Who Killed The Coffee?

Submitted by SmartGropu:
Alexander Arrieta
Lauren Cramer
Katelyn Hylbom
Dan Tran
Mayank Verma

University of California, Irvine
Informatics 148
March 15, 2018

TABLE OF CONTENTS

1. Project Background

The project consisted of designing, developing, and implementing our own Internet of Things technology. For the project, we identified a problem and designed a device with the intention of eliminating or reducing that problem. In this case, the problem consisted of empty coffee pots in a communal setting and the solution, a reminder-based, internet-connected coffee pot.

1.1 The Project

A common issue, especially in office settings, is someone killing a communal pot of coffee and not brewing a new pot. In other words, an empty coffee pot when there should always be coffee readily available. The issues are addressed with the creation of a “smart” coffee maker. People can get very frustrated when they go to get a cup of coffee only to find that it’s empty. Most coffee pot users are not actively aware that their cup may be the last. Whoever killed the coffee pot wouldn’t want anyone to know they did it because it would frustrate and upset the other coffee drinkers. We determined that shaming and raising public awareness is the best way to deter someone from not brewing a new pot of coffee after finishing the previous pot. Our solution to this problem was to create a device that takes a picture of the coffee maker user and emails the photo to their coworkers. The interface would allow for custom mailing lists and messages that are sent out when someone kills the coffee. The coffee pot gives the user time to refill the coffee pot before taking their picture and sending it out to people on the mailing list. The goal of our “smart” coffee maker is to shame the user if they don’t brew a new pot of coffee by letting all of their coworkers know that they are the ones who killed the pot.

1.2 Our Device

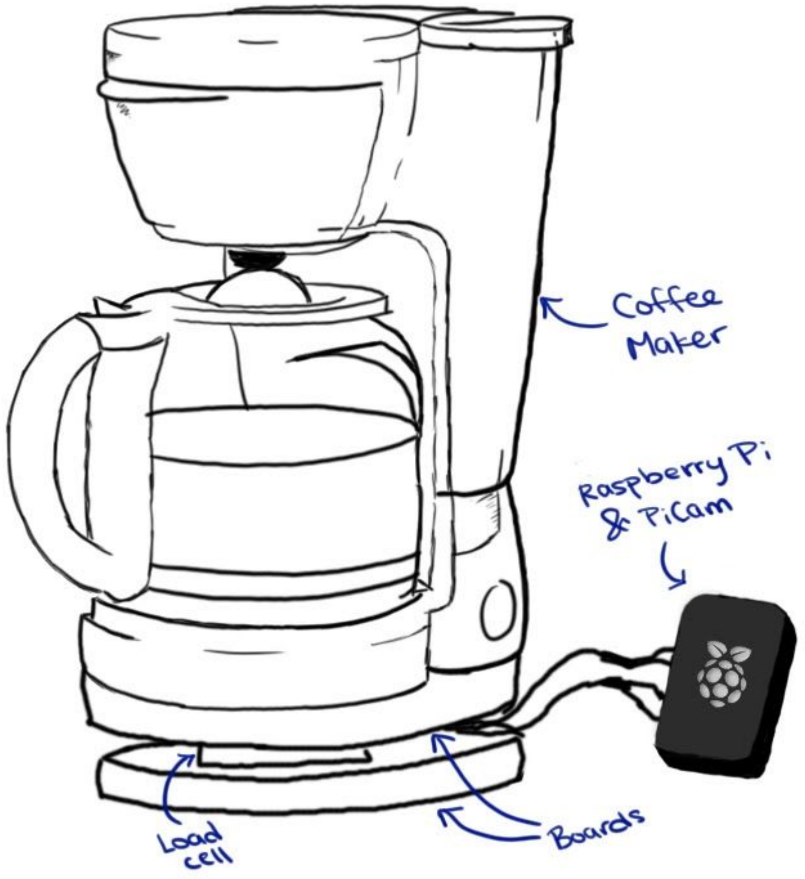
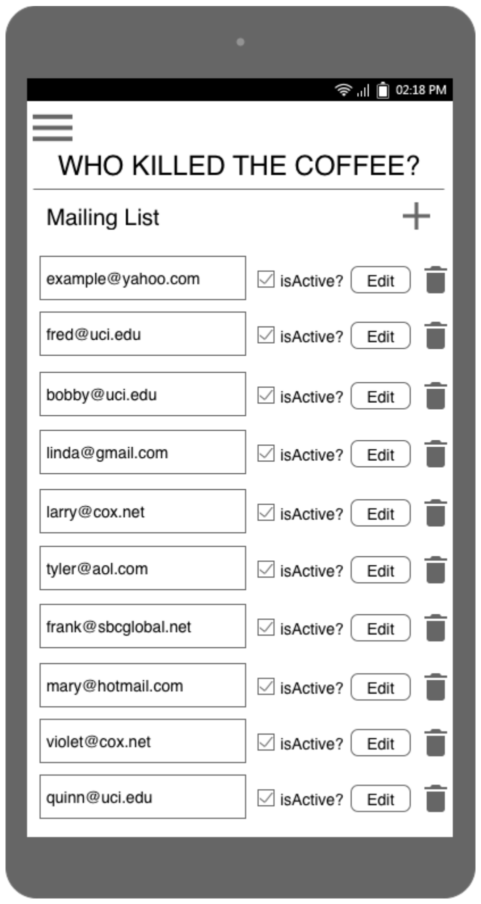


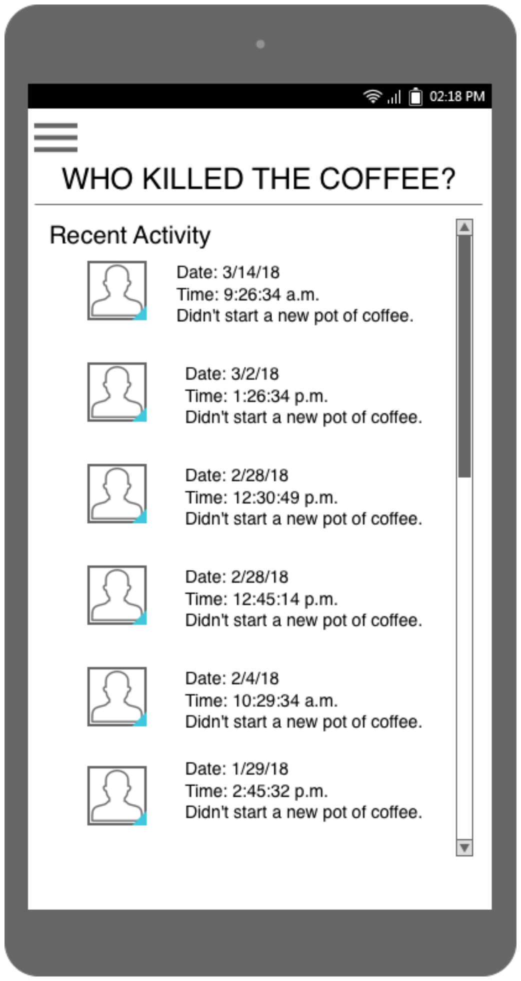
Figure 1: Sketch of the device.

Our device features a coffee maker that sits atop two boards separated by a load cell. This allows for measurement of the weight of the carafe, which the attached Raspberry Pi receives. The attached Pi Cam, located next to the coffee maker, is tilted upwards at an optimal angle for taking a picture of the user. The boards under the coffee maker include bolts and washers to create space between the wood and the load cell to create a stable base and get an accurate reading.

1.3 User Interface



The user interface will include a method for users to set up the mailing list. With the "Manage Addresses" interface, users will be able to manually delete and add email addresses from their own list, as well as import a list of emails from other sources. Email addresses can be individually altered or removed based on the users' preferences. The email addresses will be viewable as a basic list that can be edited or deleted from the same page. It will also allow users to see more information about who the email address is associated with and when they were added to the coffee pot's mailing list.



WHO KILLED THE COFFEE?

Recent Activity

Date: 3/14/18
Time: 9:26:34 a.m.
Didn't start a new pot of coffee.

Date: 3/2/18
Time: 1:26:34 p.m.
Didn't start a new pot of coffee.

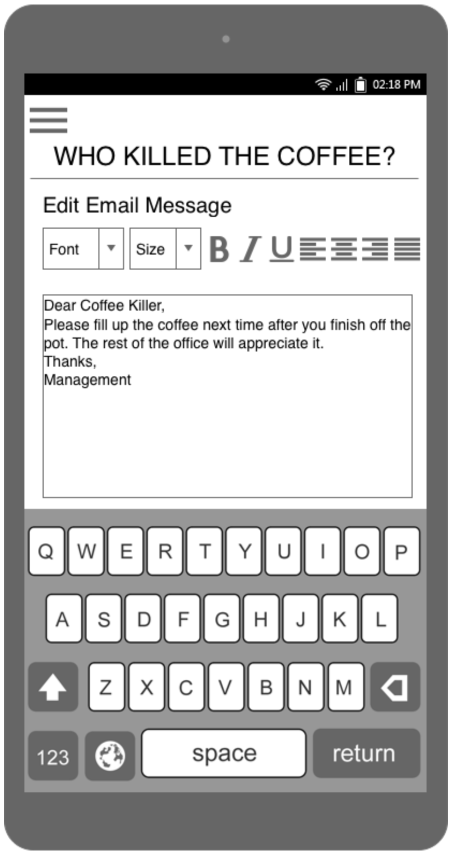
Date: 2/28/18
Time: 12:30:49 p.m.
Didn't start a new pot of coffee.

Date: 2/28/18
Time: 12:45:14 p.m.
Didn't start a new pot of coffee.

Date: 2/4/18
Time: 10:29:34 a.m.
Didn't start a new pot of coffee.

Date: 1/29/18
Time: 2:45:32 p.m.
Didn't start a new pot of coffee.

The user interface will also feature an activity feed that compiles the images and data to provide updates of whoever killed the coffee. This will appear similar to a social media feed with images and timestamps marking when the coffee pot was last emptied and not refilled and who was responsible.



Dear Coffee Killer,
Please fill up the coffee next time after you finish off the pot. The rest of the office will appreciate it.
Thanks,
Management

Thirdly, our device's user interface will include a settings interface from which the users can customize the message that is sent along with the image of who killed the coffee. This will be similar to existing email composition interfaces, with basic font types and formatting. Users will be able to adjust email settings and preferences for all future emails sent from the coffee maker.

1.4 How It Works

Our coffee maker features a Raspberry Pi, a scale, and a camera. The Pi knows the weight of the carafe when the pot is empty. When a user removes the carafe, the device takes their photo. After taking the photo, a 90-second timer starts, giving the user a grace period to put the carafe back and/or brew a new pot of coffee. If the user puts the carafe back and the device detects, by weight, that the carafe is empty, the photo of the user is saved locally and emailed to a lists of email addresses along with a note to shame them for not making a new pot. Similarly, if the user does not put the carafe back or make a new pot in the 90-second grace period, their photo will also be emailed out along with a note to shame them for forgetting. If the user puts the carafe back and it still has (enough) coffee it in or if the user puts the empty carafe back and starts brewing a new pot, their photo is deleted and is not emailed.

The device provides the user with status updates through the form of LED indicators. Below are explanations of what the color combinations mean, listed in the order in which they would occur.

1. Light will blink red. Remove the pot from the maker and push the button to tare it.

2. Tared. Yellow light blinking. Waiting for you to make some coffee.

3. Make some coffee. Yellow light will stop blinking and green will turn on.

4. Have a cup of joe!

5. When you remove the coffee pot, the green light will turn off and a red light will turn on, indicating it is taking your picture.

6. The red light turns off after a picture is taken. Then the yellow light blinks, waiting for the coffee pot to be returned to the warmer.

7a. If the coffee pot is replaced and is acceptably full, the light turns green and the photo is deleted.

7b. If the coffee pot is replaced and is not full enough, the red light turns on and the yellow blinks to indicate there is time left to make a new pot of coffee.

8. All 3 lights on. The green light turns on only while connection to the SMTP server is open.

1.5 Similar Technology

Smart and wireless coffee makers do exist. Mr. Coffee has the Smart Optimal Brew coffee maker that allows brewing control and status updates via a mobile app. This allows someone to check on their phone through an app if the coffee pot is empty or not empty. This coffee maker is convenient in the sense that you can check the status of the coffee pot from their desk. However, this doesn't address the issue of having to brew a new pot if someone else forgot to. You won't be surprised when you walk up to the coffee maker and see that it's empty. but you will still have to wait for the new pot to brew. Orenda's Smart Coffee Maker has very similar functionality as the Mr. Coffee one. Orenda's focuses on personalization, tracking your coffee and brewing time preferences. However, it still does not address the issue of someone not starting a new pot of coffee. The BrewGenie Smart Coffee Maker does the best at addressing the issue. It sends notifications via the app to the user to let them know when a pot is currently brewing and when the pot has finished brewing. The BrewGenie coffee maker also uses Bluetooth connection, which is not as secure and is also inconvenient because the user needs to be within 20 feet of the coffee maker Similar to the other coffee makers, it doesn't present a reminder to start a new pot of coffee after finishing the last.

While these have some overlapping intentions, our device is the only one that encourages remembering to start brewing a new pot of coffee. Our device is a "smart" coffee maker but satisfies a need that current smart coffee makers do not.

1.6 Security

The Raspberry Pi connects to the Internet, but is configured properly to make attacks difficult. The Pi isn't serving as a web server itself but instead uses TTLS to establish a secure connection. We changed the default username and password of the Raspberry Pi, as well as the sudo password. SSH was disabled. A potential location of breach and acces is the Pi's onboard camera interface. For the email protocol, the Pi is connected to the email server only when actively being used.

The Pi takes a photo of the user, which is stored locally on the Pi and is otherwise physically inaccessible. The Pi needs to collect the user's photo because that is the device's main functionality. If the carafe still has coffee or the user starts brewing a new pot, the image is deleted from the Pi. Pictures are only shared with others if the user consumes the last of the coffee and does not brew a new pot.

1.7 Future Development

Upon observation of the existing market of coffee makers, future development would entail a revision of the entire product. As of this initial iteration, the product is a supplemental "accessory" to coffee pots as opposed to an all-in-one "smart" coffee pot like those that currently exist. This product would be developed to act as an add-on to any normal coffee pot and add "smart" functionality. The decision was made in order to allow for maximum affordability and accessibility.

With regard to future developments of features; increased security and features could be added. Security could be increased with firewalls, data encryption, and stronger password protocols.

Additional features could include more options, such as the ability to text instead of email. An onboard interface that includes user-friendly button operation, physical or touch-based, could be implemented. The current red, yellow, green light scheme may be confusing.

More over, features such as a web, onboard, and mobile app interface could be implemented to streamline the process of shaming the perpetrator. A stronger, higher pixel, wide-lense camera could be added to the system for both more accurate photos but for optimal positioning of the camera - it was seen that clever positioning or obtaining of the coffee pot made for ambiguous photos.

2. Appendix

A.1 Source Code

Additional source code can be found at goo.gl/RPjLCf. wktc.py is the core logic behind the device.

wktc.py

```
import RPi.GPIO as GPIO
import os, subprocess, sys, time
from hx711 import HX711 as HX711Base
from modules.hx711 import HX711
from modules.led import Led
from emailer import mail_all

# define constants
WEIGHT_OF_CARAFE = 371
MIN_CUPS_SAFE = 2
# (min cups * 6oz/cup * 30g/oz) + weight of carafe in g
MIN_WEIGHT_SAFE = (MIN_CUPS_SAFE * 6 * 30) + WEIGHT_OF_CARAFE
GRACE_PERIOD = 30  # time to make a new pot in seconds
PHOTO_DIR = './photos'

# init hx711 and load cell
hx = HX711(5, 6)
hx.set_reference_unit(-410)

# prepare led
LED_RED = Led(23)
LED_GREEN = Led(24)
LED_YELLOW = Led(25)

BTN = 3

# setup button
GPIO.setmode(GPIO.BCM)
GPIO.setup(BTN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

def cleanAndExit():
    GPIO.cleanup()
    delete_photo()
    sys.exit()

def setup():
    while True:
        LED_RED.blink()
        input_state = GPIO.input(BTN)
        print(input_state)
        if input_state == False:
            # print('button pressed')
            hx.reset()
            hx.tare()
            break
        else:
            pass

def main():
    try:
        setup()
        while True:
            # make sure all the lights are off
            LED_RED.off()
            LED_YELLOW.off()
            LED_GREEN.off()

            # wait for someone to start brewing coffee
            wait_for_brew()  # BLINKS YELLOW
            while True:
                if mainloop() is False:
                    break

    except (KeyboardInterrupt, SystemExit):
        cleanAndExit()

    except Exception as e:
        GPIO.cleanup()
        delete_photo()
        raise e


def wait_for_brew():
    print('waiting for new brew')
    while True:
        LED_YELLOW.blink(1, 0.5)
        if get_reading() >= MIN_WEIGHT_SAFE:
            break

def mainloop():
    LED_GREEN.on()
    while True:
        # monitor weight
        reading = get_reading()
        # if we detect weight lower than the weight of empty carafe
        # then someone must have removed the pot from the warmer
        if reading < WEIGHT_OF_CARAFE:
            LED_GREEN.off()
            LED_RED.on()
            timestamp = time.localtime()  # save timestamp when take_photo()
called
            take_photo()

            # start timer
            start = time.time()
            LED_RED.off()

            while True:
                LED_YELLOW.blink(1, 0.5)
                timer = time.time() - start
                # print(round(timer))
                if timer < GRACE_PERIOD:
                    reading = get_reading()
                    # poured a cup with left to share or made a new pot
                    if reading > MIN_WEIGHT_SAFE:
                        delete_photo()
                        LED_GREEN.on()
                        print('***** enjoy your coffee! *****')
                        break
                    # had the nerve to put empty or near-empty pot back
*shame!*
                    elif reading >= WEIGHT_OF_CARAFE:
                        LED_RED.on()
                        LED_YELLOW.on()

                # time's up
                else:
                    # report the last drop bandit!!!!
                    LED_RED.on()
                    LED_YELLOW.on()
                    filename = save_photo(timestamp)
                    if filename is not None:
                        LED_GREEN.on()
                        email_photo(filename)
                        time.sleep(2)
                        LED_GREEN.off()
                        return False

def take_photo():
    subprocess.run('./take_photo.sh')  # saves as ./photos/tmp.jpg
    path = os.path.join(PHOTO_DIR, 'tmp.jpg')
    if os.path.isfile(path):
        return True
    else:
        return False
```

```
def delete_photo(filename='tmp.jpg'):
    # print('deleting... ' + str(filename))
    path = os.path.join(PHOTO_DIR, filename)
    if os.path.isfile(path):
        os.remove(path)
        if os.path.isfile(path):
            return False
    return True

def save_photo(timestamp=time.localtime()):
    """
    rename the temporary photo at PHOTO_DIR/tmp.jpg
    with the appropriate timestamped filename in same directory
    param: timestamp: seconds since the epoch when photo taken (default now)
    type: timestamp: struct_time such as return from time.localtime()
    rtype: filename of newly saved file, or None if save failed
    """
    # print('saving...')
    src_path = os.path.join(PHOTO_DIR, 'tmp.jpg')
    filename = str(time.strftime('%Y-%m-%d_%H%M%S', timestamp)) + '.jpg'
    dst_path = os.path.join(PHOTO_DIR, str(filename))
    if os.path.isfile(src_path):
        os.rename(src_path, dst_path)
        if os.path.isfile(dst_path):
            return filename
        # else:
            # print('oops! unable to rename file')

    # else:
    #     print('could not rename file because ' + str(src_path) + ' does not
exist')

    return None

def email_photo(filename):
    mail_all(filename, PHOTO_DIR)

def get_reading():
    hx.reset()
    time.sleep(0.5)
    return round(hx.get_weight())

if __name__ == "__main__":
    main()
```

A.2 References

[1] Avia Semiconductor, "24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales," HX711 datasheet (n.d). [Online], Available: http://www.aviaic.com/Download/hx711_brief_en.pdf.pdf.

I’m just going to start adding some stuff down here that I know you guys won’t be adding, and we can find a place for it to live later lol.

Table __ Materials blah blah

The HX711 load cell amplifier is a small breakout board for HX711 IC that allows to easily read data from load cells to measure weight. The DT and SCK were connected to GPIO 5 and GPIO 6 of the Raspberry Pi.

Schematic Diagram
[Katelyn inserts diagram here when she cleans up the diagram. The breadboard make it a mess!]