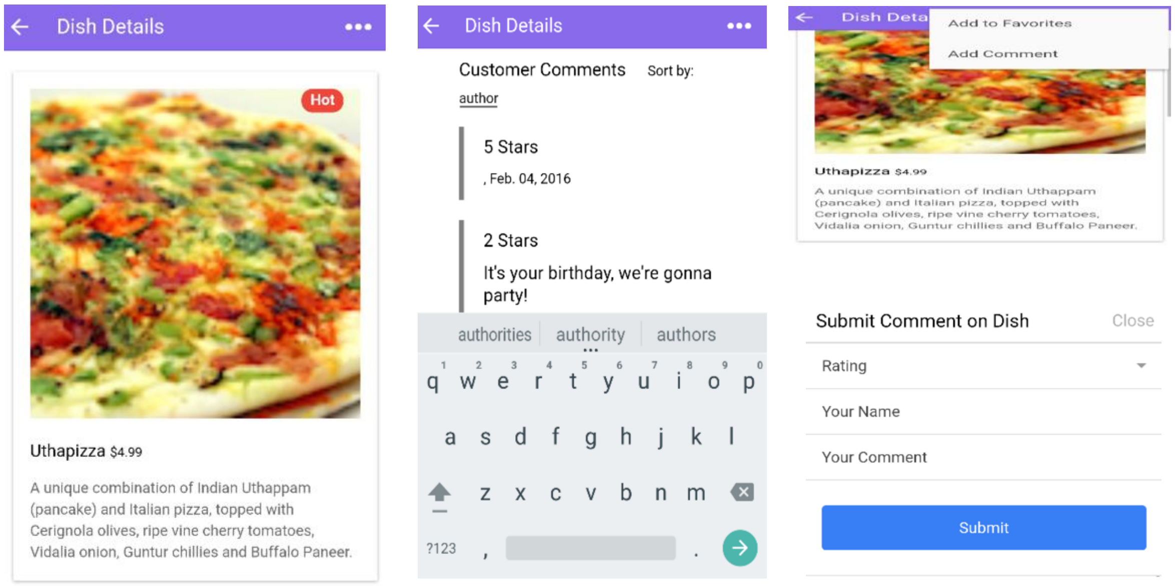


Fig 3-17 Add Favorite

Dishdetail.html
It has three parts. The first part uses "card" to display dish detail as following. The second part shows the comments of this dish which can sort by date, author and star. The third part includes two buttons, one is "Add to Favorites" which has similar functionality

described in menu.html, and the other is “Add comment” which will show a comment form for customers to comment this dish.


Dish Details

Hot

Uthapizza $4.99

A unique combination of Indian Uthappam (pancake) and Italian pizza, topped with Cerignola olives, ripe vine cherry tomatoes, Vidalia onion, Guntur chillies and Buffalo Paneer.

Dish Details

Customer Comments Sort by: author

5 Stars

, Feb. 04, 2016

2 Stars

It's your birthday, we're gonna party!

Dish Details

Add to Favorites

Add Comment

Uthapizza $4.99

A unique combination of Indian Uthappam (pancake) and Italian pizza, topped with Cerignola olives, ripe vine cherry tomatoes, Vidalia onion, Guntur chillies and Buffalo Paneer.

Submit Comment on Dish Close

Rating

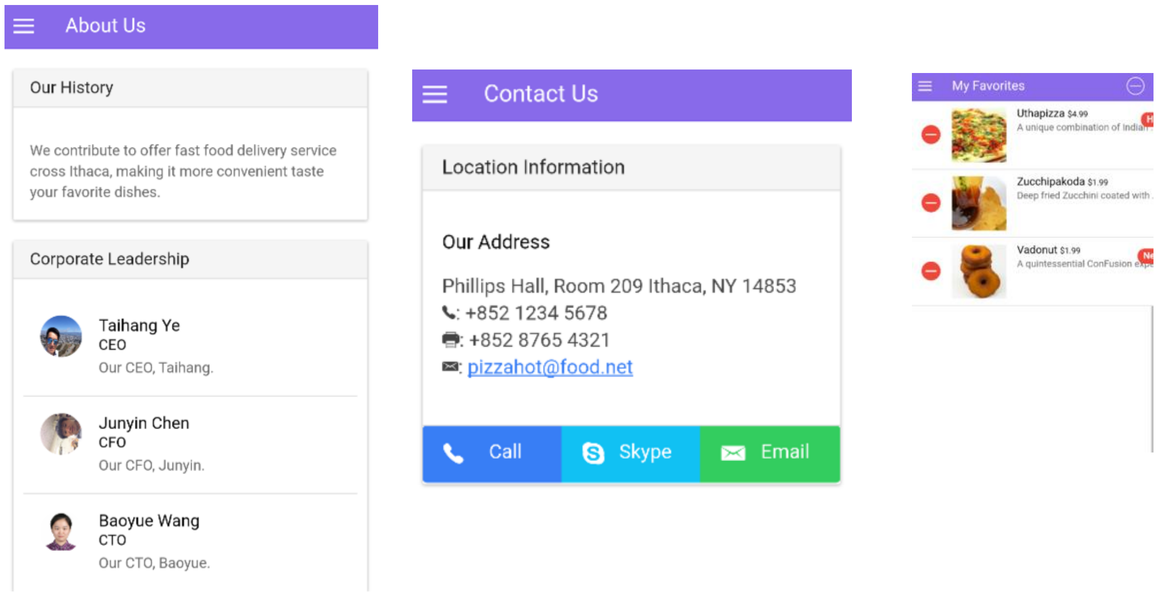
Your Name

Your Comment

Submit

Fig 3-18 Dish Detail

Aboutus.html & Contactus.html
These two templates will just show the contact and restaurant information.


About Us

Our History

We contribute to offer fast food delivery service cross Ithaca, making it more convenient taste your favorite dishes.

Corporate Leadership

Taihang Ye
CEO
Our CEO, Taihang.

Junyin Chen
CFO
Our CFO, Junyin.

Baoyue Wang
CTO
Our CTO, Baoyue.

Contact Us

Location Information

Our Address

Phillips Hall, Room 209 Ithaca, NY 14853
: +852 1234 5678
: +852 8765 4321
pizzahot@food.net

Call Skype Email

My Favorites

Uthapizza $4.99
A unique combination of India

Zucchipakoda $1.99
Deep fried Zucchini coated with

Vadonut $1.99
A quintessential ConFusion expe

Fig 3-19 About us

Favorites.html
This template shows the favorite dishes that customer selects using favoriteFilter in controller.js. It can remove the dishes by swiping the dish item left or click button on the top-right corner, which will show a delete button.

Login.html
This template shows the login form which include Username and password fields which is used for users to login in.

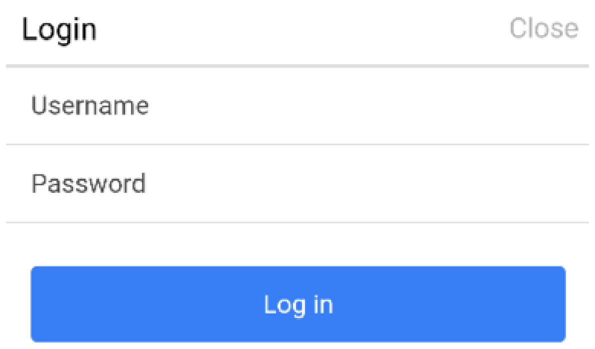


Fig 3-20 Login

Register.html
This template is to provide a registration functionality for user, including picture, firstname, lastname, username, telephone number and email address.

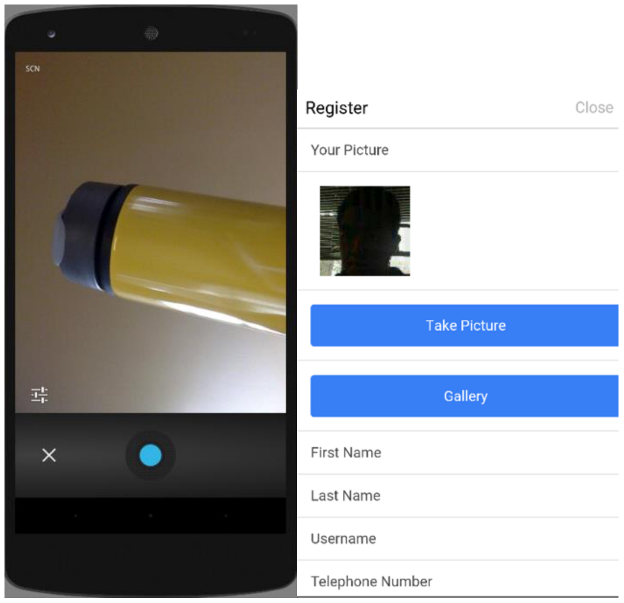


Fig 3-21 Register Page

Reserve.html
This template provides customers a reserve functionality to reserve a table in the pizza restaurant if they would like to go there and have hot pizza!

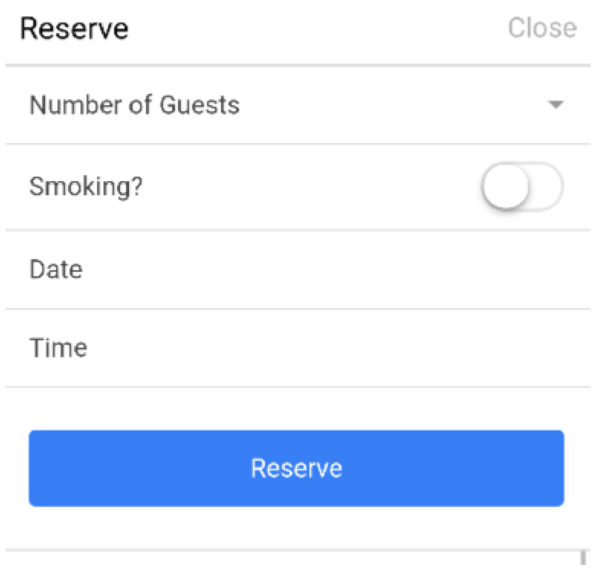


Fig 3-22 Reserve page

Sidebar.html
Here is navigation side menu template, which allows user having access to different templates in the application.

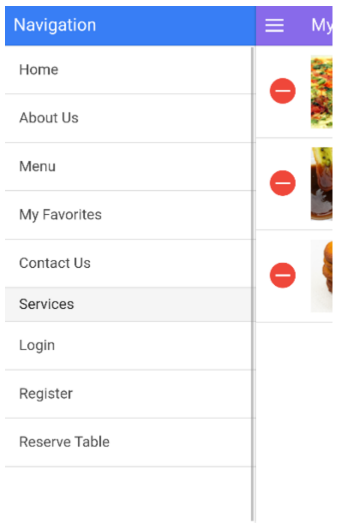


Fig 3-23 Sidebar page

3.2.3 Mobile application for delivery staff implementation

This mobile application combined with smart box helps delivery staff to navigate from one delivery place to another.

It has mainly three functionalities. The first one calls arduino.connect(IP, PORT, callback) to set up a TCP connection between the smart box (server) and this mobile application (client), which is used for transferring data from client to server.

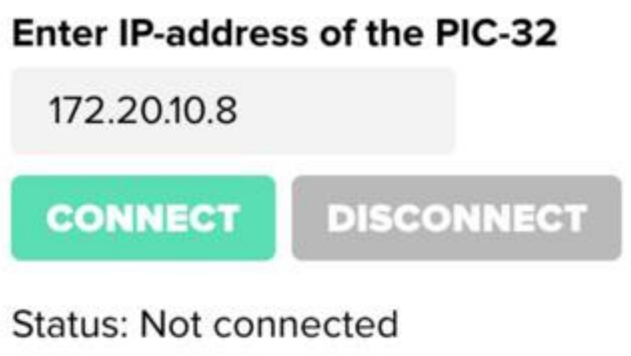


Fig 3-24 Driver's app, connection to WIFI

The second part gets start location and end location from the input, then call the google map direction API [16] to get the overview path from start location to end location.

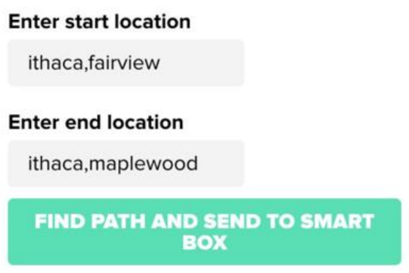


Fig 3-25 Sending the Route to the Box

The third part just transforms the overview path to the format that microcontroller needed and calls writeLocationData(location) to send data to Wi-Fi module of smart box with format "lat,40.000000,lon,45.000000\n".

4. Tests and Results

4.1 Hardware Test and Results

For hardware, interfacing WIFI module, GPS module, PIC begins at reading character level. It takes us long enough to find the frame error problem and WIFI module buffer problem.

For audio output, after converting the audio wave to an array, we plotted the simulation audio waveform in MATLAB to guarantee the wave we synthesized is correct. Below are simulation waveforms plotted in MATLAB, very close to sound wave, but from the wave, because the DAC is only 4 bits, we can anticipate the sound would not be very clear as well.

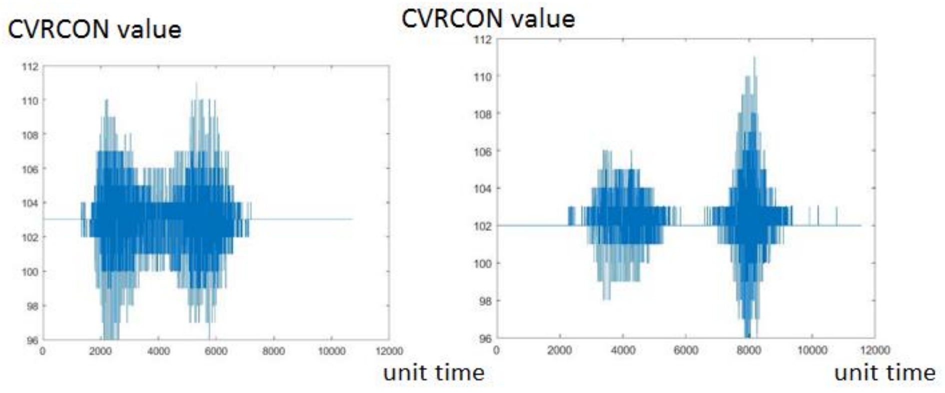
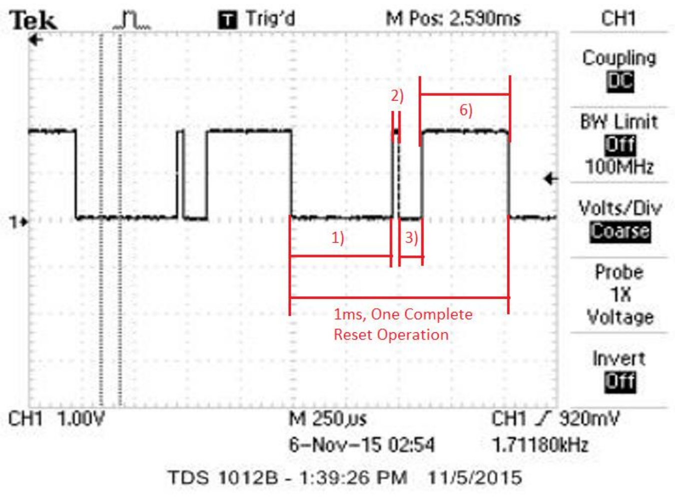


Fig 4-1 Audio simulation waveform

For temperature sensor, we are testing the wave generated by PIC IO port on oscilloscope to see whether it matches with waveform specified in the datasheet. And two images below are example images on oscilloscope.



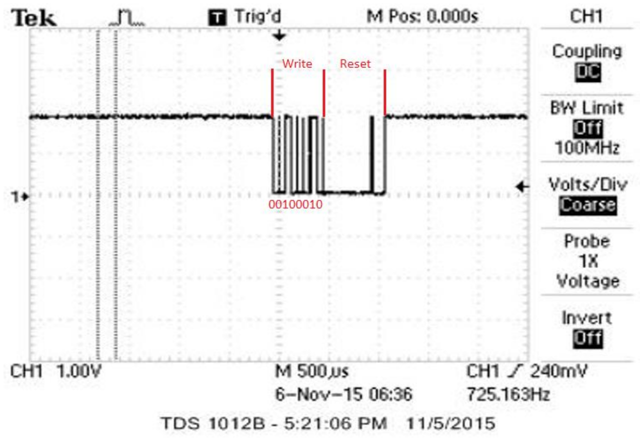


Fig 4-2 Reset and Write 0x44 Waveform

Recall from section 3.1.2, the reset timing of temperature sensor goes like:

1) Host (PIC32) pulls down the bus for at least 480 µs.
2) Host releases the bus for at least 15 – 60 µs.
3) If client (Temperature sensor) is present, the client pulls down bus for 60 – 240 µs.
4) The host samples the bus.
5) Client releases the bus.
6) An interval of at least 480 µs for the recovery of bus.

The first fig of fig 4-2 is the waveform for reset. Time unit is 250 µs one square. The waveform and data sheet specification matches.

The second fig of fig 4-2 is the waveform of write 0x44 and reset. Since two devices need to handshake before doing write, so the waveform is derived in a reset-write loop. In the write part of waveform, pulse width for each bit is about to be 70 µs. The glitch between two 0 bits being written is because of the 5 µs recovery delay time as mentioned in section 3.2. Note we are writing from the least significant bit, so the oscilloscope might read 00100010, but we need to read in the reversed order to know the actual data being written, which is 01000100 in binary, or 0x44 in hexadecimal. The right part of second fig of fig 4-2 is just one reset operation, compared to the first fig of fig 4-2, again it matches the specification well.

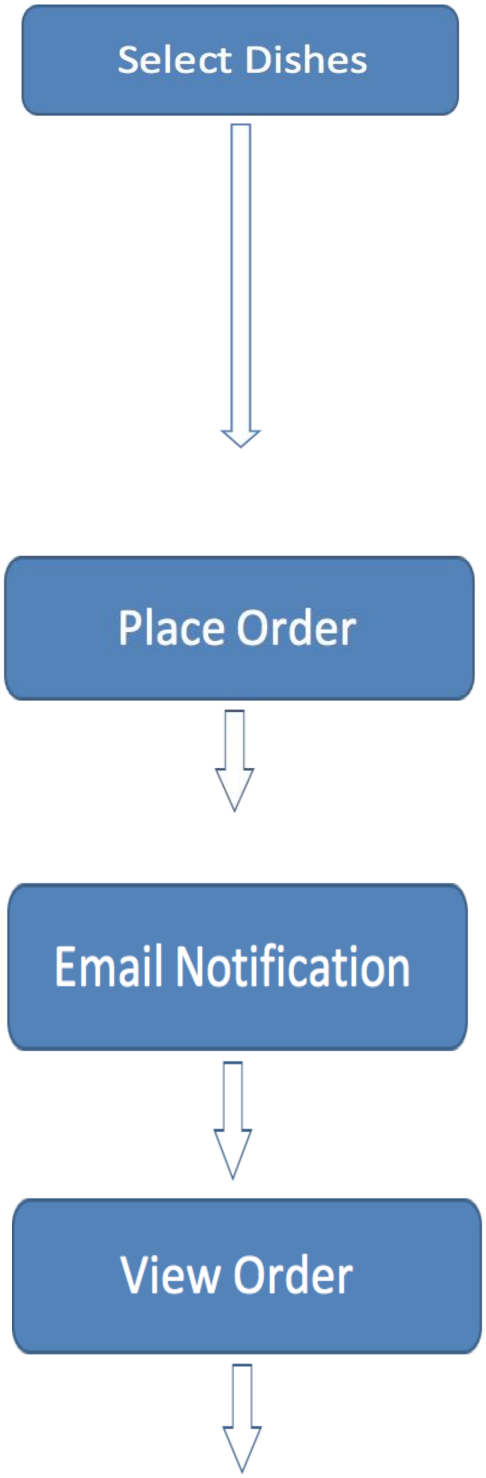
For other results, see 4.3 for integrated results.

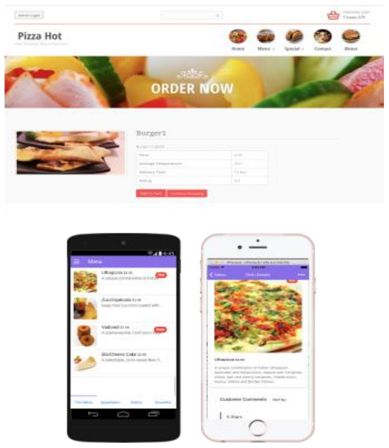
4.2 Software Test and Results

For the software, especially website we developed, we performed a bunch of test on that including model test, controller test, integration test and mailer test. These test data is in ruby test folder.

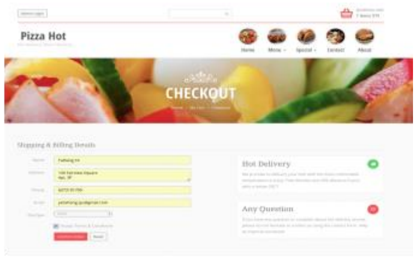
1) In the model test, we performed integrity constraint tests like “product attributes must not be empty” as well as syntax test like checking the image format.
2) In the controller test, we check the business logic of each function like create, update, show, delete is working correctly as expected.
3) In the integration test, we design a scenario to mimic the purchasing process of a customer and test if there is anything wrong during the whole process.
4) In the mailer’s functionality test, we test whether after receiving the order, the order notifier will send an email with subject “Pizza Hot receives your order” and other fields like mail from and mail to. We also write the similar test after shipping the order.

4.3 Integrated Results

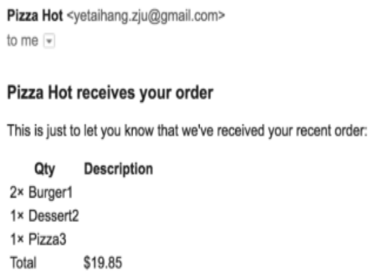




1. Select Dishes
Customer can select dishes both on our website and mobile applications on either Android or iOS platforms



2. Place Order
Customers can place their order entering their address, phone number, email then checkout



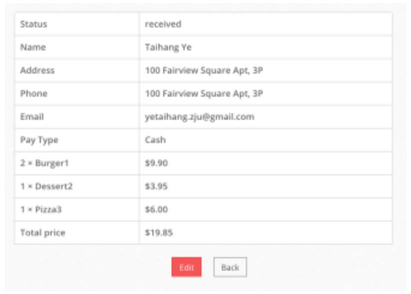
Pizza Hot <yetaihang.zju@gmail.com>
to me

Pizza Hot receives your order

This is just to let you know that we've received your recent order:

Qty	Description
2× Burger1	
1× Dessert2	
1× Pizza3	
Total	$19.85

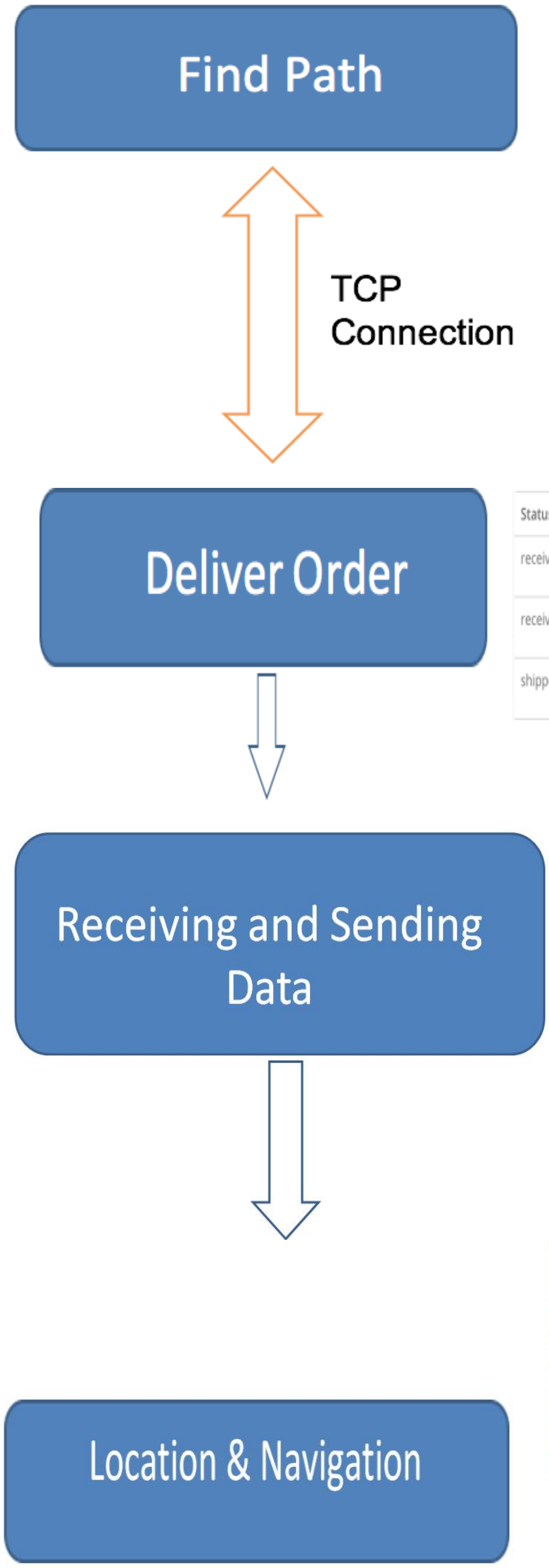
3. Email Notification
Right after customer checkout, they will receive a confirmation email to confirm what they have ordered, and

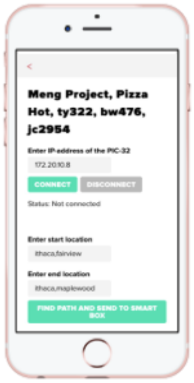


Status	received
Name	Taihang Ye
Address	100 Fairview Square Apt, 3P
Phone	100 Fairview Square Apt, 3P
Email	yetaihang.zju@gmail.com
Pay Type	Cash
2 × Burger1	$9.90
1 × Dessert2	$3.95
1 × Pizza3	$6.00
Total price	$19.85

4. View Order
The pizza restaurant owner can check the orders' detail.

5. Deliver Order
The pizza restaurant owner can decide when to ship the orders.





Meng Project, Pizza Hot, ty322, bw476, jc2954

Enter IP-address of the PIC-32

172.20.10.8

Status: Not connected

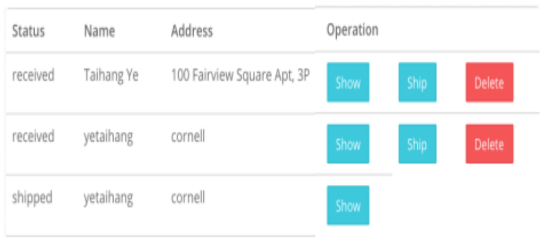
Enter start location

ithaca,fairview

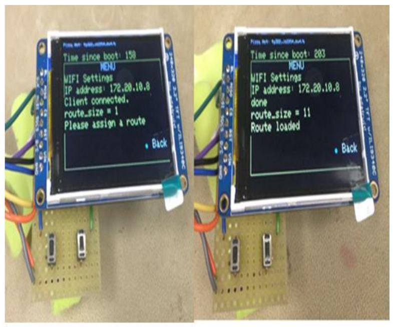
Enter end location

ithaca,maplewood

6. Find Path
Driver will enter the start and end location in the mobile application which will automatically get path data, set up TCP connection with WIFI module in microcontroller, and automatically send data to WIFI module.



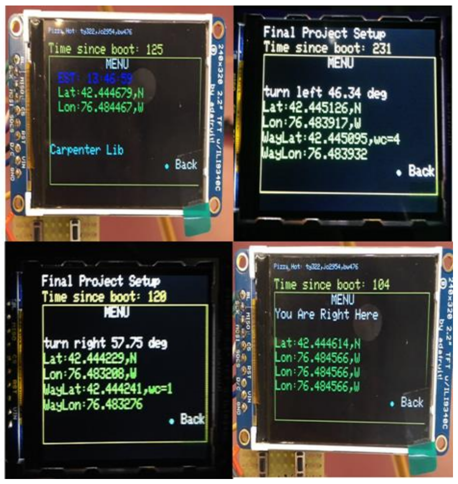
Status	Name	Address	Operation		
received	Taihang Ye	100 Fairview Square Apt, 3P	Show	Ship	Delete
received	yetaihang	cornell	Show	Ship	Delete
shipped	yetaihang	cornell	Show		



```
Time since boot: 150
MENU
WIFI Settings
IP address: 172.20.10.8
Client connected.
route_size = 1
Please assign a route
• Back
```

```
Time since boot: 203
MENU
WIFI Settings
IP address: 172.20.10.8
done
route_size = 11
Route loaded
• Back
```

7. Receiving and Sending Data
WIFI module would receive the data from driver's mobile and buffers the data. Then at about 100 characters or so, it sends the data through UART to microcontroller all at once.



```
Time since boot: 125
MENU
EST: 13:46:59
Lat:42.444679,N
Lon:76.484467,W
Carpenter Lib
• Back
```

```
Final Project Setup
Time since boot: 231
MENU
turn left 46.34 deg
Lat:42.445126,N
Lon:76.483917,W
WayLat:42.445095,wc=4
WayLon:76.483932
• Back
```

```
Final Project Setup
Time since boot: 120
MENU
turn right 57.75 deg
Lat:42.444229,N
Lon:76.483208,W
WayLat:42.444241,wc=1
WayLon:76.483276
• Back
```

```
Time since boot: 104
MENU
You Are Right Here
Lat:42.444614,N
Lon:76.484566,W
Lon:76.484566,W
Lon:76.484566,W
• Back
```

8. Location and Navigation
After microcontroller receives the routing data, it parses the stream, then it loads the route and it starts navigation.

5 Issues we meet

5.1 Hardware issues

Issue1: Concurrency

UART transmission must be non-blocking so PIC would not be blocked when other tasks need to be executed. We used protothreads for solving this problem.

Issue2: Frame Error

UART transmission of GPS module and PIC is not well compatible, after a while a frame error will occur. If the PIC does not check the frame error and clean the error bit, UART on PIC would be disabled.

Issue3: abs vs absf

In C program, abs function in built-in C library returns an integer while absf returns a floating number. When we are trying to determine whether we are close enough to a waypoint, if anyone is using abs to compute the distance would always have 1 degree offset and the navigation would never work.

Issue4: Interfacing WIFI UART with PIC32

Though the UART of WIFI is 3.3 V, its output current is strong, and under certain circumstances pins on PIC32 would be burnt. Thus a resistor of 3-5K to limit the current is strongly recommended.

Issue5: Noise suppression

Though DC value would be filtered by our amplifying circuit, DC offset must be added to raise noise to higher voltages so that noise could be properly filtered.

Issue6: WIFI module configuration

WIFI module is unable to receive the character from mobile phone and immediately send it to PIC. A buffer must be placed for WIFI module to receive 100 characters or so, then the WIFI module could send characters in the buffer all at once.

Issue7: Optimization level of PIC compiler

Sometimes pieces of codes like delay would be optimized away by the compiler. Also, reading operation might always report the same value. Setting the compiler optimization level to 0 and setting key reading variable as volatile type could solve the problem.

Issue8: Waypoint Implementation

Our navigation is based on waypoints, and our original implementation was inserting waypoint by hand. It makes sense in local area, but it’s very tedious and time consuming, even connecting Engineering Quad is very hard. However, it gives intuition of how navigation should be implemented. Then we move on to call Google Map's API for computing the route and allocate memory dynamically. This saves memory and is much more efficient. However, routes computed by Google Map might be invalid if original place and destination is very close, and it contains more waypoints than you actually need. For very local connection, for example, from Phillips Hall to Carpenter Hall, manually inserting waypoints and doing navigation statically might be more efficient.

5.2 Website issues

Issue1: Front-end pipeline configuration

Solution: There are some tutorial online taught me how to configure the whole pipeline for the frontend using bootstrap, jQuery, AngularJs and other items like font-awesome.

Issue2: User cases not defined clearly leads to redo the whole project.

Solution: First, we should ask ourselves what are the user cases, which would help us to design the whole project. While doing the project, there must be more idea, so we can combine them into the existed framework. At first, we just consider the dishes and orders, as project progressing, we found we can add more functionalities like feedback, my favorite dish list and reservation a table in restaurant.

Issue3: Email notification issue, like Google blocks out auto login and some other tricky problems like "undefined method protect_against_forgery?"

Solution: try to setup google account not to block this application and substitute the button_to with link_to solve "protect_against_forgery?" problem.

5.3 Mobile applications

Issue1: Don’t know how to use cordova camera and cordova image picker services

Solution: check API again and find how to trim the picture by using specific options.

```
options = {
        quality: 50,
        destinationType: Camera.DestinationType.DATA_URL,
        sourceType: Camera.PictureSourceType.CAMERA,
        allowEdit: true,
        encodingType: Camera.EncodingType.JPEG,
        targetWidth: 100,
        targetHeight: 100,
        popoverOptions: CameraPopoverOptions,
        saveToPhotoAlbum: false
 };
```

Issue2: Don’t know how to add a block of buttons

Solution: Check online tutorial and solutions to similar practice [17].

Issue3: Google map direction API returns thousands of data point in a json object, don’t know how to limit the number of data points.

Solution: After checking all the attributes in json object, there is “overview_path” attributes which contains primary points, so using this attributes and send it to microcontroller. Below is an example of overview_path, which contains much less waypoints.

overview_path: [{"lat":42.47437,"lng":-76.54163000000001},{"lat":42.474180000000004,"lng":-76.542},{"lat":42.4741,"lng":-76.54223},{"lat":42.47401000000001,"lng":-76.54288000000001},{"lat":42.473870000000005,"lng":-76.54345},{"lat":42.473760000000006,"lng":-76.54362},{"lat":42.473710000000004,"lng":-76.54365},{"lat":42.473710000000004,"lng":-76.54371},{"lat":42.47361,"lng":-76.546},{"lat":42.47359,"lng":-76.54618},{"lat":42.473850000000006,"lng":-76.54627},{"lat":42.47417,"lng":-76.54644}….]

Issue4: Evothings platform is updated recently, which makes TCP client on phone fail to make a socket connection with TCP server on smart box.

Solution: There isn’t an effective solution to this problem, and the only way to solve this is use laptop to try to set up TCP connection to server to see if it is working or not. If it is working, which means Evothings platform version update is the only possibility causing this problem. In the future, we should not arbitrarily update platform of mobile application, especially for the beta version.

6. Future work

The system could get improved in several aspects in the future. The system could be improved by enabling order delivery for many restaurants and customers. By now, our project is able to deliver one order to one customer at a time. Next step could be for one restaurant, the website sends several orders it receives in a time period. Then the hardware system could figure out the best route based on customer's location to deliver several orders in one deliver process. A further improvement could be for several restaurants which joined Pizza Hot, the hardware system would be able to collect orders from them in a time period and deliver in one process. This could be more efficient. In addition, the hardware system could be more portable and user-friendly. In each delivery process, the delivery man picks several orders and get all of the information in the hardware system. After an order is delivered successfully, the system would have the ability to pick up another order automatically from the selected orders in this process. In order to increase efficiency dramatically, the system would have the ability to choose the best route of delivering all the orders. In addition, the system would be able to send back real-time data to customers. In this way customers could get to know where their dishes are and what the temperature of the food.

7. Conclusion

Pizza Hot is a highly integrated and advanced pizza order and delivery system. From customers' perspective, they have multiple choices to access our service either by website, mobile application or telephone and get hot pizza in a relatively shorter time. From the restaurant manager's perspective, they will get orders information easily from the management console, have access to add or modify the dishes' detail and manage the feedback from customers. From the delivery person's perspective, they will get navigation help from smart delivery box's audio output and not worry the temperature of pizza in the delivery box.

As described in executive summary, we realized most of the functionalities we designed at the beginning and the passed tests successfully.

In conclusion, Pizza Hot food delivery system would provide a convenient, thoughtful and reasonable way for customers to order pizza.

Acknowledgement

We would like to thank Professor Joseph Skovira for his instruction and advice on our project. His insights always help us make our project unique and increase the commercial value. Professor Skovira always response to our question with great enthusiasm. We really appreciate his instruction to our project.

We would also like to thank Professor Bruce Land for his effort in helping us debug. He has a very incisive intuition towards the bug and his advice is always very constructive.

Reference

[1] Microchip. (2016, April 11). *PIC32MX1XX/2XX Family Data Sheet:* [Online] Available: http://ww1.microchip.com/downloads/en/DeviceDoc/60001168J.pdf

[2] Microchip. (2006, Dec. 08). *MCP1727 Data Sheet*: [Online] Available: http://ww1.microchip.com/downloads/en/DeviceDoc/21999b.pdf

[3] Adafruit. (2016, May 17). *Adafruit Ultimate GPS:* [Online] Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ultimate-gps.pdf

[4] NMEA 0183. (2016, April 22). In *Wikipedia, the Free Encyclopedia*. [Online] Available: https://en.wikipedia.org/w/index.php?title=NMEA_0183&oldid=716577264

[5] GPSInformation.org. *NMEA data* (v3.01) [Online]: http://www.gpsinformation.org/dale/nmea.htm

[6] S. T. Mahbub. (2014, Oct. 30). *Tahmid's Blog: Interfacing a color TFT display with the PIC32MX250F128B:* [Online] Available: http://tahmidmc.blogspot.com/2014/10/interfacing-color-tft-display-with.html

[7] Apache Cordova. (2015). *Apache Cordova API Reference:* [Online] Available: https://cordova.apache.org/docs/en/2.4.0/

[8] Evothings. (2016 May 2) *Evothings Studio*: [Online] Available: https://github.com/evothings/evothings-studio

[9] Adam Dunkels. (2005 April 08). *Protothreads*: [Online] Available: http://dunkels.com/adam/pt/

[10] B. R. Land. (2015 Oct. 14). *Cornell University ECE4760, ProtoThreads and Timers , PIC32MX250F128B*: [Online] Available: https://people.ece.cornell.edu/land/courses/ece4760/PIC32/index_Protothreads.html

[11] Maxim Integrated Products, Inc. (2015). *DS18S20 High-Precision 1-Wire Digital Thermometer Datasheet:* [Online] Available: http://datasheets.maximintegrated.com/en/ds/DS18S20.pdf

[12] Microchip. (2011 Nov. 29). *Comparator Voltage Reference Manual*: [Online] Available: http://ww1.microchip.com/downloads/en/DeviceDoc/61109G.pdf

[13] B. R. Land. (2015 Aug. 12). *Cornell University ECE4760, Vref as a DAC, PIC32MX250F128B:* [Online] Available: https://people.ece.cornell.edu/land/courses/ece4760/PIC32/index_VREF_DAC.html

[14] Rubyonrails.org. *Ruby on Rails Guides (v4.2.6)*. [Online] Available: http://guides.rubyonrails.org/

[15] Google. *Google design guiline*: [Online] Available: https://www.google.com/design/spec/components/cards.html

[16] Google. *Google Map Direction API:* [Online] Available: https://maps.googleapis.com/maps/api/js?key=AIzaSyApQwIu4qtwE2uQGjYjaNLFa5_DwrgYNNI&signed_in=true&callback=initMap

[17] Leuo-Hong Wang. *Hybrid Apps Development (v3.3):* [Online] Available: http://hybridap.blogspot.com/2015/04/hybrid-apps33-buttons.html

Appendix

Our Code is at github: https://github.com/taihangy/PizzaHotFinalVersion
The code structure:
1. Website Code: Website_Ruby_On_Rails
2. Mobile Application for customers: Mobile_App
3. PIC32 smart box code: PIC32_Smart_Box
4. PIC32 driver application: PIC32_Driver_App