School of Computer Science

Assessment Cover Sheet

Student

Student name	Student number
Ryan Hafizh Indrananda	10852565

Class

Tutor	Dakshi Kapugama Geeganage
Practical Day, Time & Room	Tuesday, 11am, GP VIRT-OL66 (Virtual)

Link to Video Demonstration: https://youtu.be/wj8eE5dgzlM

Statement of Contribution

This project was done on an individual basis. All extra components and research were acquired and done individually. No other physical external help was obtained.

Project Objectives

Utilizing a Raspberry Pi backed by the knowledge obtained throughout this semester, an endeavour to create a bot to automate tweets on the Twitter social media platform was ideated. As creating such a bot would bring about the question of triviality of what would be the subject of tweeting, the bot is to be designed to tweet content obtained from an external source, both on a software and hardware level.

Music is a driving force in life. Whatever mood, whatever circumstance, however the pressures of life may stack up, music has incredibly powerful effects. With the realization of this sentiment and after some deliberation, the bot was thus designed to act as a sort of music diary; tweeting on a specified basis or time increment the music a user is listening to, or had been listening to, in that moment. This thus would require the obtaining of such information from a software source, communicating with it through the utilization of API keys. To apply external hardware into this project, the bot was also designed to implement an external webcam to document a picture of the user every time the bot tweets as well as an LED connected with a resistor to a breadboard which physically notifies the user every time a tweet containing a set search term is detected by the Raspberry Pi. The set search term will be ‘music’, to continuously provide content for the user to keep up with the trends in current, modern-day music.

In full: creating a music diary Twitter bot tweeting logs of what a user is or has been listening to accompanied by an image of said user, as well as a physical notifier of when a music-related tweet is detected on the Twitter platform.

Review and Discussion of Technologies Used

Raspberry Pi 3B+

On a hardware level, there were multiple technologies used within this project. Arguably the most important of these technologies was the Raspberry Pi, specifically the Raspberry Pi 3B+ model. While a newer model could have been utilized, it was evident that the 3B+ model consisted of ample power in terms of specifications to carry out this project at a lower price tag. This consists of a 64-bit system-on-a-chip rated at 1.4GHz, 1GB of LPDDR2 SDRAM, a 40-pin GPIO header, and 4 USB 2.0 ports among other specifications of the model. The SoC is of course to compute commands written for the bot (which will be discussed later) through the fetch-execute cycle, while the SDRAM temporarily allows our bot to store and access data and instructions to be used (Raspberry Pi, 2016).

The Raspbian operating system (OS) is the recommended OS for use on a Raspberry Pi. It comes with over 35,000 packages precompiled for use for the user. The Raspbian image was burned using Etcher with a single 16GB SD card inserted in the SD card port of the Raspberry Pi (Raspberry Pi, 2021).

The GPIO header allows for two main things. One, access the Raspbian operating system (OS) within the Raspberry Pi through SSH (secure shell protocol) with the Putty software on Windows 10. To do this, power, ground, receive, and transmit jumper cables connected to a single USB 3.0 male output were utilized to connect the GPIO header to a laptop (in this case the Asus ROG Zephyrus G15). Then, the Putty software was used by determining the serial line of the Raspberry Pi, by which after a CLI is presented to interface with the Raspberry Pi, simplifying the communication between the two operating systems. Creating a directory separate from the home directory of the Raspbian OS was done using the mkdir command, then moving to that directory was done using the cd command. An alternative to this method of communication was to directly connect peripherals such as mouse, keyboard, and monitor to the Raspberry Pi. This alternative was not carried out due to the unavailability of spare peripherals.

Breadboard, 330 ohm 1/6 watt resistor, and LED

The GPIO header also allows us to communicate with a breadboard which has a single 330 ohm 1/6 watt resistor and a 5mm LED connected to it. A breadboard is a device which allows users to create and build circuits without the need for soldering. The holes on the breadboard contain conductive metals which depending on how you connect hardware on it allows it to electronically communicate multiple pieces of hardware together (in this case the jumper cables connected to the GPIO header, the resistor, and the LED) (Sparkfun, 2014). Resistors are passive electrical components with a specific electrical resistance to complement active electrical components (Blom, 2014) such as the LED, which is a semiconductor that emits light when a current is passed through it (Poole, 2014). The connection of previously mentioned hardware does the following: detect code ran by the Raspberry Pi, executes code which detects when a Tweet containing a specific search term is tweeted, pass a current through the LED when a tweet is detected.

Webcam (Ausdom AF640) and USB ports

The USB ports provide an interface and line of communication between two hardware technologies. As previously mentioned, a webcam was utilized for this project to capture images of a subject. The webcam used is the Ausdom AF640, capable of capturing footage at 1080p at 30FPS. This webcam was connected to one of the four USB 2.0 ports available on the Raspberry Pi. Then, the Raspberry Pi can detect if the webcam has been successfully connected using the *ls /dev/video** command. This command lists down all files detected within the /dev/video directory within the Raspbian OS, and the existence of *video0* signifies that the webcam has been connected properly to the Raspberry Pi. In the code written for the Twitter bot, the *pygame* library, combating the complexity of computer systems with hierarchichal abstraction, was used to connect to the webcam, get an image, and paste said image onto the current directory which can later be used in following lines of code which will be discussed later.

Networking, HTML, API keys, and Last.fm

The Raspberry Pi was connected to the internet through wi-fi. Internet connection is specifically crucial for this project, as it will require methods of web scraping to obtain information about the user's listening history (discussed later). This was setup by configuring the *wpa_supplicant* file within the OS to identify the network SSID and password.

HTML, or hypertext markup language, is a language that describes web pages, mainly through the use of formatting markups, such as *<p></p>* to identify a paragraph within a web page. Our methods of web scraping will make use of this language; opening a website, observing the page source, and obtaining important aspects of the page using regular expressions, a method to search for character patterns on a text available as a library in Python.

A major part of this project is API keys. Application Programming Interfaces are libraries that hide the complexity of computer systems to users. Many applications, desktop and web, provide APIs do developers to enable them to develop tools through the use of data and libraries provided in said application. This application makes use of two main applications, thus access to their APIs are needed. These are, of course, Twitter and a web application called Last.fm.

Last.fm is an online music service that tracks and logs the listening history of a user. This includes previously listened to songs (defined as scrobbles on the website), albums, and artists of a user. The user can then view lists and graphs of their most listened to music over a certain period, such as the past week. The last.fm API allows developers to view a specific user's listening history log formatted as a json file using the *user.getRecentTracks* method. This method obtains a list of recent tracks listened to by a user, which is specified in a URL format given by the last.fm API documentation. If a user is currently listening to a track, a variable within the json file called '*nowplaying*' is set to true. Using this parameter, we can therefore code our web scraping Python script to differentiate between a user's currently listening to song from a user's last listened to song. (Last.fm, 2021).

<u>Design and Implementation</u>

Dissecting Code Details and How It Works 1: Tweeting music diary with images

The code first starts with a few import statements:

```
#!/usr/bin/env python
import sys
from twython import Twython
import os
import pygame
import pygame.camera
from pygame.locals import *
import re
import urllib
```

This imports important libraries and modules that will be utilized in the functioning of the Twitter bot. The *twython* library is arguably the most important import here, as it provides a wrapper for the Twitter API within Python, allowing for the query of user information and tweeting tweets with images. The *pygame* library imports will allow the ability to take images with our webcam and save said image for later use onto the current directory, in this case */home/pi/RyanTweetbot*, where all project files are stored. The last three import calls are important for our Last.fm web scraping. The *re* library enables the us to use regular expressions, which will be discussed as we go down the code. The urllib library will us to open the json file which contains the user’s listening history. The *#!/usr/bin/env python* comment on top of the code ensures the code interpreter used is the first on a user’s environment path, accounting for multiple versions of Python being installed.

```
# Initializing API keys
CONSUMER_KEY = 'VSCpiXrN57ENtRBBNNVdhH6AM'
CONSUMER_SECRET = 'ycLCXoeQYZ7r1cRNwVAVOR6lAnUBAR5G49ZMRtJCK
bsfklcwCM'
ACCESS_KEY = '1397494044196970499-dXq1e7hVBTPF0Wll0SgBxry4qcQCjk'
ACCESS_SECRET = 'JJcxr4EyzHMgCGPpnNJ9bMMcbfu8yasXig2JPB8AXioGm'
api = Twython(CONSUMER_KEY,CONSUMER_SECRET,ACCESS_KEY,ACCESS_
SECRET)
```

This code block determines the API keys generated by the Twitter developer tool. For clarity, a new Twitter account under the handle @RyanTweetbot was created for our purposes. Assigning these keys under the variable ‘api’ will allow us to then tweet messages on the account with the Twython library.

```
# Last.fm web scraping
last_fm_source = 'http://ws.audioscrobbler.com/2.0/?method=user.getrecenttracks&u
ser=swishe&api_key=97f2ecff84ac3401c14d220b48c52791&format=json'
last_fm_page = urllib.urlopen(last_fm_source)
last_fm_bytes = last_fm_page.read()
last_fm_data = last_fm_bytes.decode('UTF-8')
```

```
playing_artist = re.findall('artist.*?text":"(.*?)"', last_fm_data)

try:
    playing_track = re.findall('nowplaying.*?name":"(.*?)"', last_fm_data)
    playing_album  = re.findall('nowplaying.*?text":"(.*?)"', last_fm_data)
    playing = ('Ryan is currently playing:  ' + playing_track[0] + ' by ' + pla
    ying_artist[0] + ' from ' + playing_album[0] + ' #RyanTweetbot')

except:
    playing_track = re.findall('name":"(.*?)"', last_fm_data)
    playing_album  = re.findall('album.*?text":"(.*?)"', last_fm_data)
    playing = ('Ryan\'s last played track: ' + playing_track[0] + ' by ' + pla
    ying_artist[0] + ' from ' + playing_album[0] + ' #RyanTweetbot')
```

This piece of code is what allows us to web scrape a user's listening history. It first opens the last.fm json webpage containing the listening log. It then searches for the string '.artist', passes, the string 'text":", then assigns the variable playing_artist to everything afterwards until a " is detected. The following try and except block does the following: try to find 'nowplaying' then assigns the variable playing_track to what comes after the string 'name":"' until a " is detected, then finds the playing_album through a similar process of string matching with regular expressions, then finally assigning a multiple strings and calls the first item in each list generated by the web scraping for each of song name, album, and artist into a single variable named 'playing'. If the string 'nowplaying' is not found, this means that the user is not currently listening to a song, and then pulls the same things as under the try block but altering the final 'playing' variable such that it mentions 'last played track' instead of 'currently playing'.

```
# Initializing webcam picture tweet
pygame.init()
pygame.camera.init()
cam = pygame.camera.Camera("/dev/video0",(1920,1080))
cam.start()
image = cam.get_image()
pygame.image.save(image,'webcam.jpg')
photo = open('webcam.jpg','rb')
response = api.upload_media(media=photo)
api.update_status(status=playing, media_ids=[response['media_id']])
```

This final code block calls the pygame library to take a picture with a camera then saves that image onto the current directory. It first initializes the pygame and pygame.camera modules. Then starts the camera defined as /dev/video0, which as mentioned before is the webcam connected directly into the Raspberry Pi USB port, and sets the resolution as 1080p. Afterwards, it takes an image, saves it as webcam.jpg. Once it is saved, it opens that image in binary format for reading (rb), defines it as media and assigns it to the 'response' variable. Finally, it calls the api variable with the Twitter API keys and tweets a tweet with the status set as the 'playing' and the image as the media to accompany it.

Running *sudo crontab -e* and inputting */5 * * * * python /home/pi/RyanTweetbot/RyanTweetbotv1.py* will tell the Pi to run the Python script every 5 minutes (Bruce, 2013).

Dissecting Code Details and How It Works 2: LED when tweet with search term is found

```
import time
import RPi.GPIO as GPIO
from twython import TwythonStreamer

TERMS = 'music'

# GPIO pin number of LED
LED = 25

# Twitter application authentication
CONSUMER_KEY = 'VSCpiXrN57ENtRBBNNVdhH6AM'
CONSUMER_SECRET = 'ycLCXoeQYZ7r1cRNwVAVOR6lAnUBAR5G49ZMRtJCKbsfklcwCM'
ACCESS_KEY = '1397494044196970499-dXq1e7hVBTPF0Wl10SgBxry4qcQCjk'
ACCESS_SECRET = 'JJcxr4EyzHMgCGPpnNJ9bMMcbfu8yasXig2JPB8AXioGm'
```

The start of the code is very similar to the first, whereby it imports several libraries to be used by the code, assigns some variables determining the search term and the GPIO pin connected to the LED, and initializes the Twitter API keys.

```
class MyStreamer(TwythonStreamer):
        def on_success(self, data):
                if 'text' in data:
                        print data['text'].encode('utf-8')
                        print
                        GPIO.output(LED, GPIO.HIGH)
                        time.sleep(0.5)
                        GPIO.output(LED, GPIO.LOW)

        def on_error(self, err, data):
                print err, data
```

The following class, which calls for the TwythonStreamer class within the Twython library, does either of two things. It first makes sure that there is am actual text field when scanning for a tweet with the specified search term, in this case ‘music’. When a tweet with the term is found, it prints said tweet in the console then flashes the LED for 0.5 seconds. The time increment used here is to maximize the amount of tweets found when scanning.

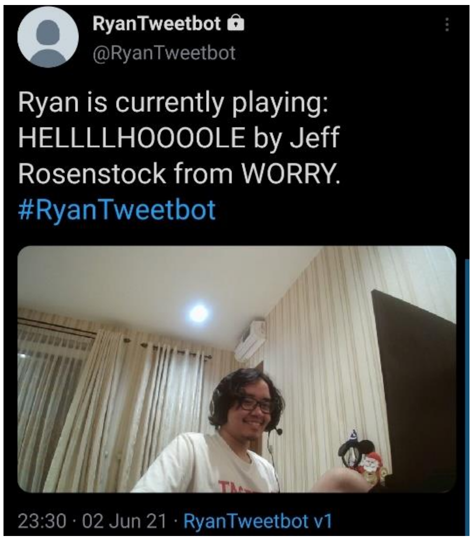
```
# Setup GPIO as output
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED, GPIO.OUT)
GPIO.output(LED, GPIO.LOW)
```

This code block sets up the GPIO. It initializes the Raspberry Pi and tells it that we are using GPIO numbering by pins (we are using pin number 25), we are using an LED and the pin directly connected to it is our output, and finally that the LED should be off initially.

```
try:
        stream = MyStreamer(CONSUMER_KEY, CONSUMER_SECRET, ACCESS_KEY, ACCESS_S
        ECRET)
        stream.statuses.filter(track=TERMS)
except KeyboardInterrupt:
        GPIO.cleanup()
```

Then the final try and except block runs the previously mentioned class and informs it of our previously defined Twitter API keys. Since 'KeyboardInterrupt' is called, this means that the code will run ad infinitum until it is manually stopped by a user (CTRL + C). The code is ultimately run by running *sudo python TweetDetector.py*, our designated name for this python script. Since our previous script to tweet has already been defined by the crontab method to run every five minutes, it will continue to do so while the TweetDetector continues to search for tweets, thus running the scripts in parallel (Hymel, 2014).

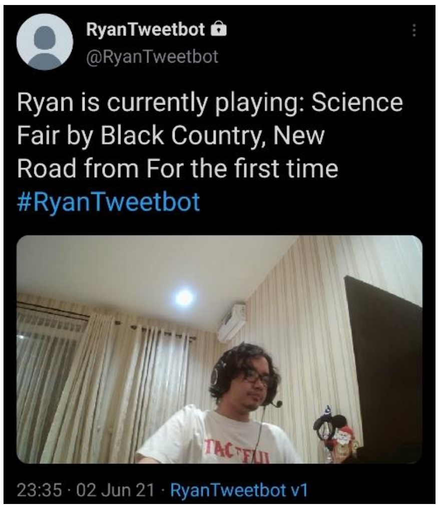
Sample Results



RyanTweetbot
@RyanTweetbot

Ryan is currently playing: HELLLLHOOOOLE by Jeff Rosenstock from WORRY. #RyanTweetbot

23:30 · 02 Jun 21 · RyanTweetbot v1

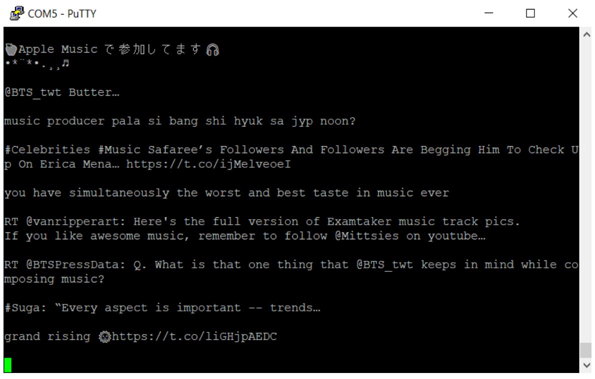


RyanTweetbot
@RyanTweetbot

Ryan is currently playing: Science Fair by Black Country, New Road from For the first time #RyanTweetbot

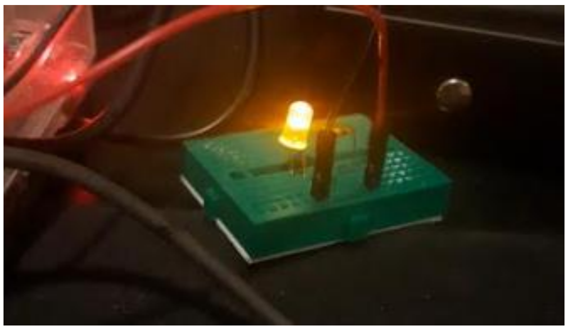
23:35 · 02 Jun 21 · RyanTweetbot v1

5 minute increment between tweets, with content being music currently being listened to and an image. Every time tweet with search term is found, the LED blinks for 0.5 seconds. The account used for this project is @RyanTweetbot.



COM5 - PuTTY

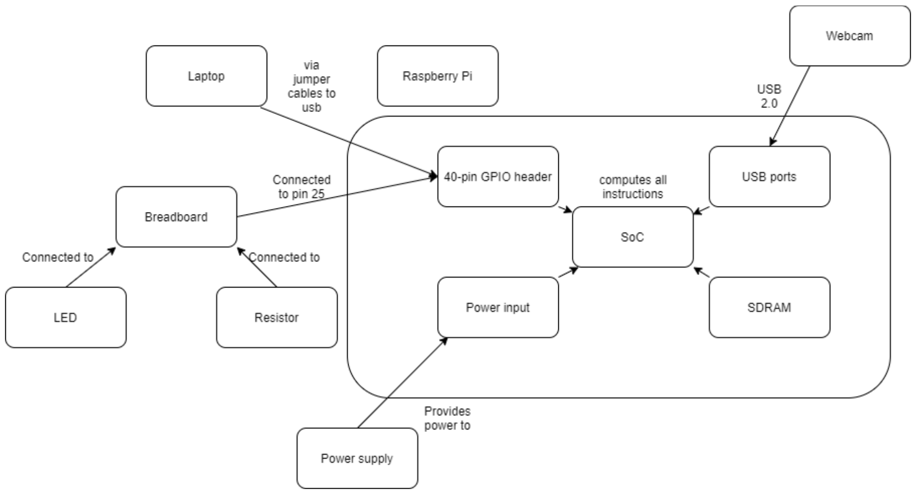
```
🍎Apple Music で参加してます🎧
•*¨*•.¸¸♬

@BTS_twt Butter…

music producer pala si bang shi hyuk sa jyp noon?

#Celebrities #Music Safaree’s Followers And Followers Are Begging Him To Check U
p On Erica Mena… https://t.co/ijMelveoeI

you have simultaneously the worst and best taste in music ever

RT @vanripperart: Here's the full version of Examtaker music track pics.
If you like awesome music, remember to follow @Mittsies on youtube…

RT @BTSPressData: Q. What is that one thing that @BTS_twt keeps in mind while co
mposing music?

#Suga: “Every aspect is important -- trends…

grand rising 🌞https://t.co/liGHjpAEDC
```



Challenges Faced and Experimentation

On initial experimentation with the Twython library, it was found that the Twitter API blocks bots, and other relevant developer programs, to tweet duplicate tweets. When trying to do so, it returns a Twitter API 403 forbidden error. The idea of the music diary bot further alleviated this issue as, for the most part, a user would listen to various pieces of music throughout the day. At this point, when running the code with the crontab method to tweet every 5 minutes, there was still small periods where the bot fails to tweet and returns the same error. This could have been caused either by the user listening to the same song on repeat, or if a certain song is over 5 minutes in length and the increment between tweets was not enough to capture the next song. With this in mind, the idea of a camera to take a picture and accompany tweets with said picture came to fruition. This is due to the fact that image data is inherently different for every image taken, thus solving the prior issue of duplication when tweeting. In the experimentation of web scraping to obtain the listening history of a user, some challenges regarding the differentiation between the detection of a 'currently listening to' song and a 'previously listened to' song arose. To alleviate this, a try-except block was used to find said 'nowplaying' variable first using regular expressions, as failure to find it would mean that the user is not currently listening to a song and the code under the except block would be run to detect the last listened to song. Small challenges such as API key errors and low image resolutions were alleviated in a trivial manner by a simple regeneration of API keys and changing code variables to define the webcam resolution as 1080p instead of the default 480p.

System Diagram and Future Improvements



For future improvements, it would be beneficial for the bot to also include the ability to detect whenever a tweet sent by the bot is interacted with by a user on Twitter. This would include replies, likes, and retweets. The updating of the music diary can also be done automatically once the user listens to a new track instead of every 5 minutes as done by crontab. This keeps the tweets more concise in terms of the ability to defend against the possibility of duplication.Some images in the tweets suffer from extreme exposure. In an ideal scenario, we may further introduce the ability to adjust the camera settings directly from the Raspberry Pi to ensure all images taken by the bot is up to standard.

References

- Blom, J. (2014). Resistors. https://learn.sparkfun.com/tutorials/resistors.
- Bruce, J. (2013, September 6). How to Build a Raspberry Pi Twitter Bot. MUO. https://www.makeuseof.com/tag/how-to-build-a-raspberry-pi-twitter-bot/.
- Hymel, S. (2014). Raspberry Pi Twitter Monitor. https://learn.sparkfun.com/tutorials/raspberry-pi-twitter-monitor/all.
- Last.fm. (2021). API Docs. Last.fm. https://www.last.fm/api/show/user.getRecentTracks.
- Poole, N. (2014). Light-Emitting Diodes (LEDs). https://learn.sparkfun.com/tutorials/light-emitting-diodes-leds.
- Raspberry Pi. (2016). Raspberry Pi OS. Raspberry Pi OS - Raspberry Pi Documentation. https://www.raspberrypi.org/documentation/raspbian/.
- Raspberry Pi. (2021). Raspberry Pi 3 Model B+. Raspberry Pi. https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.
- Sparkfun. (2014). How to Use a Breadboard. https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard.