

# Private Text Message Controlled Coffee Maker for Remote Brewing

---

Mitchell Hoppenstedt

Prepared for

ECE 4220 – Real Time Embedded Systems, University of Missouri

April 1, 2015

**Abstract** – People are becoming busier and busier every year it seems like now. Rushing around in the morning trying to get ready for the day, eating breakfast to fuel them for the day, hurrying to watch the news to catch the weather, and trying to do all this as fast as possible to get to work or wherever they are going on time. Most people would like to lighten their morning load a little and make it easier on themselves. With 83 percent of adults drinking coffee in the U.S. [1] people don't want to waste time messing around with brewing a cup of coffee in the morning. They also don't want to wait in line or pay for expensive coffee in the morning either. The idea of the project is to create a coffee maker that is controlled by a user who will send a text message that will behave a signal that will then start brewing coffee that was put into the machine the night before. This way the user will just have to send a text message when the user wakes up and the coffee will start to brew, saving time and energy in the morning.

**Index Terms** – Remote systems, Internet of Things, embedded systems.

## Introduction

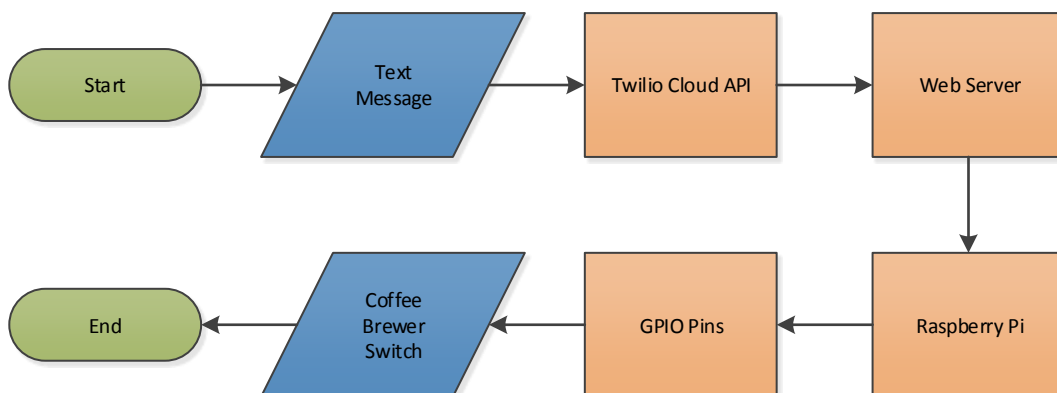
As people are in a hurry in the morning, it is important to save time as much as possible. For this reason it would be more convenient to be able to text and have a cup of coffee ready to go when you are ready to walk out the door or to have with breakfast. This would be much more efficient also so you can do other things such as take a shower or make breakfast or a magnitude of other things.

Many people sleep next to their phones. This is a benefit for a text based way to notify the coffee maker to make coffee. The user can wake up, turn over to their phone and immediately start to make a cup of coffee. This way the user can walk into the kitchen with the smell of fresh coffee and be ready to start off the day. This is the most effective way because no matter what time you wake up, you can just send a text to the coffee maker and have a fresh cup of coffee, not one that has been sitting on the burner for an hour or you didn't have time to make one because you were running behind.

The proposed solution will let the user be able to send a text message to a phone number which will send a signal to the coffee maker telling to begin brewing the coffee. The short term goals will involve the back end of the system to work while the long term goals will involve allowing a computer to control the coffee maker when a signal is sent.

## Proposed Implementation

Figure 1 shows the overall layout and flow of the system.



*Figure 1 - The overall flow of the system*

## Flow and Description of System

### Text Message

This is where the beginning of the system is. The user will send a text message to a phone number that will be designated to the user through the Twilio [2] web service. The user will send a text message to this number saying any message with the keyword “coffee” in it. The message will be sent to the Twilio servers for processing.

### Twilio Cloud API

After the text has been sent to the Twilio servers for processing, it produces an XML formatted file that holds a variety of information about the message such as what number sent the message, who it was to, the message content, time sent, etc... This information will be used to determine if the message that was sent was a valid number. Only certain numbers can send messages to the Twilio number. This is what makes the system private so no one else can send the Twilio number a message and make a cup of coffee. When the XML file is created, it is sent to a web server of the user’s choice.

### Web Server

When the Twilio Cloud API sends the XML file to the web server, the web server will gather it and send it to the Raspberry Pi. There is a way to simulate a web server on a Linux system using *ngrok*. This may be the best way to do it because I may not have access to a dedicated web server.

### Raspberry Pi

The Raspberry Pi will be the main source that takes in the XML file and will interpret it to see if the keyword “coffee” is in the message. If it is in the message, a signal will be sent to a GPIO pin on the Raspberry Pi telling the coffee to start brewing. This will only happen if the message contains the word “coffee” or else the system will not turn the coffee maker on. For interfacing with the GPIO pins, the C library WiringPi [3] will be used. This will make interfacing with the coffee maker switch much easier.

## GPIO Pins

By using the GPIO pins on the Raspberry Pi, it will be easy to be able to send the signal to the coffee maker switch telling it to brew coffee. I will be using GPIO pins for sending a logic high signal to the Coffee Brewer Switch, telling the coffee make to start brewing. WiringPi makes it easy to set pins to input and output and also writing values to them to tell them if they are high or low. Functions such as shown below [4] are easy to use and will allow the switch to turn on and off with ease.

```
void pinMode(int pin, int mode);           Function 1
```

```
void digitalWrite(int pin, int value);     Function 2
```

Function 1 will set the pin mode of the GPIO pin. “pin” is the pin number that wants to be written to or from. “mode” will be INPUT, OUTPUT, or PWM\_OUTPUT which will be OUTPUT in this system.

Function 2 will write to a value of high (any positive number) or low (0) to the pin designated. “pin” is the pin to write to and “value” is either high or low as specified before.

## Coffee Brewer Switch

This is a simple switch that is controlled by the Raspberry Pi. When no voltage is applied to this switch, it is logic low, meaning that the switch will be off. When a voltage is applied to the switch, it is logic high, meaning that the switch will be on, allowing the coffee to start brewing.

## Timeline

April 6<sup>th</sup> – April 10<sup>th</sup>: Make Twilio account and start using the Twilio Cloud API to understand how to use it.

April 13<sup>th</sup> – April 17<sup>th</sup>: Start interacting with the Twilio Cloud API and the Raspberry Pi and be able to receive messages to tell coffee maker to brew coffee.

April 20<sup>th</sup> – April 24<sup>th</sup>: Use the Raspberry Pi to control the switch of the coffee maker to start brewing coffee.

April 27<sup>th</sup> – May 1<sup>st</sup>: Combine all aspects of the system together so when the user sends the message, the coffee will start to brew.

## Class Related Objectives

This project will cover many class aspects that have been discussed in class. One aspect would be task communication. The web server will be listening and waiting for an incoming message from the Twilio Cloud API and have that communicate with the process that will write the signal to the GPIO pins to turn the coffee maker on. Another aspect would be network communication. To interact with text messages, Twilio will help tremendously. Communicating between the Twilio service and process that will drive the coffee switch will have to communicate over the internet. Lastly, the main focus of the system is to turn on a coffee maker, a peripheral device. This will simply just turn on a switch that will allow the coffee maker to brew coffee.

## Works Cited

- [1] K. Fernau, "USA Today," 9 April 2013. [Online]. Available:  
<http://www.usatoday.com/story/money/business/2013/04/09/coffee-mania/2069335/>.  
[Accessed 30 March 2015].
- [2] Twilio, "The Company," 30 March 2015. [Online]. Available:  
<https://www.twilio.com/company>. [Accessed 30 March 2015].
- [3] WiringPi, "WiringPi," 31 March 2015. [Online]. Available: <http://wiringpi.com/>. [Accessed 31 March 2015].
- [4] WiringPi, "Core Functions," 31 March 2015. [Online]. Available:  
<http://wiringpi.com/reference/core-functions/>. [Accessed 31 March 2015].