2016年度 卒業論鞭

IoTを用いた食品管理システム

大阪産業大学 デザイン工学部 情報システム学科
情報教育システム研究室

13H043 佐味谷深雪

目次

1　背景

毎日入れ替わる冷蔵庫の中身を把握できていない人は少なくない。そして急な外出やふと立ち寄った店で買い物をしようとして食材の残りを家族に電話し、残りを確認したこともある。献立を考えるとき冷蔵庫の中身を見れる環境であるとは限らない。そこで、どこにいても材料を確認して買い物を出来るようなモノがほしいと感じた。まず必要なのは冷蔵庫の中身のモノ、そして残量を確認できるということである。そして、材料はいつもどれくらいで消費できているかというグラフも欲しいと感じた。

第 2 章では目的、第 3 章では IoT の歴史と現在について。第 4 章ではシステムの概要、第 5 章ではシステムの詳細を述べる。第 6 章では研究成果を述べ、第 7 章では今後の課題についてまとめる。

2 目的

本研究の目的は、外出中でも冷蔵庫の材料や分量を知ることができ、家の食材に配慮した買い物が出来るシステムを開発することである。開発にあたり、近年 Raspberry Pi や Arduino などシングルボードコンピュータを用いた、モノとインターネットを繋ぐ IoT(Internet of Things) が人気を博していることに注目し、本研究も Raspberry Pi を用いて開発をした。外出中でも手持ちの端末で確認出来るという点について、2016 年ではスマートフォンの普及率は 70% を超え、タブレット端末の普及率は 30% を超えたことに目を向けた。スマートフォンやタブレットはアプリをインストールし使用するが、iPhone なら iOS のアプリ、Android のスマートフォンなら Android アプリとどちらも使えるようにするには 2 つのアプリを作る必要があり、更にノートパソコンの小型化からノートパソコンを持ち歩く人も増えたことから、Web サイトから確認できることが幅広く使ってもらえるのではないかと考えた。表示する Web サイトは冷蔵庫の中身のモノを確認するため画像を使用し、残量を確認するために圧力センサーを使用した。本研究ではこれらを考え、いつでも冷蔵庫の中身を確認出来るシステムの作成をした。

3 IoT

前章でも述べたように、モノとインターネットを繋ぐ IoT という言葉をよく目にする。また、機器同士をネットワークで繋ぎ、人が操作することなくネットワークを通じて機器間で情報を収集したり機器を作動させたりすることを M2M（Machine to Machine）といい、IoT と並んで広がりを見せている。IoT とはここ数年で出来ているように感じる方もいるかもしれないが、歴史は 1980 年代まで遡る。1980 年代の東京大学情報科学科助教授だった坂村健博士が TRON プロジェクトを提唱した。組み込み分野に向けての OS 開発を目指しており、最終的な目標はどこでもコンピュータを内蔵した機器同士がネットワークと通じて動作するというものである。その後、1991 年にはゼロックスパロアルト研究所のマークワイザー博士が The Computer for the 21st Century で、ユビキタスコンピューティングという用語を使用している。坂村健博士やマークワイザー博士は IoT と同じ目標を持っていたのである。この事から IoT は 1980 年代には考えられていたと言えるだろう。
IoT とは言え大きく個人向けとメーカーがあると考えており、まず個人向けとしては、本研究でも使用している Raspberry Pi や Arduino、IoT デバイス向けのデータ通信サービスの SORACOM Air、子供向けプログラミング教材のレゴ WeDo 2.0 など様々な年齢を対象としていろんな IoT デバイスがある。また、お家ハックや IoT 縛り勉強会などさまざまな勉強会も広がりつつある。

次にメーカーとしては、パナソニックのスマート家電や東芝の家電コンシェルジュサービス、農林水産省のスマート農業などがある。また、東京電力と日立製作所とパナソニックの 3 社で住宅内の電気の使用状況などを情報を収集, 蓄積する IoT に関する共同実験が開始されるという。Softbank Award 2015 で孫正義氏が、ソフトバンクが考える情報革命を牽引する 3 つの分野のうちの 1 つが IoT と今後の経営方針を語った。
M2M や IoT という考えは古くから考えられていたが、ネットワークの接続することにコストが生じ商品化がされていなかった。だが、スマートフォンやタブレットが普及し WiFi に関連する部品のコストが下がり各家庭にインターネットが引かれるのが当たり前になり、IoT が一気に普及したと考えられる。大手メーカーでは既にスマート家電と称し、家電の IoT 化が進んでいる。スマート家電で有名なのはパナソニック [1] である。冷蔵庫や洗濯機、電子レンジや炊飯器など多岐に渡る家電があり、テレビやビデオカメラなどもスマートフォンで操作出来るという。

国内メーカーの東芝ライテック株式会社の東芝冷蔵冷凍庫 VEGETA [2] があり、インターネットに繋いで外出先からでも冷蔵庫の内部を確認することが出来る。下記の図のように冷蔵庫内部にカメラが設置されており、設定された時間にカメラで撮影されサーバから確認できる。他にもスマートフォンから撮影指示も出せ、現在の状況も確認出来るということが図 1 に示されている。

また国内メーカーだけではなく海外メーカーも IoT と家電を用いた商品を開発してるという。ドイツの大手家電メーカー BOSCH [3] は Home Connect サービスを開始し、冷蔵庫だけではなく洗濯機や乾燥機、コーヒーメーカーなどを商品をフルコントロールでき、すべて外出先から操作できるのが強みだという。タブレットで監視している所を図 2 示した。

そこで Raspberry Pi3 を用いてシンプルで誰でも使えるように Web ページを用いて表示をした。

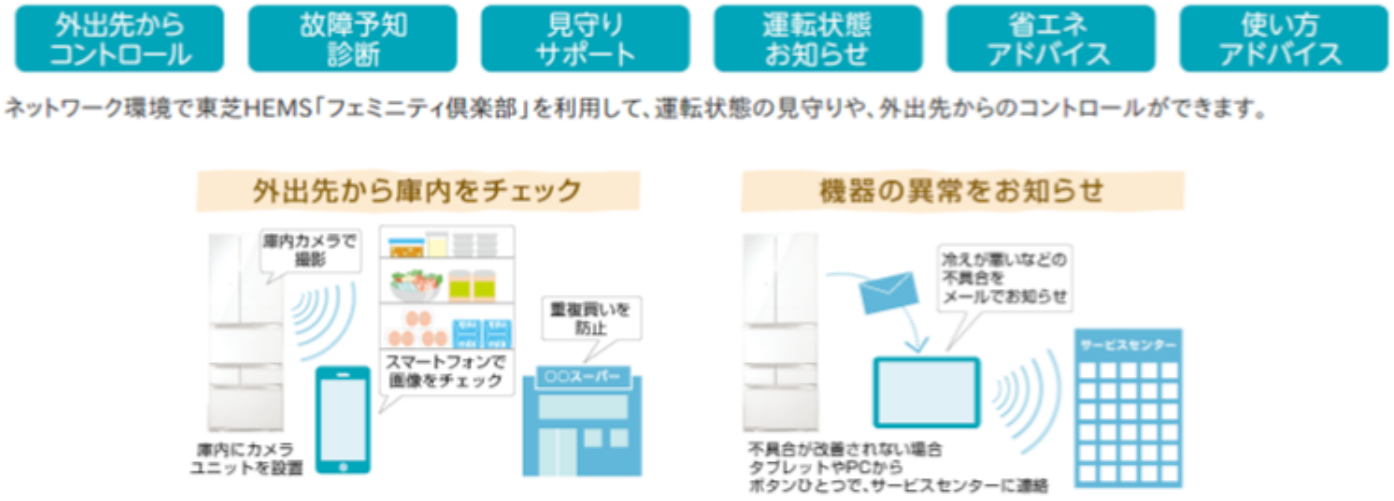

図 1　東芝ライテック株式会社の東芝冷蔵冷凍庫 VEGET

図 2　BOSCH の Home Connect サービスを使用してタブレットで監視している

4　システム概要

本研究では Web ページを用いて外出していてもスマートフォンやパソコンなどで現在の冷蔵庫の残量の確認をし、適切な買い物が出来る事を目的としている。Web ページには現在の冷蔵庫内の画像、材料の重さ、グラフが表示されている。本研究では、Raspbery Pi3 と Debian がインストールされている仮想の Linux サーバを用意した。現在の冷蔵庫内の画像は Raspberry Pi3 に繋がっている Web カメラで撮影し、その画像を Linux サーバに転送し Web ページに表示している。材料の重さは Raspberry Pi3 に圧力センサーを繋げ、そこで重さを測っている。測った重さは Linux サーバのデータベースに保存され、Web ページには最新のデータが表示されるようになっている。今までの重さの経過を表すグラフは Linux サーバのデータベースを元に、過去約 1 週間の重さの推移が確認できるようになっている。Web ページを更新すると画像、重さ、グラフが最新のものに更新される。

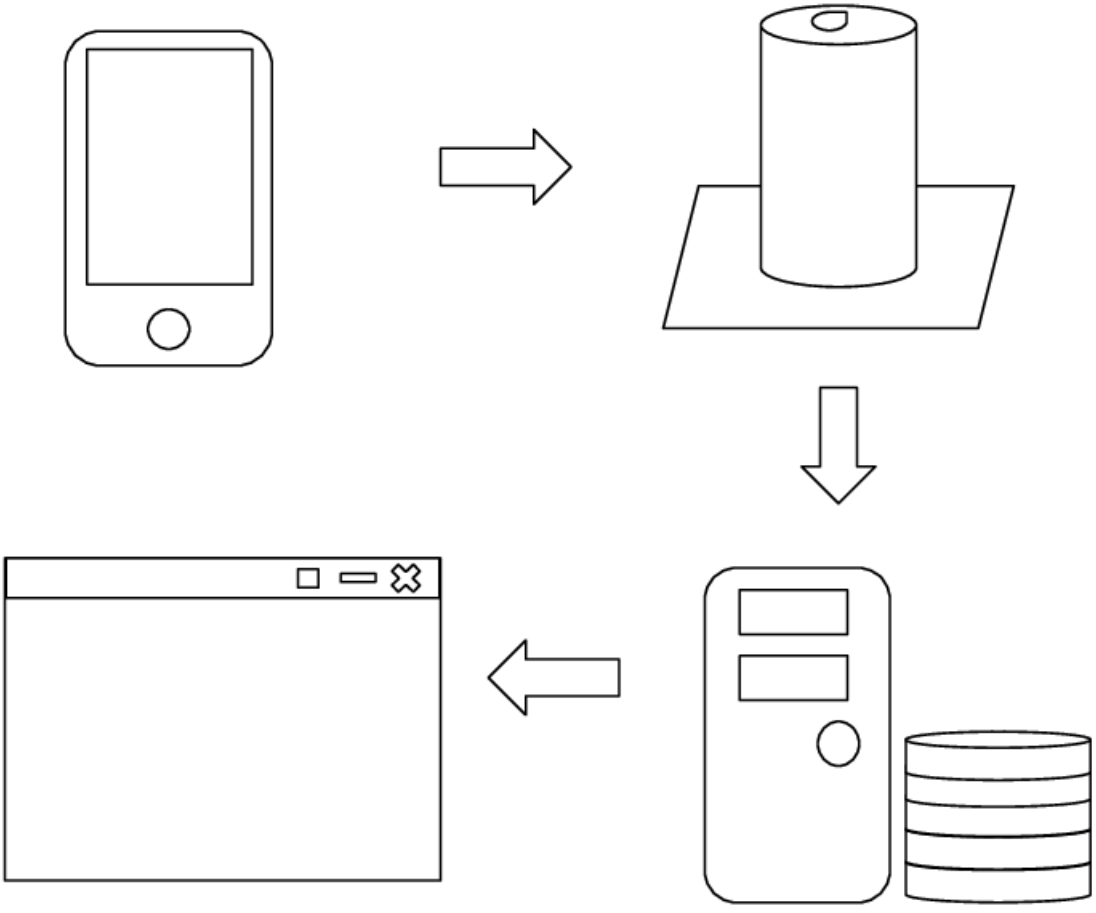

図 3　スマートフォンからアクセスすると Raspberry Pi3 が重さを測り、その重さのデータを Linux サーバに蓄積され Web に表示される

スマートフォンから Web ページにアクセスし、Web サーバが Raspberry Pi3 を稼働させ、重さの値が Web ページに戻ってきてクライアント側のスマートフォンで表示されるという流れを図 3 で示す。

5 システム詳細

本研究で開発したシステムは、現在の冷蔵庫内部の画像、材料の重さ、過去約一週間分のグラフを Web ページに表示する。

5.1 画像の取得

冷蔵庫内部の画像を Web ページに表示するため、Raspberry Pi3 に USB カメラを取り付けた。Web カメラで写真を撮り、画像として保存するため Raspberry Pi3 に fswebcam をインストールした。fswebcam とは繋いだ USB カメラを使用しコマンドを実行するだけで写真を撮影、保存するソフトである。Raspberry Pi3 で写真を撮るコマンドを実行するプログラムを書いた。撮影、保存された画像は Raspberry Pi3 から Web サーバのある Linux サーバに ssh で転送され、Linux サーバで保存される。また、画像は重さのデータと違い、毎回上書きで保存されることはない。

5.2 重さの検知

1. 圧力センサーの接続
   重さを測るため、圧力センサーを使用した。本研究で使用した圧力センサーは FSR406、A/D コンバーターは MCP3002、抵抗は 10k Ωのものを使用した。センサー部のサイズが 43.7mm と小さく、圧力センサーを 4 枚用いて 1 つのセンサーと動作するように制作した。Raspberry Pi3 には 40 本のピンがあり、プログラムで各 GPIO ピンに役割を設定すると出力電圧制御や入力電圧制御をすることができ、データ通信を行うことが出来る。本研究では圧力センサーからのデータの取得にシリアル (SPI) 通信を用いた。Raspberry Pi3 ではアナログ端子がないため、圧力センサーなどのアナログ信号と受け取るために A/D コンバーターを使用して 2 進数のデジタル表記にする必要があるため、A/D コンバーターを使用した回路を組んだ。
2. データの取得
   Raspberry Pi3 で圧力センサーから重さの値のデータを取得する。GPIO を制御するパッケージもあり、Raspberry Pi3 で標準インストールされている Python を用い、プログラムを書いた。
   各圧力センサーから A/D コンバーターに 0ch と 1ch に分かれてアナログ信号が入力される。A/D コンバーターではアナログ信号をデジタル信号に変換し、変換された値が Raspberry Pi3 に入力される。Raspberry Pi3、圧力センサーなどを繋いだ回路は以下の表 2 に示す。

表 1　Raspberry Pi3 の GPIO ピンと圧力センサー、抵抗や A/D コンバーターの接続を表す表

Raspberry Pi3 3.3V(1)	→	ブレッドボード + 列	→	MCP3002 VDD/VRFF(8)
Raspberry Pi3 SPI0_MOSI GPIO10(19)		→		MCP3002 DIN(5)
Raspberry Pi3 SPI0_MISO GPIO9(21)		→		MCP3002 DOUT(6)
Raspberry Pi3 SCLK GPIO11(23)		→		MCP3002 CLK(7)
Raspberry Pi3 SPI0_CE0_N GPIO8(24)		→		MCP3002 CS/SHDN(1)
Raspberry Pi3 GND(39)		→		MCP3002 Vss(4)
MCP3002 CH0(2)		→		抵抗 10k Ω.1
MCP3002 CH1(3)		→		抵抗 10k Ω.2
抵抗 10k Ω.1		→		ブレッドボード-列
抵抗 10k Ω.2		→		ブレッドボード-列
FSR406.1-1 3.3V, FSR406.1-2 3.3V		→		ブレッドボード + 列
FSR406.1-3 3.3V, FSR406.1-4 3.3V		→		ブレッドボード + 列
FSR406.2-1 3.3V, FSR406.2-2 3.3V		→		ブレッドボード + 列
FSR406.2-3 3.3V, FSR406.2-4 3.3V		→		ブレッドボード + 列
FSR406.1-1 GND, FSR406.1-2 GND		→		抵抗 10k Ω.1
FSR406.1-3 GND, FSR406.1-4 GND		→		抵抗 10k Ω.1
FSR406.2-1 GND, FSR406.2-2 GND		→		抵抗 10k Ω.2
FSR406.2-3 GND, FSR406.2-4 GND		→		抵抗 10k Ω.2

実際の Raspberry Pi3 や圧力センサーなどを配置した回路図は以下の図 4 に示す。

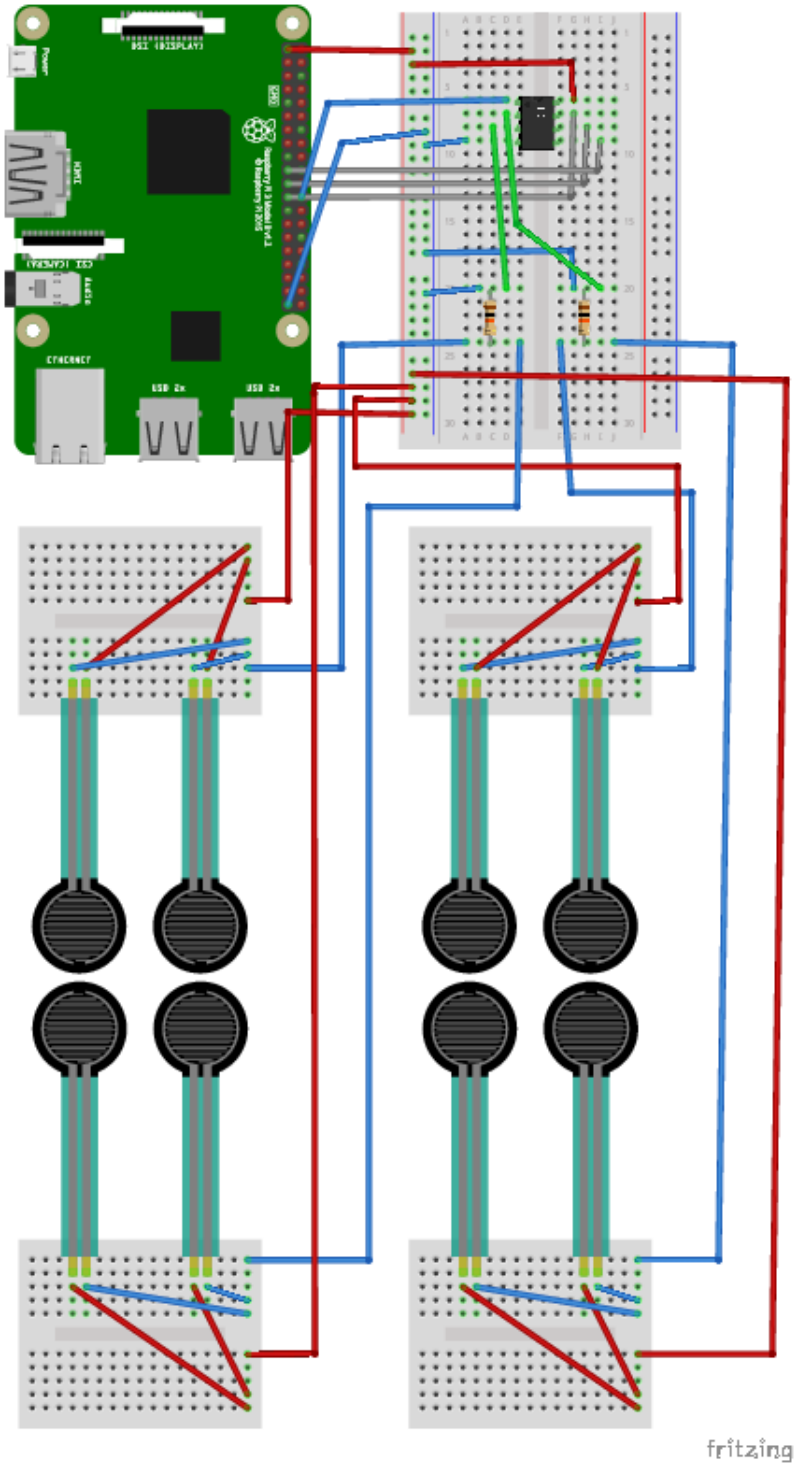

図 4　左上に Raspberry Pi3、右上には抵抗や A/D コンバーターが接続されたブレッドボードがあり、右下と左下の圧力センサーから出た値を抵抗や A/D コンバーターを通して Raspberry Pi3 に伝えられる

左上部に Raspberry Pi3 を示した。Raspberry Pi3 から右上部のブレッドボードに A/D コンバーターや抵抗に配線を伸ばした。各圧力センサーにブレッドボードから電源と各 ch の抵抗からコードが伸びている。圧力センサーが 2 枚ずつ向かい合わせになるように 4 枚 1 組にし、それを 1 枚とするように並べ配線をした。4 枚 1 組を 2 組作り、2 つの物を測れるようにした。4 枚 1 組にした理由は、圧力センサーの大きさ上 1 枚 38mm × 38mm と小さすぎたので 4 つ並べ、76mm × 76mm として動作させ、冷蔵庫に入れる大抵の物なら測れるように考えた。また、圧力センサーに載せるもので小さすぎて 4 枚の圧力センサーに載らない場合でも、1 枚のセンサーに載れば重さを測ることが出来る。その場合、4 枚の圧力センサーに載せて測った場合でも値は変わらない。冷蔵庫の底部が網目状で正確な数値を出せない可能性があり、アクリルのクリップボードを用いて底部が平らになるようにし、圧力センサーを貼り付けた。Raspberry Pi3 を冷蔵庫外に設置し、圧力センサーから Raspberry Pi3 までコードを伸ばした。ソフトの関係で圧力センサーが FSR402 になっているが、実際に使用しているのは FSR406 である。

SPI 通信 (Serial Peripheral Interface) は、マスターとスレーブの間でデータ通信を可能し、この場合はマスターは Raspberry Pi3 でスレーブは A/D コンバーターである。SPI 通信では SCLK、 MOSI、MISO、CE を使用する。SCLK はマスター (Raspberry Pi3) からの同期を取るためのクロック、MOSI はマスター (Raspberry Pi3) からスレーブ (A/D コンバーター) へのデータ送電線、MISO はスレーブ (A/D コンバーター) からマスター (Raspberry Pi3) への送電線、CE はスレーブを選択する信号機である。したがって、Raspberry Pi3 の SCLK は A/D コンバーターの CLK(クロック) に、MOSI は A/D コンバーターの Din(シリアルデータ IN) に、MISO は A/D コンバーターの Dout(シリアルデータ OUT)、CE は A/D コンバーターの CS(チップセレクト) と繋げるということである。Raspberry Pi3 側と A/D コンバータ側に分けて表にしたのが以下の表 2 に示した。

表 2　マスターとスレーブの動きを表した表

Raspberry Pi3 側	-	A/D コンバーター (MCP3002) 側
SCLK	→	CLK
MOSI	→	Din
MISO	←	Dout
CE	→	CS

MCP3002 の 0ch の先には 10k Ωの抵抗を挟んで 4 枚の圧力センサー FSR406 の GND と繋がっている。MCP3002 の 1ch の先も同様に FSR406 の GND と繋がっている。Raspberry Pi3 から 3.3V を + 列に GND を-列に繋げ、抵抗や圧力センサー、A/D コンバーターの電源や GND も + 列や-列に繋げている。MCP3002 の 0ch と 1ch はアナログ入力であり、圧力センサーから入ってきた値をデジタルにし Dout から Raspberry Pi3 に送っている。以下の図 5 が MCP3002 の各ピンの役割を表した図である。

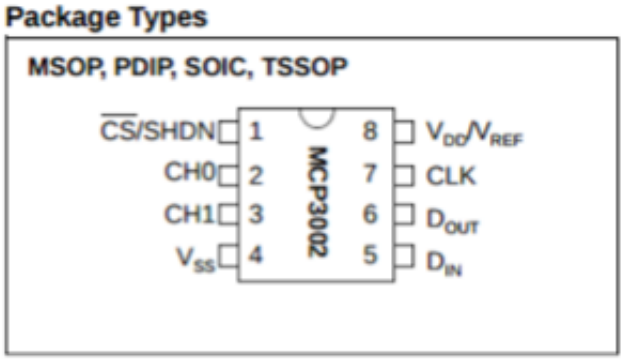

図 5　MCP3002 の各ピンの役割

図 4 では SPI 通信を表すのは Raspberry Pi3 からブレッドボードの伸びている灰色の線である。赤の線は 3.3V の電源を表し、青の線は GND を表している。緑の線は圧力センサーがどの ch の値を検出するのかを表している。図 5 から 2 ピンがが 0ch、3 ピンが 1ch ということがわかる。

5.3　重さの蓄積

圧力センサーを通じ、Raspberry Pi3 に送られてきた重さの値のデータを蓄積するにあたり、データベースを使用した。使用したデータベースは Mysql5.5 である。2 組の圧力センサーのデータの値と値を取得した日時を記録するデータベースを作成した。データベース名が pi、テーブル名を rapi で作成した。テーブルの構造を表 3 に表した。

表 3　データベースのテーブル構造

Field	Type	Null	Key	Default	Extra
no	int(11)	YES	-	NULL	-
no2	int(11)	YES	-	NULL	-
created	detetime	YES	-	NULL	-

圧力センサーから検出された値を保存するテーブルでは no カラムに ch0 の値を、no2 カラムに ch1 の値を、created カラムに登録した時点での日時が登録されるように設定した。データベースは Linux サーバにあるが、登録は Raspberry Pi3 からである。データベースに接続し圧力センサーから出た値を保存するプログラムを重さを検出するプログラムと同じファイルに記述し、重さを検出したと同時に値のデータをデータベースに保存する。重さを検出する度にデータベースに接続しカーソルを取得した後、レコードを挿入してコミットする。

5.4　自動で重さを測る

本研究では 24 時間の監視を目的にしているが、重さを測るのは Web サイトにアクセスした時だけであり、アクセスしない限り重さを検知しデータを蓄積することが出来ない。それではアクセスした現在の重さを確認することは出来るが、どのくらいのペースで食材を消費できているか知ることが出来ない。消費するペースを把握し前もって買っておけば在庫がなくなるという心配もないと思われる。そこでペースを把握するためには重さをグラフで表すのがわかりやすいと考えた。ただ、アクセスする場合に限りデータが蓄積されるとしたら、日時に偏りが生じると考え、自動的にデータを取得しデータベースに保存するよう設定

した。

(a) 自動でプログラムを実行

決められた時刻にコマンドを定期的に実行させるためのデーモンプロセスである cron を使用し、重さを測るプログラムを 1 時間に 1 回実行させた。Raspberry Pi3 で重さを測るプログラムを書き、Raspberry Pi3 の cron を使用し毎時 1 分ごとにプログラムを実行するように書いた。研究室のネットワーク上、Raspberry Pi3 が 1 時から 7 時までネットワークに接続出来ず、データベースでの登録にエラーが生じるのでこの実験では 1 時から 7 時を除いたその他の時間で毎時 1 分ごとにプログラムを実行するように設定した。1 時間に 1 回プログラムを実行し重さを測り、データベースに登録することでペースを知ることが出来ると考えた。

(b) データの消去

1 時から 7 時を除いた 18 回とアクセスした回数分だけデータベースに重さのデータが登録され蓄積される。しかし登録するだけではいつか膨大な量のデータになってしまう。1 日自動で登録される 18 回と 15 回以上アクセスすると考え、1 週間に約 300 データ蓄積される。データベースの中にデータが 300 データ以上あると古いデータから消去し最新の 300 データを残すようなプログラムを書き、常に 300 データがデータベースに保存されているように重さを測ると同時に 300 データを残し古いデータを消去するプログラムを実行するように設定した。

5.5 グラフを作成

Web サイトでグラフを作成するために gnuplot を使用した。これはデータベースからデータを取得し、グラフ化をしている。グラフを使用する目的は、センサーに乗せてある食材はどのようなペースでなくなるかを把握するためである。グラフでは 1 日での減り具合も確認でき、買い物をする際にペースを考えて買い物が出来ると考えられる。Web ページが表示される度にデータベースから最新のデータを取得し、グラフに保存している。

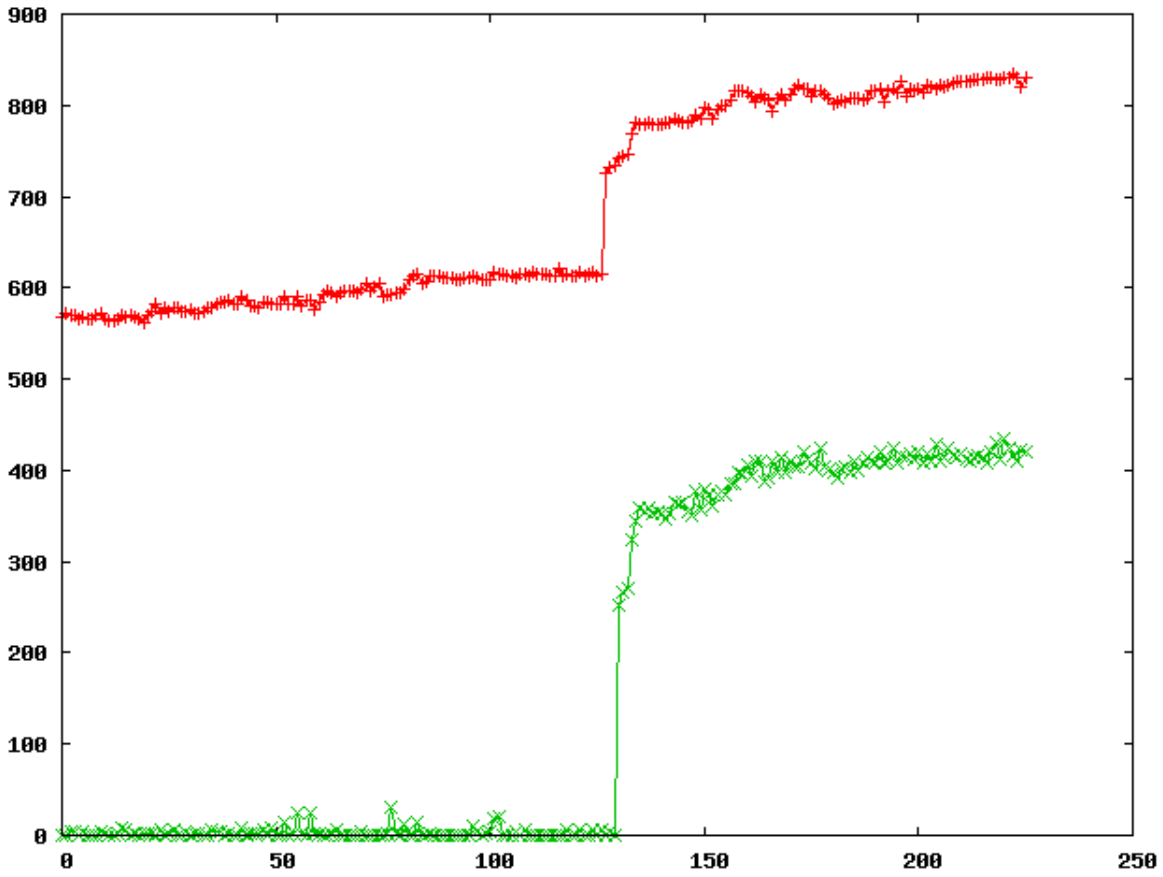

図 6　重さをグラフ化したもので、横軸が時間経過を表しアクセスした時と 1 時間に 1 回重さを測ったのが 1 つの点でありそれが何個あるか表し、縦軸が重さ (g) を表す

実際のデータをグラフ化したものを図 6 に示した。赤が 0ch の重さの推移、緑が 1ch の重さの推移である。縦軸が重さの軸で横軸が大きくなるにつれて重さが重いことを表し、横軸が時間経過を表す軸で縦軸が大きくなるにつれて最新のデータである。このグラフでデータベースにデータがある分、つまり約 1 週間分のデータがあるのでひと目でどれくらいの早さで消費されているか確認でき、買い物の食材も考えやすくなるだろうと考える。

5.6　Web ページに表示

Web ページを表示させるために、apache を使用している。表示されるまでのプロセスはクライアント側の PC から http://ks133.a4w にアクセスすると Linux サーバの apache が動き、表示するプログラムが読まれるというものである。pi.php に記述されているように Raspberry Pi3 の/home/pi/denn.py を実行するように記述したシェルスクリプトが動き実行されることで、アクセスされると重さを測りデータベースに送る、また写真を撮って Linux サーバの/var/www/picture に送るという動作が実行される。その結果、Linux サーバの/var/www/pi.php ではデータベースから最新データを取得し、/var/www/picture から最新の画像を取得し、Web ページに表示出来る。画像を使用することによりリアルタイムで重さを測っている物体を確認することが出来る。

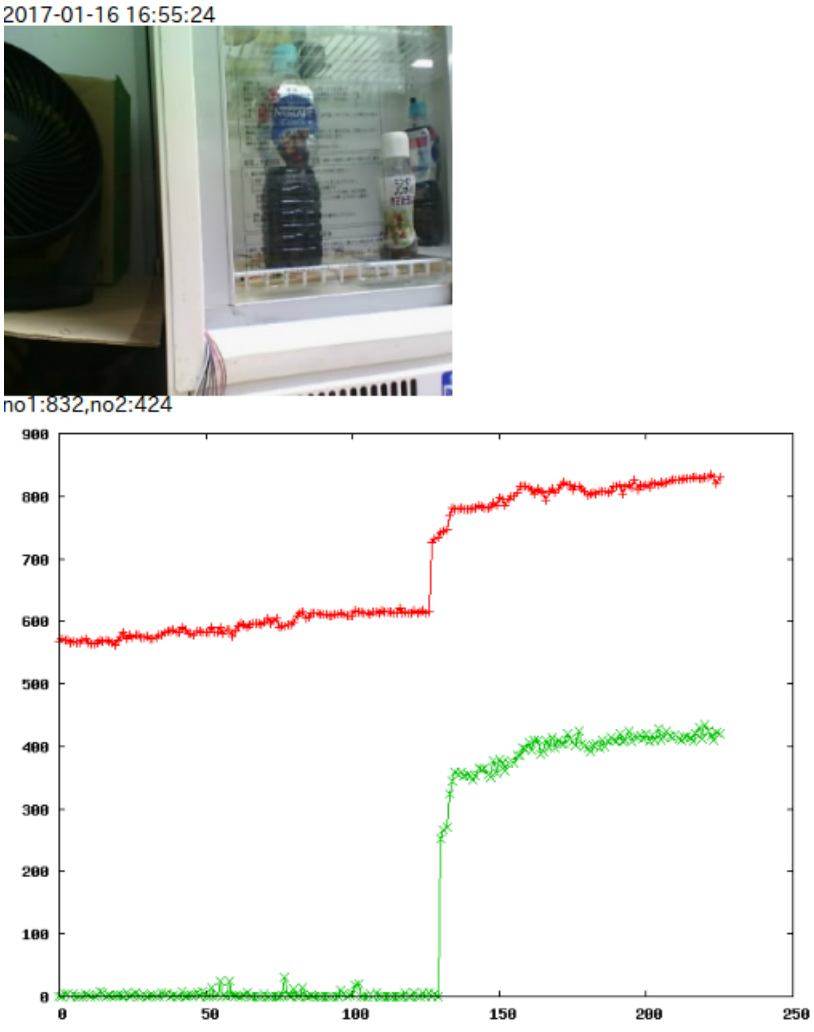

図 7　Web サイトに表示されている画面

実際の Web ページの様子を図 7 に示した。リアルタイムで撮影した画像が表示されて圧力センサーの上に置かれているものを確認することが出来る。その下には 0ch と 1ch に分かれて重さの表示がされていて、どちらがどれくらいの重さがあるかをひと目で確認でき、グラフも表示されることでペースもわかりより良い買い物が出来ると考える。

5.7　設置図

Raspberry Pi3 を冷蔵庫の外部に設置、センサーを貼り付けたクリップボードは冷蔵庫の内部に設置し、Raspberry Pi3 とセンサーを繋ぐコードは冷蔵庫の扉に挟んだ。また、冷蔵庫はガラスが透明でカメラでも確認しやすい業務用の冷蔵ショーケースを使用した。Raspberry Pi3 は無線が内蔵されているため、WiFi でネットワークに通信をしている。

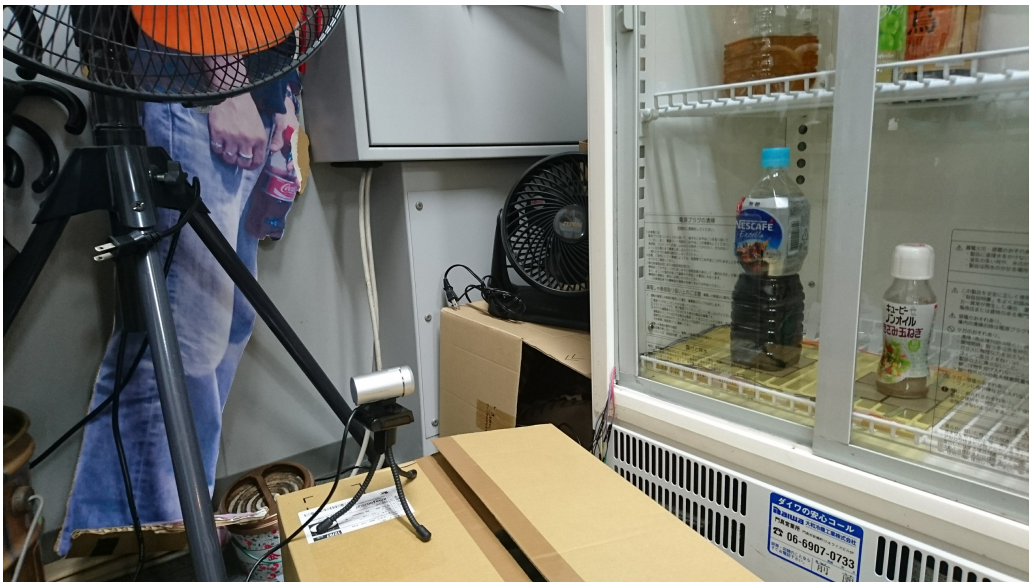

図 8　全体図

全体図を以上の図 8 に示した。画像を撮影する USB カメラが図の中央部にあり、USB カメラは Raspberry Pi3 に繋がっており、図の右には冷蔵庫とセンサーが見られる。図の中央部には熱対策として USB ファンを挟んで真ん中に Raspberry Pi3 が設置されている。Raspberry Pi3 のピンは電子回路基板はジャンパーワイヤーで繋がっていて、電子回路基板には A/D コンバーターの MCP3002 や 10k Ω の抵抗がハンダ付けされている。そこからセンサーに向けてジャンパーワイヤーを繋いでいる。

図 9　センサー周辺を拡大したもので圧力センサー 4 枚で 1 つのモノが載っていると確認できる

センサー周辺を拡大して撮影したものを以上の図 9 に示した。センサーの上部と上部をくっつけて 4 つのセンサーを隙間なく敷いた。センサーから伸びている 3.3V や GND などのジャンパーワイヤーはクリップボードの端に寄せ、圧力センサーの上に載せるものの邪魔にならないように考慮した。さらにジャンパーワイヤー同士バラバラにならないようにテープで等間隔にクリップボードと貼ってある。

6　結果と考察

本研究の目的は外出中でも冷蔵庫の材料や分量を知ることができ、家の食材に配慮した買い物が出来るシステムを開発することである。

今回は実際に冷蔵庫にある残りの分量を確認せずに買い物に赴いた。この実験で使用したものは 1L の紙パックの牛乳と 1L のペットボトルの飲み物である。牛乳は内容量が変わっても見た目に現れることがなく、ペットボトルの飲み物は透明なので見た目で確認することが出来た。買い物をしてみると内容量に変化があっても見た目でわからない牛乳より、内容量に変化があると見た目でも確認することの出来るペットボトルの飲み物の方がわかりやすく、必要な量の買い物が出来ると感じた。その点でカメラの導入は良かった。実際に買い物をし、このシステムを使用して買い物をした結果、必要な分の買い物が出来ると考えた。

7　今後の課題

今後の課題としては以下の 7 点である。

1. 使用した圧力センサーでは軽いものが正確に測れないことがある
   使用した圧力センサー、FSR406 の感圧範囲は 100g から 10kg である。今回の実験対象としたものは 1L の牛乳と 1L のペットボトルの飲み物で重さの変化は 0mg から 1000mg である。その中で 100g 以下の重さになってしまうと正確な重さが測れないことが多い。本研究の目的では残りの分量を知ることができ、それに配慮した買い物が出来るシステム作りなので少ないと理解できても正確性は必要だと考えられる。
2. 圧力センサーの数が少ない
   今回の実験では圧力センサーを 8 枚用いて 4 枚を 1 組とみなし 2 組作った。したがって、今回重さを測れるのは 2 組であった。冷蔵庫に入っている食材が 2 つというのはほぼありえない。よって、より圧力センサーの数を増やす必要があると感じる。しかし、今回は 1 枚の圧力センサーのサイズが小さかったため組み合わせて作ったが、今後はより大きい圧力センサーの使用を検討する必要もあると感じる。
3. 合計値しかわからない
   家庭の冷蔵庫でも、タッパーなど重ねることが出来るものは重ねて冷蔵庫に入れてある光景をよく見る。今回のシステムでは重さの合計値しか出せないので重ねた際の個々の重さが測定出来ない。
4. 底が平らなものを置かないと正確な値が出ない
   今回の実験対象は牛乳とペットボトルの飲み物である。牛乳パックは底が真っ平らなので問題なく重さを測れたのだが、ペットボトルは容器への圧力を吸収して変形を防ぐ役割で側面や底面に凸凹のデザインになっている。炭酸飲料などのペットボトルは特に接地面が少ないので反応しないので、値が出ないことがある。また、コップの底やお茶碗などの高台といったものも接地面が少なく、反応しないしないことが少なくなかった。
5. 部屋が暗くなると画像の撮影が出来ない
   冷蔵庫の中に入っている対象物の撮影で部屋が暗くなると明かりがなくなるので撮影しても何も映らない。撮影する際に冷蔵庫の内部から明かりをつけるなど工夫が必要だと感じる。フォトトランジスタを使用すれば明かりを検知出来るので、撮影できるくらいの明かりがないときのみライトをつける設定が出来ると考えられる。
6. 残量が値で出るが、感覚的に残量がわかりにくい
   Web サイトに実験対象の画像と重さが表示されるが、実際持っていないので感覚的にわかりづらい。そこで一番最初に実験対象を置くときを定めておいてそこからどれだけ減ったかを視覚的に表すことが出来ればもう少しわかり易くなるのではないかと考える。
7. スマートフォンでは見づらい
   スマートフォンではグラフの細かい点が見えづらく正確な数値を確認することが難しいを考える。だが、グラフはペースを確認するためのもので正確に分からなくてもペースを知ることは出来ると考えられる。また、表示した重さの数値なども見えづらいと感じることもあることからレスポンシブデザインを採用したら改善されると感じた。

謝辞

本研究の研究作業を進めていく上で大垣斉准教授からご協力及びご指導頂き、深く感謝致します。また、情報教育システム研究室のメンバー及び卒業生の方々に深く感謝の意を表します。

参考文献

[1] パナソニック株式会社. スマート家電. http://panasonic.jp/pss/.

[2] 東芝ライテック (株). フェミニティ対応冷蔵庫. http://feminity.toshiba.co.jp/feminity/service/concierge_refrigerator.html, 2016.

[3] 山本敦. ボッシュ、外出先で冷蔵庫の中身が分かるスマート家電など. http://akizukidenshi.com/download/ds/microchip/mcp3002.pdf, 2015.

付録 A　ソースコード

A.1　ソースコード.1

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python
# coding: utf-8

import MySQLdb
import spidev
import time
import subprocess

import commands
import os

#デバイスオープン spi
spi = spidev.SpiDev()
spi.open(0, 0)

try:
        res = spi.xfer2([0x68, 0x00])
        re1 = spi.xfer2([0x78, 0x00])

        pe = (res[0] * 256 + res[1]) & 0x3ff
        pe1 = (re1[0] * 256 + re1[1]) & 0x3ff

        print pe
        print pe1

except KeyboardInterrupt:

    spi.close()



# に接続しカーソルを取得するDB
connect = MySQLdb.connect(host='192.168.122.133', port=3306, user='pi1', db='pi', charset=
cursor = connect.cursor()
```

```
# 写真を撮る
os.system('fswebcam␣--no-banner␣-S␣20␣sample.jpg')

#写真を送る
os.system('sh␣sousin.sh')

#レコードの挿入
sql = u"insert␣into␣rapi␣(created,␣no,␣no2)␣values(now(),␣%s,␣%s)" % (pe, pe1)
cursor.execute(sql)

connect.commit()      # コミットする

cursor.close()
connect.close()       # データベースオジェクトを閉じる
```