

# CS413 Embedded Systems ACE Group One

## Phase One Report

Andi Anderson      Fraser George      Adam McGhie  
Aidan O'Grady      Kristine Semjonova

We confirm and declare this report and the assignment work is entirely the product of our own efforts and we have not used or presented the work of others herein.

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Assessment of Capabilities</b>	<b>4</b>
2.1	Arduino . . . . .	4
2.1.1	Specification . . . . .	4
2.2	Raspberry Pi . . . . .	5
2.2.1	Specification . . . . .	5
<b>3</b>	<b>Development Environment</b>	<b>6</b>
<b>4</b>	<b>Additional Resources</b>	<b>6</b>
4.1	Coffee Pot . . . . .	6
4.2	Barduino . . . . .	6
<b>5</b>	<b>Design</b>	<b>7</b>
5.1	Overall . . . . .	7
5.2	Hardware . . . . .	7
5.3	Software . . . . .	7
5.3.1	Website . . . . .	7
5.3.2	Java Spring Service . . . . .	7

# 1 Introduction

The initial scope of the assignment is to design and implement an embedded systems ‘gadget’ with the use of an Arduino and/or a Raspberry Pi.

After much deliberation and debate, we have decided to implement the Hyper Text Coffee Pot Control Protocol (HTCPCP)[1], with a system that allows the user to send requests for coffee over internet for either immediate or future consumption. A server will be implemented to handle the requests to be sent to our Arduino controlled coffee pot, which will then create coffee on demand. In addition, we hope to include the functionality of adapting the Barduino, a project from a previous year, to supply milk or other substances to a user’s drink as well.

## 2 Assessment of Capabilities

### 2.1 Arduino

The Arduino is an open source microcontroller primarily aimed at electronics prototyping. It can read and interact with various components such as LEDs, motors, etc.

As it is a microcontroller it is designed to interact with analogue inputs of the outside world, which the Raspberry Pi does not have the ability to do as it can only deal with digital I/O (via GPIO). This allows easy collection of data through various sensors.

The Arduino Uno is the microcontroller we have decided to use for this project, as it meets the requirements for our project, listed below:

- Has sufficient size of onboard flash to hold the main program.
- Can be easily extended through the use of shields for additional hardware interface requirements.
- Uses 5V supply, meaning it does not require any power transformers to be compatible with other components (i.e. Raspberry Pi).

The primary disadvantages of the Arduino when compared to the Raspberry Pi is it's lack of network capabilities, and the Arduino's lack of processor power. The Pi allows us to run the necessary independent web server, and make it accessible to an external client. Due to this the Pi shall be do any computationally intensive tasks and use the Arduino to interact with the other hardware.

#### 2.1.1 Specification

<b>Microcontroller</b>	ATmega 318
<b>Operational Voltage</b>	5V
<b>Input Voltage</b>	7-12V
<b>Input Voltage (Limits)</b>	6-20V
<b>Digital I/O</b>	14 (6 PWM)
<b>Analogue Input</b>	6
<b>DC Current per I/O Pin</b>	40mA
<b>DC Current per 3V Pin</b>	50mA
<b>Flash Memory</b>	32KB (0.5KB reserved for bootloader)
<b>SRAM</b>	2KB
<b>EEPROM</b>	1KB
<b>Clockspeed</b>	16 MHz

## 2.2 Raspberry Pi

The Raspberry Pi is a single-board computer that is small in size (85.60 mm x 53.98 mm) and was originally developed as an aid to teach the basics of computer science in schools. It commonly uses the Linux operating system, and has a Broadcom system on a chip, which includes an ARM1176JZFS, with floating point, running at 700 MHz, and a Videocore 4 GPU. It is also capable of 24 GFLOPS of general-purpose compute and features several texture filters and DMA infrastructure.

For the purposes of our project, we are using a Raspberry Pi 2 Model B to host a server and the Java Spring application that will be used to handle user requests for coffee.

### 2.2.1 Specification

<b>System on Chip</b>	Broadcom BCM2836
<b>CPU</b>	900 MHz quad-core ARM Cortex-A7
<b>Memory</b>	1 GB
<b>Ethernet (Limits)</b>	10/100 Mbits/s
<b>USB 2.0</b>	4 Ports
<b>On Board Storage</b>	MicroSD slot

### 3 Development Environment

The development environment that we shall use for the Arduino is the Arduino Integrated Development Environment (or Arduino Software (IDE)). For development on the Pi we will be using Raspbian with a Java Spring application acting as our web service for the system.

We all also be using a Git version control system in addition for easier transferring of work between group members. For the purposes of sharing other documents and files (for example, this report), a shared Google Drive folder was created to allow us all to add to each other's notes with ease before being formatted and organized properly.

### 4 Additional Resources

#### 4.1 Coffee Pot

Since the HTCPCP is all about sending requests for coffee, a coffee pot is naturally required for the whole thing to work. We have decided on the Andrew James 1100 Watt DIgital Filter Coffee Maker.

**Price:** £27.99

**Purchased From:** Amazon UK

#### 4.2 Barduino

As part of our project, we have found the potential of adapting the Barduino, a previous year's Embedded Systems project into our own, for the use of supplying the coffee with milk and other condiments if desired. Since the project owners have not claimed it, it has been deemed available for appropriation.

**Price:** £0.00

**Purchased From:** N/A

## 5 Design

### 5.1 Overall

### 5.2 Hardware

### 5.3 Software

There are three primary components of the software side of this project. Those are:

- A website through which users can order coffee.
- A Java Spring application to handle the above requests and pass to Arduino.
- The Arduino software triggering the coffee brewing and serving.

The typical user will only be exposed to the website aspect of this stack, which will be able to provide all necessary information through the website, such as when there is no coffee available.

#### 5.3.1 Website

The website is relatively simple. In order to obtain coffee, all they need to do is complete the form that is provided to them. Given the current set up, the URL is likely to be off the form IP:8080/brewduino, where IP is the address of the Pi. The system will ideally provide the user information such as the amount of coffee available right now, and how many requests are currently in a queue to be filled.

#### 5.3.2 Java Spring Service

The Raspberry Pi will be hosting a Java application utilising the Spring framework to allow for us to take the form submission and act upon it in Java. This has the advantages of allowing communication between Java application and the user, as well as between the application and Arduino.

The Java application is made up of four core classes, this system is a spin-off of Spring's guide on Handling Form Submissions[4] (due to the structure of this, a class diagram was deemed unsuitable for provision):

- Application
  - Main method runs the system here, through the Spring framework, making it an extremely simple class.

- Brewduino
  - This class acts a data structure representing the form the user fills out on the website. Each field in the class represents a part of the website's form.
- BrewduinoController
  - This class handles the requests sent from the user to the Pi, and acts upon them. It will validate that the input is OK, and then communicate with the Arduino to fulfill the request.
- Arduino
  - This class handles all communication with the Arduino, acting as a middleman between the Arduino and the BrewduinoController.
  - The Arduino class will make use of a suitable framework, such as JArduino[3] or ZXTX[2] to handle communication with Arduino.



## References

- [1] L. Masinter. Rfc 2324 - hyper text coffee pot control protocol (htcpcp/1.0). <https://tools.ietf.org/html/rfc2324>, 1998.
- [2] Rxtx. Rxtx. [http://rxtx.qbang.org/wiki/index.php/Main\\_Page](http://rxtx.qbang.org/wiki/index.php/Main_Page).
- [3] SINTEF-9012. Sintef-9012/jaruno. <https://github.com/SINTEF-9012/JArduino>.
- [4] Spring. Getting started - handling form submissions. <http://spring.io/guides/gs/handling-form-submission/>.