Volume 15, No. 2
March 2017
ISSN: 1545-679X

INFORMATION SYSTEMS EDUCATION JOURNAL

Special Issue – Teaching Cases

The **Information Systems Education Journal** (ISEDJ) is a double-blind peer-reviewed academic journal published by **EDSIG**, the Education Special Interest Group of AITP, the Association of Information Technology Professionals (Chicago, Illinois). Publishing frequency is six times per year. The first year of publication was 2003.

ISEDJ is published online (http://isedj.org). Our sister publication, the Proceedings of EDSIGCon (http://www.edsigcon.org) features all papers, panels, workshops, and presentations from the conference.

The journal acceptance review process involves a minimum of three double-blind peer reviews, where both the reviewer is not aware of the identities of the authors and the authors are not aware of the identities of the reviewers. The initial reviews happen before the conference. At that point papers are divided into award papers (top 15%), other journal papers (top 30%), unsettled papers, and non-journal papers. The unsettled papers are subjected to a second round of blind peer review to establish whether they will be accepted to the journal or not. Those papers that are deemed of sufficient quality are accepted for publication in the ISEDJ journal. Currently the target acceptance rate for the journal is under 40%.

Information Systems Education Journal is pleased to be listed in the 1st Edition of Cabell's Directory of Publishing Opportunities in Educational Technology and Library Science, in both the electronic and printed editions. Questions should be addressed to the editor at editor@isedj.org or the publisher at publisher@isedj.org. Special thanks to members of AITP-EDSIG who perform the editorial and review processes for ISEDJ.

2017 AITP Education Special Interest Group (EDSIG) Board of Directors

Leslie J. Waguespack Jr Bentley University President	Jeffry Babb West Texas A&M Vice President	Scott Hunsinger Appalachian State Univ Past President (2014-2016)
Meg Fryling Siena College Director	Lionel Mew University of Richmond Director	Muhammed Miah Southern Univ New Orleans Director
Rachida Parks Quinnipiac University Director	Anthony Serapiglia St. Vincent College Director	Li-Jen Shannon Sam Houston State Univ Director
Jason Sharp Tarleton State University Director	Peter Wu Robert Morris University Director	Lee Freeman Univ. of Michigan - Dearborn JISE Editor

Copyright © 2017 by the Education Special Interest Group (EDSIG) of the Association of Information Technology Professionals (AITP). Permission to make digital or hard copies of all or part of this journal for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial use. All copies must bear this notice and full citation. Permission from the Editor is required to post to servers, redistribute to lists, or utilize in a for-profit or commercial use. Permission requests should be sent to Nita Brooks, Editor, editor@isedj.org.

INFORMATION SYSTEMS EDUCATION JOURNAL

Editors

Jeffry Babb
Senior Editor
West Texas A&M University

Thomas Janicki
Publisher
U of North Carolina Wilmington

Donald Colton
Emeritus Editor
Brigham Young University Hawaii

Cameron Lawrence
Teaching Cases Co-Editor
The University of Montana

Anthony Serapiglia
Teaching Cases Co-Editor
St. Vincent College

Nita Brooks
Associate Editor
Middle Tennessee State Univ

Wendy Ceccucci
Associate Editor
Quinnipiac University

Melinda Korzaan
Associate Editor
Middle Tennessee State Univ

Guido Lang
Associate Editor
Quinnipiac University

George Nezlek
Associate Editor
Univ of Wisconsin - Milwaukee

Samuel Sambasivam
Associate Editor
Azusa Pacific University

2016 ISEDJ Editorial Board

Samuel Abraham
Siena Heights University

Teko Jan Bekkering
Northeastern State University

Ulku Clark
U of North Carolina Wilmington

Jamie Cotler
Siena College

Jeffrey Cummings
U of North Carolina Wilmington

Christopher Davis
U of South Florida St Petersburg

Gerald DeHondt II
Kent State University

Audrey Griffin
Chowan University

Janet Helwig
Dominican University

Scott Hunsinger
Appalachian State University

Mark Jones
Lock Haven University

James Lawler
Pace University

Paul Leidig
Grand Valley State University

Michelle Louch
Duquesne University

Cynthia Martincic
Saint Vincent College

Fortune Mhlanga
Lipscomb University

Muhammed Miah
Southern Univ at New Orleans

Edward Moskal
Saint Peter's University

Monica Parzinger
St. Mary's University

Alan Peslak
Penn State University

Doncho Petkov
Eastern Connecticut State Univ

James Pomykalski
Susquehanna University

Franklyn Prescod
Ryerson University

Bruce Saulnier
Quinnipiac University

Li-Jen Shannon
Sam Houston State University

Jason Sharp
Tarleton State University

Karthikeyan Umapathy
University of North Florida

Leslie Waguespack
Bentley University

Bruce White
Quinnipiac University

Peter Y. Wu
Robert Morris University

Teaching Case

Ding Dong, You've Got Mail!
A Lab Activity for Teaching the Internet of Things

Mark Frydenberg
mfrydenberg@bentley.edu
Computer Information Systems Department
Bentley University
Waltham, MA 02452, USA

Abstract

Connecting ordinary devices to the Internet is a defining characteristic of the Internet of Things. In this hands-on lab activity, students will connect a wireless doorbell to the Internet using a Raspberry Pi computer. By modifying and running a program on the Raspberry Pi to send an email or text message notifying a recipient that someone is at the door, students will gain an appreciation for how the Internet of Things enables devices to work together to produce new products or solutions. The activity also serves as a brief introduction to interacting with the Linux operating system on a Raspberry Pi computer. Working in groups of 3 or 4, students will demonstrate their understanding by completing a collaborative lab report that summarizes the key takeaways of each step. Little or no programming skills are required. The experience of physically connecting an everyday object to a device online gives an appreciation of the potential for automating tasks using the Internet of Things.

Keywords: Internet of Things, Raspberry Pi, Python, programming, automation.

1. INTRODUCTION

The Internet of Things continues to become apparent in daily life as the number of everyday items connected to the Internet constantly increases. Sensors, wearable and mobile devices, and apps capable of transmitting, receiving, and modifying data over the Internet allow for new possibilities and business opportunities.

IoT has become common in home automation through thermostats such as Nest that can be adjusted via a mobile app, washing machines in college dormitories whose availability is published online through a service such as laundryview.com, and wearable devices such as FitBit whose sensors monitor heart rate and share that data with a smartphone or computer connected over the Internet.

This lab activity is based on a project by technology blogger Martin Harizanov. (Harizanov, 2013). Harizonov's blog at http://harizonov.com shares several do-it-yourself home automation projects using simple tools and equipment. This lab guide describes how connect a wireless doorbell to the Internet using a Raspberry Pi computer. Upon pressing the doorbell's buzzer, a program running on the Raspberry Pi will send an email or text message to the recipient's computer or mobile device notifying that someone is at the door.

A Raspberry Pi (Raspberry Pi Foundation, n.d.) is a very inexpensive, single-board computer with ports for connecting a keyboard, mouse, and display device. It runs a version of the Linux operating system such as Raspbian, connects to the Internet, and supports developing applications using programming languages including Python, C, C++, Java, Scratch, and Ruby. A Raspberry Pi has pins which enable it to obtain input from a variety of external sensors and devices, such as weather stations and motion detectors. A Raspberry Pi computer is about the

size of a deck of playing cards. (See Appendix 1, Figure 2.) Raspberry Pi computers are popular among hobbyists, students, and educators who want to build solutions as they learn about computing.

You will complete this project in several steps:

- Create a lab report as a Google Doc and share it with your group and instructor
- Identify all of the equipment needed
- Wire the doorbell to the Raspberry Pi
- Connect the keyboard, mouse, camera, Ethernet cable, and display to the Raspberry Pi, and power it on
- Test the camera to make sure it takes pictures and identify the location where they are stored
- Create a folder to store, and examine the source code for the programs that wait for the doorbell to be pressed, and then send the notifications
- Run a program to test the connection between the doorbell and the Raspberry Pi
- Modify and test the program to take a picture and send an email message
- Modify the program to send a text message

You will work in groups of 3 or 4 to complete this assignment. Discuss the **For Your Lab Report** questions with your group as you complete each step, and take turns recording your answers in a lab report document that you create as a Google Doc. As you proceed through the exercise, one group member may serve as the "recorder", typing your group's answers to the lab report questions directly in the Google Doc, or on paper to enter into the Google Doc at a later time. Except for the last two questions, which ask you to reflect on your own personal experience with this activity, your group will receive the same grade for your lab report. Each group member is responsible to ensure that the lab report shows your group's answers to all of the questions, so remember to review and edit the entire lab report document after completing the activity.

Learning Objectives
After completing this activity, you will be able to:

- Explain ideas related to the Internet of Things
- Identify the components and ports of a Raspberry Pi Computer
- Describe how a Raspberry Pi computer can receive data from an external object
- Recognize and modify simple code statements in Python
- Navigate to files and folders, and run programs in the Linux operating system using both graphical and command line interfaces
- Describe how a program can automate common tasks

Prerequisites
While no programming knowledge is required to complete this activity, some basic digital literacy skills are required:

- Read and follow written instructions
- Identify ports on a Raspberry Pi computer and connect computing devices to them
- Create, edit, and share a Google Document
- Capture and save a screenshot using the Snipping Tool or Print Screen on a Windows PC, or Command-Shift-4 on a Mac computer
- Modify and save files using a text editor
- Navigate and interact with files and folders using an operating system's File Explorer or Finder app
- Examine incoming and sent mail using a Gmail account
- Examine text messages sent to a mobile phone

Preparing for the Lab Activity
Complete these steps before working on this activity:

- Read this description before coming to class on the day that this lab activity is scheduled, so that you will be familiar with the steps. Doing so will save you a lot of time.
- Identify the three other people who are in your group.
- Create a new Google Doc for your group's lab report. Share it with your group members and with your instructor at the email provided.
- Copy and paste the lab report template document into your blank Google Doc. You can find the lab report template at http://bit.ly/doorbellreport

2. GETTING STARTED

Examine the kit containing the equipment for this lab activity. Make sure it contains all of these items, shown in Appendix 1, Figure 1:

- Raspberry Pi with camera installed (or USB webcam)
- Power supply with micro-USB cable
- Ethernet cable
- Wireless doorbell and buzzer with batteries installed
- 1 220-ohm resistor
- 3 clip lead wires (two of the same color for positive, and one of a different color for ground)
- 2 jumper wires already connected to Raspberry Pi pins

In addition, you will need:

- An HDMI display and cable
- A USB keyboard and mouse
- A Gmail address and password so your Raspberry Pi can send email (provided by your instructor)
- A personal Gmail account for receiving email
- A mobile phone at which you can receive a text message (fees from your carrier may apply)

Know Your Pi

A Raspberry Pi is a small, inexpensive single-board computing device that runs a variation of the Linux operating system built specifically for the Pi platform. Many programs that run on a Raspberry Pi are written in the Python programming language.

Raspberry Pi computers have on-board memory and a CPU and graphics processing unit on the circuit board. Processing speed and amount of memory vary depending on the Raspberry Pi model used. SD (Secure Digital) cards store the operating system and other files. Most models have wired Internet, Wi-Fi, and Bluetooth connectivity. For simplicity, this activity uses a wired Internet connection, although it is possible to access the Internet on a Raspberry Pi using Wi-Fi via a USB dongle or on-board Wi-Fi, depending upon the model. (Sims, 2014)

A Raspberry Pi contains several ports and plugs to connect a variety of cables and devices, as shown in Appendix 1, Figure 2. The Raspberry Pi connects to other devices through its general purpose input/output (GPIO) pins which may send or receive signals from connected devices. You can program the pins to interact with devices, sensors, and other signals. Some pins are "ground" pins that do not receive current.

Appendix 1, Figure 3 shows the pins and their names. To orient the Raspberry Pi correctly, hold it so that the GPIO pins are at the right edge.

For Your Lab Report: Know Your Pi What type of cables or devices might you connect or attach to each of the ports or pins, numbered 1 – 8 of Appendix 1, Figure 2?

For Your Lab Report: GPIO Pins Your Raspberry Pi has two pre-connected wires. What are the pin numbers and GPIO names of the pins to which they are connected?

3. CONNECT THE DOORBELL TO THE RASPBERRY PI

Appendix 1, Figure 4 shows the inside of the wireless doorbell receiver. To simplify the process of connecting the doorbell to the Raspberry Pi, and avoid damaging the wiring inside the receiver box, a red (positive) and black (ground) wire have been soldered to the corresponding wires inside the doorbell receiver.

You will connect the doorbell chime to the Raspberry Pi via the soldered red (positive) and black (negative) wires soldered to the doorbell's speaker so that you can easily connect them to the Raspberry Pi without concern about damaging the fragile wiring inside. When the speaker chimes after pressing the doorbell buzzer, wires attached to the speaker will carry the current to the Raspberry Pi through the clip wires that you attach to connect both devices.

Your kit contains three clip lead wires: two of one color, and one of another. The two clips of the same color will serve as the positive wires transferring the electrical current, the one clip of the other color will serve as a "ground" wire completing the circuit.

Clip one end of each positive wire (for which you have two clips of the same color) to the ends of a 220 ohm resistor. A resistor reduces current flow, and, acts to lower voltage levels within circuits. Use the clip of the same color connected to the other end of the resistor, to the wire coming from the Pi that is connected to the GPIO pin closest to the end of the Raspberry Pi that has USB ports.

Clip the other (Ground) wire to connect the speaker with the wire attached to a Ground pin (the pin closest to the end of the Raspberry Pi that does not have USB ports) on your Raspberry Pi. The completed setup is shown in the diagram in Appendix 1, Figure 5.

4. SET UP THE RASPBERRY PI

Follow these steps to set up your Raspberry Pi.

1. Connect the keyboard, mouse, Ethernet, and HDMI cables to the appropriate ports (Connect an external USB webcam if you are not using the built in Pi camera).
2. Verify that the MicroSD card is inserted.
3. Turn on the monitor, and make sure input is set to come from the attached HDMI cable.
4. Connect the power cable to the Raspberry Pi and watch it boot.
5. Sign in to the Raspberry Pi using username *pi* and password *raspberry* .
6. Type the command *startx* to load a graphical user interface.
7. Open a file manager app and navigate to Desktop. You should see a folder named StarterFiles. Copy those files to a new folder on the desktop named Doorbell. Be careful that you name the Doorbell folder with a capital D.
8. Open the Doorbell folder and verify that it contains three files: webcam.jpg, sendnotify.py, wait_doorbell.py .

For Your Lab Report: Path A path describes the location of a file in a computer's storage. What is the path to the Doorbell folder that you created, stored on your Raspberry Pi?

For Your Lab Report: User Interface How does the user interface for the version of Linux on the Raspberry Pi resemble that of other operating systems with which you are familiar? How is it different?

5. TEST THE CAMERA

Locate the LXTerminal application's icon on the Raspberry Pi desktop. Click on the icon to open a window running LXTerminal. You can type commands into the LXTerminal app to run programs on the Raspberry Pi.

Type one of these commands to instruct the Raspberry Pi to use its camera to take a picture. Be careful to include spaces after the command name and the letter after each option specified by a minus sign (-). Be sure to smile!

- If you are using the built-in Raspberry Pi camera, type this command all on one line in the terminal window to take a picture: *raspistill -h 300 -w 400 -n -o /home/pi/Desktop/Doorbell/webcam.jpg*
- If you are using a USB webcam, type this command all on one line in the terminal window to take a picture: *fswebcam -r 960x720 -d /dev/video0 /home/pi/Desktop/Doorbell/webcam.jpg*

Look in the Doorbell folder and click the webcam.jpg file to view the picture. Make sure it has the photo you just captured.

If you receive a permission denied error when running either webcam command, try typing it again, and preceded with the keyword *sudo.* In Linux, *sudo* ("super user do") is a program that gives you permission to run the command that follows with "super user" (often higher) privileges.

For Your Lab Report: Camera Where is the picture stored? What happens to the picture that was stored previously after you take a new one?

6. CODE CHECK: wait_doorbell.py

In this step you will examine the source code from wait_doorbell.py, shown in Appendix 1, Figure 5, to get a sense of how it works. You can open this file in a text editor app installed on the Raspberry Pi.

The wait_doorbell.py file is a program written in the Python programming language. Its job is to wait until someone presses the buzzer on the doorbell, and when that happens, take a photo and send it in an email message. The program runs forever. The only way to stop it is to press the Control key and the C key (written as Control-C) at the same time.

The code statements to take the photo (19 or 22) are *commented out* – that means, they have a # symbol at the beginning which tells the computer to ignore those lines. As part of the exercise you will remove the # symbols from one of these lines so that the program runs them. Be careful not to change the indentation of any of the lines of program code.

For Your Lab Report: wait_doorbell.py Which line of the wait_doorbell.py program causes it to run forever until you press Control C? Which line's statement checks to see if somebody pressed the buzzer? What does the program do when that happens?

7. CODE CHECK: sendnotify.py

In this step you will examine the source code from sendnotify.py shown in Appendix 1, Figure

7 to get a sense of how it works. You can open this file in a text editor app on the Raspberry Pi.

The sendnotify.py program contains steps to set up the contents of an email message, and then send that message. The program calls sendMail (in line 51) to send an email message with an attachment to a recipient. Lines 14-46 specify the steps for the Raspberry Pi to send an email message from a sender's email address. Lines 55-59 specify the content of that email message.

For Your Lab Report: sendnotify.py In addition to a sender's email address, what four pieces of information are necessary for the sendnotify.py program to send an email message to a recipient? Where does the program get this information?

8. TEST THE CONNECTION BETWEEN THE DOORBELL AND THE RASPBERRY PI

In this step you will test the connection between the doorbell receiver and the Raspberry Pi.

1. Open the wait_doorbell.py file in a text editor. (The text editor app has an icon on the desktop.)
2. Change line 6 to replace 999 with the GPIO number of the pin receiving the positive charge. Look at Figure 1 to find the GPIO number. (This is not the pin number!)
3. Save this change in the text editor.
4. In an LXTerminal window, type *cd* (for change directory) and a space, followed by the path to the Doorbell folder that you created. Press the ENTER key to complete the command. This will change directory in the command window so you can directly run the programs in the Doorbell folder.
5. Type the command *sudo python wait_doorbell.py* and press the ENTER key to run the program. This command instructs the Raspberry Pi to run the Python program whose code is in the file wait_doorbell.py.
6. What do you see on the screen in the terminal window when the program runs?
7. Press the button on the doorbell buzzer. (It is possible and likely that your buzzer may cause several receivers in the room to chime if they operate on the same frequency. If this happens, press the sound button on the back of the receiver unit.) What information displayed in the terminal window changes when you sound the doorbell?
8. The program will run forever. Press the Control key and the C key at the same time (Control C) to end the program and return to the terminal prompt.

For Your Lab Report: wait_doorbell.py Terminal Window When running the wait_doorbell.py program, what does the terminal window show when you are not pressing the buzzer? How does the contents of the terminal window change when you press the buzzer?

9. TEST SENDING AN EMAIL MESSAGE WITH A PHOTO ATTACHED

If the values in the terminal window change when you press the doorbell's buzzer, this should convince you that the Raspberry Pi is receiving input from the doorbell's receiver through the wires you connected to it. Now you will configure the sendnotify.py program to send an email message containing the photo as an attachment.

1. Open the file sendnotify.py in another text editor window.
2. Replace the values for USERNAME and PASSWORD in lines 11 and 12 to show the sender's username and password. This will be your doorbell email sender address and password provided by your instructor.
3. Select one member of your group as the person to receive an email notification. Replace the recipient's email address at the bottom of the program the recipient's personal Gmail address.
4. Save the changes to send_notify.py.
5. The # symbol starting a line of code in a Python program indicates a comment. The program will skip that line when it runs. In the wait_doorbell.py program, remove the # symbols at lines 19 or 22 (depending on which type of camera you are using).
6. Remove the # symbol starting line 26 to instruct wait_doorbell.py to run the sendnotify.py program. Also remove the # symbol starting the line that prints "Photo captured." This will instruct the program to print a notification message when the camera takes a photo.
7. Save these changes to the wait_doorbell.py file.

These steps will verify that the program sends, and the user receives email messages.

For Your Lab Report: Commented Out What does the statement at line 19 or 22 that is not commented out, do?

On another computer or mobile device, one person in your group should sign into Gmail using the doorbell email sender address and password, provided by your instructor. *Please note: If you already signed into Google with your own account because you are working on the lab report, then you will need to sign in using a different browser or using private browsing, so you can access your account and the doorbell email account in its own browser window.*

Delete all sent mail from the doorbell sender account, so you can determine easily if the program is sending mail when you run the program. Navigate to the SENT folder and monitor it when you run the program.

The program on the Raspberry Pi should be able to send email messages from the doorbell sender Gmail address because security settings for that address are set to allow apps to send email messages. *Please note: If it is not working, you may need to check the settings on that account. From the doorbell sender Gmail account, visit https://www.google.com/settings/security/lesssecureapps and turn on access for less secure apps.*

On the recipient's own computer or mobile device, the recipient should sign into his or her personal Gmail at the address specified to check if the email messages from the doorbell email account have arrived.

Navigate to the recipient's INBOX folder in Gmail and monitor any new messages that may appear when you run the program.

Click in the LXTerminal window, press the UP ARROW key to reissue the *python wait_doorbell.py* command (or retype it) to run the program and press ENTER.

10. TEST THE DOORBELL CONNECTION

With the wait_doorbell.py program running, press the buzzer on the doorbell.

Check the sender's sent mail and the recipient's received email. How many messages were sent when you ring the bell? If you received several messages, try adjusting value specified in the time delay of line 6 of wait_doorbell.py.
Check the sent mail and received mail accounts to see if the program sent an email message. Take a look at the picture.

For Your Lab Report: What values do you see in the LXTerminal window when the wait_doorbell.py program runs? What information does the program display in the terminal window before and after you press the buzzer? When does a message appear? Why?

11. SEND A TEXT MESSAGE

To send a text message via email, you must use a SMS (simple messaging service) to email gateway. By sending an email message to an address containing the mobile number of the recipient, the gateway will forward the email-message as a text message. Substitute a 10-digit cell number for 'number' for each carrier's name. Table 1 shows email gateways for popular cellular carriers. If you have a different carrier, use a search engine to find the gateway email address for your carrier.

Table 1. Carriers and Email Gateway Addresses

Carrier	Email Gateway Address
AT&T	number@txt.att.net
T-Mobile	number@tmomail.net
Verizon	number@vtext.com
Sprint	number@messaging.sprintpcs.com

Copy the code at lines 55 to 58 of sendnotify.py that sends an email message to a recipient. Paste it at the bottom of the file, after the existing code. Replace the recipient's email address with an email address containing your ten-digit phone number (with no dashes, spaces, or parentheses) and the correct email gateway for your mobile provider.

Run send_notify again, press the doorbell buzzer, and see if you receive both an email message and a text message on your phone. *Please note: Some carriers may not forward the attached photo, but will send the text message.*

For Your Lab Report: Add these screenshots to your lab report:

- The email message sent from the sender's Gmail account
- The email message and attached photo received in the recipient's personal Gmail account
- The text message received on the recipient's mobile device

12. CLEAN UP

Complete these steps to remove the files you created, then disconnect and power off the Raspberry Pi.

1. Delete the Doorbell folder on the Raspberry pi desktop containing the files you modified and the webcam photo.
2. Power off the Raspberry Pi.
3. Disconnect the wires, power supply, and connected devices from the Raspberry Pi. Disconnect the clip wires from the doorbell. Place all of the parts back in the plastic bag as you received it.

For Your Lab Report: (To be answered individually) What was the most difficult part of this lab activity for you? How did this activity help you learn about the Internet of Things? What did you learn?

For Your Lab Report: (To be answered individually) What everyday items have you seen or would you like to see connected to the Internet? What does or would connecting those items to the Internet enable you to do?

Review and discuss the answers with your group, and share the Google Doc with the instructor at the Gmail address provided.

13 ACKNOWLEDGEMENTS

The author acknowledges Tejas Shah, MSIT student, who helped debug this activity; Prof. Bill VanderClock, whose adeptness with a soldering iron increased durability of the doorbell equipment and simplified the activity for students; and Jake OConnell and Vivek Dak, CIS tutors at Bentley University, who prepared a video demonstration of this lab activity, available at https://youtu.be/cgYpnU-pdnI.

14. REFERENCES

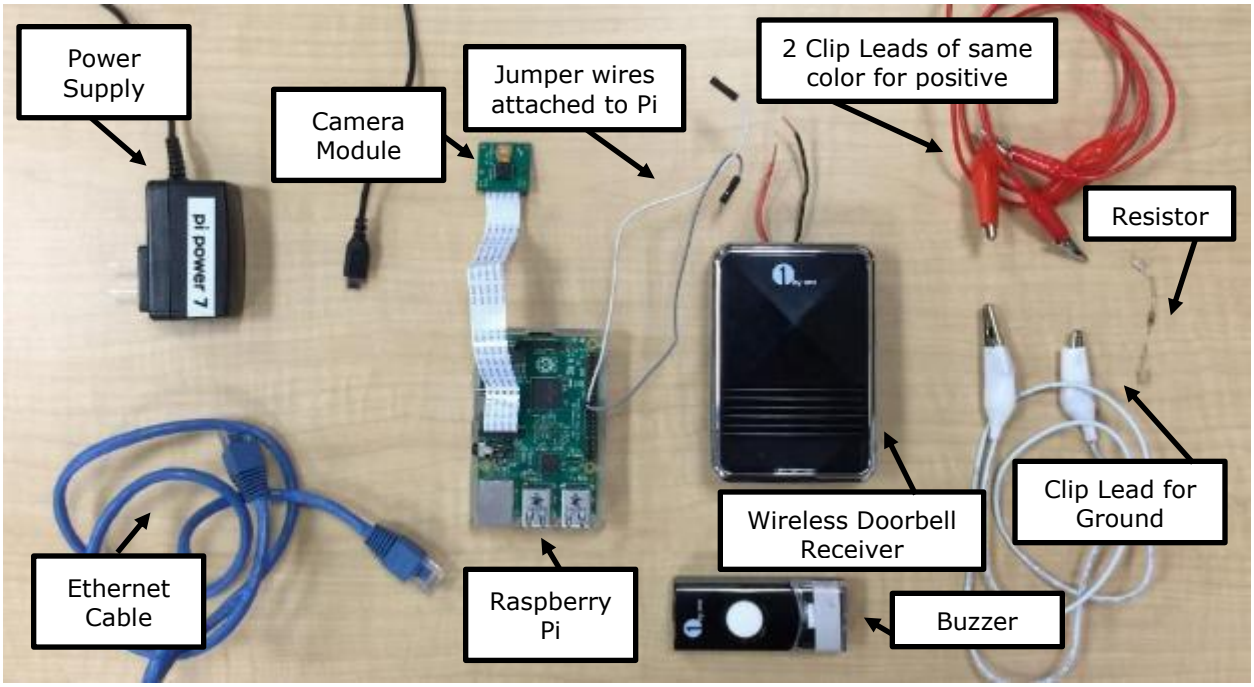
Harizanov, M. (2013, July 17). *Raspberry Pi email/SMS doorbell notifier + picture of the person ringing it*. Retrieved from Martin's Corner on the Web: http://harizanov.com/2013/07/raspberry-pi-emalsms-doorbell-notifier-picture-of-the-person-ringing-it/

Sims, G. (2014, January 17). *How to Set Up Wi-Fi on a Raspberry Pi*. Retrieved from Make Tech Easier: https://www.maketecheasier.com/setup-wifi-on-raspberry-pi/

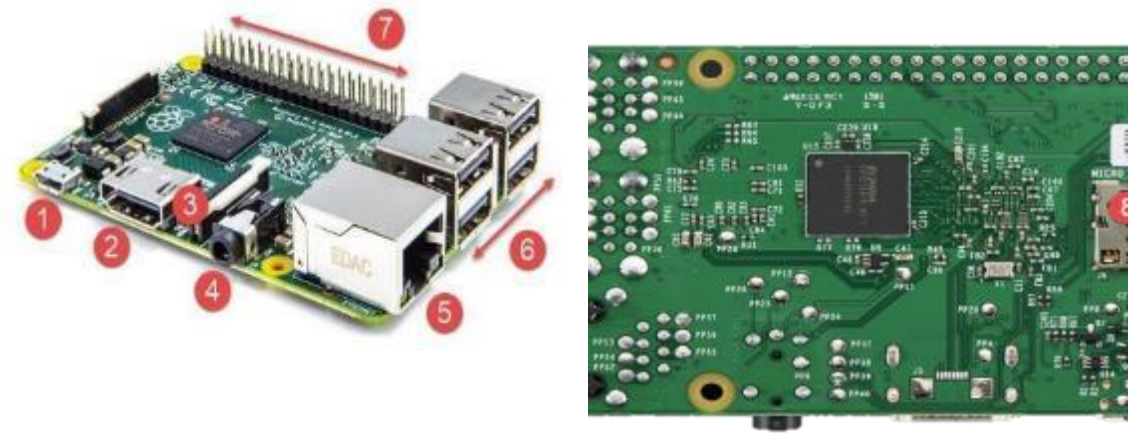
Editor's Note:

This paper was selected for inclusion in the journal as the EDSIGCON 2016 Best Teaching Case. The acceptance rate is typically 2% for this category of paper based on blind reviews from six or more peers including three or more former best papers authors who did not submit a case in 2016.

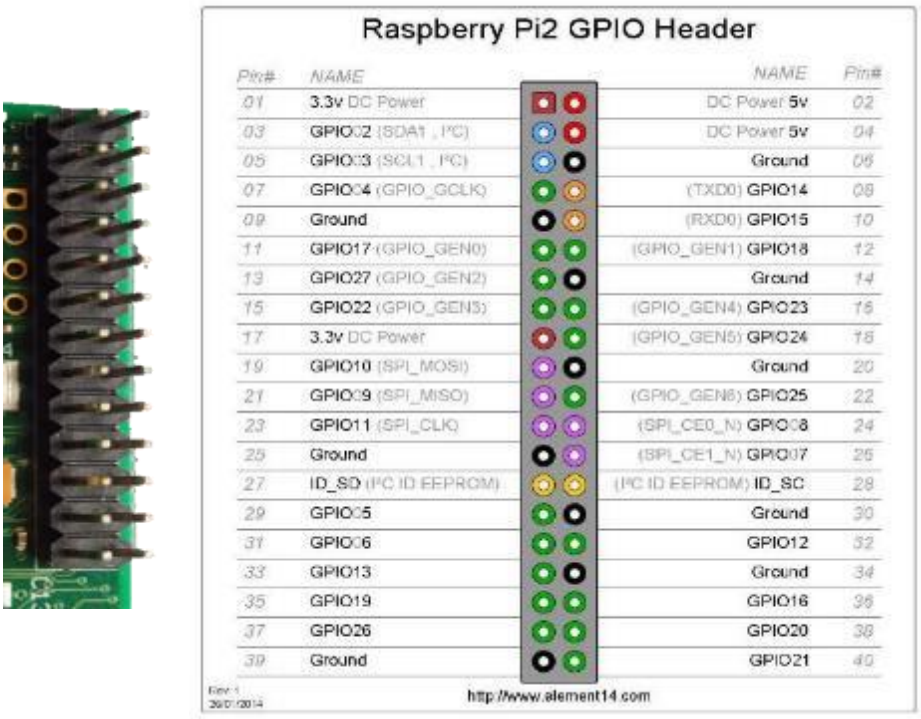
Appendix 1. Additional Figures



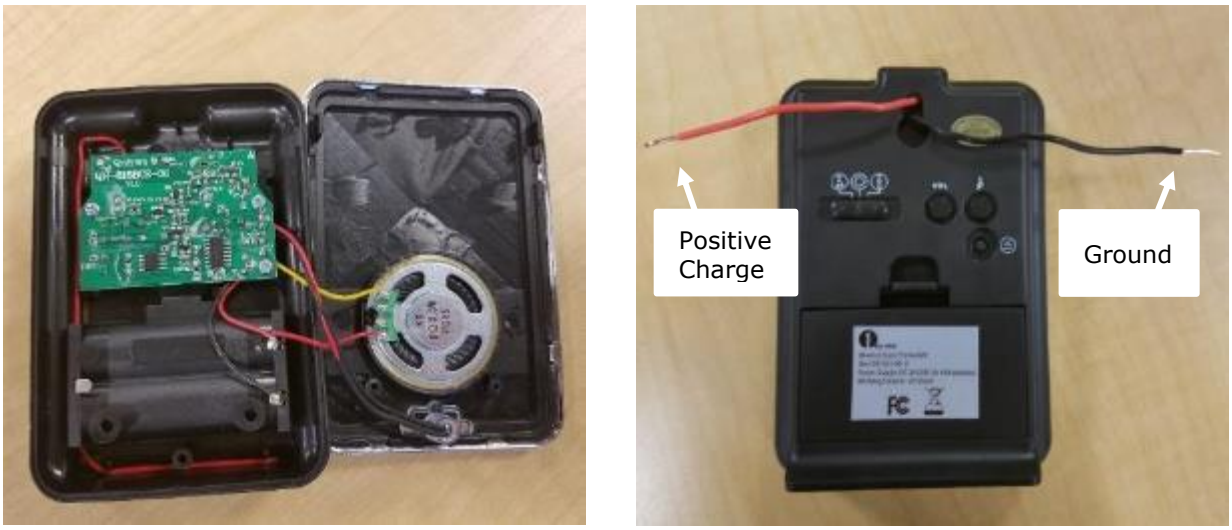
Appendix 1, Figure 1. Doorbell Lab Kit



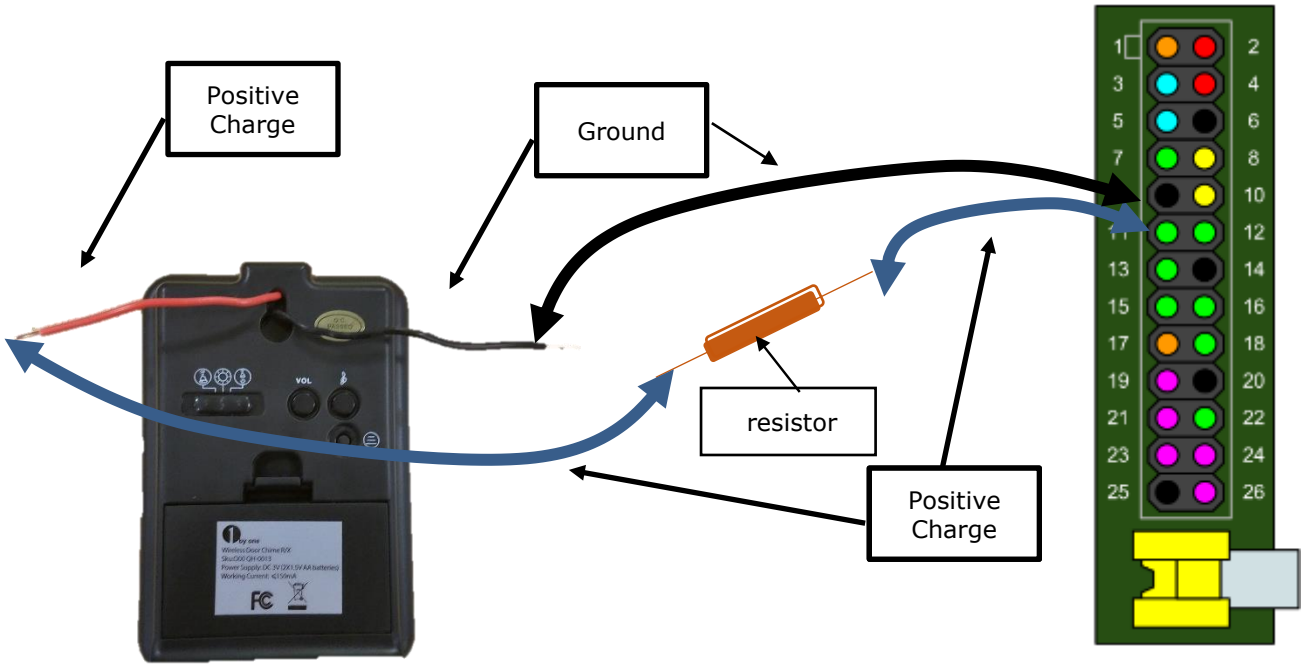
Appendix 1, Figure 2. Raspberry Pi Ports and Pins.



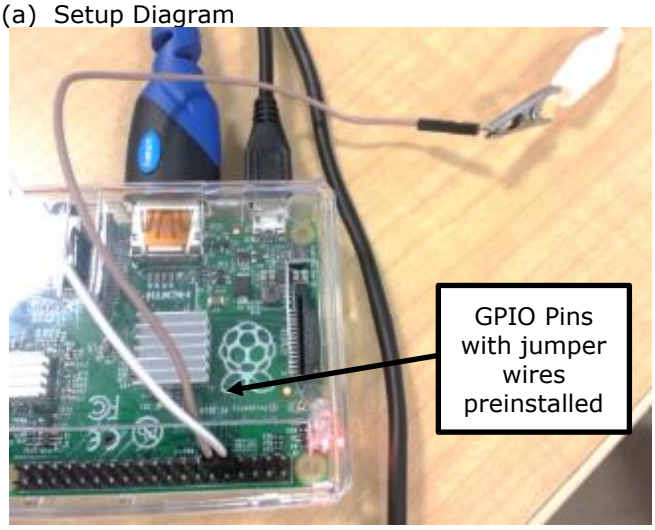
Appendix 1, Figure 3. Raspberry Pi GPIO Pins. (Diagram reference: http://element14.com)



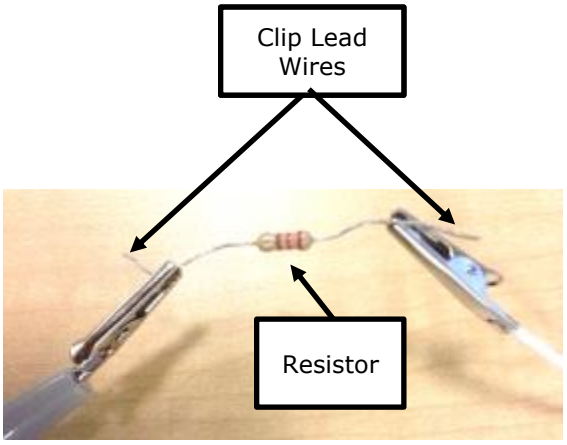
Appendix 1, Figure 4. Inside a wireless doorbell receiver. Soldered wires connect to ground (black) and positive charge (red).



(a) Setup Diagram

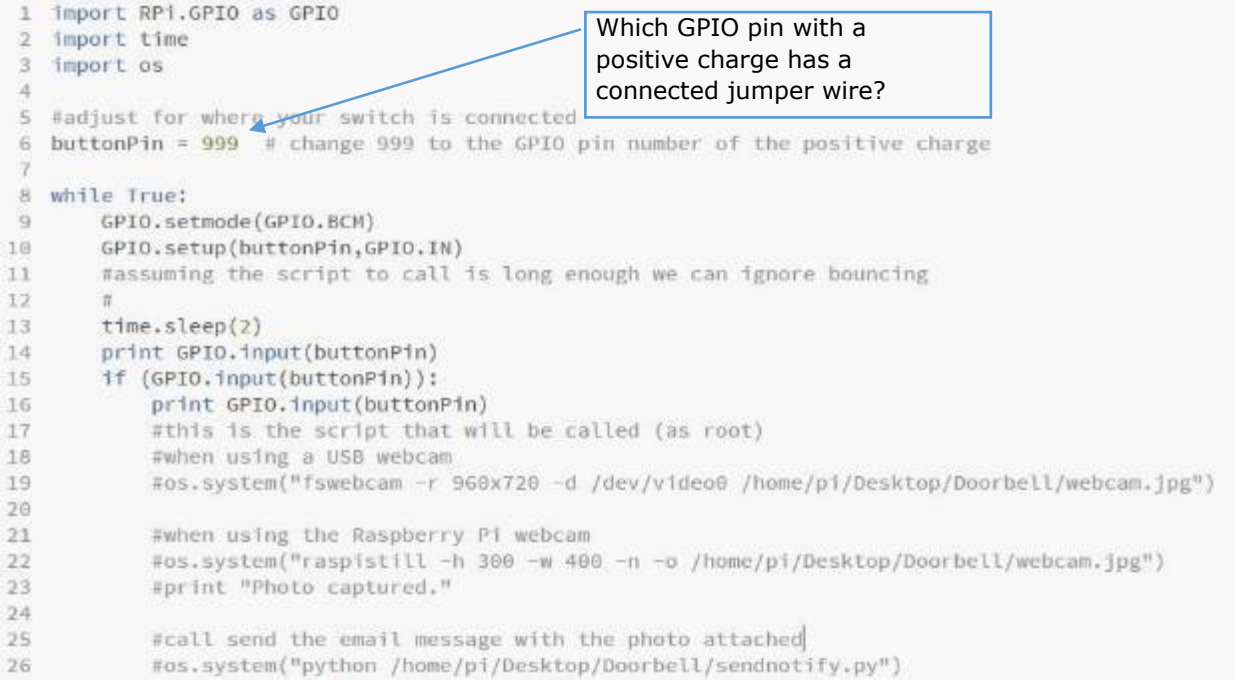


(b) Wires connected to GPIO pins of Raspberry Pi



(c) Clip leads connected to a resistor

Appendix 1, Figure 5. Connecting the doorbell to the Raspberry Pi.




```
import RPi.GPIO as GPIO
import time
import os

#adjust for where your switch is connected
buttonPin = 999  # change 999 to the GPIO pin number of the positive charge

while True:
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(buttonPin,GPIO.IN)
    #assuming the script to call is long enough we can ignore bouncing
    #
    time.sleep(2)
    print GPIO.input(buttonPin)
    if (GPIO.input(buttonPin)):
        print GPIO.input(buttonPin)
        #this is the script that will be called (as root)
        #when using a USB webcam
        #os.system("fswebcam -r 960x720 -d /dev/video0 /home/pi/Desktop/Doorbell/webcam.jpg")

        #when using the Raspberry Pi webcam
        #os.system("raspistill -h 300 -w 400 -n -o /home/pi/Desktop/Doorbell/webcam.jpg")
        #print "Photo captured."

        #call send the email message with the photo attached
        #os.system("python /home/pi/Desktop/Doorbell/sendnotify.py")
```

Appendix 1 Figure 6. Code for wait_doorbell.py

```python
#!/usr/bin/env python
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email.Utils import COMMASPACE, formatdate
from email import Encoders
import os
import time

USERNAME = "doorbellemailXX@gmail.com"
PASSWORD = "password"

def sendMail(to, subject, text, files=[]):
    assert type(to)==list
    assert type(files)==list

    msg = MIMEMultipart()
    msg['From'] = USERNAME
    msg['To'] = COMMASPACE.join(to)
    msg['Date'] = formatdate(localtime=True)
    msg['Subject'] = subject

    msg.attach( MIMEText(text) )

    for file in files:
        part = MIMEBase('application', "octet-stream")
        part.set_payload( open(file,"rb").read() )
        Encoders.encode_base64(part)
        part.add_header('Content-Disposition', 'attachment; filename="%s"'
                        % os.path.basename(file))
        msg.attach(part)

    server = smtplib.SMTP('smtp.gmail.com:587')
    server.ehlo_or_helo_if_needed()
    server.starttls()
    server.ehlo_or_helo_if_needed()
    server.login(USERNAME,PASSWORD)
    server.sendmail(USERNAME, to, msg.as_string())


        print "Mail sent."

        server.quit()

# end sendMail



localtime = time.asctime(time.localtime(time.time()))


#  change to recipient's email address

sendMail( ["recipient@gmail.com"],
        "Doorbell notification",
         localtime +": " + "Someone is ringing the doorbell, picture attached",
        ["/home/pi/Desktop/Doorbell/webcam.jpg"] )

```

Type doorbell email sender address and password here

Type recipient email address here

Appendix 1 Figure 7. Code for sendnotify.py