FINAL REPORT: SMART COFFEE

EECE 425: Spring 2017

MONDAY, MAY 1, 2017

Ebrahim Karam 201401059
Firas Haidar 201503236
Bilal Ghader 201503696

Table of Contents

Introduction 2
Hardware Design 3
Software Design 4
Afterword 6

List of Figures

Figure 1: Hardware Flowchart 3
Figure 2: Software Flowchart 4
Figures 3 & 4: User Interface ..... 5 & 6

Introduction

Today, human utilities are migrating towards what is called the Internet of Things, read: IoT.

Our aim was to integrate a simple coffee machine into this IoT by using the TIVA C series TM4CGXL microcontroller and a few of its sensors and booster-packs. Although making coffee the old-fashioned way is not a problem at all (some even find the routine of it relaxing), it is evident that even making coffee should move forward by at least creating a more efficient option for the people who would rather spend that time on something else. Our project thus allows users to brew their coffee over a wireless connection through an application on their phone.

Compared to earlier coffee machine models, the option for the user to make coffee without having to physically interact with the machine is a functionality we have not seen before. Although this is the only innovation worth mentioning featured in our finished product, we were also planning on programming the coffee machine to start brewing as soon as the user's alarm rings, as well as allowing the user to input his or her own presets through the phone app, and finally interface the coffee machine itself with the user's calendars so that it can read his or her tasks out loud while the user drinks his or her coffee.

Hardware Design

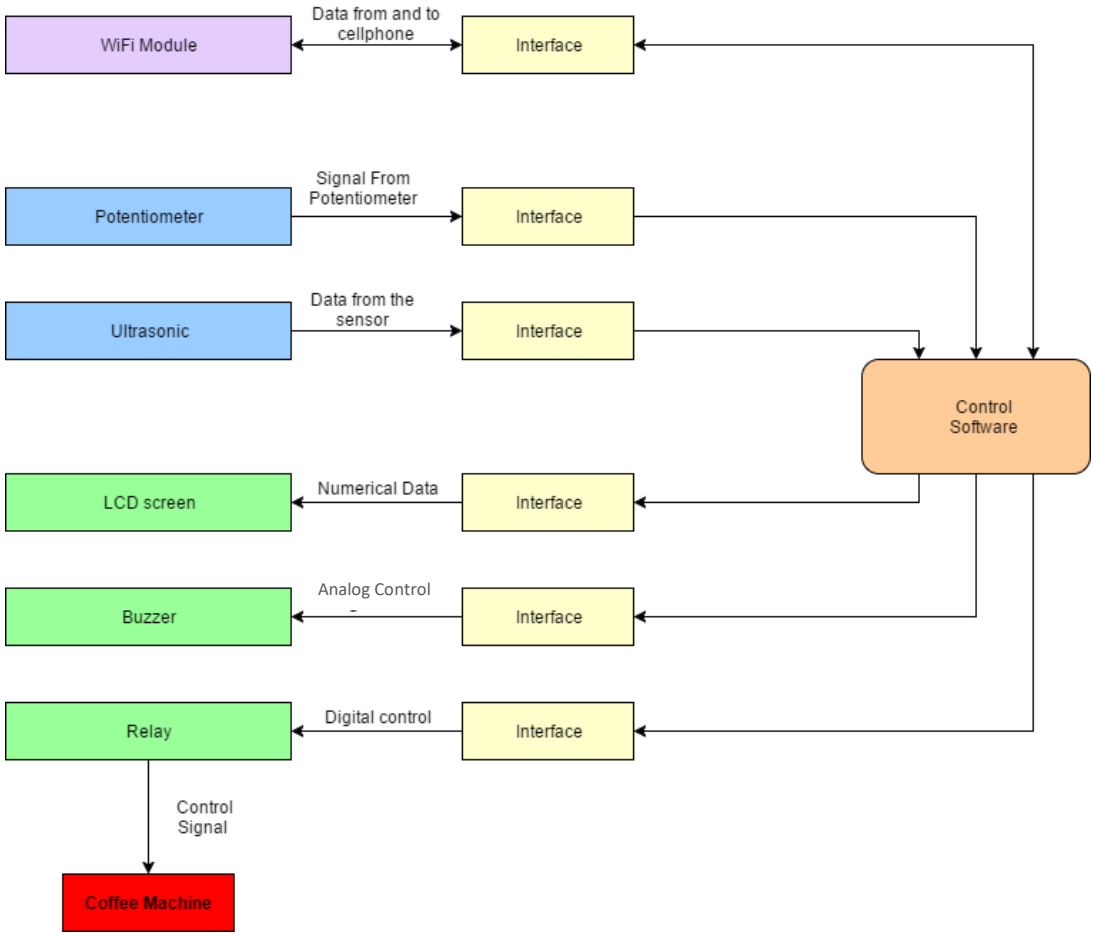

Figure 1 Hardware Flowchart

1. **Wi-Fi Module:** This component is used to establish a wireless connection between the phone and the TM4C. It was initially intended to turn the TM4C itself into a server but we ended up creating a separate server that acts as a middle point between the microcontroller and the end user. This block acts as both an input (to the controller) and an output (to the user), by receiving commands from the user and inputting them into the controller, and by collecting data from the sensors and displaying them on the user interface.
2. **Ultrasonic Ranger:** This sensor's functionality was to detect the level of the water inside the coffee machine. This allowed the machine to alarm the user when water was running low, as well as monitoring how many cups the user could still make.
3. **LED Screen (typo in the flowchart):** The screen displayed the number of cups the user could still make. Its output was dependent on the readings of the ultrasonic ranger.
4. **Potentiometer:** This acted as an analog input for the microcontroller, used to control the volume of the buzzer.
5. **Buzzer:** Indicated when the coffee brewing process is over and when the tank was

empty.

6. **Relay:** The relay turned the coffee machine on and off, depending on either a signal from the user or from the ultrasonic ranger.

The user mostly interacts with the hardware through its interfacing software, with the exception of the potentiometer which the user has to control manually.

Software Design

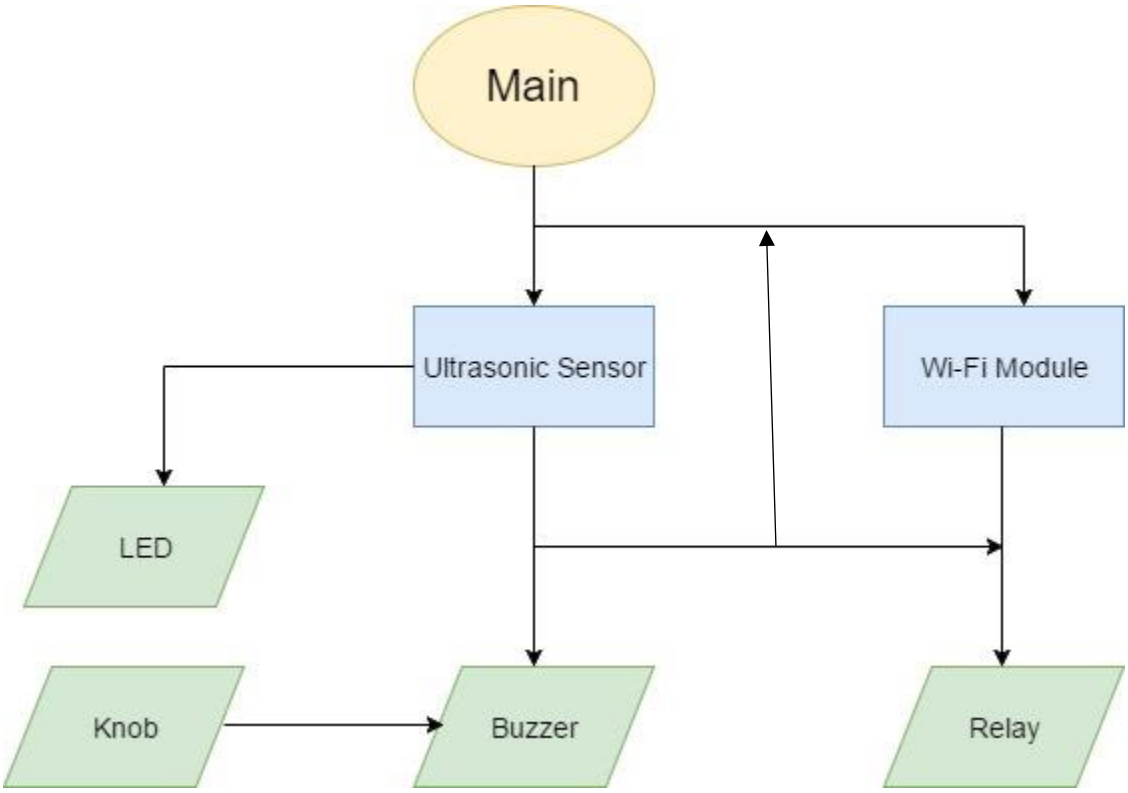

Figure 2 Software Flowchart

1. **Ultrasonic Sensor:** The libraries needed for this function were already available in Energia, but we needed to download a couple of complimentary one that were not already installed. The code for this block corresponds with what was described in hardware. What we had to do was interface it with the LED display, configure its pin setup, and interface it with the relay to control it. The Ultrasonic sensor also updated the WiFi Module by passing it the number of cups that are available.
2. **Wi-Fi Module:** The libraries needed for this function were all available in Energia. We had to configure this by inputting the name and address of the server we were using within the code. In addition we had to modify the HTML and CSS to give a more pleasant looking interface and give a few more functionalities. The first extra functionality is displaying the number of cups left which was received from the

Ultrasonic sensor. The second was allowing the user to pass a parameter “Cups wanted”. The third was the ability to turn on and off the Coffee machine.

3. **LED:** As mentioned in the ultrasonic sensor block.
4. **Knob & Buzzer:** The libraries needed for this function were all available in Energia. The buzzer used a Boolean dependent on the ultrasonic ranger and relay inputs. The knob controlled the voltage to control the volume.
5. **Relay:** The libraries needed for this function were all available in Energia. The relay used inputs from the Wi-Fi module (user through app) and the ultrasonic sensor to turn the machine on and off.

The user interacted with the software using the HTML page the app linked him or her to using the URL. The page had 3 different inputs for the user to use: an on button, and off button, and an integer control so that he or she can specify the number of cups he or she wants.

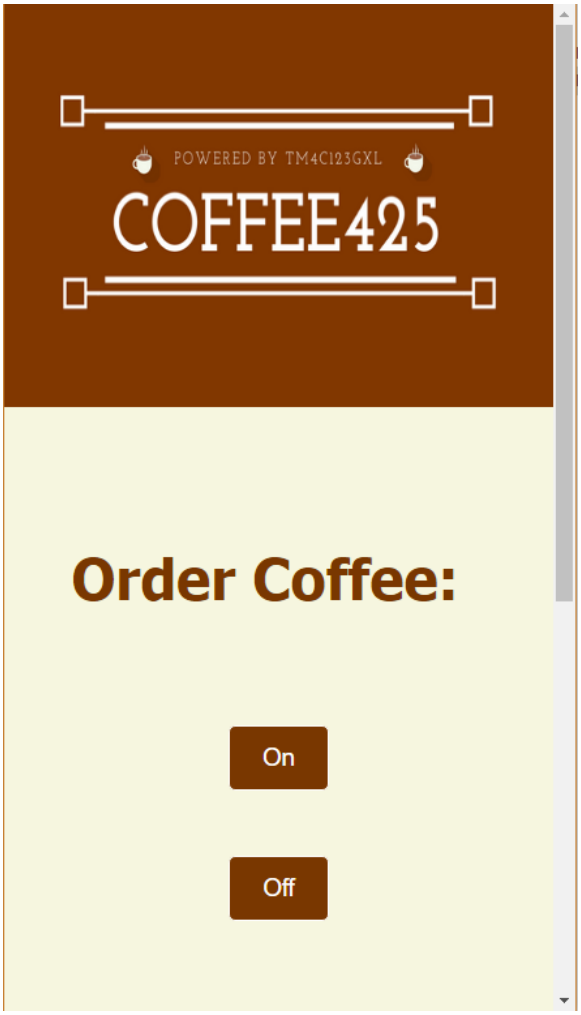

Figure 3 User Interface

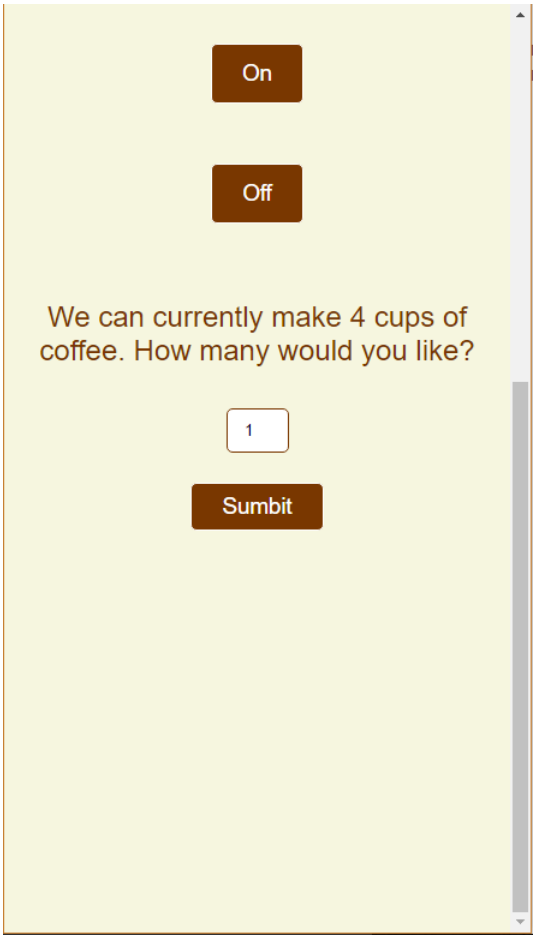

Figure 4 User Interface

Afterword

When we look at this project in retrospect, we come to realize that we may not have been especially methodical in our development. We had this big idea that we all liked very much and we wanted to see it done as soon as possible, so we went around doing some of the harder tasks before the easier ones were done. For example, we started working on the touch LCD screen, the stepper motor, the Android app, and the Google APIs since day one, which cost us a lot of time because all of these tasks were actually wants, not needs, and yet required more time and effort to be properly implemented. If we were to start over, perhaps we would have built what we had now before working on all of these cool add-ons, then build them sequentially, not in parallel, one by one until we either finish all of them or reach our deadline.

These components, in fact, were the main and only technical difficulties we faced. After spending so much time on the touch LCD screen, we found out it was not functional. We then spent more time working on the Joystick LCD screen only to find out that we could not use it and the Wi-Fi module together.

The stepper motor also caused us a ton of trouble because we originally started working on a servo motor code, but then dropped it when we found out we only had stepper motor available. The stepper motor code was tiring and time consuming because the Stepper libraries found for Arduino controllers, for example, were not available for the TM4C. We had the research and create new libraries until we were able to code something that works. We eventually dropped the functional stepper motor because we needed to alter the whole case of the coffee machine to add the extra tank and its corresponding gauge. We had no time to do that.

The Android app never finished because we found out that learning Android development from scratch was a harder task than expected. An original design was functional but we were unable to interface it with the Wi-Fi module, which rendered the whole thing obsolete. We ended up building an app that simply redirects the user using our page's URL.

That said, we still think these three ideas are important to make our project more efficient and user-friendly. Which is why these three steps would be our first towards expanding on our project: Implement them, then go beyond. Other modifications would include using a water level sensor instead of the ultrasonic, implementing an adaptive software so that the coffee machine can learn of the user's habits and start accommodating them, adding different presents (amount of water, amount of coffee, amount of sugar, amount of cream) allowing the user more freedom in brewing his or her coffee, etc.

The list goes on, and with each addition, an idea to go further pops up. Considering how much coffee-making techniques have developed over the years, we believe this project is prone to constant updates and modifications.

The human race's coffee consumption is timeless, and so is our project.