Week 5 – Update (22/10/2018)

This week we began researching how we would complete this project. Following the lecture on Friday during which we set up the Raspberry Pi's and began playing with Node-Red. I started having a look and playing around with the Google Cloud vision API and seeing if I could get Node-Red to send an image to this in order to see if it could be recognised. I managed to get this working after copying some sample code and then modifying it to suit my needs. I started of by entering a URL to an image I found online and then sending this to the Cloud Vision API and the recognition seemed to be pretty good. I had a go at trying to get it to work with an image from the live webcam although unfortunately I had problems getting the web cam to work with the browser on the Raspberry Pi which made it difficult to send the image. I did however manage to create a Python Script that took an image locally and then sent that to the Node-Red program which was communicating with Google. During the Friday lecture Stavros also showed us IBM's image recognition feature for us to consider which appeared to work similarly to the Google Cloud Vision API (although his example didn't seem to be working). If we have any problem with the Google Cloud Vision API we will be able to consider this and use it as a back up although we will probably stick with Google because I have a fair bit of experience using Google Cloud plugins and as this API is used fairly widely with similar software being used in the Google Lens feature on Android I felt it would probably be the better API.

Week 6 – Update (29/10/2018)

This week I returned to working on my computer as the Raspberry Pi screen was a bit small to program on and also the Pi was starting to struggle with the number of tasks I was trying to run. My focus for this week was to get the scanner system working properly. My first task was to get the browser sending an image to Google Cloud Vision whenever the user scanned an image. To do this I started off trying to send the video feed directly to a HTML canvas element as I knew then that I would be able to start modify the image and also save it in a base64 format that could be sent to Node-Red. I had difficulty getting this to work before I found a tutorial that showed me how to read a camera feed into a standard video element and then save a screenshot from that feed into the canvas element on the call of a function. During this time, following a discussion with Stavros we also decided that most of the functionality would be done in the cloud as he had suggested using the data on a wide scale to help organisations such as the NHS collect data on how healthily people were eating. Because this would involve storing all the data from the fridge in the cloud I decided it was probably easier and better to stick with the Node-Red session I had set up with IBM Bluemix as this would make it easier to communicate with other elements already stored in the cloud and would reduce the workload on the Raspberry Pi as we would not be needing to run Node-Red locally. It also means that we can reduce the data being sent to the Raspberry Pi as we will be able to only send information that is required by the interface.

Week 7 – Update (05/11/2018)

This week having got my computers web cam working and sending an image to Node-Red I now needed to get the Raspberry Pi to do the same. I spent a long time trying to get the

web cam working with the browser on the Raspberry Pi which was difficult because the browser wasn't recognising the Camera Module we were using as a webcam although we were able to make it work using a Python Script. I tried several methods to try and get the camera working with the browser including broadcasting the feed to a server which could then be accessed by the browser although this would probably have made if difficult to save the image in the canvas element. In the end I found a command that I could run on the Raspberry Pi[1]. This basically changed the software running the camera module meaning that it was now working the same as a webcam and could be recognised by the browser.

After doing this I started looking into how I could get a barcode scanning system to work. This was harder than I thought as a lot of scanners that are built on the mobile platforms rely on the autofocus feature which a lot of webcams and also the camera module for the Raspberry Pi, do not have. I tried a number of different JavaScript plugins[2] [3] as well as having a go at building a Processing sketch which could be used in a canvas element using P5. Eventually I decided to go with the Dynamsoft API which seemed to work quite well when dealing with a still image. I then had to modify the code I already had in place to take an image so that it would also scan this image to look for a barcode. I decided to check if there was a barcode before sending the request to Google Cloud Vision so that I could limit the number of server requests as I assumed that if there was a barcode in the image then this would probably be able to identify the product. I eventually managed to get this working and so was able to start working on the other features of the site. As I was waiting on Yuzhou to complete the interface designs I put together a basic template which included all of the features but didn't look very appealing. I then set to work on the other backend features of the site.

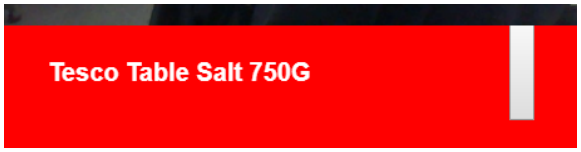

barcode Functionality.mp4


Camera functionality.mp4

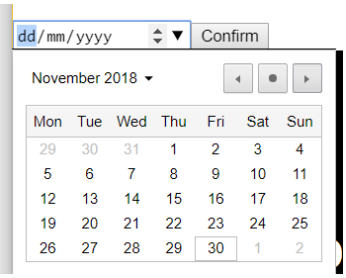
[1] https://forum.astroprint.com/t/raspberry-pi-camera-support-solved/62
[2] https://github.com/andrastoth/WebCodeCam
[3] https://www.dynamsoft.com/Products/barcode-recognition-javascript.aspx

Week 8 – Update (12/11/2018)

During this week I was able to start work on the backend database part of the fridge. I decided for this that I would create a number of PHP scripts which would control all of this functionality as I knew I would be able to easily connect these to the SQL database I was using and because it would all be running on the same server, I knew it would be more secure which is essential when dealing with user information such as this. I started by building the system that would store the food items that had been scanned into the fridge. I did this by creating a button in the interface that would allow the user to add in their scanned food once it had been identified.



Because one of the features of the fridge that we had discussed had been the idea of encouraging users to use up food before it's use-by date I needed to find a way of getting data about a products use-by date. I had originally hoped that I would be able to get this data through the Tesco API however I found that this data was not available. I therefore had to find another solution to this issue. I ended up building a system that would ask the first 3 users to scan in a particular item to also enter the sell-by date on the packaging. Using this data as well as the date that the item was entered, I am able to assume how many days the item is expected to last. Using this data, I can then work out the average life-expectancy of individual food items which can then be used to work out the use-by date. Obviously, this solution isn't perfect as it assumes that the item is being put into the fridge whilst it is still fresh. There could also be issues if the date is entered incorrectly and I found that I had to develop a special system for entering the date to ensure that it could distinguish between the UK and US format of the date. A solution to this problem would be to develop a system that would look out for any outliers in the average and then remove those from the calculation. I also had to work around the limitations of the JavaScript Date() function which I knew would struggle to know if the user was entering the date in the format dd-mm-yy, mm-yy or even mm-dd-yyyy. I therefore built a system that would present the user with a calendar that they could enter the sell-by date into.



Week 9 – Update (19/11/2018)

During this week I finished off the backend scripts for adding food into the fridge. I tried to build the scripts so that the interface would call the same script but would call different functions within that script based on what was required. The reason behind this was to try

and keep the code tidier as I would only have to connect to the database once. I however did put some of the code in separate files which were linked to the main script that was being called. Once I had finished all the scripts for adding items to the database, I started looking at getting all the items that were in the database to display them to the user. One problem I found with this is that because all the data was in different tables I had to make several different queries. I also needed to format the data to make sure that everything relevant was sent to the user. One of the major things I had to think about was the date because this had to be the estimated expiry date which was worked out by adding the average life expectancy of an item to the date that it was added into the fridge. Fortunately PHP's date() tool is very good for formatting dates how you want and you can also easily add a number of days onto a specific date by simply adding " + X days".

Week 10 – Update (26/11/2018)

During this week I continued working on display the items on the interface. I also started working on the interface using the designs that Yuzhou had designed as well as sitting down with Rupert who was able to help with any other design elements that hadn't been thought about in the original designs. One of the problems that I has was getting the systems that I had built for the functionality of the fridge to work with the new interface designs. This was difficult because I had to make sure that everything would still work as intended.

One of these issues was the video feed because I needed this to work with the barcode scanner that I was using. I had to change all the keyboard triggers that I had created for starting up the camera and scanner to trigger when the user clicks on the icon to scan an item. I will in the future probably change this again so that it gets triggered when the user opens the fridge as it will be likely that this means that they intend to put something into the fridge.

Also during this week I started working the Edaman API so that I could start fetching recipes that the user could use with the items in the fridge. One of the issues that I had with this is that the original plan had been to get the recipes based on what items where in the fridge. The problem was that the data from the Tesco API wasn't compatible with the ingredient lists in the Edaman API. An example of this was a recipe that required salt but the Tesco API had listed the only salt as Tesco Table Salt 750g. This meant that I had to find a way of connecting the 2 different data formats. I got around this by splitting every word in the name. I then made an SQL query of the database with all the recipes in using the LIKE command to find recipes with ingredient items that were similar to the item in question. This method seemed to work fairly well although it did mean making a lot of requests to the database which I had wanted to avoid. I however decided that this was probably the best solution as otherwise I would have to find some way of manually linking the two recipes. If we were launching this as a commercial product we would probably have to team up with Tesco to find a way of linking the ingredients to what was in the fridge.

As I wanted to avoid making too many API requests, I decided to limit it so that requests where only made when the user wanted to add a new recipe. I stored any recipes that came back in the database so that they could be accessed by the system again. This also made it

easier for me to access the data and meant that I could guarantee that it would always be available. If this was to be launched as a product I would however need to make sure that this was ok as I wouldn't own the data and this would probably breach the terms and conditions for the API.

We also spent this week preparing for the formative assessment. This was a concern for us as although we had done a lot to the product it still wasn't in a position to be shown. We got a lot of feedback on our project including a suggestion to use a tablet instead of the Raspberry Pi to display the interface and to also work on the design of the interface.

Week 11 – Update (03/12/2018)

As Rupert had been working on the designs for the visualisations that would show off the data that was collected by the fridges in people's homes. I decided to work on implementing this. I had started with the backend scripts for pulling the data from the database and had also had a look at designing the map using an SVG image which could then be edited with JavaScript and CSS. I continued working on this during the start of this week and eventually got a basic map that would change the colours based on Nutrient information of the items in the fridge.

Also following on from the formative assessment we decided to take Stavros' advice and use a tablet instead of the Raspberry Pi to display the interface. The reason for this was because the Raspberry Pi had been struggling to handle the interface, specifically the features that used the camera. We found that the CPU was constantly getting close to 100% and this was affecting the quality very badly. We had discussed previously using a more powerful computer however we preferred the idea of using a tablet because it would be easier to integrate into the interface and we would be able to use the forward-facing camera already built into the tablet. We also decided that we would be able to use the Raspberry Pi to link the interface to the fridge, possible hosting the website from the Pi so that the whole system would work independently. The Raspberry Pi would also be able to deal with any of the data coming from the fridge such as the weight of the fridge and when the door opens. Because of this I decided that we would need to get the two systems communicating with each other, which I decided to do using websockets.

I was initially going to use MQTT to do this, using a Paho-mqtt plugin which would allow me to communicate with an MQTT broker in JavaScript and spent a lot of time trying to set up my own MQTT broker using Node.js and Node Red. The reason for this is that I wanted to have more control over the broker and was also worried about the security of using a third party broker, especially if this was sold as a commercial product. I was also worried that I could end up bringing down a broker as I would have to make a lot of requests although MQTT requests are more efficient. Although I managed to get this working, I had difficulty getting both systems to communicate with the broker.

One of the reasons that they weren't working was because I was still using the Universities' Eduroam network which was blocking my requests. This was annoying because it took me several hours before I realised that I hadn't changed this. Unfortunately, although this meant that the Raspberry Pi was communicating with the broker, I was unable to get the

interface to work with it. I therefore decided to try getting it to work with a third party broker. I first tried the i-DAT broker[4] which I managed to get working with the Raspberry Pi but again I couldn't get it to work with the interface. I then tried with another broker, HiveMQ[5]. With this broker I managed to get it working with the interface but not the Raspberry Pi. After talking to Stavros about this, he suggested using Socket IO which would allow me to communicate directly with the Raspberry Pi using WebSockets. I however had trouble getting this to work however whilst looking into it I found out about JavaScripts native WebSockets tool which was being used by Socket IO. I therefore decided to try and adopt this to see if this would work and I managed to easily get both systems communicating with each other.

Week 12 – Update (10/12/2018)

Having got the Raspberry Pi communicating with the interface I decided that we needed to start looking into building the hardware for the fridge. One of these features that we were going to use was a weight module so that we could read the weight of the fridge and find out if an item had been removed. I had already order a HX711 chip to work with the module that we already had.

At the start of the week I worked with Stewart who helped me to solder the chip to the module and then connect this to the Raspberry Pi. I then looked at setting this up to work on the Raspberry Pi using a Python script that I found only and then modifying this. The idea with the module is that it is designed to flex when under pressure meaning that when more weight was added to the scale, the metal bar would flex more and this would output a higher figure. The reason behind the HX711 chip is to convert the minute changes that would be made by the flexing of the metal and increase that to a larger figure that could be read by the Raspberry Pi

Once I had finished this we needed to build the scales that the fridge would sit on so that it had a base and the weight module would work properly. Me and Rupert went to the Brunel Lab to work on this, building a base that would be strong enough to support the fridge and then integrating the weight module into it. We had a few problems getting everything to line up properly but once this was sorted, we were able to carry on with the project.

Over the Christmas break we continued working on the project. One of the problems we had with the weight module was that when it was initially set up it was not grounded properly meaning that we were getting a lot of varied figures coming through. This was very annoying as it meant that we would not get accurate data coming through which would limit what we could do with the weight module. I had a look at several different solutions, including building a faraday cage type box around the HX711 element which would block out any electromagnetic interference which was causing the varied figures. I however found that one of the wires that hadn't been connected was designed to ground everything which

[4] mqtt://broker.i-dat.org:80
[5] broker.mqttdashboard.com

should sort the problem. Unfortunately, I will need to wait until I am back at university to fi this meaning that I had to leave this over the Christmas break.

Although I couldn't work on the weight module, I was able to work on the programming that would update the database when I removed or added an item. I built the system so that it would store the weight when an item was added into the fridge and then when an item was removed the weight difference would be detected by a Python script on the Raspberry Pi that is read by a system in Node-Red and then sent to the interface via the WebSockets system that I set up earlier. The interface would then make a request to the database to get a list of all the items that were a similar size to the item removed and then the user would be prompted to say which item they have just removed which will then be removed from the database.

Week 13 – Update (07/01/2019)

This week was the final week of working on the Smartfridge. Early in the week I started off fixing the issues I had been having with the HX711 chip over Christmas with the help of Stewart. I managed to reduce the impact that interference was having by plugging the Raspberry Pi directly into the mains and avoiding crossing any wires which might have caused some interference. I also recalibrated the fridge using a tub of paint and improved the data coming in through the load cell by tightening the bolts that were holding it together as it was still very loose and therefore not all the pressure was being placed on the load cell. All of this managed to reduce the interference although there was still some impact. I therefore modified the code to only look for bigger changes and also to only look when the door was open as this would be the only time that the user would be adding or removing an item from the fridge. As I had been working on the interface over the Christmas holidays this week was mainly dedicated to putting together the hardware for the fridge. I used a Infrared Sensor to test the proximity of an item to the camera meaning that whenever an item was held just above the sensor and therefore in front of the camera it would know to take an image. If the sensor was triggered but there was nothing in front of the camera, such as if someone was leaning over the fridge, then it would still take an image but because there is nothing to recognise it will just ignore this data and return to the normal interface. I also considered using an ultrasonic sensor to identify the user however because I had an issue with a dodgy sensor I ended up opting to go with the infrared sensor on the basis that it would be easier to wire in and also was designed to deal with data from close range which would be what was required on the fridge. I also needed to build a system for detecting if the door was open or closed. To do this I decided to build a simple button that would be activated when the door was closed and then this would be able to send a message to the Raspberry Pi which would update the interface. Both this button and the Infrared sensor were originally going to be connected directly into the Raspberry Pi in order to keep it tidy however because the Raspberry Pi didn't have a pin to read analogue data and there wasn't enough time to order the chip that would be able to make this conversion I ended up using an Arduino to read the data coming in from these sensors. The Arduino is connected to the Raspberry Pi through the serial port and sends 3 different messages dependent on what data the Arduino was reading. These messages are 'OPEN' which is

called once when the door is opened and the circuit for the button is disconnected. There is also ‘CLOSE’ which is called once when the door is closed and the circuit is reconnected. The last message is ‘FLASH’ which is called every second whilst the Infrared sensor is detecting something in front of the camera. These messages are sent to the interface through the Raspberry Pi and the interface will then react accordingly to this data. Once this had been fixed and I was happy with the interface we could then start putting everything together. We 3D printed a stand for the tablet so that it could sit neatly on top. The idea was that everything would sit inside the base so that the users wouldn’t be able to see it. One issue we had was that there were a lot of wires and because I didn’t want the wire from the load cell to touch any other wires as this could cause interference we ended up with a lot of wires sticking out of the back although we did manage to conceal most of these inside the base and if this was to be launched as an actual product we would be able to use shorter wires in order to conceal everything properly. We also cut a couple of holes in the base so that we could get wires through for the button for the door and also for the Infrared sensor. Because we didn’t want the users to be able to see everything that was inside the base we covered the base with a cloth. The reason for opting to use a cloth was so that it wouldn’t offer any resistance for the load cell and it would cover everything up easily. It also meant that it could easily be removed if necessary.