Senior Project

Design Document

DRINC
Dynamically Refreshing Interplexing Number of Cordials

Brandon Arnold
BArnold@vtc.edu
Owen Ledvina
OLedvina@vtc.ed

Hoang Phan
HPhan@vtc.edu
Kyle Timins
KTimins@vtc.edu

January 28, 2014

Revision History

Revision Number	Date	Comment
1.0	Nov. 15, 2013	First version of document available
1.1	Jan. 13, 2014	Converted document to $\LaTeX$

Contents

List of Figures

1 Introduction

This design document provides documentation which will be used to aid in software development by providing the details for how the software should be constructed. This design document will also provide some documentation about the hardware that will be used to aid in the hardware development.

1.1 Purpose

The purpose of this document is to describe the design of DRINC and itâĂŹs use cases. The purpose of the Design Document is to provide a description of the design of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to be built. This document provides information necessary to provide description of the details for the software and system to be built.

1.2 Scope

This software design document is for the base level system which will work as a proof of concept for the use of building a system the provides a base level of functionality to show feasibility for large scale production use.

1.3 Acronym/Definitions

DRINC Dynamically Refreshing Interplexing Numbers of Cordials

2 | Use Cases Model Survey

2.1 Logging In

In this senario the user logs into the system.

2.1.1 Actors

- User
- System

2.1.2 Assumptions

- User already has valid login information.
- User is at the login page.

2.1.3 Procedure

- System prompts user for login information
- User enters correct login information
- User clicks "Login"
- The system logs the user in

2.1.4 Expected Outcome

- The user is now logged in.

2.2 Loggin Out

In this senario the user logs out of the system.

2.2.1 Actors

- User
- System

2.2.2 Assumptions

- User is already logged in

2.2.3 Procedure

- User clicks the “Logout” button
- The system logs the user out
- The system prompts the user for the login information

2.2.4 Expected Outcome

- The user is now logged out, redirected to the login page and is prompted to login.

2.3 Create a Custom Drink

In this scenario the user creates a drink from scratch, selecting what ingredients he wishes to use, as well as their amounts. The user then saves this recipe for future use.

2.3.1 Actors

- User
- System

2.3.2 Assumptions

- User is already logged in
- User is already at the create a draink page

2.3.3 Procedure

- System presents user with a list of available options to make a drink
- User selects which drinks to use
- User inputs how much of each ingredient to use
- User chooses a name for the new drink
- User clicks “Create Drink”
- The system stores the new drink in the database

2.3.4 Expected Outcome

- The drink is correctly placed into the database for future retrieval.

2.4 Create a Drink

In this scenario the user selects which premade drink he/she wants DRINC to produce.

2.4.1 Actors

- User
- System

2.4.2 Assumptions

- User is logged in.
- There are valid drinks in the database.
- There is a cup in the DRINC machine.

2.4.3 Procedure

- System prompts user for drink selection
- User selects drink
- User clicks “Create”
- The system makes the drink and increments the number of drinks consumed
- User consumes drink

2.4.4 Expected Outcome

- The drink is created and consumed. User is positively affected by drink.

2.5 Delete a Drink

In this scenario the user deletes a drink from the database so that it will be removed from the list of valid drinks.

2.5.1 Actors

- User
- System

2.5.2 Assumptions

- User is logged in.
- There is already at least one drink in the database.

2.5.3 Procedure

- System displays a list of saved drinks
- User selects a drink
- User clicks “Delete Drink”
- The system pops up with a confirmation “Are you sure you want to delete [Drink x]?”

- The user clicks “Yes” or “No”
- If the user clicks “Yes” the system drops the drink from the database

2.5.4 Expected Outcome

- The drink is no longer listed under saved drinks.

3 | Hardware Design Overview

3.1 Description

The DRINC project's hardware consists of all physical devices used to transport, pour, and support drinks.

- The frame (made up of the following subcomponents):
  - The track
  - The valves
  - The support structure
- The PSU
- The Raspberry Pi server
- The Arduino microcontroller
- The Nexus 7 frontend

3.2 Support Structure

DRINC will be held in a frame constructed from high quality ABS plastic capable of supporting a minimum of 9 liters of liquid in bottles. It will also have space to house the drink transportation track, the backend server, the Arduino, and the PSU.

3.3 Track

The track is made up of a pair of toothed rails running the length of the frame attached to rotation controlled servos. The track will hold a cup of varying size, as well as detect whether or not a cup is in place.

3.4 Valves

Each mixer be equipped with a custom built valve consisting of a shaped tube connected to a position controlled servo. The Arduino microcode will control each valve.

3.5 PSU

DRINC will be powered by a 300W ATX PSU. This will be enough to power all valves, the track, the backend server, and the microcontroller.

3.6 Arduino

An Arduino Mega256 microcontroller will be used to control all track and valve functions required by DRINC. The microcontroller will draw power from the DRINC PSU. The microcode running on the is structured as follows:

- The microcontroller turns on and initializes itself
- Wait for a serial data packet from the backend
- Upon receiving all serial data, check to make sure it is a valid drink recipe
- If drink is valid, move track to first valve in recipe
- Calculate time valve must be open based on recipe
- Open valve, then close it after calculated time
- Repeat process for remaining valves in recipe
- Move track to finished drink position
- Wait until cup is removed and replaced
- Send backend a ready signal
- Return to wait for data state

4 Software Design Overview

4.1 Description

The DRINC project uses various pieces of software to control the pouring mechanisms and front end features. the DRINC project will consist of the following pieces of software:

- The microprocessor code
- The server code to communicate with the drink flow
- The server code to present the API
- The client code to render the display for the user

The microprocessor code is documented in the hardware section.

4.2 Display Backend

The generalplan for the backend is to write modules in PHP. These modules will either be running in the background and communicating data to and from the database through means of an API.

PHP will allow the ease of use in communication between the PHP and PostgreSQL database by use of sanitized Structured Query Language (SQL) queries and ability to escape all output. This will eliminate the chances of SQL injection (SQLi) and Cross-site scripting (XSS) attacks on the system.

4.3 Display Frontend

The front end will be displayed using mainly HTML, with some PHP and Javascript for anything dynamic. The frontend will communicate with the backend using API.

4.4 Program Flow

This program will have three main “threads”. One thread would be responsible for listening to user input coming from the Android tablet and another would be responsible for communicating with the Arduino device controlling the dispensary of fluids. The last thread will be responsible for communicating with the database and receiving or adding data.

The user input thread will be responsible for receiving all input via Android tablet or the website. It will communicate with the API which will inturn communicate with either the database or fluid dispensary thread when needed. Therefore, all this thread needs to do is listen for user commands and send the processed commands to the correct thread.

The Arduino thread will be responsible for communicating to the Arduino device controlling the fluid dispensary. It will take input from the API, originating from the input thread, and control the fluid flow as such. When a user request is handled and the drink recipe is collected via the other threads, a listing of array positionings and the amount for each will be passed to the arduino using this thread.

The database thread will be responsible for communicating with the database. Its job will be to lookup things in the database along with inputting/changing/deleting things. The things consists of: types of base drinks, recipes for mixed drinks, and the current positioning of the currently loaded drinks. It will receive its information, along with forward responses, to/from the API.

4.5 Data Flow

The applications will use PostgreSQL to store all of the data. Both the website and the Android app will be able to access the information in the database via an API, which will interact with the database as needed for the applications.

4.6 Potential Problems

One potential problem is that their is a limited time constraint. Since we will be working with different technologies and languages that we have not currently worked with, a bulk of our time will be used by learning such.

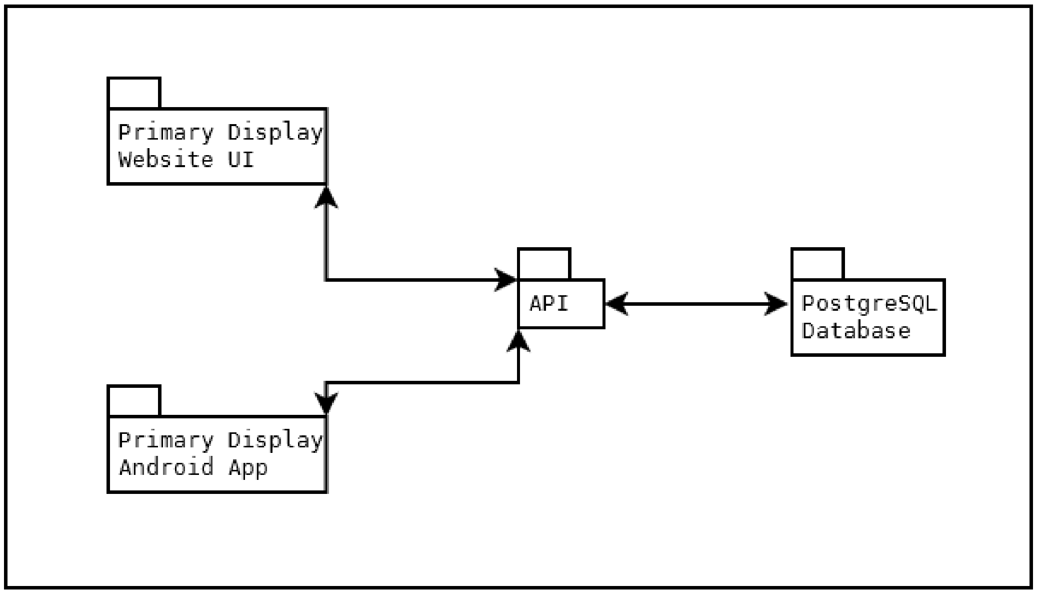

Figure 4.1: Diagram of Data Flow

4.7 The Backend

4.7.1 Authentication

The authentication module will be used every time a user attempts to log in. This module will use the user commands thread to communicate with the API. It will send the login information given by the user to the API, in which the API will check to see if the user information is valid. If it is, the API will tell the module to log the user in. Otherwise, it will tell the module to deny the login attempt and print an error.

4.7.2 API

The API will deal with all of the backend communication between the website/Android app and the server. This includes authentication, deleting a drink from the user's list, adding a drink to the user's list and requesting access to the drinks in the userâĂŹs list when the user wishes to create a drink.

5 Server Configuration

5.1 Description

The server will be responsible for hosting the web interface, as well as holding the PostgreSQL database. The server will be attached to the device.

5.2 Hardware

5.2.1 Mainboard

Our server will be a Raspberry Pi, Model B. It was chosen due to the small size, availability, price, and versatility. The Raspberry Pi specs include:

- 700MHz ARM CPU
- 250MHz Broadcom VideoCore IV GPU
- 512 MB SDRAM
- 3.5âĂİ by 2âĂİ
- Various connection interfaces such as
  - Composite RCA
  - HDMI
  - 3.5 mm audio jack
  - 10/100 Ethernet LAN
  - GPIO
  - Two USB 2.0

- SD Reader

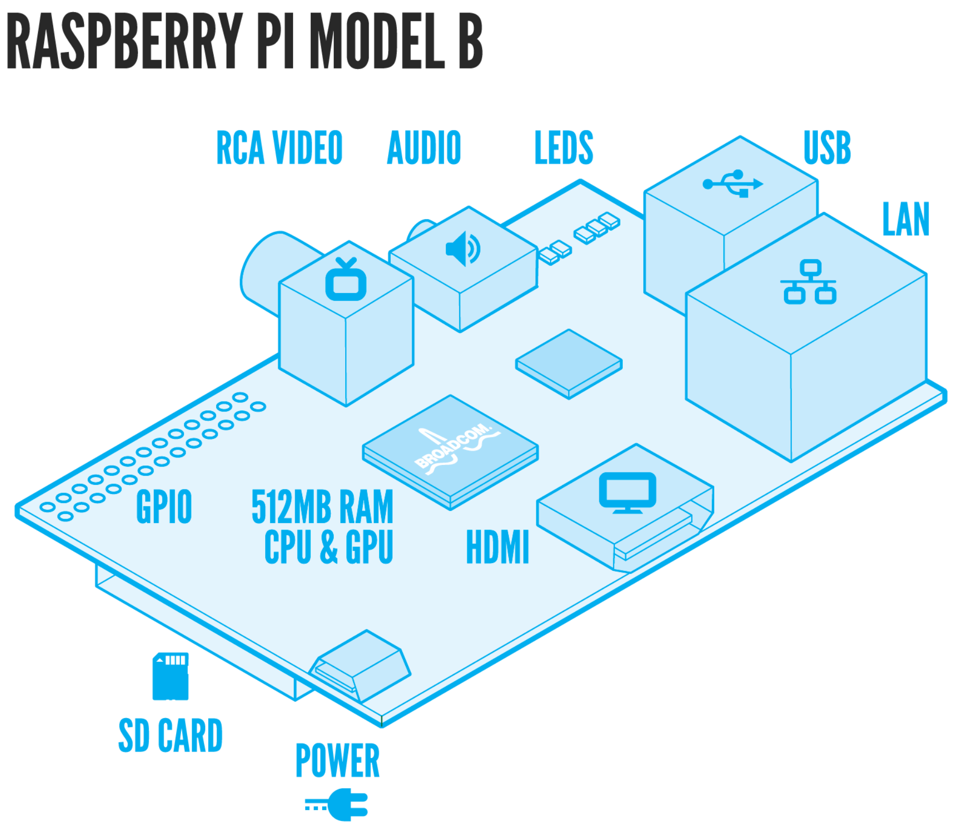

Figure 5.1: Diagram of Raspberry PI

5.2.2 Storage

Onboard storage will be comprised of a 8GB Element 14 branded SD card. This is the Raspberry PiâĂŹs primary boot device, and will contain the Operating System and all necessary software. Additional storage can be obtained through USB storage devices if necessary.

5.2.3 Power Supply

The Raspberry Pi will be powered via a 5 volt Element 14 branded MicroUSB adapter.

5.3 Software

5.3.1 Raspian

Raspian is a Raspberry Pi optimized version of the Debian Linux operating system. We have chosen Debian for its stability, and its ability at handling the services we will be utilizing. Additional packages will be handled using apt-get.

5.3.2 Apache2

We will use the Apache2 server software to handle the web interface. We have chosen this due to its ease of use and stability. Apache2 will be integrating into PostgreSQL while handling web information.

5.3.3 PostgreSQL

We will be using PostgreSQL to handle database information, such as user credentials and drink customizations. Any sensitive information will be hashed and salted before being placed into the database.

6 Non-Function Requirements

6.1 Server Requirements

- Enough disc space to hold an 3 years worth of system updates, and 500 users data.
- Ability to maintain internet connection and reconnect automatically if network is lost.
- Small and unobtrusive case.
- User will be unable to gain root access.