

Doorbell Monitor for Engineering 240 Design Project

By: Kendall Knapp and Nazmus Shakib Khandaker

Consultation by: Joseph Leeper

Instructor: Steven Portscher

Product Documentation

May 2014

Contents

Introduction	3
Doorbell Monitor – Device Documentation	4
Device Schematic	4
Operations Overview	4
Receiving input.....	5
The counter.....	5
Sending Email Notification.....	6
Sending SMS (i.e. text message) Notification	6
Time Delay	7
User Configuration.....	8
Important Security and Privacy Information	8
Formatting Description	8
Formatting Example.....	8
Hardware Stop Button	9
Practical Applications.....	10
Limitations	11
Cost Analysis	12
What We Learned	13
Conclusion.....	14
Appendix	15
Code	15
References	19

Introduction

Digital technology is prevalent in today's world, and it serves as a foundation for a flexible mechanism in innovation. It allows one to play with concepts and test ideas easily and quickly. Moreover, since such technology has become so affordable, we could not help but jump at this great opportunity to create the doorbell monitor we envisioned.

The catalyst of our project is the recent development in affordable computers—namely the Raspberry Pi—designed for students, enthusiasts, and hackers alike. Priced at only \$35, a fully functional desktop-class computer that fits in the palm of one's hand was a dream just a decade ago. It was a dream that has become a reality today. The potential of such a device is as large as our minds can imagine. The accessibility of the technology allowed us to execute our vision in a matter of weeks. In fact, while the entire process spanned about three months, most of the time we spent was in careful planning rather than building the Doorbell Monitor.

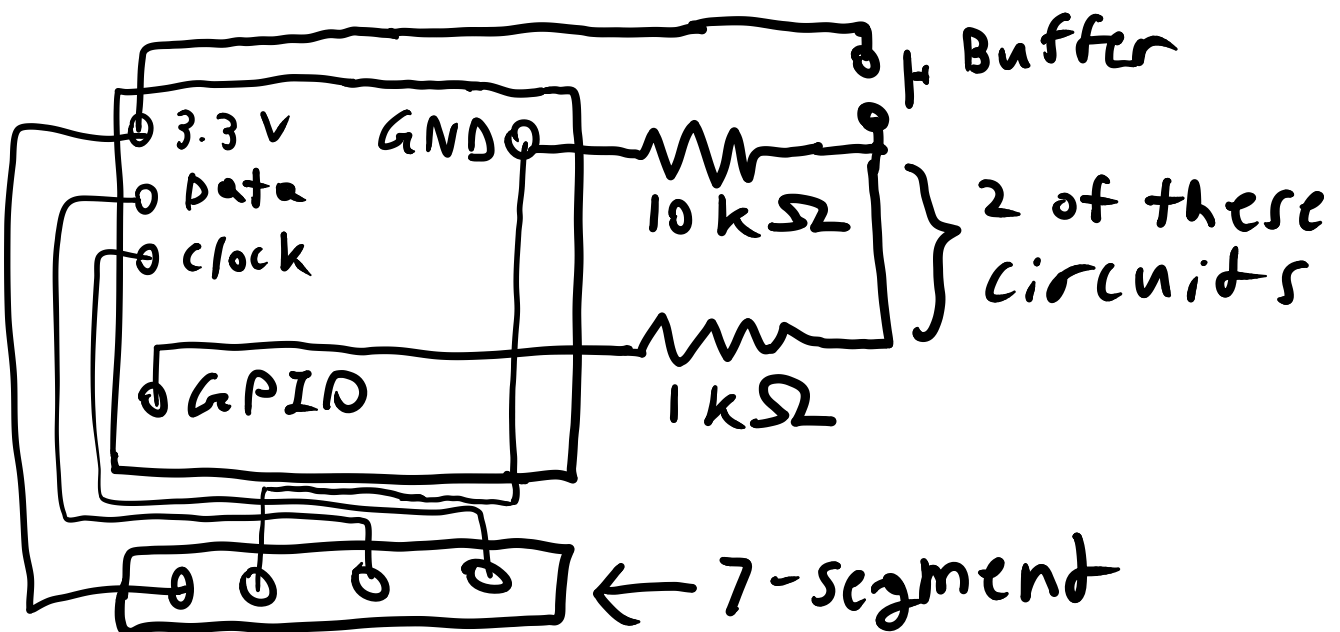
The design project demonstrates what two engineering students in their second year in college course can do in just a few months with the help of modern technology. The aim of this paper is to describe how the doorbell-monitoring device works, talk about the engineering processes that went into creating the device, and analyze the practical scenarios for which this device may be used.

Doorbell Monitor – Device Documentation

For our design project, we opted to creating a doorbell-monitoring device, creatively named “Doorbell Monitor”. The Doorbell Monitor serves two purposes. It keeps track of how many people rings the doorbell in a set time interval. The device also sends an email and SMS (text message) notification to the user of the device at the time in which someone presses the doorbell. We describe the various aspects of the device and its operation in more detail.

Device Schematic

The Doorbell Monitor consists of a Raspberry Pi computer that is connected to the internet, an Adafruit seven-segment display with four digits, a doorbell button, and a reset button.



Operations Overview

We programmed the doorbell monitor to carry out a sequence of operations when someone rings the doorbell. Below is a brief summary of what the device does. Each item will be explored in more detail following this overview.

If the doorbell push button is activated, the following steps occur.

1. Takes input (doorbell push button)
2. Determines whether to take action by going on to the next step (See Time Delay)
3. Increment Counter in Seven-Segment Display
4. Sends Email or SMS notification
5. Starts Time Delay
6. Awaits next input

If the reset button is activated, the following steps occur.

1. Takes input (reset button)
2. Resets counter and clears seven-segment display
3. Resets time delay
4. Checks if the reset button is still activated for the next five seconds; if so, the program quits

Receiving input

To get the raspberry pi to receive an input, we connected a GPIO pin initially to ground. When a button is pushed, the pin is tied to the power, driving the pin high. Then, our program reads it using the built in GPIO library. Two resistors keep the current from getting to large. We obtained the information to do all of this from a University of Cambridge tutorial [1]. Adafruit's GPIO library made reading the input easy [2]. We have two separate inputs. One is for the doorbell being rung; the other is for the reset button.

The counter

Every time the doorbell button is pushed, the counter displays increments by one. Pushing the reset button changes the counter display back to zero. We used a built-in Adafruit LED backpack and Adafruit's 7-segment library to do this easily [2].

Sending Email Notification

When the doorbell button is pushed, the device sends an email to the user's email address. The email notification displays the date and time at which the doorbell was rung. Figure 1 shows a screenshot of the email.

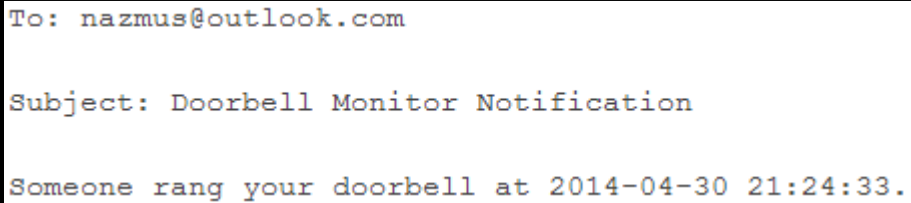
A screenshot of an email notification displayed in a monospaced font. The text is as follows:
To: nazmus@outlook.com
Subject: Doorbell Monitor Notification
Someone rang your doorbell at 2014-04-30 21:24:33.
The text is enclosed in a black rectangular border.

Figure 1: Screenshot of the email notification

We got the basic coding to do this from computer programmer and blogger Mike Driscoll [3]. At first, we had difficulties with the program freezing if the email server could not be connected to. Joseph Leeper was instrumental in helping us solve this problem, by introducing a timeout parameter.

The email address to which the notification is sent is configured by the user in a text file. Both the sending and receiving addresses set by the user. The user name and password of the sending email is also needed. This causes a potential security problem.

Sending SMS (i.e. text message) Notification

If the user chooses, he or she can opt to have a Short Message Service (SMS) sent to them. (Note: SMS is the technical term for “text messages” used in North America.) If the user opts for this, he or she will receive an SMS notification on their cell phone instead of an email.

This is still done using the email code. Instead of sending an email to another email address, the email is sent to a phone number. Every provider has a different service that does this, always free. All you need

is the 10-digit phone number and the service's email address. For example, Verizon's is 0123456789@vtext.com.

The SMS will display the same content as that of the email notification, including the date and time at which the doorbell was rung.

Time Delay

The device is programmed to have a time delay between which the doorbell button push is counted. This is done to prevent counting frequent ringing of the doorbell by the same person as discrete rings by unique visitors. It is natural that when a visitor comes to the door, he or she will ring the bell several times if no one answers the door. The device is programmed to deal with this scenario.

We set our default time delay to be two minutes because we are assuming that a visitor will wait less than two minutes for an answer before leaving. The visitor is likely to press the doorbell multiple times during this two-minute period. Therefore, during this two-minute delay, the counter does not update nor is any email or SMS notification sent if the doorbell button is pressed.

This time delay is simply written into our code, and can be changed quickly from inside it. It would be easy to make a separate text file that the user can modify to adjust the length of the time delay.

User Configuration

Important Security and Privacy Information

It is not safe to store passwords in plain text file. You can and will very likely get your account hacked!

Use two-factor authentication and an application specific password. For more information on application specific passwords and two-factor authentication, search the internet using your favorite search engine. DO NOT upload your `doorbell_config.txt` file on cloud storage without absolutely making sure it is not viewable to the public. In fact, it is best not to upload the file to the internet at all.

Formatting Description

The `doorbell_config.txt` file must have four line of values in specific order.

- Line 1: Sender's email address (must be a Gmail address)
- Line 2: Recipient's email address (can be any email address). This is the person that will be alerted via email when someone rings the doorbell
- Line 3: Username of the sender's email address (likely the same as Sender's email address, but not necessarily)
- Line 4: Password of the sender's email address. If you use two-factor authentication (highly recommended), please create and use an application specific password.

Formatting Example

The `doorbell_config.txt` file must be formatted exactly as shown below. Replace the generic values with ones specific to you:

- `yourname@gmail.com`
- `yourname@example.com`
- `yourname@gmail.com`
- `password`

Hardware Stop Button

The reset button can act as a hardware stop button to exit the program that runs the doorbell monitor.

Since the doorbell monitor program runs at the startup of the Raspberry Pi computer, the user will need to stop the program in order to boot into the Linux shell and log in to the desktop environment of the Raspberry Pi. If the program is running from within the Linux desktop environment, the hardware stop button can be used to exit the program and return to the shell.

To activate the hardware stop button and quit the program, press and hold the reset button for five seconds. This will stop all operation regarding the doorbell monitor.

Important: If the user wants to boot the Raspberry Pi computer to Linux shell, he or she needs to power on the device, *wait* for the seven-segment display to turn on, and then press and hold the reset button for five seconds to get to the login screen. A monitor and keyboard must be connected to the Raspberry Pi computer in order to operate within the Linux shell.

Practical Applications

The most obvious use for the doorbell monitor is in a home setting. If someone rings your doorbell while you are away, you will be notified on your email or cell phone. If you were expecting someone but had to leave for a short time, this would be useful because you could let him or her know you will be back soon. In addition, if you are in the house but do not hear the doorbell, a text message would let you know to go answer the door. The counter tells you how many people have tried to reach you while you were away. This could be reset every time, or left on to keep a record of how many people have visited your house. For businesses, the counter would be the most useful feature. With modifications, it could be used to count how many customers enter a store.

If the time delay was set to zero, and the phone number of someone you do not like was used, the monitor would also make an excellent spam machine. It would even tell you exactly how many messages were sent. However, we do not advocate this use.

Limitations

Currently, our doorbell monitor operates using DC current. A doorbell for a house would use AC current.

To use our device on an actual doorbell, we would need to convert the buzzer's input to DC current and lower the voltage. However, the program should work the same.

The device is also fragile. If the connection with the 7-segment display is broken, the counter will stop working. Plugging it back in does not make it work again; the entire program has to be restarted. A more secure connection would be needed. Additionally, a protective outer shell for the wiring would be helpful and aesthetically pleasing.

Cost Analysis

Table 1 breaks down the cost for our prototype.

Raspberry Pi	\$30
USB Wireless Adapter	\$10
7-Segment Display	\$10
Board, wires, and resistors	\$5
Total	\$55

Table 1: Cost of parts

Mass production would significantly lower the cost of each product. We estimate we could bring the cost down to \$40 per unit.

Additionally, there is the labor cost. Collectively, the two of us spent approximately 50 hours designing the monitor. At \$100 an hour cost to an employer, this means a design cost of about \$5000.

Reasonably, we could sell the monitors for \$50, making a \$10 profit on each one. So, we would need to sell 500 monitors to start making a profit. Of course, this is just a rough estimate.

What We Learned

This project was a very useful experience for us. Throughout the design process, we learned how to use a Raspberry Pi, and saw that it can be a very powerful tool. It is very cheap, but can function as a basic computer. It also provides internet connectivity. At the same time, it can read analog signals, providing countless possibilities.

Along with this, we learned how to work with Linux. The command line was not something we were familiar with using. Thanks to our constant work with the Raspberry Pi and Joseph Leeper's frequent assistance, we now have the ability to operate Linux from the command line.

The bulk of our design time was spent working on the python code. The process of making our program gave us a much better understanding of coding with python. Even more importantly, we gained experience with debugging and solving unexpected problems with our code.

Conclusion

In this project, we developed the Doorbell Monitor to record the visitors that come to one's door. It can count the total number of visitors in a set time as well as notify the user at the exact time someone rings the doorbell. The device has built in safeguards to prevent duplicate counting and to prevent itself from freezing if the email server cannot be reached. We added a mechanism to quit the program that runs the Doorbell Monitor and let the user boot to the Raspberry Pi Computer's Linux desktop environment by just pressing and holding the reset button.

The device has potential practical applications that range from business to consumers. Some of these applications we thought of while others are waiting to be imagined by the customer.

In creating a simple device to monitor recipients at the door, we were once again reminded of the fact how far technology has taken us in just a few years. During the project, we discussed with our Instructor about past senior design projects from the University of Illinois and Bradley University. Indeed, the projects from just a few years ago involved creating devices that are not easy to make in a few days of work, and they can now be affordably purchased from any technology retailer. It is a remarkable pace of technological evolution, one that we are most excited to experience.

Appendix

In this section, the entire source code for the program that operates the Doorbell Monitor is provided.

This code is licensed under Creative Commons with attribution to the authors of this document. For more information on Creative Commons Licensing, please visit <http://creativecommons.org>. Please note that calls to the Adafruit Library run code produced by Adafruit. This license does not apply to the third party code. Please consult Adafruit in regards to their licensing policy for their programs.

Because the code calls for particular Adafruit Libraries, these exact libraries must be installed on the Raspberry Pi computer in order for the code to work. Please visit <http://www.adafruit.com> for installation instructions.

Code

```
# ***** SOURCE CODE FOR DOORBELL MONITOR (ENGR240) ***** #

# ***** DEVICE DESCRIPTION *****
# The Doorbell Monitor records the time at which the doorbell button was pressed and the
# number of time it was pressed.
# It displays the number of time the button was pressed to the 7-segment display
# It sends an email to the user with the recorded time of the doorbell pressed

# ***** CODE DESCRIPTION *****
# The code is designed to work with the raspberry pi and select Adafruit hardware and
# associated libraries.
# The doorbell push button is attached to GPIO pin 17. The 7-segment decoder is attached as
# output
# Each time the push button is pressed, the counter variable count up and system time recorded
# Counter variable is outputted to the 7-segment decoder and email is sent to user with
# recorded time
# Reset button (input connection to be determined) resets counter to 0, and counter is
# outputted to 7-segment decoder

# ***** LINKS *****
# Adafruit 7-segment python library on GitHub: https://github.com/adafruit/Adafruit-Raspberry-
# Pi-Python-Code/blob/master/Adafruit_LEDBackpack/Adafruit_7Segment.py
# Using SMTPLIB with Gmail as outgoing server:
http://codecomments.wordpress.com/2008/01/04/python-gmail-smtp-example/

#Import modules
import RPi.GPIO as GPIO      #Enables the use of the input pins (GPIO)
import time                  #Enables time delays and accessing system clock
```

```

import urllib2
import smtplib          #SMTP library: Internet Mail Transfer protocol. Needed for email
import string           #Enables the use of strings in variable
from sys import path
path.append("/home/pi/doorbell/Adafruit-Raspberry-Pi-Python-Code/Adafruit_LEDBackpack")
import Adafruit_7Segment
import Adafruit_LEDBackpack

#Set-up and variables
GPIO.setmode (GPIO.BCM)
GPIO.setup (17,GPIO.IN)          #Sets pin 17 as an input pin (Used for doorbell push
button)
GPIO.setup (22,GPIO.IN)          #Sets pin 22 as an input pin (Used for reset button)
display = Adafruit_7Segment.SevenSegment()
reset = Adafruit_LEDBackpack.LEDBackpack()
count = 0
dig_1 = 0
dig_2 = 0
dig_3 = 0
dig_4 = 0
wait = 0
stop = 0
pause = 0
display.writeDigit(4, 0, False)

def internet_on():              #Checks if email server responds. If no response received,
email notification will not be sent.
    try:
        response=urllib2.urlopen('http://74.125.228.100', timeout=1)
        return True
    except urllib2.URLError as err: pass
    return False

def send_email():\
#The following snippet is the code for using the SMTP library, taken from
http://www.blog.pythonlibrary.org/2010/05/14/how-to-send-email-with-python/
#It has been modified to work with gmail as outgoing server, courtesy of information provided
in http://codecomments.wordpress.com/2008/01/04/python-gmail-smtp-example/
#The function makes use of an external text file which contain email account information,
including username and password (or application specific password if two-step authentication
is enabled)
#Please see documentation for information on how to set up the config file. You may also refer
to the dedicated text file available on this program directory

    configFile = open('c:\users\NazmusShakib\Desktop\doorbell_config.txt','r')
    #IMPORTANT: Replace this directory to the directory where the doorbell_config.txt file
is located
    lineCount = 0
    for line in configFile:
        lineCount = lineCount + 1

        if lineCount == 1:
            FROM = line
        elif lineCount == 2:
            TO = line
        elif lineCount == 3:
            USERNAME = line

```



```

        elif lineCount == 4:
            PASSWORD = line
        else:
            print "ERROR: line count of doorbell_config.txt exceeds standard
bounds. Make sure the doorbell_config.txt is formatted exactly as specified. Please see \"How
to setup doorbell_config.txt\" for instructions. Line count = " + lineCount + "."

    lineCount = 0
    configFile.close()

    SUBJECT = "Doorbell Monitor Notification"
    text = "Someone rang your doorbell at " + time.strftime('%Y-%m-%d %H:%M:%S') + "."
    #Leave the following BODY block intact. Only change the values above.
    BODY = string.join((
        "From: %s" % FROM,
        "To: %s" % TO,
        "Subject: %s" % SUBJECT ,
        "",
        text
    ), "\r\n")

    try:
        server = smtplib.SMTP('smtp.gmail.com', 587)
    except smtplib.SMTPServerDisconnected:
        print "Connection timed out:"
        return False
    server.ehlo()
    server.starttls()
    server.ehlo()
    server.login(USERNAME, PASSWORD)
    server.sendmail(FROM, [TO], BODY)
    server.close()

while stop == 0:
    #The program will continuously loop until the user invokes the
    Hardware Stop button, wich turns the "stop" variable to 1
    if wait == 0:
        #Time Delay (See documentation)
        #The following code will set the seven-segment display
        if (GPIO.input(17)):
            dig_1 += 1

            if dig_1 > 9:
                dig_1 = 0
                dig_2 += 1

            if dig_2 > 9:
                dig_2 = 0
                dig_3 += 1

            if dig_3 > 9:
                dig_3 = 0
                dig_4 += 1

            display.writeDigit(0, dig_4, False)
            display.writeDigit(1, dig_3, False)
            display.writeDigit(3, dig_2, False)
            display.writeDigit(4, dig_1, False)
    #This controls the time delay
    wait = 1

```

```

        pause = time.time()
        if internet_on() is True:
            send_email()

if time.time() > pause + 10:
    wait = 0

#This code runs if the reset button is pushed
if (GPIO.input(22)):
    display.writeDigitRaw(3, 0)
    display.writeDigitRaw(1, 0)
    display.writeDigitRaw(0, 0)
    dig_1 = 0
    dig_2 = 0
    dig_3 = 0
    dig_4 = 0
    display.writeDigit(4, 0, False)

    wait = 0    #Resets time delay

#Kill switch: Hold reset for 5s
holdReset = time.time()
while (GPIO.input(22)):
    if time.time() > holdReset + 5:
        print "Program Ended"
        stop = 1

```

References

- [1] University of Cambridge Computer Laboratory, web page. Available at:
https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/buttons_and_switches/.
Accessed April 2014.
- [2] Adafruit Raspberry Pi Python Code, web page. Available at:
<https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code>. Accessed April 2014.
- [3] Driscoll, Mike. The Mouse vs. The Python, web page. Available at:
<http://www.blog.pythonlibrary.org/2010/05/14/how-to-send-email-with-python/>. Accessed
April 2014