University of Waterloo
Faculty of Engineering
Department of Electrical and Computer Engineering

Brew It Yourself

An Automated Single Vessel Home Brewery System

Group Number: 2016.019
Consultant: Douglas Harder

Kevin Nause (20413332)
Mathieu Tremblay (20420813)
Scott Wood (20379649)
Steve Jung (20411563)

Date: July 2, 2015

Contents

1 High-Level Description of Project

1.1 Motivation

The art of home brewing has been steadily gaining popularity over the past 35 years alongside the rise of craft breweries in North America, so much so that in 2010 there were over 2000 craft breweries in the United States, after starting with only 8 in 1980 [1]. The traditional method for homebrewing requires various components, constant monitoring and heavy maintenance. There should be a solution which reduces complexity, making it much more affordable and practical for home use. We hope to create a single vessel system that would make the home brewing process precise, automated and compact, all at a reasonable price.

1.2 Project Objective

The objective of this project is to combine homebrewing experience with engineering design, and construct a single vessel brewing system. By maintaining a strict control of key parameters, the brewing process is regulated using a combination of fluid mechanics, heat transfer, digital controls, power systems, embedded robotics and mobile development. The Robotics Operating System (ROS), allows for a design where sensors can be added to a modular setup and provide feedback. By receiving feedback from temperature readings, density measurements and pH monitoring, the brewing process can be accurately recorded, shared, and automated by the system.

1.3 Block Diagram

Figure 1 shows relevant links between the physical, mechanical design of the brewing system, as well as the computer systems and their underlying software.

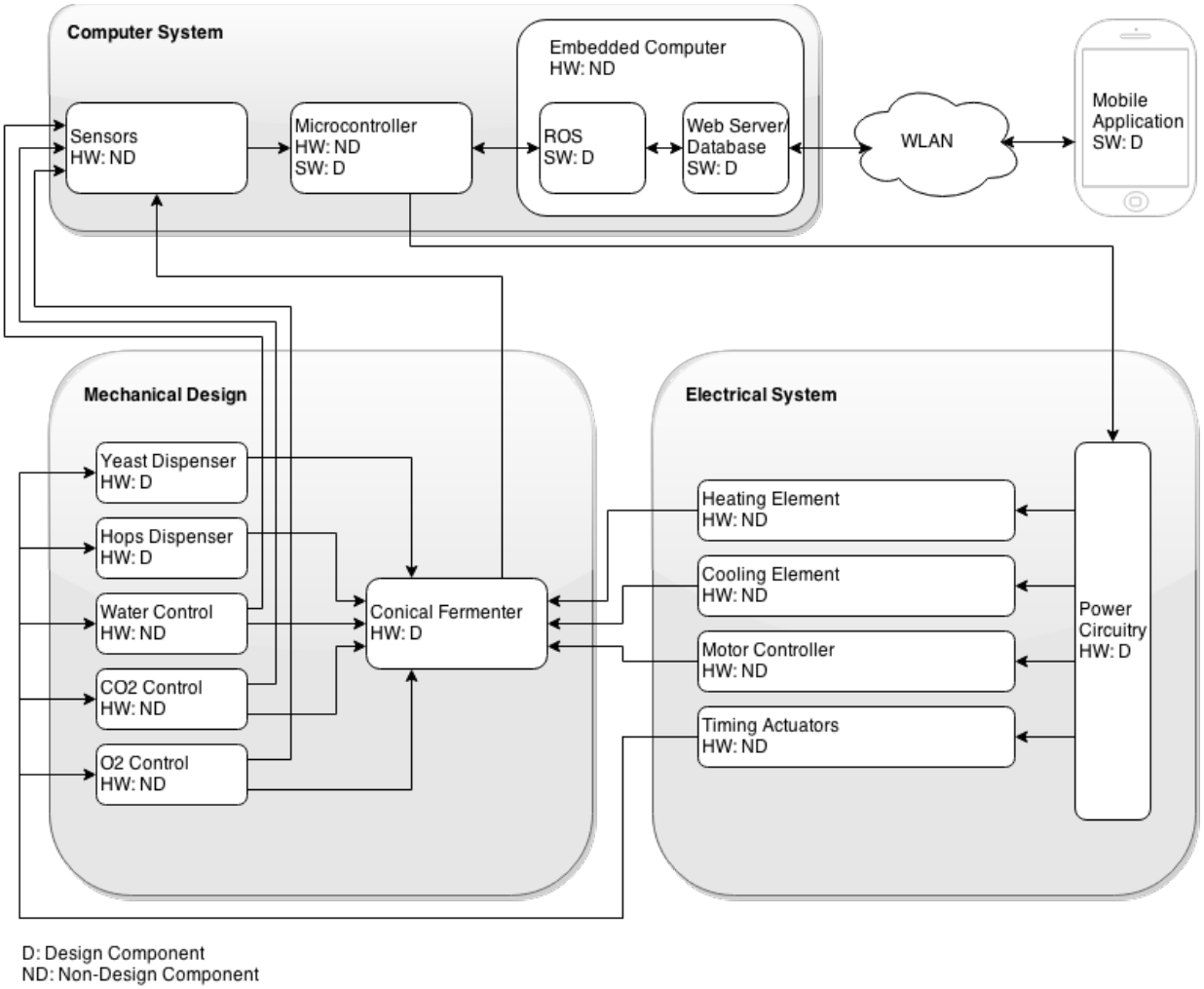

Figure 1: Block Diagram outlining the interactions between the computer, electrical, and mechanical systems

1.3.1 Description of System

The single vessel brewing system, as shown in Figure 1, contains various sensors which forward the environmental parameters to the microcontroller. The data from the various sensors accurately represents the state of the fermenter and feed into digital control loops running on the microcontroller. Data from sensor modules is obtained by a microcontroller via an I2C interface. This data is then sent to an embedded computer system over USB where it is logged in a database and used to control relevant subsystems. Control commands are sent from the embedded computer to the microcontroller, where the appropriate component can be communicated with in order to regulate the brewing environment. The embedded computer system is then able to send diagnostic information and push notifications to a mobile device through a WLAN connection.

1.3.2 Designing and Not Designing Components

The following is an outline of the components in the block diagram and the requirements associated for each Designing (D) and Non-Designing (ND) component.

- **Sensors**
  ND: Sensors being used in the vessel are not being designed. Instead, off-the-shelf sensors are being purchased and integrated into the system. Relevant sensor types for this system include, but arent limited to, pH sensors, volume sensors, flow meters, temperature probes, etc.

- **Microcontroller(s)** ND: The hardware of the microcontroller is not being designed since thats outside the scope of this project. The system instead uses an off-the-shelf microcontroller. D: The software running on the microcontroller is being designed. This software mainly consists of digital control loops for interfacing with the sensors and power electronic circuitry.

- **Conical Fermentation Vessel**
  D: The mechanical features and dimensions of the conical fermentation vessel are being designed to hold various sensors and other electrical and mechanical components.

- **Power Electronic Circuitry**
  D: The power electronic circuitry is being designed to power the pumps, motors, actuators, and heating/cooling mechanisms. The circuitry takes input from the digital controller running on the microcontroller and outputs the appropriate power to the end devices.

- **Heating and Cooling Elements**
  ND: The heating coils and refrigeration unit used to regulate temperature of the vessel are not being designed. Instead, off-the-shelf or salvaged and adapted components from existing appliances (e.g. hot water tank coils, home air conditioning heat pump) are being used and the power electronic circuitry is being designed to power these devices.

- **Motor Controller**
  ND: The motors, pumps, and mixing mechanisms used in our system are not being designed. Instead, off-the-shelf or salvaged and adapted components from existing appliances (e.g. blender mixing prop, aquarium pumps) are being used and the power electronic circuitry is being designed to power these devices.

- **Timing actuators**
  ND: The actuators used in our system are not being designed. Instead, off-the-shelf solenoids and servo motors are being used and the mechanical subsystems that they will actuate are being designed. The power electronic circuitry is also being deigned to be able to power the actuators.

- **Embedded Computer System**
  ND: The hardware and operating system of the embedded computer system is not a design objective as it is outside the scope of this project. Instead an off-the-shelf embedded computer is purchased and a Debian based Linux operating system is installed to satisfy the requirements for the development environment.
  D: The embedded computer system is configured with a Web Server, Database, and the Robotics

Operating System (ROS). Microcontrollers can be modularly added to the system via USB and automatically configured as ROS nodes through negotiations.

- **Mobile Application**
  D: The mobile application is designed such that the user can receive push notifications and see various sensor data during the brew process.

2 Project Specifications

This section outlines the functional and non-functional requirements of the project design.

2.1 Functional Specifications

Table 1 describes each functional requirement and highlights whether it is essential or not to the completion of the project.

Specification	Classification	Description
Completion of the Brewing Process	Essential	The device automatically completes the brewing process, consisting of these steps: • Mash • Sparge • Boil Wort • Dispense Hops • Aerate the Wort • Pitch the Yeast • Kegging and Dispensing in its entirety and in the correct order. For more information on the brewing process and the terms used, please see the Glossary.
Heating Unit	Essential	Mashing the grains, sparging the grist, and boiling the wort all require high water temperatures. The system is able to accurately heat the contents to a minimum of 110°C within 1°C of error.

Cooling Unit	Essential	Yeast pitching and fermentation happen immediately after boiling the wort, but require temperatures a specific temperature (typically 20°C). The system is able to rapidly cool the wort to a temperature within 1°C of the target temperature so that the yeast may be pitched safely.
Temperature Regulation	Essential	The system is able to control temperature for each step in the brewing process. An error of 1°C is the target specification.
Trub Removal	Non-Essential	Remove the undesired sediment and other byproducts at the bottom of the fermenter so that ageing may take place within the vessel. The user does not have to maintain the trub accumulated by the brewing process.
Application Notifications	Non-Essential	The system provides notifications to the user updating them on the current state or action being taken. Additionally errors or warnings can be sent, in cases of emergency such as clogs, low oxygen or carbon dioxide, or an improper environment.
Database	Essential	The system is able query the specific steps it needs to automate for a specific brew. Additionally, it can store various data involved in brewing including temperature, pH levels and density measurements to generate logs and reports.
Reproducibility	Non-Essential	The system is able to record recipes, ingredients and steps for various brews. The user can recreate the same type of brew if they select one of the recipes.

Table 1: An overview of each functional specification of the project

2.2 Non-Functional Specifications

Table 2 describes each non-functional requirement and highlights whether it is essential or not to the completion of the project.

Specification	**Classification**	**Description**
Temperature Accuracy	Essential	The system accurately regulates the temperature required for the brewing process within 1°C of the targeted temperature.
Volume Control	Essential	The end result must be greater than or equal to the target yield volume. Target yields typically consist of 15.5 gallons, 7.75 gallons, and 5.16 gallons. These are the standard sizes of a half barrel keg, quarter barrel keg, and a sixth barrel keg respectively.
Size	Essential	The dimensions are limited such that the vessel can fit within a standard 36 inch residential door frame.
Mobility	Essential	The system is able to be relocated by a single person using the aid of caster wheels and or a standard 18 inch wide utility dolly.
Sanitation	Essential	The system employs SUS 304 stainless steel to maintain food-grade sanitation conditions. The vessel is self cleaning since a sanitary environment is crucial for the brewing process.

Table 2: An overview of each non-functional specification of the project

3 Risk Assessment

Table 3 describes each risk associated with the project and highlights methods and actions to take to mitigate any adverse effects.

Risk	**Impact**	**Probability**	**Mitigation**
Funding this project proves costly and infeasible due to the requirement of expensive components	High	High	Sources of additional funding and sponsorship are being investigated at an early stage of this project to mitigate this risk.
The on-campus machine shops have insufficient manufacturing capability and precision to create the conical vessel	High	Medium	Communications with the on-campus machine shop is happening at an early stage of the project, so that if necessary there is time to investigate alternative manufacturers.
Latencies in the supply chain caused by shipping overseas cause crucial parts to arrive too close to the prototype deadline	High	Low	Local suppliers for necessary parts and components are being considered first before any overseas suppliers.
Poor quality or malfunctioning sensors cause unreliable control of the brewing process	Medium	High	The use of higher quality sensors or multiple redundant sensors can lower this risk.

Table 3: An overview of each risk associated with the project

4 Detailed Design

4.1 Mechanical Design

4.1.1 Material

4.1.2 Tank Structure

4.2 Electrical System

4.2.1 Heating and Cooling Controller

4.2.2 Motor Controller

4.2.3 Timing Actuators

4.3 Computer System

The computer systems consist of three main components, the embedded central computer, the microcontroller and sensor pairs, as well as a Web Server and Database pair to allow for storage and communications. The detailed description of each subcomponent and the methods of interaction between them is discussed in this section.

4.3.1 Embedded Computer

The primary function of the embedded computer system is to aggregate data in close to real-time from the multiple sensors used in the design. Analysis of this data is crucial for determining the current progress in the mash, sparge, and boiling of the wort. For specific analysis of sensor data please see subsection 4.3.2.

For ease of repeatability, availability, and cost the Raspberry Pi was selected. At the time of design and creation of this report, this is the most recent hardware revision of the Raspberry Pi. The relevant hardware specifications for the Raspberry Pi is a quad-core Acorn Reduced Instruction Set Computing (RISC) Machine (ARM) processor, 1GB of Random Access Memory (RAM), and a Bluetooth dongle for local wireless communications (rated to 50m). Since the outer shell of the fermenter is aluminum, the casing for the Raspberry Pi can act as a heat sink, allowing it to be safely overclocked to 1GHz, 500MHz, and 500MHz for the ARM processor frequency, sdram frequency, and L2 cache (core) frequency respectively. The configuration file below allows for a substantial performance boost with negligible thermal impact.

```
arm_freq=1000
sdram_freq=500
core_freq=500
over_voltage=2
arm_freq_min=400
sdram_freq_min=250
core_freq_min=250
```

For a list of stock specifications and configurations for the Raspberry Pi, please see Appendix **??**. The operating system chosen was the ARM build of Ubuntu 14.04 Long Term Support (LTS), codename Trusty Tahr, as well as Robot Operating System (ROS) Jade Turtle for sensor coordination and communication. Ubuntu 14.04 was chosen over the default Raspbian operating system for increased security due to the use

of SELinux policies. Since the Raspberry Pi will be acting as a LAMP server as well for mobile interaction and remote notifications, enforcing read only access over the configured port of choice is a necessity. For an in depth overview of the mobile application please see subsection **??**. Additionally, the LTS version of Ubuntu was chosen for a maximum of five years support, as well as increased compatibility and reliability for repository packages specific to the trusty distribution.

4.3.2 Sensors and Micro-Controllers

Every sensor in the system is paired with a micro-controller to enable a modular design and a plug and play like support. By adding more sensors to the system, more information can be provided as feedback to enable better logging and regulation of the brewing environment. The only mandatory sensors for the system are a main temperature sensor for the wort and flow meters on the liquid inputs and outputs of the system. By introducing additional sensors, such as pH and density sensors, certain yield thresholds of interest in the sparge and fermentation procedure can be met rather than approximated. Additionally, this will allow for a tier based configuration for the automated vessel to reduce cost and complexity if desired. As the end goal for this project is to have a community influenced input, having a customizable sensor configuration with open hardware and software pairs is encouraged. For maximum affordability the Arduino compatible Teensy-LC microcontroller is used to poll the sensors and report data, however any Arduino compatible microcontroller may be used. Technical specifications for the Teensy-LC microcontroller can be seen in Appendix **??** The bridge between the ROS instance on the Raspberry Pi and the Arduino compatible microcontroller is achieved through the use of the ROSserial package. Example code for obtaining data from a TMP102 temperature sensor over Inter-Integrated Circuit (I2C) at address 0x91 [2] can be seen below [3].

```
#include <Wire.h>
#include <ros.h>
#include <std_msgs/Float32.h>

//Set up the ros node and publisher
std_msgs::Float32 temp_msg;
ros::Publisher pub_temp("temperature", &temp_msg);
ros::NodeHandle nh;

int sensorAddress = 0x91 >> 1; // From datasheet
long publisher_timer;

void setup()
{
  Wire.begin();      // join i2c bus (address optional for master)
  nh.initNode();
  nh.advertise(pub_temp);
}

void loop()
{
  if (millis() > publisher_timer) {
  // step 1: request reading from sensor
```

```
    Wire.requestFrom(sensorAddress,2);
    delay(10);
    if (2 <= Wire.available()) // if two bytes were received
    {
      byte msb;
      byte lsb;
      int temperature;

      msb = Wire.read(); // receive high byte (full degrees)
      lsb = Wire.read(); // receive low byte (fraction degrees)
      temperature = ((msb) << 4); // MSB
      temperature |= (lsb >> 4); // LSB

      temp_msg.data = temperature*0.0625;
      pub_temp.publish(&temp_msg);
    }
  publisher_timer = millis() + 1000; //publish once a second
  }
  nh.spinOnce();
}
```

Additionally, controllers can be linked to the system via Arduino compatible micro-controllers and receive inputs based on sensor data gathered via ROS publications. To improve the resolution or accuracy of data and controls, the polling interval of each of the micro-controllers can be calibrated appropriately. Since most sensors will remain idle for a majority of the brewing process, their sampling rates can be scaled using a simple algorithm as suggested in the code sample below.

```
#include <std_msgs/Float32.h>

int sample_rate = MIN_SAMPLE_RATE;
int sample_count = 0;

void scale_rate(std_msgs::Float32 prev, std_msgs::Float32 curr)
{
    if(prev == curr)
    {
        sample_count++;
    }
    else
   {
        sample_count--;
   }

   if(sample_count == SCALE_THRESHOLD)
   {
         sample_rate = increase_rate();
      sample_count = SCALE_THRESHOLD >> 1;
```

```
    }
    else if(sample_count == 0)
    {
        sample_rate = decrease_rate();
        sample_count = SCALE_THRESHOLD >> 1;
    }
}
```

The two fundamental controllers that require communications for the operation of the system are the water controller and the heating and cooling controllers. Figure 2 demonstrates a state diagram for the water controller and Figure 3 demonstrate the state diagrams for toggling the heating and cooling states of the temperature controller. The variables volume, temp, target and offset represent the current volume, current temperature, desired resultant, and amount of allowed deviation from target respectively.

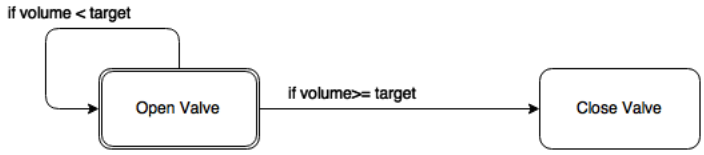

Figure 2: State diagram for the water controller component

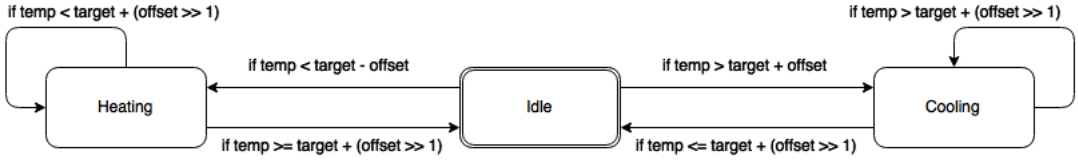

Figure 3: State diagram for the temperature controller component

4.3.3 Web Server and Database

A Lamp Apache MySQL PHP (LAMP) server is the current solution for delivering information from the embedded computer to the mobile application. All communications and parameters are to be communicated using a JavaScript Object Notation (JSON) format. The database will consist of relational connections between recipes, instructions, ingredients, sensors, and logged data as demonstrated in Figure 4.

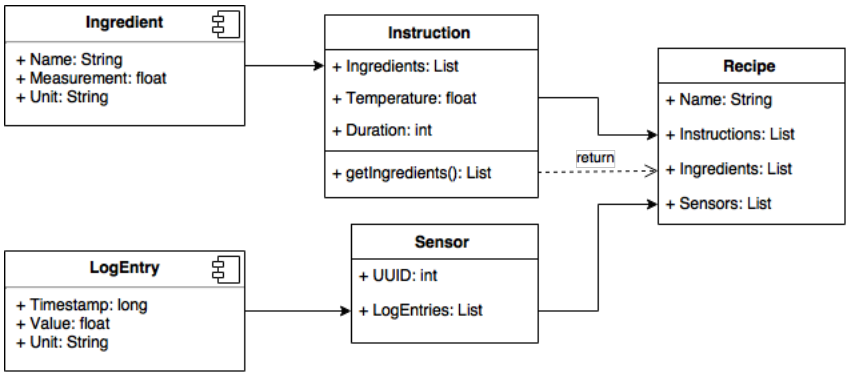

Figure 4: A UML diagram showing the relation between tables and their entries in the logging database

4.4 Mobile Application

4.4.1 Real Time Data Analysis

4.4.2 Input Controls and Interaction

5 Discussion and Project Timeline

5.1 Evaluation of Final Design

5.2 Use of Advanced Knowledge

5.3 Creativity, Novelty, Elegance

5.4 Student Hours

5.5 Potential Safety Hazards

5.6 Project Timeline

Acronyms

ARM Acorn RISC Machine.

I2C Inter-Integrated Circuit.

JSON JavaScript Object Notation.

LAMP Lamp Apache MySQL PHP.

LTS Long Term Support.

RAM Random Access Memory.

RISC Reduced Instruction Set Computing.

ROS Robot Operating System.

Beer Terminology

Grist The combination of milled grains to be used in a particular brew. Also sometimes applied to hops [**?**]..

Mash (Verb) To release malt sugars by soaking the grains in water. (Noun) The resultant mixture [**?**]..

Pitch To add yeast..

Sparge To spray grist with hot water in order to remove soluble sugars (maltose). This takes place at the end of the mash [**?**]..

Trub The layer of sediment that appears at the bottom of the fermentation vessel upon the completion of fermentation..

Wort The solution of grain sugars strained from the mash tun [**?**]..

Technical Terminology

Raspberry Pi 2 Model B The Raspberry Pi 2 Model B is the second generation Raspberry Pi. It replaced the original Raspberry Pi 1 Model B+ in February 2015..

References

[1] A. Tepedelen, "History of homebrewing," 2013. `http://allaboutbeer.com/article/power-to-the-people/`. [Accessed 2015-05-19].

[2] T. Instruments, "Tmp102," 2015. `http://www.ti.com/product/tmp102`. [Accessed 2015-07-01].

[3] ROS, "Measuring temperature," 2014. `http://wiki.ros.org/rosserial_arduino/Tutorials/Measuring%20Temperature`. [Accessed 2015-07-01].