COOLS

Cooling Orginary Objects Lickety Split

Phase 1 Report

CS413 Embedded Systems

Paul McGurk, Alex McBride, Daniel Rafferty,
Andrew Mortimer, and Scott Henderson

October 18, 2015

Contents

1 Introduction

The project is to create a product using either an Arduino and/or Raspberry Pi that will be a part of the Internet of things. The Internet of things is about taking everyday objects and embedding them with software, electronics and networking capabilities so that they can sense the environment around them and then send and receive data. The idea, after much thought and discussion will be to create a smart fridge. This idea will not only have a physical benefit to its application but also an environmental one, as from the start the design is centred on trying to reduce food waste and energy consumption.

2 Assessment of Capabilities

2.1 Requirements

The device must be able to support a touchscreen, have a thermometer connected, camera to act as barcode scanner, wireless capabilities and be able to run a web server.

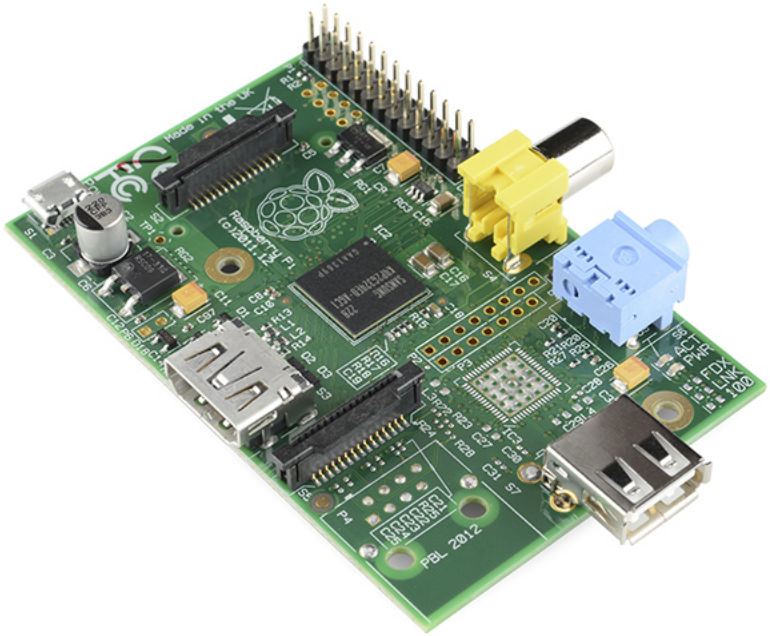
2.2 Arduino

The Arduino Uno V3 is a small microcontroller board sporting an ATmega328p processor. It is designed for prototyping embedded products and as a result has good support for driving ICs and other low-level peripherals. The 6 PWM outputs allows analogue components to be operated easily, and the lean 16MHz processor is extremely low power meaning it could potentially be powered from a battery.

TODO: table

2.3 Raspberry Pi

Figure 1: Raspberry Pi



The Raspberry Pi is a small form-factor ARM based computer. It is a very popular system that has a healthy development ecosystem. The RPi can run a custom Linux-based operating system, called Raspbian, allowing fairly high level development and integration with lower-level peripherals such as GPIOs. The Raspberry Pi also has official peripherals or "modules" for devices such as a camera or a touchscreen.

There are several models of Raspberry Pi available:

TODO: table

2.4 Choice

We have chosen to go for the Raspberry Pi 2. The reason we chose a RPi over an Arduino was that we needed the higher performance that the RPi line offers in order to serve our web pages and do the barcode scanning in software. Furthermore we wanted internet connectivity and we felt this would be easier to achieve in the Linux based Pi. The reason we chose to use the model 2 instead of the model A was the better connectivity for peripherals in terms of USB ports and available GPIOs, with the higher performance an added bonus.

3 Hardware Design

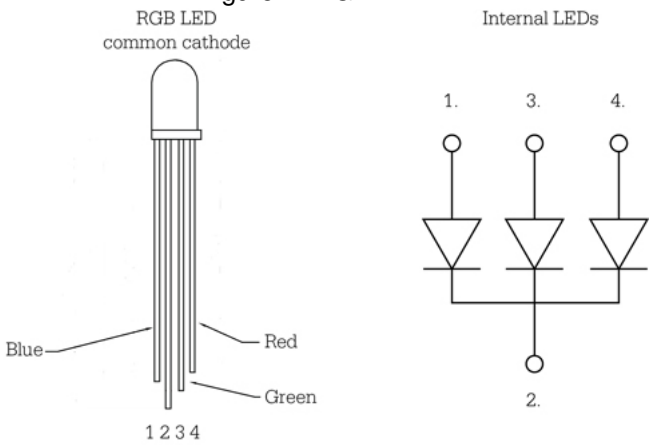
3.1 Main Parts

RGB LED

An RGB LED (Red Green Blue Light Emitting Diode) is a small light which has three different diodes within it, capable of emitting three types of light. Each pin controls a different colour, with the longest pin being the PLUS. Using PWM (Pulse Width Modulation), we can dim and lighten the different colours to give a range of colours. The operating temperatures are also well within the range of the fridge.

This is being used within the project as a visual display of the temperature of the fridge, which Blue representing when the temperature is near the ideal temperature, and red when at room temperature.

Figure 2: RGB LED



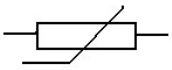
TODO: table

Resistor

A resistor is a device which reduces the flow of current within a circuit. This can be used to protect elements from high current, as well as other things.

This is being used within the project to protect the RGB LED, and within the temperature part of the circuit to work out the resistance of the themistor (see below).

Figure 3: Resistor



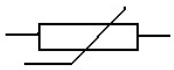
TODO: table, change image

Thermistor

A themistor is a resistor which is sensitive to temperature. The resistance of the thermistor is used to work out the temperature of it's surroundings using a variety of different math techniques, notably the Steinhart-Hart equation.

This is being used within the project to work out the temperature within the fridge, allowing us to monitor it in real-time.

Figure 4: Thermistor



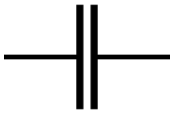
TODO: table

Capacitor

A capacitor is a device which stores electrical energy temporary.

This is used within the project to work out the temperature of the fridge, which is done by measuring how long it takes to empty, with the resistance of the thermistor being dependent on temperature.

Figure 5: Capacitor



TODO: table

Raspberry Pi Camera

The Raspberry Pi Camera Board connects to any Raspberry Pi or Compute module and allows for high definition photography. It has several useful features, including: high data capability, 5 mega-pixel fixed focus, support of 1080p, 720p60 and VGA90. It also has automatic control functions including white balance, exposure control, luminance detection. The Pi Camera Board has a 15cm ribbon cable which slots into the Pi Camera Serial Interface Port. Via Raspbian, there are several applications that can be used to take photos, including Raspistill.

Figure 6: Raspberry Pi Camera

TODO: table

The minimum operating temperature of -30ÂřC means we could mount the camera inside the fridge if we so desire.

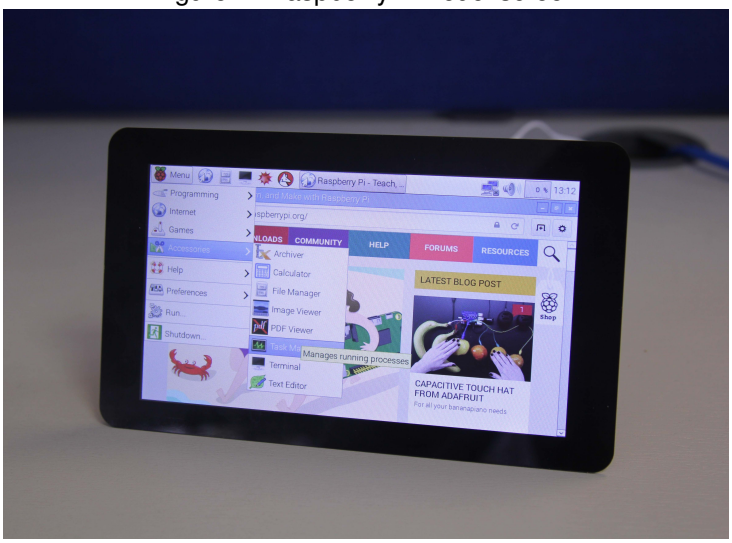
This will be used within the project to scan the barcode of the products being entered into the fridge. It could possibly automatically detect the use-by date as well.

Raspberry Pi Touch Screen Display

The Raspberry Pi 7" Touchscreen Display connects to a raspberry pi, and allows it to display graphics in a self contained package. It also allows the user to interact with the GUI through touch, ommiting the need for additional peripherals such as mouse and keyboard.

We choose this screen as others were seen as too small to be able to display a useful amount of information, and too small to be able to allow the user to type on.

Figure 7: Raspberry Pi Touchscreen



TODO: table

This will be used within the project to display a custom GUI which will have a number of uses, such as; display all products which are near their use-by date, adding custom products (homemade food, products which do not scan properly, etc). From this screen, the user will also be able to see the temperature of the fridge, and possibly change the temperature within the fridge, if that is implemented.

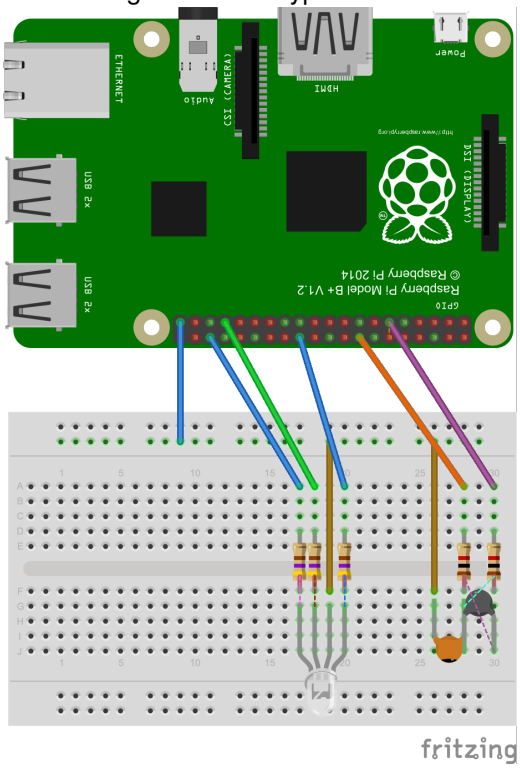
3.2 Component List

TODO: this

3.3 Prototype Circuit

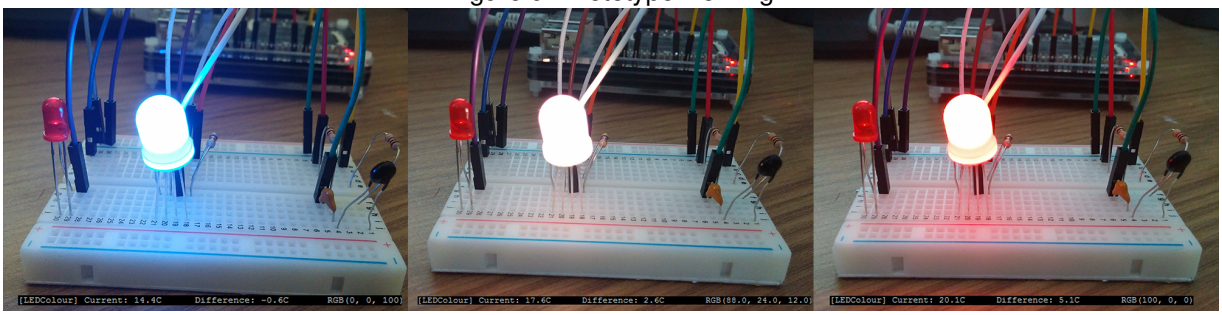
TODO: text about the prototype here

Figure 8: Prototype Circuit



Result

Figure 9: Prototype working



4 Software Design

4.1 Operating System

Raspbian is a Linux Debian-based operating system which is designed to run on a Raspberry Pi. It optimizes full power of the Linux desktop environment for the Raspberry Pi's hardware. It allows us to focus on writing the functionality of the software that is unique to our smart fridge, leveraging the rich software ecosystem of 35000 packages that are offered by Raspbian to implement some of the more "standard" technologies such as the web server.

4.2 Accessing the hardware

RPi.GPIO is a library that allows high level access to the GPIOs on the board through python bindings. Since we currently do not have plans to utilize the hardware PWM, SPI, or I2C functionalities we are not limited by the fact that RPi.GPIO does not support these. If we choose to use any of these functionalities then we can implement them in C and call down from python fairly easily.

4.3 Web server

Why use a web server?

Most of the functionality of our project requires us to have an interface to the fridge to administrate and view information. We have chosen to go with a web-based approach, with the RPi itself hosting the web server. Going with this approach allows us to write the client application once and access it from a phone or a desktop without having to write client applications directly for each.

Technology stack

Since the RPi.GPIO library provides access to the GPIOs via Python, we thought it prudent to serve the dynamic content of our application from Python also. We will achieve this using theFlask framework. Flask features a web server and a templating engine to allow us to write static HTML with wildcards that allow us to dynamically insert relevant data. Although none of us have had any experience with this technology before, we thought it was a natural fit and do not anticipate much trouble in utilizing it.

4.4 Temperature Measurement

blah

4.5 Barcode Scanning

Barcode Scanning Library

Zbar is an open source barcode reading library that supports a wide range of barcode formats. It has Python bindings allowing us to read the barcodes in Python.

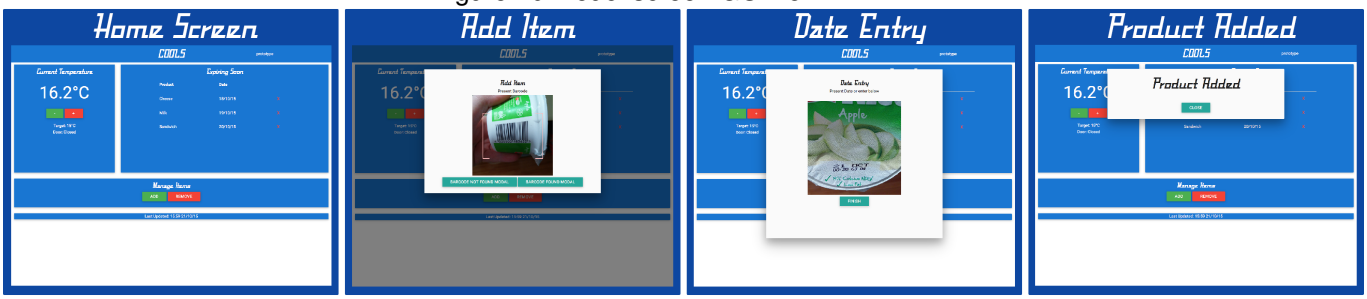
Identifying Products using the Barcode

Outpan is a web service that provides an API for identifying products from their GTIN. It is free to use and has a database of millions of products. We can make simple HTTP requests with the barcode we just scanned and get information such as the name of the product back.

4.6 Interface

Touchscreen The touchscreen interface will be written in HTML5, JavaScript and CSS3, using the Materialize CSS for the visuals. As for the keyboard for entering data, we will use matchbox keyboard.

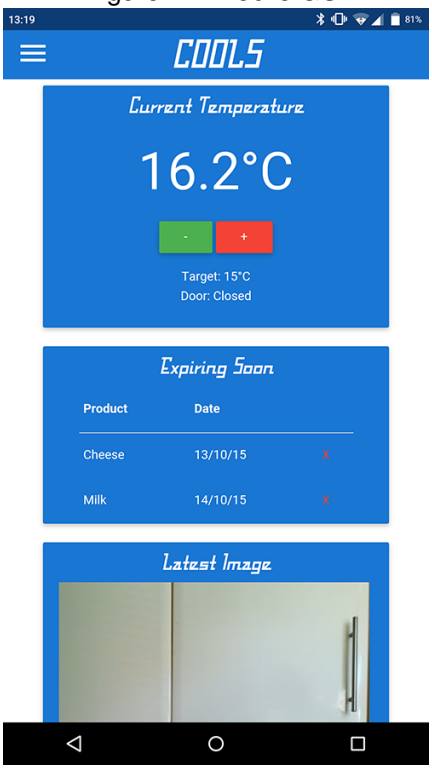
Figure 10: Touchscreen GUI flow



Mobile

The touchscreen interface will be written in HTML5, JavaScript and CSS3, using the Materialize CSS for the visuals. As for the keyboard for entering data, we will use matchbox keyboard.

Figure 11: Mobile GUI



5 Current Progress

Our excellent group organisation has allowed us to have already made good progress on Phase 1 of our project. To begin with, we have researched all of the necessary components to complete our project, including deciding on a Raspberry Pi 2 as the basis for our embedded system. We have also created a demo to sense temperature and change the colour of an LED using this Pi, as this is one of the main features of our project. Another feature that we have already implemented is a Flask web server running on the Pi. Flask will give us an easy way to store data and check the values of the sensors remotely via a web interface. We have also planned a prototype that will allows us to identify the major challenges in our design/software before moving onto the final build of our project. The basis of a web interface has been created to allow a user to view/change temperature, display current contents of the fridge, including expiration dates, and also to show the latest image from the PI camera.

6 Conclusion

We have outlined a solid plan to go forward and start developing our smart fridge. By staying as organised as we have been so far, we should meet all the deadlines (submission ones as well as ones we have set ourselves) with no trouble. Building a smart fridge will be very interesting, not only because we will have a useful product by the end of the project, but also because of the environmental implications. By being able to regulate the temperature of the fridge, as well as alerting a user if the door has been left open, we will ensure the fridge will use the least amount of power possible. Secondly, by giving the user an obvious visual representation of which foods are going out of date soon, there is the hope that this will reduce food wastage. Furthermore, it is generally a useful system which will be applicable to the real world. We look forward to making good progress, and our strong group dynamic will allow us to overcome any obstacles with minimal stress.

TODO: BELOW IS JUST OLD STUFF FROM THE TEMPLATE, REMOVE WHEN TABLES ARE DONE Use the `callout` command:

By the shores of gitchee gumee
by the shining big sea waters
stood the wigwam of Nokomis
brother of the moon, Nokomis.

7 Old stuff, ignore below

7.1 Writing in a box

Use the `frame` command:

'Twas brillig and the slithy toves did gyre and gimble in the wabe
all mimsy were the borogroves, and the mome raths outgrabe.
Beware the Jabberwock, my son, the claws that bite, the jaws that snatch
Beware the Jubjub bird, and shun the frumious Bandersnatch.

8 Presenting data

This describes more general stuff. If you are experienced with LaTeX, you may already know this.

Figure **??** shows a fat horse. You might use a more informative graphic. If you make it in excel, make sure the dimensions are right before you export it, or it will be fuzzy. I do that, and then copy it into paint and save it as a *.png (or *.jpg). For Stata graphics, you can either save them as vector graphics (*.eps) and use `epstopdf` or save them as a normal graphic. The advantage of vector graphics is that you can scale them without fuzziness.

8.1 Tables

Table 1 was produced using `est2tex` to get results directly out of from Stata. The tables were beautified using the `booktabs` package. Both of these things are worth doing, and will save time and enhance appeal.

Table 1: Effect of Codes

	Natural Gas		Electricity		Gas & Electricity	
	(1)	(2)	(3)	(4)	(5)	(6)
Log Electric Price	.412***	.410***	-.307***	-.334***	.048	.028
	(.111)	(.053)	(.048)	(.027)	(.055)	(.028)
Log Gas Price	-.314***	-.472***	.132***	.202***	-.073***	-.099***
	(.047)	(.032)	(.013)	(.013)	(.021)	(.014)
Percent Code Units	.470*	.799***	-.001	-.147***	.215	.267***
	(.281)	(.110)	(.099)	(.044)	(.145)	(.051)
Log Heating Degree Days	.697***	.621***	.113***	.146***	.383***	.370***
	(.072)	(.058)	(.024)	(.021)	(.043)	(.027)
Log Cooling Degree Days	.011	.008	.058***	.061***	.015	.015*
	(.017)	(.019)	(.012)	(.010)	(.010)	(.009)
Log Median Income	.089	.116*	.031	.008	.034	.030
	(.119)	(.065)	(.040)	(.028)	(.047)	(.028)
Instrumental Variables	No	Yes	No	Yes	No	Yes

Confidence Level: *90% **95% ***99%