Availability Prediction Based on Multi-Context Data

DESIGN DOCUMENT

Group: sdmay19-33

Client: Goce Trajcevski

Team members:
Justice Wright - Report Facilitator
Shane Impola – Scribe
Noah Chicchelly – Meeting and Communications Facilitator
Nick Schmidt – Software Systems Engineer
Tristan Anderson – Network Systems Engineer
Brendon McGehee – Hardware Systems Engineer

Team email: sdmay19-33@iastate.edu

Team Website: https://sdmay19-33.sd.ece.iastate.edu

----------Temporary----------
Document: http://seniord.ece.iastate.edu/resources/Design_Document.pdf
Rubric: http://seniord.ece.iastate.edu/resources/Design_Document_Rubric.pdf
---

Version 1.0
Revised:October 12, 2018

Table of Contents

List of Figures

List of Tables

List of Definitions

MySQL - My Structured Query Language
AWS - Amazon Web Services
VM - Virtual Machine

1 Introductory Material

1.1 Acknowledgement

The Availability Prediction team would like to thank Dr. Trajcevski for all of his technical and project advice given throughout the duration of this project.

1.2 Problem Statement

Currently in a restaurant atmosphere there are multiple data sources that could be better used to predict a better recommendation for table wait time.

Our goal is to create a device that can collect and organize data from a restaurant environment such as, how long people take to eat, how many people are sitting at a certain table, and what tables are being used. The data can then be analyzed to answer many questions about the restaurant.

1.3 Operating Environment

The expected operating environment of this project will be inside of restaurants. The system should not be exposed to any harsh weather conditions because it will remain inside in a room temperature environment. The only condition that the system might be exposed to is some dust and debris over a long period of time.

1.4 Intended Users and Intended Uses

Our intended final project users are going to have a wide range of technical knowledge. The individuals that are going to be using this are both customers and employees of the restaurant that it will be implemented in. Although customers and employees will be using this in an app based form, the purpose of the app for each will be very different.

The employees will be using this app to input resturant data such as when the food has been given to the table, when the food has been removed from the table, and when the check has been given to the table. This information will then be used to better predict table wait time, which brings me to customer use. The customer will use the app to see available tables as well as receive restaurant wait times.

1.5 Assumptions and Limitations

Assumptions:
The final product will only be implemented indoors - the hardware of the project does not need to stand up to any harsh weather conditions.

Hardware component will not have any power limitations - the hardware component will have access to electrical outlets.

Limitations:
Sensors need to be unnoticeable inside a restaurant - The sensors used to collect data need to be implemented into a restaurant setting without impeding the normal operation of the restaurant.
The application needs to be usable by all types of technical backgrounds - The app that we will be implementing in our project can not be overly difficult to use because people of all technical backgrounds will be utilizing the application.

1.6 Expected End Product and Other Deliverables

The final product of this project will include sensor with a microcontroller to collect data about tables, a networking component to organize and analyze the collected data, and an app component for either an employee or a customer. The components need to be able to communicate with each other but they do not need to communicate individually. The sensor component needs to be able to communicate with the networking component and the networking component needs to be able to communicate with the software component.

2 Specifications and Analysis

2.1 Proposed Design

We have started implementation on the 3 main states of our project. Hardware, software/database, and front-end application. For hardware we are finding the best possible devices and sensors to take in the data we require in the fashion we want. There are many possibilities when it comes to this, and we are balancing out our options the best we can. As for database and front-end we are in the process of designing the architecture and communication relationship that those will have. Once we have a relationship well defined, implementation will ensue.

Our general design will be to collect data with hardware, send it to our AWS, organize and analyze it with algorithms within our MySQL database, and then output it to our application. Queries involved will be based on the functionality of our front-end. It will vary on the customer version of the app, and the employer version. Basic functions like “what tables are full/empty” and “how long is the wait” will be our base queries to accomplish.

2.2 Design Analysis

While our project is very much in the beginning stages, we have made some palpable progress with hardware, and a lot of research has been done for server/db and front-end. Hardware has made a lot of progress, which is what we needed done first. There was a large focus on this area first, because without data, it will be hard to do much of anything. Once our hardware is in place, we will be able to establish communications with it and start taking in data and generate queries.

We have tested multiple sensors and they are all giving us significant data and are working successfully. Our job now is to determine the best cost, efficiency, communication, scalability, etc. for the hardware we will use in the future.

As we continue, we will need to get more hard progress completed before taking significant steps forward. Getting a rough front-end up that can operate queries, a server that can handle them, and hardware that can give us data, we will be on the way to expanding our scope and really making progress.

Strengths with our design flow is keeping a narrow scope while we start work on our project. As we get successful states completed, we will be able to expand scope slowly and really broaden the horizons for this project. Weaknesses are that off the bat we had to find the starting scope, so things were a bit slow to start. However, now that we have found our starting scope, we can really grow quickly.

3 Testing and Implementation

3.1 Interface Specifications

To test our project, we need to test the software and hardware interfaces to make sure they work as expected and function reliably in a variety of situations. There multiple interfaces to test in our project, including the human-sensor, sensor-controller, controller-server, server-database, app-server, and human-app (user) interfaces. This means we need to test that the guests at a restaurant will properly actuate the sensors (human-sensor), the sensors can properly communicate with the arduino (sensor-controller), the arduino can communicate data to the server (controller-server), the server can store and retrieve data in the database (server-database), the app and server can communicate requests and information with each other (app-server), and the humans using the app can get and view any information they request (human-app).

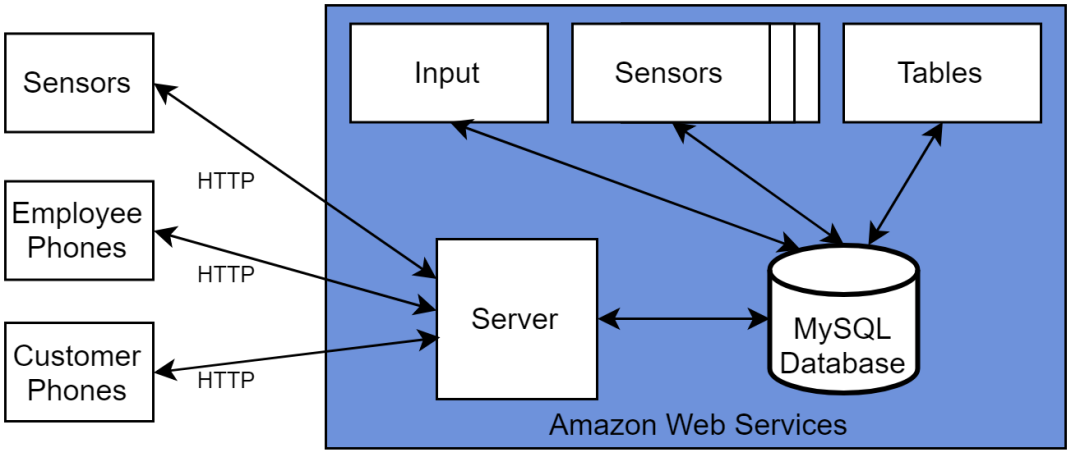

Figure 1 Task Approach

The server will be sending queries and data to the database using MySQL. We will be implementing a RESTful API on the server, so the controller-server and app-server interfaces will be using HTTP for communications. The arduino will process data from the sensors they are actuated by the users.

3.2 Hardware and Software

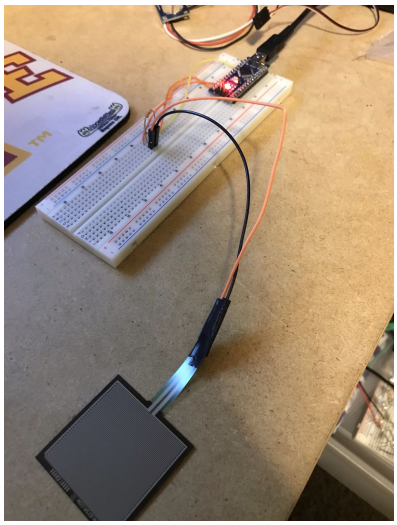

Figure 2 Force Sensing Resistor

Force sensing resistors exhibit a decrease in resistance as increasing force is applied to the the surface of the resistor. This was one of our primary occupancy sensors with the intent to be placed under the seats of diners.

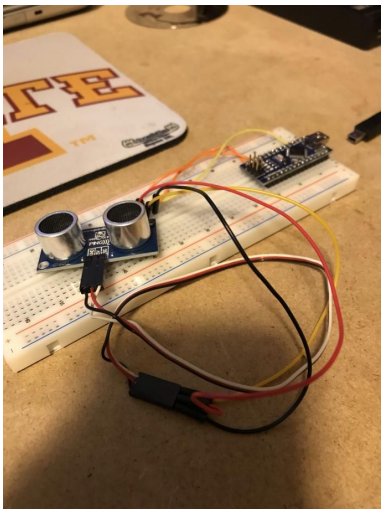

Figure 3 Ping Sensor

The ping sensor sends out a sonic pulse and waiting to to hear when the echo from the initial pulse reaches the device. Based on the amount of time between initial pulse and the resulting echo the distance of the object causing the pulse to echo back can be determined. This sensor was tested alongside a mock table for occupancy determination purposes.

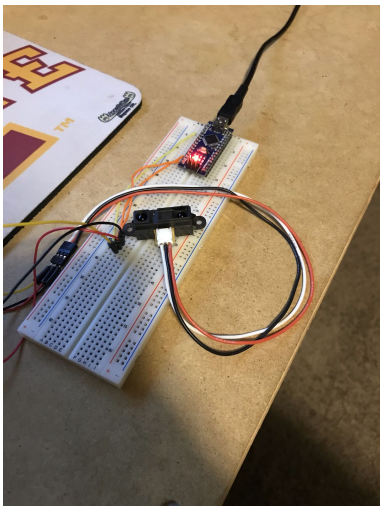

Figure 4 IR Sensor

An IR sensor works very similarly to a ping sensor. It has both an IR transmitter and an IR receiver. The transmitter emits radiation and if an object is struck by this it will reflect this radiation back towards the receiver. The more intense the radiation observed by the receiver the more closer the object. These are another occupancy sensor used, and were placed under the table facing the diners.

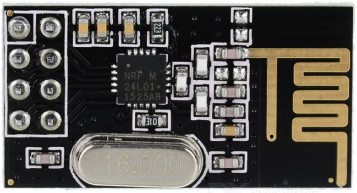

Figure 5 Transceiver Module

These transceiver modules transmit and receive data in the 2.4-2.5GHz band. They will be implemented on each arduino unit to send data from the sensors to the aggregation unit.

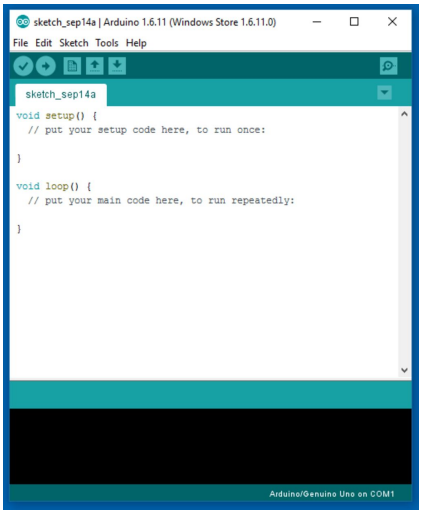

Figure 6 Arduino IDE

Arduino IDE is a piece of software that allows us to directly program our individual arduino microcontrollers. This allows us to instruct it on how to read and store sensor data as it becomes available.

3.3 Testing

Unit Testing:(individual component testing)

Tested Unit	Requirements	Status
1A Force Sensitive Resistor	Detect occupancy with a clear low potential for false positives.	Passed
1B Ping Sensor	Detect occupancy with a clear low potential for false positives.	Failed
1C IR Sensor	Detect occupancy with a clear low potential for false positives.	Passed
1D Transceiver module	Send and receive data across modules	Passed

Test 1A Force Sensitive Resistor:

1. Attach force sensitive resistor to arduino microcontroller.
2. Configure arduino to record all data received by this sensor.
3. Place sensor underneath a cushion on the surface of a stool.
4. Start collecting data and at regular time intervals sit on the seat and stand up from it.

5. Observe the data collected and see if there are clear thresholds that can be established for when the seat was occupied.
6. Repeat steps 1-3.
7. Start collecting data and bump, move, and place a hand along the surface of the cushion and the stool at recorded intervals.
8. Observe the data and determine how close the recorded activity was to the data obtained in step 5.

Success criteria: There is a reasonable distinction between actual occupancy data and noise/disturbance data.
Failure criteria: The disturbance/noise data too closely mimics the occupancy data and the two cannot distinctly be differentiated.

Test 1B Ping Sensor:

1. Attach ping sensor to arduino microcontroller.
2. Configure arduino to record all data received by this sensor.
3. Plase sensor facing a stool at a mock.
4. Start collecting data and at regular intervals, have someone approach, take a seat, and leave.
5. Observe the data collected and see if there are clear thresholds that can be established for when the seat was occupied.
6. Repeat steps 1-3.
7. Start collecting data while approaching the table without seating and walking by the table at recorded intervals.
8. Observe the data and determine how close the recorded activity was to the data obtained in step 5.

Success criteria: There is a reasonable distinction between actual occupancy data and proximity data.
Failure criteria: The proximity data too closely mimics the occupancy data and the two cannot distinctly be differentiated.

Test 1C IR Sensor:

This test is identical to test 1B, but uses an IR sensor in place of ping sensor. It shares the same success and failure criteria.

Test 1D Transceiver Module:

1. Assemble a simple circuit with an led configured to a receiver module and a transmitter connected to a signal generated by a pushbutton on the other side.
2. Activate the circuit with the pushbutton.

Success criteria: The LED lights up on the other side, proving a signal was sent through the transceiver module.
Failure: No signal is received and the LED fails to light.

Test #2 App Tests

Test	Requirements	Status
2A Android Interface	The app's interface should be functional and correctly displayed on Android using Flutter.	Partial-Pass Test interfaces display
2B Apple Interface	The app's interface should be functional and correctly displayed on Iphone using Flutter.	Untested
2C Wait Time Display	The app should be able to communicate with the Server to receive the average wait time.	Untested
2D Details View	If the user is the restaurant owner or has permission they should be able to view the detailed information about the restaurant.	Untested
2E Restaurant Selection	The App should be able to choose from available restaurants to view that information.	Untested

2A Android Interface:

1. Load App onto various Android phones and emulators.
2. Move from the homepage to the restaurant selection.
3. View the wait time.
4. Login as owner and view detailed view.

Success Criteria: All the interfaces should display in an acceptable way on each of the selected devices. The interfaces should remain responsive and usable throughout all the pages.
Fail Criteria: The interfaces have display or responsiveness issues.

2B Apple Interface:

1. Load App onto various apple products and emulators.
2. Move from the homepage to the restaurant selection.
3. View the wait time.
4. Login as the owner and view detailed view.

Success Criteria: All the interfaces should display in an acceptable way on each of the selected devices. The interfaces should remain responsive and usable throughout all the pages.

Fail Criteria: The interfaces have display or responsiveness issues.

3C Wait Time Display:

1. Setup values to show for wait time for a few restaurants locally.
2. Navigate to a restaurant in the app and view wait time.
3. Wait time should be displayed correctly.
4. Switch to the homepage and navigate to a new restaurant.
5. The wait time should show the new restaurants wait time.

Success Criteria: The wait time correctly updates on a per restaurant basis and should show correctly.
Fail Criteria: The wait time is not displaying correctly, does not contain the correct value, or does not update on a per restaurant basis.

3D Details View:

1. Setup the data view with locally generated or available data.
2. Navigate to the details view for a restaurant you own.
3. Ensure all data is correct for the restaurant and is displayed correctly.
4. Switch to a new restaurant and open the details view.
5. If you don't have permission you should not be allowed to view.
6. If you do have permission the data should be correctly displayed.

Success Criteria: The details for the restaurant are correctly displayed and only users with permission are allowed to view details for a restaurant.
Fail Criteria: The details are not displayed correctly or do not update per restaurant or you can view pages without permission.

2E Restaurant Selection:

1. Go to the restaurant selection page.
2. Ensure that the restaurant list is up to date.
3. Search for a restaurant and ensure interface narrows down results.
4. Select a restaurant and ensure the data displays.
5. Go back to the selection page and ensure all the data is correct still.

Success Criteria: The restaurant selector should have all the available restaurants and should be narrowed down with searches.
Fail Criteria: The restaurant list isn't narrowed down correctly, doesn't display all the restaurants, or doesn't allow selection of the correct restaurants data.
Integration Testing:

Test	Requirements	Status
3A Arduino → Pi	Arduino submits sensor data to Pi with 0% packet loss.	Untested
3B Pi → AWS	Pi submits sensor data to	Untested

	AWS with 0% packet loss.	
3C App -> AWS	App make requests to AWS.	Untested

Test 3A Arduino → Pi:

1. Arduino equipped with both IR and force sensors is powered and instructed to collect and store sensor data before transmitting it through the attached transceiver module.
2. A Raspberry Pi is equipped to receive the transmission and store the data locally.

Success Criteria: The data on the Pi matches the data on the Arduino with nothing lost in transmission.
Failure Criteria: One or more datapoint is not transmitted and stored to the Pi.

Test 3B Pi → AWS:

1. A Raspberry Pi will be preloaded with dummy sensor data in its memory.
2. The Pi will attempt to establish a connection to a blank RDS instance.
3. The Pi will upload its stored data to the database.

Success Criteria: The database is populated with exactly the information stored on the Pi.
Failure Criteria: The database has observed differences when compared to the source data.

Test 3C App -> AWS

1. Make a wait time request to the server.
2. Make a restaurant request to the server.
3. Make a details request to the server.

Success Criteria: The data should all be delivered and displayed in the app.
Failure Criteria: The data is not delivered to the app correctly.

System Testing:

Test	Requirements	Status
4A Proof of Full Integration	Analytics adjusts to real time data and relays accurate predictions to the app in a lab environment.	Untested
4B Full Service Integration	Analytics are taken in industry setting and tested in real-scenarios to verify accuracy and usability.	Untested

Test 4A Proof of Full Integration:

1. Database will be loaded with known dummy data.

2. App will access database and confirm dummy data.
3. Sensor circuits will be set up at two mock tables.
4. Data collection will be initialized.
5. Sensors will be triggered at controlled intervals to roughly match a predetermined input stream.
6. Data will update in real time and be checked against our calculated expectancies.
7. The app will continue to access the data and ensure real time adjustments are accurately being delivered to the user.
8. Sensor collection will be turned off and data progression through the testing process will be reviewed.

Success Criteria: Prediction analytics gave expected output at all times. Data was constantly updated and current on the app. Sensors continuously updated database as expected.
Failure Criteria: Prediction analytics give unexpected output. Data is unavailable in app at any time. Sensors do not update database as expected.

Test 4B Full Service Integration:

1. Arrange with a local restaurant to test this service in 2+ tables.
2. Setup sensors at allocated tables.
3. Teach staff how to use app to access data and what it means.
4. Observe and support operation of sensors for allotted time period.
5. Compare prediction analytics to actual times.
6. Confer with staff for feedback.

Success Criteria: System remains operational during entire trial. App is intuitive and requires minimal training. Users found value in the app.
Failure Criteria: System fails or malfunctions during testing period. App is confusing to users. Users report no gained value from experience.

3.4 Process

In order to progress in this project we will break up into small subteams consisting of hardware, software, and platform operations. These teams will continue to practice AGILE development, having sprints defined by according to the gantt chart provided at the end of this section. As the year progresses we will continually tie the subteams work together in efforts to meet full integration testing standards.

We will update reports to have documentation on how the project is progressing and detail the experiences, triumphs, and struggles of each team as the project matures. If necessary we will also add or remove requirements to the project to better tailor the project to the vision that the client and the team share.

An outline of our expected deliverables and timeline in advancement of the prototype to these testable milestones is outlined here:

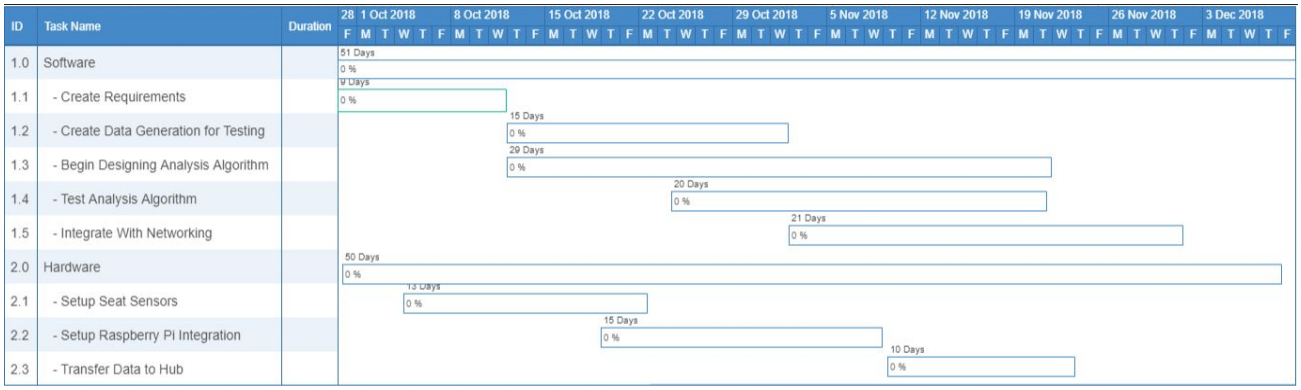

Figure 7 Process Plan

3.5 Results

As we are still rather early on in development, most of our testing that has occurred has been focused on hardware and hardware integration as the rest of the app and platform need to be built before they can be fully tested.

We initially wanted to use all only two of the aforementioned sensors in the project: ping, and force. However after going through tests 1A and 1B we found that the ping sensor was heavily affected by background movement which would lead to many false positives. Due to this unforeseen error we decided to test a similar sensor, IR, for measuring distance. We knew IR sensors generally were not good for tracking objects over great distances which is what initially steered us away from exploring it in the first place. However this shortfall ended up being an asset, given that it was much less prone to proximity interference, leading to more accurate detection.

Further testing including integration and full system testing will be added as the results become available according to the procedures outlined in the previous section.

4 Closing Material

4.1 Conclusion

Thus far, we have developed a solid foundation of the hardware portion which included acquiring reliable sensors as well as implementing them into a small restaurant-table-scale network able to communicate wirelessly. Using this foundation, we plan to build out a larger restaurant-scale network to complete the data collection portion of the project. In parallel to this network development, our current plan also involves continued development on implementing communication between our in-progress user application, as well as our AWS server. With the ability to now generate real data using our small sensor network, we will be able to begin testing as outlined above.