CS-308-2014 Final Report

Smart Kitchen Containers

Akhil Surya Vamshi, 120050067
Jagadeesh Nelaturu, 120050055
Bhargav Chippada, 120050053
Pranay Dhondi, 120050054

Table of Contents

1. Introduction

In a day where an average Indian is getting more and more health-conscious and the notion of quantifie self is stronger than ever there is no automatic way to count your calories or other protein information seemlessly and continuously. Imagine another scenario, a household kitchen, where different commodities are being used everyday. We would want to track and if any of them needs to be refilled. Our Smart Kitchen Containers help users in tracking their usage of kitchen commodities, thereby alerting the user if it needs to be refilled and also tracking calorie consumption everyday.

2. Problem Statement

There is a platform on which different containers are being placed, and then replaced. The weights of these containers are to be logged and displayed to the user, along with daily consumption and calorie consumption in an appealing and easily comprehensible manner. If the amount of a commodity goes below a certain level, the user is to be alerted.

3. Requirements

3.1 Functional Requirements

- If a container is put on the platform or taken from it, both the RFID readings and combined weight is sent to the Raspberry Pi, and is stored on the SD card.
- In the above case, the data is processed and the weights of the relevant containers are stored separately for easy retrieval.
- A HTTP server continuously runs on the R-Pi, to which the user can connect whenever desired.
- The home screen continuously shows the commodities present, and the current amount of every commodity. Also, it also alerts the user if the quantity of a commodity goes below a set minimum value.
- Whenever the user requests for the details of a particular commodity, the data pertaining to this commodity is processed and plots of its weight, consumption and calorie consumption are shown to the user.

3.2 Non-Functional Requirements

The platform should be measuring the weights of the containers to a precison of 10 grams
When the contents of a container go below a pre-specified amount level the website should be showing some sort of warning the first time the page refresh happened
The user does not have to bother about the place in which they are placing the container as long as it's on the shelf. Hence a misplace of container by few cms. is allowed

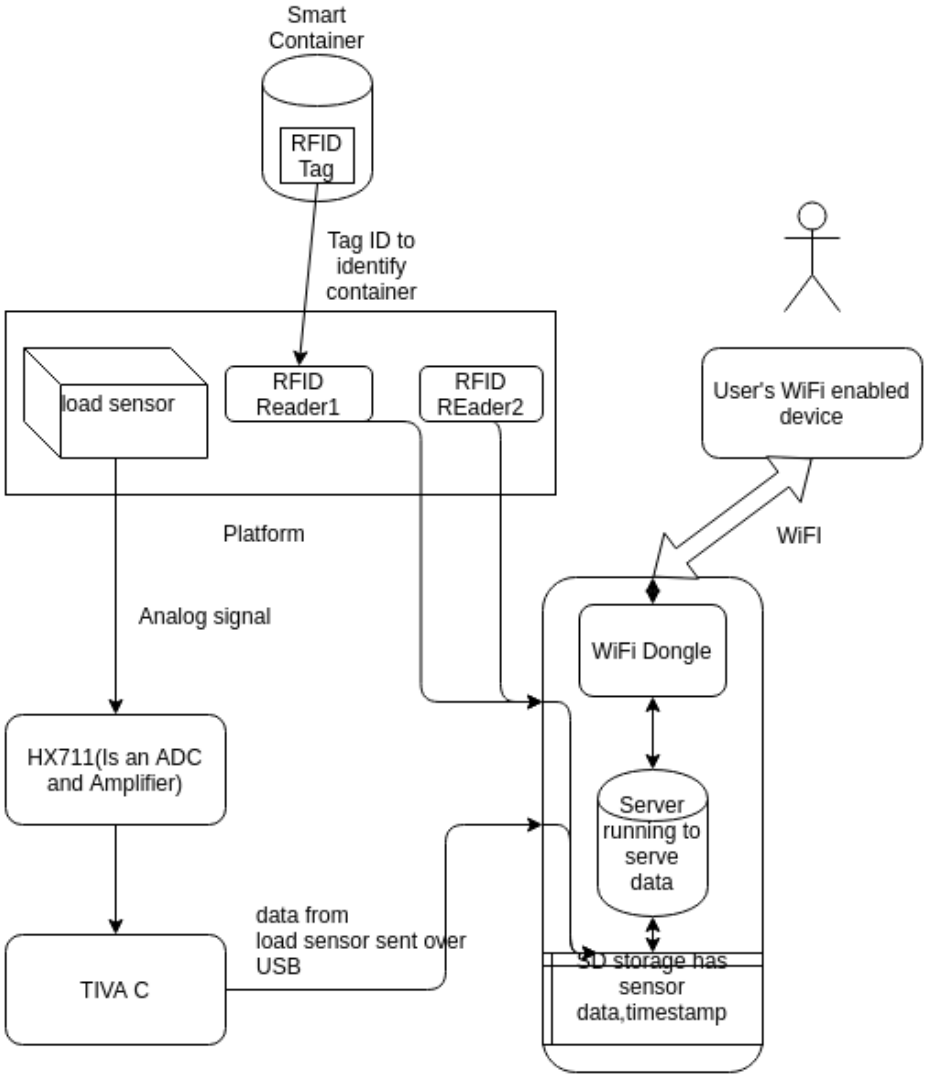
3.3 Harwdare Requirements

1) 1 Mechanical platform
2) 1 Load sensor + 1 HX711
3) 2 RFID readers
4) 2 containers and RFID tags sticked to them
5) Raspberry Pi with SD card
6) Tiva C board
7) WiFi dongle
8) Connectors

3.4 Software Requirements

1) Python and g++ on Raspberry Pi
2) Energia to program TIVA C
3) Web browser that supports HTML5, JavaScript

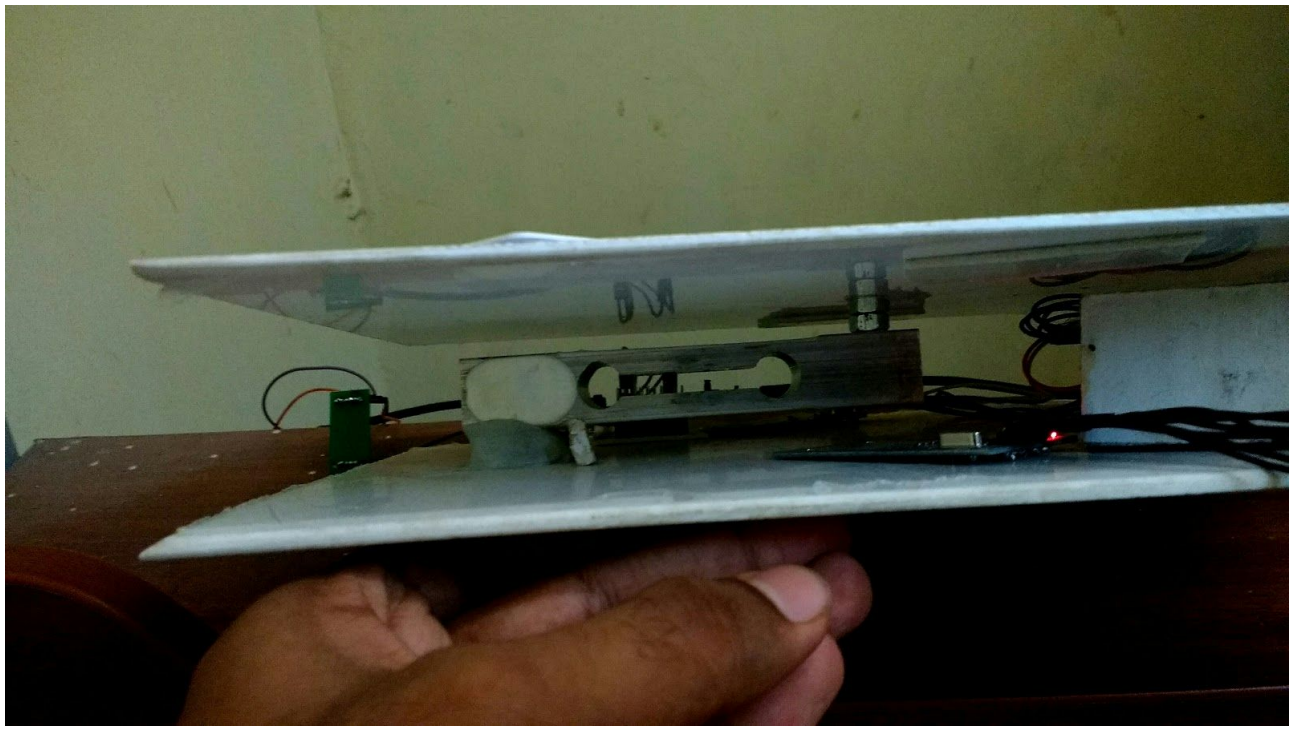
4. System Design



Shown above is an architecture diagram of our final product. We have indicated how and what each component communicates with other containers. Essentially the data flow is

- The RFID reader records the ID of the tag that comes into it's range
- The TIVA-C keeps sending the latest reading from load sensor(that is amplified and digitised by HX711) to R-Pi over USB Serial every 5 seconds.
- When a new tag is taken off and put back the software running on R-Pi checks for a weight change and adds it the change log(stored on a file on SD card) along with time stamp and the tag ID to associate the change in weight
- The HTTP Server being served from R-Pi uses the change log saved before and shows it the user through graphs for each container separately

We also have an interesting mechanical setup needed for the load cell. It can seen in the image below.



As you can see one side of the load cell is supported by the base and the other end supports the top. Hence the difference in pressure is used to drive potential across proportional to the weight placed on the platform. This prortionality means there is caliberation needed and we did by placing few known weights on the platform and calcualting the necessary constants.

5. Working of the System and Test results

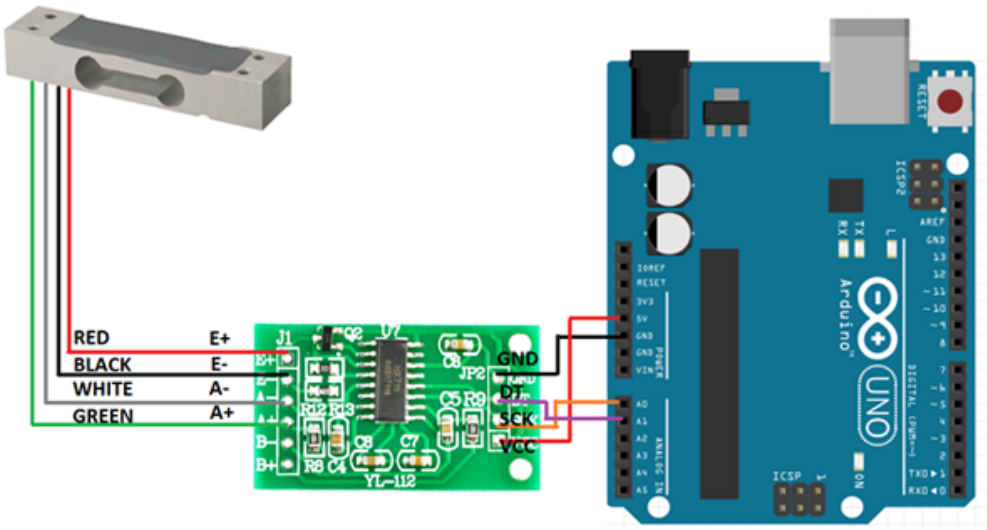
- The RFID reader records the ID of the tag that comes into it's range
- The TIVA-C keeps sending the latest reading from load sensor(that is amplified and digitised by HX711) to R-Pi over USB Serial every 5 seconds.
- When a new tag is taken off and put back the software running on R-Pi checks for a weight change and adds it the change log(stored on a file on SD card) along with time stamp and the tag ID to associate the change in weight
- The HTTP Server being served from R-Pi uses the change log saved before and shows it the user through graphs for each container separately
- We used d3.js to present the data in graphs that are updated when you refresh the browser

The functional requirements from SRS are

1. Test RFID hardware and interface with R-Pi

   We tested by connecting the reader the Reader to laptop through USB-serial communicator and found that it wasn't functional and ordered new ones which were tested and used for interfacing as they worked fine. We interface using some C++ libraries mentioned in references.

2. Interface load cell to measure weight
   We interfaced the load cell connected to HX711 to TIVA as we had timing issues with using R-Pi and HX711 together. The circuit looked something like



   We however had to map corresponding ports from aurdino to TIVA. The code uploaded to TIVA also sent the data recorded to R-Pi every 5 seconds through USB

3. GUI with data visualization and alerts

   We have implemented both visualizations through graphs and alerts by highlighting containers that are running low in amount

4. Protoype with frame to put containers
   We have used acrylic boards to construct the platform to place containers on using the mechanical structure like shown before. This can also be seen in the image

below

6. Discussion of System

a) What all components of your project worked as per plan?

1. The load sensor used required us to interface with a HX711 and connect it through a Tiva board. All other mechanical and electronic components worked as per plan.
2. There was also the RFID Reader that did not work so we had to switch for new one.

b) What we added more than discussed in SRS?

1. We couldn't time HX711 with Raspberry Pi. Hence, we had to connect it to Tiva board, and then communicate to R-Pi using USB-to-Serial.

c) Changes made in plan from SRS:

1. We initially planned to use 1 RFID reader for 2 containers and then switched to using one for one as the RFID readers we had were pretty low range.
2. We also added HX711 to amplify and digitalize the signal from load sensor
3. Like described before an extra TIVA C board entered in between HX711 and R-Pi as there issues with timing the HX711 to work with R-Pi which we learnt based disucssions on online forums

7. Future Work

- The user interface can be extended to be viewed from an Android/iOS mobile phone for the users' convenience.
- The platform can be made to accommodate a larger number of containers by using more load sensors and RFID readers.
- Visiting the website requires the IP of raspberry which hasn't been found automatically. We could customize the server to send a particular reply on particular request to pair the device with R-Pi without having to enter the IP
- The mechanical design can be improved to make the platform sleek.
- This product can be marketed in collaboration with other Smart-Kitchen and Modular-Kitchen platforms.
- When a commodity is beyond a certain minimum quantity, the user can be alerted through SMS, or the application can place an order to the seller directly.

8. Conclusions

Our final prooduct not only provides the convienience of information on groceries in your kitchen avaliable everywhere and hence saving you that time you care about, it also assits you to live a healthy life where you take informed decisions for your family through the seamless calorie counting service we provide.

9. References

1. Library to interface HX711 with TIVA and send it to R-Pi
2. Energia to program TIVA
3. Graph plotting through Data Driven Documents
4. r-pi bcm2835 library (http://bsd.ee/~hadara/blog/?p=1017) (http://www.airspayce.com/mikem/bcm2835/index.html)
5. Bootstrap(http://getbootstrap.com)