Final Development of the IoT Elderly Care Solution

Final Report

Client:
Andrew Guillemette

Advisor:
Daji Qiao

Team Members:
Jared Griffin
Nidhi Dalvi
Robert Guetzlaff
Siyuan Zeng
Tyler Borchert

Team Email: sddec19-18@iastate.edu
Team Website: sddec19-18.sd.ece.iastate.edu
Written: 2019-12-10
Revision 3

Table of Contents

List of figures

List of tables

1 Introduction

1.1 Foreword

Our original objective for this project was to pick up where two previous senior design groups left off developing a minimum viable product for a startup to take to investors. The client was not satisfied with the state of the project after the first group had finished, and while our task was to “improve” what they had left behind, frequent changes in what our client’s expectations and his general disorganization led to the majority of implementation work to be put off until well into the second semester. Coming into the second semester, we suspected that our client was not going to continue with this project as he became hard to get a hold of and did not provide the resources we asked for, either deflecting the issue or expecting us to use hardware that our advisor had for use in one of his classes. It was eventually confirmed that he had no interest in this project and had moved to another startup idea.

Two weeks into the semester, we also gained a sixth member who after getting deeply involved with our sensor development abruptly left the group. These two situations together left us in a situation where we were not able to test the full implementation of the system and had to rely on testing the modules independently.

1.2 Problem Statement

The goal of this project is to make improvements and advancements of a previous capstone project. This project is intended for elderly individuals who wish to have their health monitored through the use of sensors placed around their residence. This project is driven by the need for elderly care today. There is a large number of elderly people who live alone, and loved ones would like to monitor their quality of life and health status to prevent health issues from arising. This is important because having a personal nurse is expensive, so we hope our project can provide a cheaper and more efficient way to allow the elderly to pursue preventative healthcare.

1.3 IoT Elderly Care Solution

The premise of this project is that for an elderly person who has regular habits, deviations to their habits may be an early indication of a new health issue. The habits of the resident are tracked through sensors placed throughout the kitchen to track preparation of food, tracking power usage of various appliances, and a smartwatch to track the patient’s activity.

Our task for this project was to create the backend logic for meal preparation detection, a web application for the viewing of the recorded data, and to minimize the invasiveness of the currently existing system.

2 Requirements

2.1 Functional Requirements

1. Sensors must be wireless and have a lifespan of 1-2 years.

2. The data from the sensors must be transmitted to our server.

3. The logic system can take in a set of recorded sensor data and determine to a degree of certainty if a meal has been prepared.

4. The resident events, meals prepared or skipped, must be displayed on a web application.

2.2 Non Functional Requirements

1. As much of the previous groups work should be reused as possible.

2. The sensor system is non-invasive to the residents home.

3. Detection of prepared or missed meals is timely.

4. System can recover from a loss of power.

2.3 Considerations and Constraints

This project aims to create a minimum viable product, so the biggest consideration was cost. We were directed by our client to reuse as much of the previous groups work as possible. The server infrastructure had already been setup in Amazon Web Services, the onsite computation was already being done on a couple of Raspberry Pi's, and a set of wireless smart outlets had already been installed. We need to continue working with these.

The second most important constraint/consideration is the limitation of the sensors. Once such limitation is the cost of the sensor. Since this is a M.V.P for a startup company we had to ensure that the chosen sensor was at a reasonable price. Another is that the chosen sensor is completely wireless. Lastly, the sensor should be available off-the-shelf. Meaning that it can be readily bought.

Finally, as a minimum viable product, we were instructed by our client that security and advanced logic systems were outside the scope of the project, and that our sensors had to be available off the shelf.

2.4 Users and Use Cases

The user for this project would be a concerned party of an elderly resident living in an independent situation as well as the elderly resident themselves. The final system should be nearly invisible to the resident but provide data back to the concerned party. The web application should be easily accessible by the loved ones of the elderly resident.

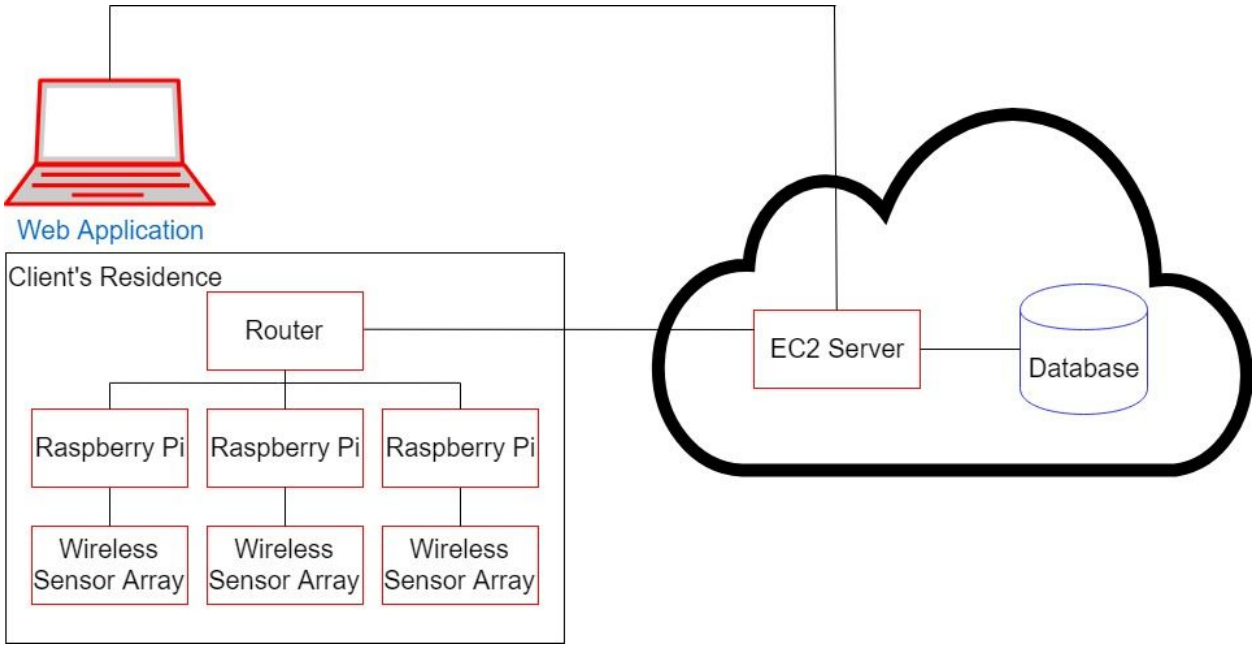
There are two main use cases for this project;
1: A family member who wants to monitor the activity of a loved one. In this case the system would be installed by the family member and can be monitored from the web application

2: An independent living facility that wants to monitor the activity of all of their residents as a service.The system can be installed at time of construction or as a retrofit of existing apartment units. A central physical location can then view all the data from the residents and check in on residents after defined warning signs occur.

System implementation for either use case is the same.

3 System Design

3.1 Final Design Plan



The planned design for the elderly individual's residence is in their kitchen. There will be arrays of sensors in cabinets and drawers. Each of these arrays connect through bluetooth low-energy to their predetermined Raspberry Pi. Once the information is received by the pi it is then transmitted to the cloud via the residence's router.

The logic server processes the data that comes from the sensors in the elderly individual's residence. This information is then fed into the web application, along with some of the unprocessed information for hydration of the web application.

3.2 Modules

3.2.1 Hardware

3.2.1.1 Kitchen Sensors

The kitchen sensors are TI CC2650STK Sensortags which we decided to use in our first semester. These sensors communicate with a Raspberry Pi via bluetooth low-energy. The data being sent are from the accelerometer and gyroscope sensors. The Raspberry Pi does processing on the incoming data to give us the desired opening/closing events to be used later.

This program is written in Python and uses gatttool to establish connections with the Sensortags.

3.2.1.2 Smart Outlets

Th
e smart outlets used by our group are three TP-Link HS110 setup by the second group to work on this project. These were added to our scope when we discovered that they were unable to communicate to the AWS server instance group two had configured for them. We were unable to gain access to the instance and had to rewrite the backend of their communication.

The outlets communicated over WiFi to the Raspberry Pis, where the pre-existing NodeJS code would intercept a notification from the outlet and send data to the server using an HTTP POST request. As were were unable to access or redeploy their backend, we rewrote it in PHP to accomplish the same task.

3.2.1.3 Data Transmission

The sensors in the kitchen log readings to an sqlite database on the Raspberry Pi. This allows them to quickly dump their data without having to check if an internet connection is present. A separate program acts as a relay, Checking to see if there is data in the local database, checking if it can communicate to our server, then transmits the data to the server and then deletes the transmitted entries.

3.2.2 Logic

3.2.2.1 Spring Logic Server

We are using an AWS Elastic Container as a server to run the logic of our project. We deployed a Spring MVC project on this server to compute the data transferred through the hardware.

3.2.2.2 Logic Algorithm

We collected surveys from our testing target, Bob, and analyzed the sensor data that was transferred through the sensors to build our logic algorithm to monitor the target's health status and daily activities.

3.2.3 Web Application

The web application is supposed to provide a way for loved ones of the resident to view the resident's eating habits. The web application was also going to integrate with the work of the previous capstone groups and ours, displaying details about power and water usage and dietary changes..

The layout of the web application was going to be a dashboard, giving an overview of if the resident was eating their meals within a set of constraints related to when a meal was started

and how long it took to finish. The loved one would be alerted if the resident was displaying behavior outside of these constraints. The loved one would also have the ability to look at information for other days along with a detailed log of events our system was sensing.

3.3 Changes from Last Semester

Two weeks into the summer, the system the second capstone group and us set up at the end of the Spring 2019 semester crashed the central hub Raspberry Pi. We were unable to get into the resident's apartment in a timely manner and were eventually unable to recover any logs indicating the reason for the crash. Because of the system structure setup by the first capstone group, the central Raspberry Pi going offline prevented all sensor data from being sent to the logic server.

To prevent such a problem in the future, we removed the central hub Raspberry Pi concept, instead allowing each Raspberry Pi to connect to the logic server. We also changed the sensor reading program to log to a local Sqlite database. A data relay program then runs at regular intervals and transmit the readings to the server only if an internet connection is present. This change removed the potential for data loss because of a lost internet connection, and in case of the failure of one Raspberry PI, we will only lose the sensors connected to it and not the entire kitchen and smart outlet system.

4 Implementation

4.1 Hardware

4.1.1 Kitchen Sensors

In order for data to be collected by the client we needed to create a program with the ability to interface with the sensors. This required us to create a python program with the following features: create connections, acquire data, manipulate data, consume data, and send data.

Connection:
This was achieved by having a process spawned for each desired device. In each of these independent processes, they create a connection using gatttool and pexpect. If a connection cannot be established at that time, then it retries the connection.

Data acquisition:
The data is received by sending commands to the clients gatttool processes. These commands include: turning on the sensors, setting its period, and turning on notifications. Once these are turned on, pexpect returns any notification the gatttool client receives from the sensortags.

Data Manipulation:
Since we only need a specific axis from the accelerometer and gyroscope data we need to extract only that part. Once that has been extracted it is sent to a function that converts it into a signed integer. For the gyroscope it is between -250 and 250. The accelerometer is between -16 and 16.

Data Consumption:
The data is consumed by open/close logic. This is sensor specific where accelerometer and gyroscope sensors have their own implementation. This takes in a signed integer and compares it to the initial value or previous values. The sensing algorithm looks for peaks and valleys in the data. For the “sensing” to begin the received value will have to be non-negative and beyond the minimal threshold. Once that has been hit, then we can assume the opening event has started and now accept negative values. These values signify that the closing event has begun. Once both of these have been found, then we can say the drawer/cabinet has opened and closed. This will report the time opened as well.

Data Transmission:
Once an open/close event has been recorded it is sent to a sqlite3 database. If the database is not already created, then it creates one. These database entries are in a format for the cloud mysql database to understand.

4.1.2 Smart Outlets

We inherited this portion of the project from the second senior design group after this portion of their project failed over the previous summer. Our project uses three of these outlets, one on a microwave, one on a water heater, and one on a toaster.

The Raspberry Pi that receives data from the outlets runs code developed by our previous group. This code is written in NodeJS and uses a third party library to intercept the notifications the outlets transmit to TP-Links servers when an event they have been configured to record occurs. The program on the Pi then transmits the event using a POST request to the AWS server.

Our contribution replaces their NodeJS backend with one written in PHP using an apache web server as we were unable to access or redeploy their code. Our program receives the post request, and stores the event in a set of database tables, recording each event, the duration of usage if the event was a power off, and the last time the outlet had an event.

4.1.3 Data Transmission

The data relay program was written in Python and creates a Sqlite database if it does not already exist that mirrors the database on the AWS server. Upon running, the program will check to see if entries exist in the local database, and if a connection to the server is possible. The program will then transmit the data from the local database entry by entry, deleting the local entry upon confirmation that the data has been successfully inserted.

4.2 Logic

4.2.1 Spring Logic Server

The Spring MVC project is composed of Models, Repositories, Services and Controllers. The models interpret the data transferred through the database to human readable format; The repositories stored the interpreted data in a good manner for the service parts to search and use. The services provides the algorithm to compute the logic results of our system. And finally the controllers mapped these services on specific paths, so the web application can call the server to request compute force and get the results.

4.2.2 Logic Algorithm

The logic algorithm the services using is implemented by the survey from our testing target, Bob. We collected his daily activities to analyze his behavior mode. For example, we concluded that Bob generally eat his breakfast during 4AM - 7AM. And then, we monitored the sensor data

of his daily activities to determine his behavior mode. For example, we concluded that Bob will open his refrigerator, bowl & plates drawer and silver drawer to make a meal.

4.3 Web Application

The web application that we developed was built leveraging React JS and was developed using test-driven development, resulting in complete code coverage by unit tests. It was also set up to use continuous integration by utilizing Gitlab Pipelines, initially running our test suite on the web application. It was also going to be responsible for deploying the web application when changes were made.

5 Testing

5.1 Hardware

5.1.1 Kitchen Sensors

These sensors were tested with the use of Iteration Testing with prototypes and Physical testing.

Physical testing:
These physical tests involved using them for their designed purpose. We place the sensors in cabinets and drawers, open, and close them. If we were able to sense the opening and closings, it passed the test.
Due to the nature of these sensors we had to resolve to testing them physically. This is because the sensor output is different each time. We could not use a sample output to determine test because they would be unrepresentative of the rest.

Iteration Testing with Prototypes:
This involved creating a prototype that served a smaller function of the planned overall design. This was used to make it easier to run the physical testing and finding where problems reside. Once the physical test have passed, then we could integrate the prototype into the overall program.

5.1.2 Smart Outlet

Testing of the smart outlets was done two fold. First by providing test events to the PHP backend we created to verify that data was being inserted into the database correctly. Second, we set the system up in an apartment connected to devices that were regularly used for two weeks to ensure the stability of the system.

5.1.3 Data Transmission

The testing plan of the data transmission is to first create sample data and ensure that it is transmitted correctly to the server. Then we will leave it to run alongside the kitchen sensors to ensure that the program is stable and correctly transmits data.

5.2 Logic

5.2.1 Spring Logic Server

The Spring Logic Server was tested using Junit test & Postman to respectively make sure each line of the code can run properly and the project can interact with Web Application correctly. The Junit test covered every class of the project and assigned mock tests case to test the classes. The Postman provides a way to mock the web application request to the server, such when the web application is complete, it can communicate with the prepared logic server.

5.2.2 Logic Algorithm

The Logic Algorithm is hard to test through the code or data, so we tested it by asking our testing target, Bob, to make sure the results and algorithm matched his activities.

5.2 Web Application

The web application was developed using test-driven development. This resulted in having a substantial unit testing suite that covered every line of code in the web application and was executed in our continuous integration pipeline, running over the web application’s code every time a change was made.

6 Results

6.1 Hardware

6.1.1 Kitchen Sensors

We are able to successfully receive, manipulate, process, and send the data with multiple connected sensors. At most we have six sensors connected to a raspberry pi. Through the use of the python script we can produce data that is representative of what happened in the testing environment. However, the functional requirement of battery life was not discreetly addressed due to losing a team member late in the project.

6.1.2 Outlet Monitors

The results of testing our smart outlet has shown that the outlet program is stable. Testing in one of our members apartments accurately recorded the times that devices were turned on and turn off along with the length of time that the device was used.

Monitoring the smart outlets for two weeks have shown that the program can run long term without crashing or generating errors.

6.1.3 Data Transmission

Our limited testing has shown that the data relay program is stable and transmits data correctly without duplicates.

6.2 Logic

6.2.1 Spring Logic Server

The Spring MVC is deployed on the AWS ec2 server. Whenever it received the proper request from web app, it will respond and send the results back to the web app. Everything looks fine and works properly for now. The server satisfied our expectations: provides compute force to analyze data, compute events, and send the results to web application.

6.2.2 Logic Algorithm

We had a prototype logic algorithm running on the logic server, but we didn’t have too many data to determine how accurate it is because we had problems to set up meeting with our client and the testing target Bob.

6.3 Web Application

The web application was primarily developed in the second half of the Fall 2019 semester because of a significant amount of time spent designing prototype mockups of what the web application would look like. Due to this, the web application only has a hard-coded meal overview section. If done differently, we should have forged ahead with insufficient mockups and made changes to the web application's look and feel as we went. This way, we would have at least had a complete web application, even if it would not have looked like what we would have desired.

7 Summary

Our goal is to help monitor the health of senior citizens.The system we built addresses this by collecting data from elderly residents' homes, analyzing their behavior against the baseline for their habits, and displaying our system's health predictions in a web application for loved ones and health professionals to view.

For the hardware, the system that we have build addresses the data from elderly residents's home. We have multiple sensors connected to a hub which we will able to collect and send accelerometer and gyroscope data to a Raspberry Pi.

The logic server provides the features to process raw data to human readable format, provides the logic algorithm for determining if a meal was eaten, and serves requests for the web application. This reduces the complexity for both hardware module and web application because they can ask for the compute force from the server instead of running hard complications on their sides.

The web application allows loved one's of the resident to keep tabs on them, allowing them to tell if the resident is regressing in terms of health, or if they are still doing well. This better allows loved ones to monitor the well being of the resident and take action when the resident's health begins faltering faster.

Appendix I - Required Hardware

Sensors

TI Sensor Tags
Tp-Link HS-110 Smart Outlets

Computation Devices

Raspberry Pi 3
Wireless Router

Appendix II - Operation Manual

Installation

Kitchen Sensors

Physical Sensor Installation:
For cabinets place the sensor on the inside as close to the hinge as possible. For drawers place the sensor at the beginning of the drawer.
First take the silicon cover off and then use some form of adhesive to stick the sensor in the desired location. Ensure that the clear portion on the sensor case is facing into the cabinet/drawer. Also make sure that it is oriented in a way that is expected.

Smart Outlet

The smart outlets are plugged into the wall and the devices you want to monitor. For best monitoring, one devices per outlet.

Raspberry Pi

Place the Raspberry Pi's in a location where you can provide it power and connect it to the sensors it will be responsible for.

Download code from the senior design Gitlab page.

Install and configure apache.

Running

Kitchen Sensors

On the raspberry pi ensure you have these required dependencies: gatttool, pexpect, and sqlite3.

To run the sensor.py program type into a terminal “python3 sensor.py”. Once it is done printing statements start turning on the sensors one at a time. Note: if a sensor does not initially connect, wait 30 seconds for the program to retry the connection.

Smart Outlet Monitoring

Run the eventScript.js program
Node eventScript.js

Data Relay

Run the data_relay.py program as a scheduled task.
Add the line
*/10 * * * * /path_to_script/data_relay.py
To the crontab table

Logic

Download the jar file and copy it to the AWS ec2 server.
Run the command:

```
nohup java -jar demo-0.0.1-SNAPSHOT.jar > demo-app.log 2>&1 &
```

Web Application

Appendix III - Alternative Solutions

Arduino Sensor

Our early concept for a wireless sensor replacement was to make one ourselves using an arduino nano, a magnetic sensor, and a battery. This package gave us the advantage of playing around with the attached battery to ensure we got sufficient sensor life. The sensor would be programmed in C++ and enclosed in a 3d printed case.

This sensor concept would have been cheaper and easier to configure the the TI sensor tags we ended up choosing but our client decided that off the shelf sensor solution was what we needed to go with.

Machine Learning

Though this project we struggled with deciding how best to identify that a meal happens when different meals could have many different steps involved. We used survey data to identify what a meal included for our one test user.

In the end, our client decided that our focus should be on boolean logic focused on one single user and that any enhanced logic would be developed by future teams.