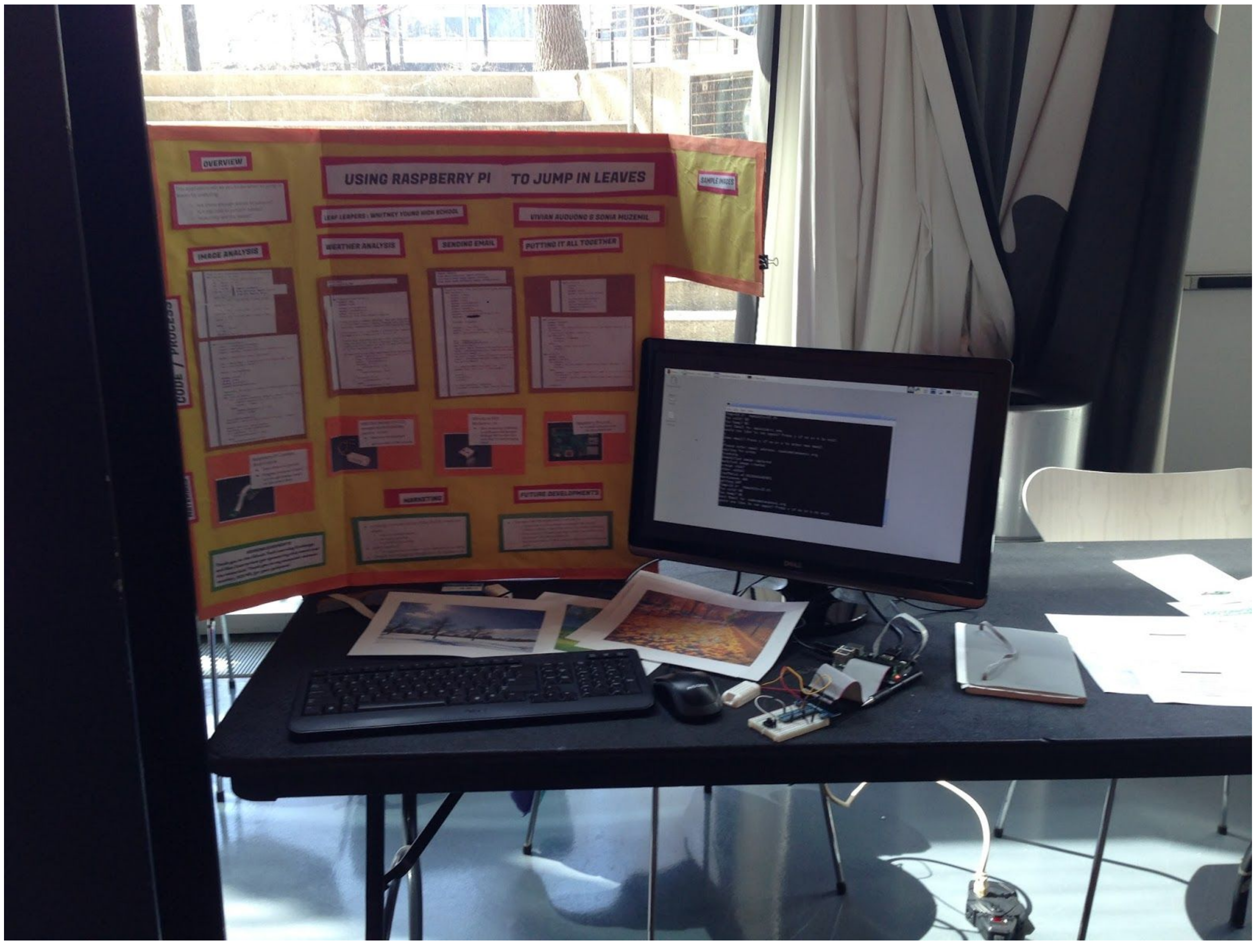


Raspberry Pi Leaf Jumping Application Writeup

Presented at Raspberry Pi BuildIT Student Challenge, April 26th 2016

Awards: 1st Place, Sensor Category

Repository: https://github.com/vauduong/Leaf-leaping

Overview

1. Introduction
2. List of materials
3. Are there leaves?
    a. Camera
    b. Python Imaging Library and the Leaf Concentration Algorithm
4. Is it warm enough? Dry enough?
    a. Temperature-Humidity Sensor
    b. Forecastio, pyzipcode
        i. Sudo apt-get requests[security] --upgrade
5. Sending a Notification
    a. Smtblib
    b. Wifi module
6. Next Steps
    a. Slipping on Ice
    b. Remote Access
7. Conclusion and Acknowledgements

1. Introduction

Jumping in leaves is one of my favorite fall activities, and I'm sure others share the sentiment. However, with such busy schedules, it's easy to forget to jump in leaves. Whenever we do remember, we might find that conditions are suboptimal. There might not be leaves at all, it might be too cold to enjoy it properly, or the leaves might be so damp that we're met with disappointing squishes instead of crisp fun.

This application will let you know when to jump in leaves by analyzing the following questions:

- Are there enough leaves to jump in?
- Is it warm enough to jump in leaves?
- Are the leaves crisp enough for maximum enjoyment?

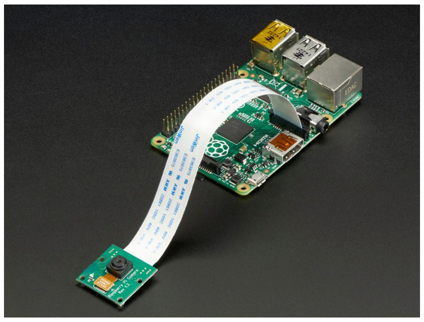
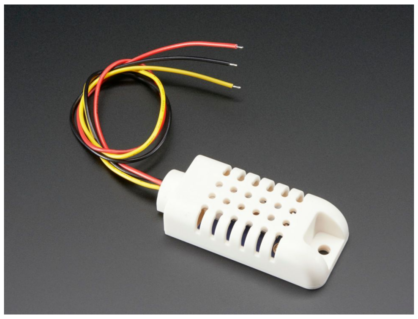
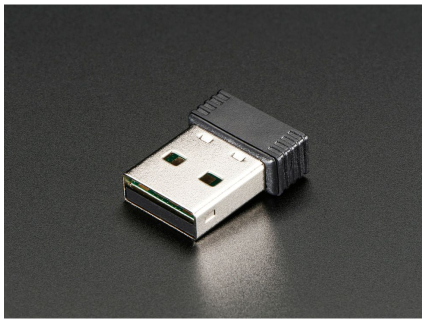
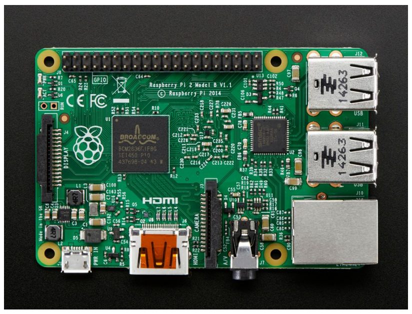
Then, the application will send you an email to let you know that optimal leaf jumping conditions are met!

It's mostly just an application to have fun, but if you don't like fun, you can also use it to tell you when to rake your lawn.

The project was created as a result of entering the Raspberry Pi Student Challenge hosted by the Illinois Tech Learning Exchange. In November 2015, I pitched the project as a silly idea. To my surprise, I won 1st place in the PitchIT competition for the sensor category, along with $500 to actually build my project! After working on building the project, I presented the prototype in April at the BuildIT competition, where it also won 1st place and a 1-week tech camp for me to learn web design.

This is my first time doing a writeup, so please bear with me! Sections are organized by the "big steps" (leaf analysis, weather analysis, notification) the application takes, and describe the process of creating the application. Note that modules are downloaded frequently, and usually the module can be obtained by entering "sudo pip install 'module-name'", but there are occasions when the module goes by a different name. If you get an error, try googling and finding the correct command to enter in the terminal prompt.

2. List of materials

Item	Description
	Raspberry Pi Camera Board $29.95 -Takes picture of ground, Program analyzes if there are enough leaves based on the pixel colors
	AM2302 (wired DHT22) temperature-humidity sensor $15.00 -Measures temperature and humidity
	Miniature WiFi Module $11.95 -After analyzing conditions, an email will be sent through WiFi to alert the user that "It's leaf jumping time!"
	Raspberry Pi $39.95 -A small computer that can do a whole lot! This is what puts everything together.
Keyboard, Monitor, Mouse	These are needed to program with the raspberry pi, and were used to demonstrate the project.

3. Are there leaves?

In order to answer this question, I wrote a pictureAnalysis() function. This is a rather simple algorithm, analyzing the concentration of leaves rather than trying to identify individual leaves and counting them. The function counts the number of orange pixels (denoted leaves) and compares it to anything that isn't orange.

```
def pictureAnalysis():
	takePicture()
	analyzeImage()
```

a) Camera

The camera for raspberry pi uses the picamera module. I also added a button to make the demonstration more interactive.

```
import picamera
from gpiozero import Button
def takePicture():
	#takes picture using camera and saves it as unmodified.jpg
	#I set the resolution to 120x120 pixels because this was a good
	balance between getting the details of an image and not taking
	too much time to analyze.
	camera = picamera.PiCamera()
	camera.resolution=(120,120)
	camera.capture('/home/pi/BUILDIT/unmodified.jpg')

	#lets the user know that the image was captured and saved
	print ("Unmodified image captured")
```

b) Python Imaging Library and converting and image to an array

Python Imaging Library (PIL) was used to analyze the images and count the orange to other pixels. However, it's important to note that PIL is outdated. The maintained version is Python Pillow, which can be obtained by running "sudo pip install Pillow".

Python Pillow for raspberry pi requires the prerequisites that can be installed with the command: "sudo apt-get install libtiff4-dev libjpeg8-dev zlib1g-dev libfreetype6-dev liblcms1-dev libwebp-dev tcl8.5-dev tk8.5-dev"

```
from PIL import Image
```

```
#initializing variables
other = 0
orange = 0
leafRatio= 0.0
def analyze():
     #PIL has commands which open an image and convert it into an
     array of RGBA points

    img = Image.open('/home/pi/BUILDIT/unmodified.jpg')
    img = img.convert("RGBA")
    datas = img.getdata()
```

c) RGB to HSV

```
#rgb to hsv conversion from Michael Fogleman on activestate code
def rgb2hsv(r, g, b):
    r, g, b = r/255.0, g/255.0, b/255.0
    mx = max(r, g, b)
    mn = min(r, g, b)
    df = mx-mn
    if mx == mn:
        h = 0
    elif mx == r:
        h = (60 * ((g-b)/df) + 360) % 360
    elif mx == g:

        h = (60 * ((b-r)/df) + 120) % 360
    elif mx == b:
        h = (60 * ((r-g)/df) + 240) % 360
    if mx == 0:
        s = 0
    else:
        s = df/mx
    v = mx
    return h, s, v
```

d) Image analysis and finding the leaf concentration

```
#Let python know that you want the global variables
```

```
    global orange
    global other
    global leafRatio
    global haveLeavesStatus

     #Modified "Using pil to make all white pixels transparent" code
from cr333 on stackoverflow
     #shows which pixels are orange in new image called Modified.jpg
    newData = []
    for item in datas:
     #converts each pixel’s rgb value to hsv and checks the hue and
     #saturation for “orange” pixels
            if (((rgb2hsv(item[0], item[1], item[2]))[0]) <70)  and
     (((rgb2hsv(item[0], item[1], item[2]))[2])>0.7):
                newData.append((255, 255, 255, 0))
                orange = orange + 1
            else:
                newData.append((0,0,0,0))
                other = other + 1

        img.putdata(newData)
        img.save("/home/pi/BUILDIT/modified.jpg", "JPEG")

        print("modified image created")
        leafRatio = float(orange/other)
        if (leafRatio > 0.3):
            haveLeavesStatus = "YES"
        else:
            haveLeavesStatus = "NO"
```

Is it warm enough? Dry enough?

a. Temperature-Humidity Sensor

```
import Adafruit_DHT
import smtplib
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart



temp = 0
humid =0
timesRun = 0
button = Button(5)
toAdd = ""
haveLeavesStatus = ""
tooColdStatus = ""
tooDampStatus = ""

def temperatureAnalysis():
    global temp
    global humid
    global tooColdStatus
    global tooDampStatus
    print("getting DHT")
    #Humidity code from Adafruit website

    # Try to grab a sensor reading.  Use the read_retry method which
will retry up
    # to 15 times to get a sensor reading (waiting 2 seconds between
each retry).
    humidity, temperature =
Adafruit_DHT.read_retry(Adafruit_DHT.AM2302, 4)

    # Un-comment the line below to convert the temperature to
Fahrenheit.
    # temperature = temperature * 9/5.0 + 32
    # Note that sometimes you won't get a reading and
    # the results will be null (because Linux can't
    # guarantee the timing of calls to read the sensor).
    # If this happens try again!

    if humidity is not None and temperature is not None:
            print 'Temp={0:0.1f}*
Humidity={1:0.1f}%'.format(temperature, humidity)
            temp = float(temperature)
            humid = float(humidity)
            if(temp> 20.0):
                tooColdStatus = "NO"
            else:
                tooColdStatus = "YES"

            if(humid> 50.0):
                tooDampStatus = "YES"
            else:
                tooDampStatus = "NO"

            print("Too cold? " + tooColdStatus)
            print("Too Damp? " + tooDampStatus)
    else:
            print 'Failed to get reading. Try again!'
```

Forecastio, python, pyzipcode

i.

ii. This was a modification made after the competition. I realized there were too many factors pointing against using the temperature-humidity sensor and placing the raspberry pi outside.

1. Minimizing water damage requires a weatherproof enclosure, which renders the humidity function of the sensor useless.
2. Where would I put the enclosure and raspberry pi? As I walked around and surveyed my yard, I struggled to find a place that offered a good view of the ground and could allow for an enclosure to be secured.

3. Nature is unpredictable. Even if I managed to secure the enclosure and build a frame to secure the camera within the enclosure, my efforts may not be enough for particularly strong wind, rain, or curious squirrels. It was entirely possible that the camera could be jostled or the raspberry pi could vanish.
4. Sending power to the raspberry pi was another problem. While working on the application, power was supplied through a micro-USB adapter which connected to your standard American power outlet. I looked into solar panel and battery options, which seemed possible but also stated that it was *extremely* necessary to regulate power consumption. According to voltaicsystems.org, a seller of solar chargers, my raspberry pi B+ model has a current consumption of 480mA and therefore a daily power consumption of 57.6Wh, while their recommended $159 9W solar panel produces 32Wh assuming 6 hours of bright sunlight (which isn't even guaranteed!).
5. Solar chargers were way out of my budget. For the convenience of the sponsors, I had to send a list of the items under the $500 budget just a week after the fall challenge in November, and I hadn't considered solar power at that point. My original plan was to rely on much cheaper power banks or power adapters, but I realized later that changing power banks was impractical, while wires leading out of the power adapters wouldn't work with the airtight project enclosures.

iii. As a result, I ditched the temperature-humidity sensor in favor of using a python module to get the weather forecast info, forecastio. To install it, run the command "Sudo pip install python-forecastio". You'll also need an API key which you can get by making an account at developer.forecast.io.

iv. , the main component of the breadboard. This also gave me the opportunity to use the pins for something other than the breadboard, and I attached a pitft screen, following the instructions to install a disk image.

v. Unfortunately, I hadn't known that installing a disk images completely rewrites *everything!* Luckily, I saved my code on a flash drive, but spent a long time reinstalling modules.

vi. Forecastio uses latitude and longitude, which most people don't know off the top of their head, so I decided to use the python module pyzipcode to convert zip codes to these lat/long values. Install pyzipcode by running "sudo pip install pyzipcode". Pyzipcode requires sqlite3, which can be installed by running “sudo apt-get install libsqlite3.dev” You might also run into the python insecure SSL issue, which can be fixed by installing the following packages: “sudo pip install requests[security] --upgrade”, “sudo apt-get install libffi-dev libssl-dev”, and “sudo pip install pyopenssl ndg-httpsclient pyasn1”. A huge thanks to helloflu.wordpress.com for writing a guide on this!

Sending a Notification

b. Smtblib

```
#sending email code modified from Gaven MacDonald on youtube and
user1292828 on stackoverflow
def sendEmail():
    global orange
    global other
    global toAdd
    global tooColdStatus
    global tooDampStatus
    global leafRatio
    global haveLeavesStatus
    smtpUser = 'wyleafleapers@gmail.com'
    smtpPass = '***************************'

    fromAdd = smtpUser

    subject = 'Leaf Data'
    header = 'To: ' + toAdd + '\n' + 'From: ' + fromAdd + '\n' +
'Subject: ' + subject
    body = 'Concentration data:' + '\n' + "orange =" + str(orange) +
'\n' + "other =" + str(other) + '\n' + "leaf ratio =" + str(leafRatio)
+ '\n' + "temp= " + str(temp) + " humidity=" + str(humid) + '\n' +
"Enough Leaves? " + str(haveLeavesStatus) + '\n' + "Too cold? " +
str(tooColdStatus) + '\n' + "Too damp? " + str(tooDampStatus)


    msg = MIMEMultipart()
    text = MIMEText(str(body))
    msg.attach(text)
    img_data= open('/home/pi/BUILDIT/modified.jpg', 'rb').read()
    image = MIMEImage(img_data,
name=os.path.basename('/home/pi/BUILDIT/modified.jpg'))
    msg.attach(image)
    img_data = open('/home/pi/BUILDIT/unmodified.jpg', 'rb').read()
    image2 = MIMEImage(img_data,
name=os.path.basename('/home/pi/BUILDIT/unmodified.jpg'))
    msg.attach(image2)
```

```
s = smtplib.SMTP('smtp.gmail.com', 587)
print("Sent Email to: " + str(toAdd))

s.ehlo()
s.starttls()
s.ehlo()

s.login(smtpUser, smtpPass)
s.sendmail(fromAdd, toAdd, msg.as_string())

s.quit()
```

c. Wifi module

Putting It All Together!

```
import os, sys
import math
def action():
    orange = 0
    other = 0
    global button
    print ("waiting for press")

    button.wait_for_press()
    print ("Scanning...")
    pictureAnalysis()
    temperatureAnalysis()
    sendEmail()

def restart():
    global timesRun
    global toAdd
     restart = raw_input("would you like to run again? Press y if so or
n to exit."  + '\n')
    if restart == "n":
       sys.exit("Goodbye, thank you!")
    if restart == "y":
        timesRun = timesRun + 1
        script()
    else:
        print("please type y or n")
        restart()

def script():
    global toAdd
    global timesRun
    if(timesRun < 1):
        toAdd = str(raw_input("Please enter email address: "))
        action()
    else:
            address = raw_input("Same email? Press y if so or n to enter
new email." + '\n')
           if address == "n":
               toAdd = str(raw_input("Please enter email address: "))
           action()

       restart()







script()
input("press any key to close the window")
```

Next Steps

d. Slipping on Ice
e. Remote Access

Conclusion and Acknowledgements

Thank you to Rico Enterprises and the Illinois IT Learning Exchange for sponsoring the Raspberry Pi Challenge and

Raspberry Pi Fall Challenge 2015
Visit the Illinois IT Learning Exchange website:
illinoistechlex.org
INFORMATION TECHNOLOGY
for updates on the Spring 2016 Raspberry Pi Student Challenge!
5:05 / 5:12

This was my first time working on such a project, and I learned a lot about downloading modules, trial and error, and how fast open source changes! I would make improvements on my leaf leaping application periodically over the summer, and find to my surprise that “sudo apt-get install --update”, used for updating packages, would take a few minutes to run each time.

There was quite a bit of frustration involved, as it took me hours to figure out that Python Pillow required prerequisites, and then also to figure out what those prerequisites were! As it turns out, this was a pretty big lesson in development environments. Stackoverflow is a community of people using all sorts of platforms to run the same module, so it may not always have the answer you’re looking for. I eventually found the solution in a single line of a section in the documentation on pypi which described prerequisites for different operating systems. Dependencies and deprecation were definitely words I learned through the process.

It was also quite the beneficial activity to do this write-up, as I learned how to better explain code and organize my thoughts. I actually went back in my program and made my code neater as a result!

Finally, I learned about the constraints in projects. If I had an unlimited amount of time and money, I could tinker around with using the raspberry pi outside.