# Twitter Mood Light Project Report

*<Basic project information/explanation/introduction>*

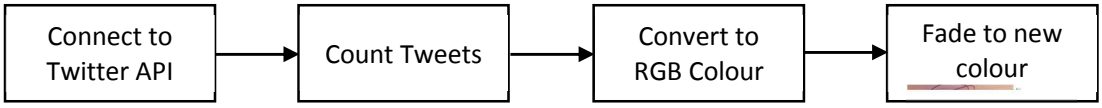
The Twitter Mood Light is an Arduino and Python project that connects to the Twitter API in Python, searches for happy, sad and angry emoticons and uses this data to generate Red, Green and Blue values for a colour which are then transmitted to the Arduino via Serial communication and used to change the colour of an RGB light emitting diode to show the mood of Twitter in real time.

## Project Design

*<Information about initial and final project proposals. How the idea came about why I liked it etc>*

I initially started this project by thinking of 3 different project ideas. These ideas were an Arduino clock, an Arduino door lock and a Twitter Mood Light. I decided that the clock was too simple and would be difficult to extend, additionally Arduino does not deal with real-time very well without extra Shields which would make the project extremely simple. The Arduino door lock would not be feasible as getting the door to lock would be very difficult and unlocking it intuitively would also be difficult with the equipment available. I finally decided to make the Twitter Mood Light as I am fascinated by web technologies, for example I have my own website, and I really liked the idea of the project not being controlled by me but by the whole world and it updating in real time.

Once I had decided on my project I split it into logical steps or objectives that would need to be completed for the project to be completed:




For the hardware I would need to make a working circuit with an analogue RGB LED so that I had full colour control being able to set each blue, red and green value from 0 to 255.

The software I wrote would need to connect to Twitter's streaming API (https://dev.twitter.com/streaming/). After connecting it would need to search for my key terms for different emotions, counting how many tweets contain these emotions. Using the emotion counts, I would need an algorithm to determine the RGB values for the RGB LED based on the data. Then I would need to find a way to change the colour smoothly. Finally I would need to make this dynamically update in real time or at regular intervals so the colour is constantly changing with the mood of Twitter.

I realised that making the program work in real time would not be possible at the University due to connecting to the University network via Ethernet would not be possible, however with my knowledge of Python I could create a Python program to collect some test data for the emoticon count, put that on an SD card which can be read via the Ethernet shield's built in SD Card reader.

I then built a project plan of what needed to be done when based on the deadlines I was given and how difficult I expected my project to be.

- Collect Twitter Mood Test Data using Python program – 28th January.
- Prepare 1 Minute Presentation on how my project proposal has changed – 28th January.

- Code a program to interpret the test data and use it to change the colour of and LED– 28th-30tt January.
- Adapt Program to take live data from Arduino Ethernet Shield – 25th February.
- Submit first draft of write-up – 18th March.
- Final submission and presentation of project to academics – 22nd April

## Project Research

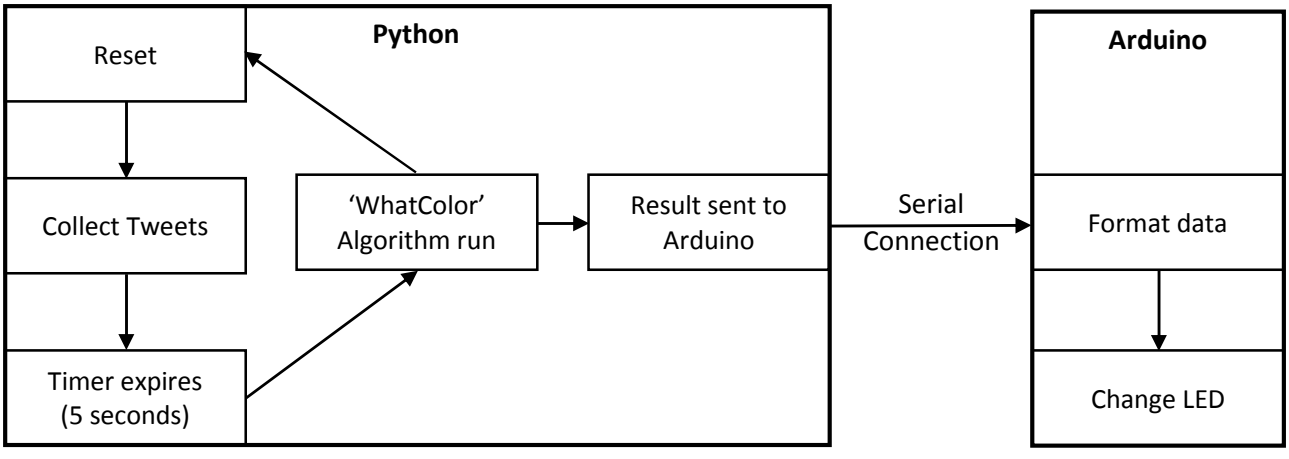
*<How the project was researched. Include links for research with explanation.>*

As there is a large Arduino community online there is a variety of online resources available to research this project. Initially I looked at similar projects of Twitter mood lights (http://arduino.cc/en/Tutorial/Twitter-controlledMoodLamp, http://www.instructables.com/id/Twitter-Mood-Light-The-Worlds-Mood-in-a-Box/ )

After looking at these I decided that the coding for the Arduino Ethernet Shield was very complex and too difficult to learn in the time frame I had been given. As a result, I decided to look for alternatives. I decided I would use my test program, coded in Python, a language which I am familiar with as it is in my A Level Computing syllabus. I found a useful, well documented library for connecting to Twitter (https://github.com/geduldig/TwitterAPI) and within this library there was an example file (https://github.com/geduldig/TwitterAPI/blob/master/examples/stream_tweets.py) which was very close to what I was trying to achieve. I decided I would use this to help create my test program. Originally I made the program so that the data would be outputted into a CSV file using Python's built in CSV library. After testing the test program and getting a set of results that outputted correctly I thought about streaming the Python data into Arduino. I found through these sites (http://www.instructables.com/id/Arduino-and-Python/?ALLSTEPS) and (http://playground.arduino.cc/interfacing/python) that sowed how Arduino can easily interface directly with Python using another library pySerial (http://pyserial.sourceforge.net/). All this information I used all this information to help me decide on the feasibility of my project as well as research different ways of getting the data from Twitter to Arduino that better utilised my programming skills.

I then looked for a suitable RGB LED. I found many cheap ones on eBay and Amazon but decided on one from Oomlout as they are the same brand as the Arduino kits we are using so I knew they were likely to be reliable and contain a lot of documentation which was especially useful for me as I had trouble making complete, working circuits with Arduino.The kit also included three RGB LED's so I knew if one of them was dead on arrival I had two backups, this was reassuring as I would not be able to test them until I received my Arduino kit at the start of the three project days and would not have enough time to get another one shipped out on time if the RBG LED was not working.

Once all my research had been done I built a simple flowchart to show what was going to be achieved:



## Project Implementation

*< Making of the Python program. All about 3 project days. How I used my time and how I went about producing the program. Use project log >*

Before the three project days I made the Python program to collect test data. This program first opens a csv file to output RGB values, then makes a call to the Twitter API searching for the terms 'angry, annoyed, happy, sad, upset'. For every tweet the program collects it is passed through an 'if else' statement which looks for the emotion which corresponds to the keyword and adds 1 to the count for that emotion. The count is an array containing 3 integer values which correspond to the 3 emotions. Then every 10 seconds the count is passed through an algorithm contained in the 'whatColour' function which takes the emotion that has the highest count and decides how strongly Twitter feels this emotion compared to the other emotions by first deciding which emotion has the highest count and then working out what percentage of the tweets counted were that emotion. Finally the algorithm expresses this percentage as a fraction of 255 as this is the highest possible 8 bit RGB value. The other 2 emotions without the highest count are then set to 0 and the 3 integers in the array are formatted so they are separated by commas and this represents the RGB value for Arduino to use, this is then written to the CSV file. This was all preparation before the three project days and receiving the Arduino kit.

**WhatColor function:**

```
def whatColour():
    global colour
    totalNum = sum(emotion) # number of tweets analysed
    largestNum = max(emotion) # mode emotion
    intensity = (largestNum / totalNum) * 255 # mode emotion / number of tweets
* 255(Maximum RGB value)
    intensity = round(intensity, 2) # Rounds intensity to 2 d.p
    selectedColour = emotion.index(largestNum) # Works out which emotion
appears most often
    colour[selectedColour] = intensity # Sets the corresponding light.
```

Once I received my Arduino kit I first tested the kit with the LED blink program by making a basic circuit with one green LED and making it blink to test the Arduino and breadboard where both working. After this I tried to set up a basic circuit with an RGB LED. I found that I couldn't get it to work after trying to set up the circuit multiple times with all three LED's but thankfully one of the student ambassadors, Ieuan realised that my problem was simply that one of my cables leading the Arduino was in the wrong pin, causing Arduino to try and register the LED to a different pin.

After getting the circuit to work with the example program (http://oomlout.com/RGBL/CODE/_RGBL_AnalogTest.txt) I decided that I would use this code as the basis of my program to turn the LED on and off and change the colours using the built in setColour and FadeToColour functions in the program. To do this I copied the relevant functions of the example code into a new program and tested that it still worked. Again I hit errors and after help from Dr Hyde and Raj I managed to get the program performing the same tasks as the example program while only keeping the relevant code. I then took a break from coding, as I was encountering a lot of problems, to set up my Laptop, connect to Eduroam wifi and install the Twitter API and pySerial libraries on my laptop so I could start implementing them into the program. Until this point I had been coding on the University computers but I saved everything I had done so far onto GitHub and the Google Drive to continue on my Laptop as I needed to utilise libraries not available on the University computers.

As I had my Python and Arduino programs running successfully while independent from each other I tried to use Serial connection to connect the two programs together. I found that the light was not turning on so something had to be going wrong. Debugging the program was very difficult as I could not see what was being transmitted through the serial connection as Arduino would not let me open the Serial Monitor while Python was using it making it frustrating to try and find the issue.

After hours of not making any progress I decided to get the Arduino program working completely independently with manual inputted numbers for the RGB values instead of numbers from the Python program to ensure this was working. Then I made sure that the Python program was outputting numbers correctly in the Python console. I then edited the Python program so it outputted to the Python console (for debugging) as well as the Serial port and changed the Arduino program so that the incoming data from Python was formatted correctly using help from this (http://stackoverflow.com/questions/11068450/ ) code. I found the program then worked perfectly.

**Arduino data formatting:**

```
    //Dealing with the serial input
    String myString = Serial.readStringUntil('\n');
    // Search for first comma
    int commaIndex = myString.indexOf(',');
    //  Search for the next comma just after the first
    int secondCommaIndex = myString.indexOf(',', commaIndex+1);
    //Set RGB values as strings
    String firstValue = myString.substring(0, commaIndex);
    String secondValue = myString.substring(commaIndex+1, secondCommaIndex);
    String thirdValue = myString.substring(secondCommaIndex+1); // To the end
of the string
    //Changing the values to integers
    int r = firstValue.toInt();
    int g = secondValue.toInt();
    int b = thirdValue.toInt();
```

However after watching the colour change and the tweets come in as well as listening to other's feedback I realised I had a few problems with the accuracy of my program. Firstly searching for keywords was not an accurate representation of Twitter's mood as a tweet containing happy could be saying I am not happy or other things that do not mean the Tweet is a happy tweet. The second problem being that every time I added a keyword to the search to try and make it more accurate I was actually counting less tweets showing that Python could not keep up, Python is not the most efficient and fast programming language so I decided to fix both problems at once by changing the search to emoticons. Now instead of searching for 'angry, annoyed, happy, sad, upset' the program now searches for ':), :(, :@, :/' This makes it more accurate as emoticons are more likely to show the mood of people and I am only sorting 4 emoticons (2 for angry) meaning that I can process more tweets at a time meaning I get a much more accurate representation of Twitter's mood.

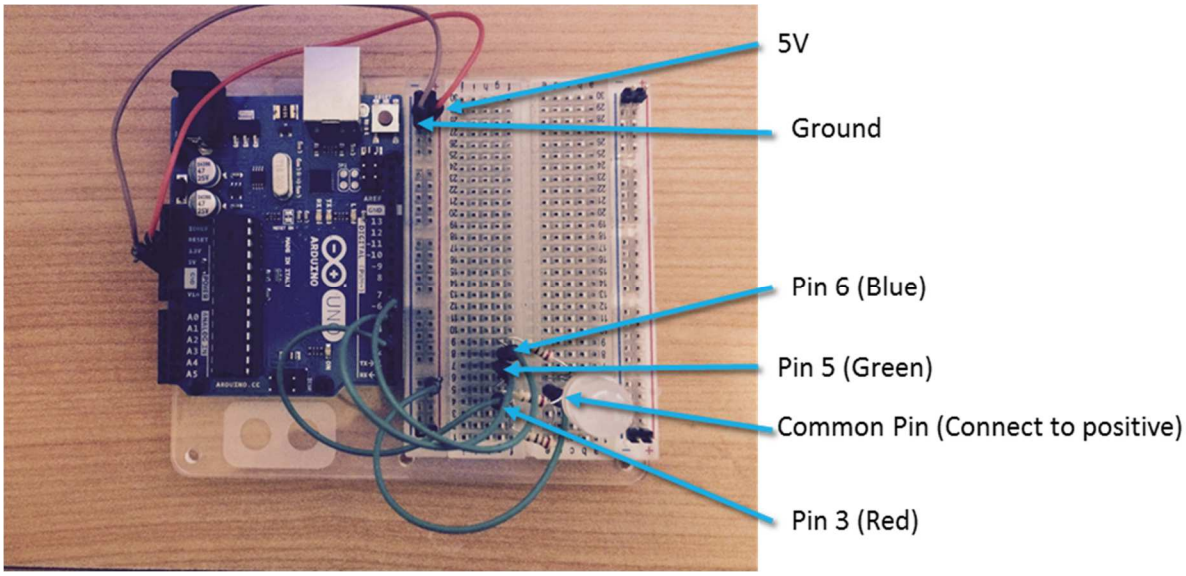
There was also problems with the accuracy of my algorithm. Using the tweet numbers as RGB values had 2 problems. Firstly it is not an accurate representation of the tweets processed and if one of the emotion readings was more than 255 the program would break. To fix this I made an algorithm that works out the proportion of each emotion and sets that to the RGB values. The algorithm is the emotion / Total emotions x 255 to work out the proportion of tweets for each emotion.

**Updated whatColor Algorithm:**

```
def whatColor():
    global emotion
    totalNum = sum(emotion) # number of tweets analysed
    for i in range(0, len(emotion)): # for each emotion count
        emotion[i] = (emotion[i]/totalNum)*255 # find proportion of that
emotion as a franction of 255
        emotion[i] = round(emotion[i], 2) #round to 2 decimal places
```

The program was now finished. To make the project look better I borrowed some Arduino jumper male to female extension cables so the LED could sit on top of a small box with the Arduino machine inside it to make it look more professional and polished as well as hide the internals of the machine.

**Finished Project:**




## Project Review

*< What went well. Successes of the project. Things that could have gone better and why. What I would have done with more time/resources. Include information about Planning and research as well as implementation. >*

In conclusion I believe my project was mainly successful. I set out to create a program which displays Twitter's mood as an RGB LED and that is what I managed to do, however the way in which I achieved this was very different then what I first planned.

My initial idea planning was not very successful as I did not consider any other ideas except from the Twitter Mood Light and I could have considered ideas that were less complex as the complexity was mentioned to me on many occasions by Dr Hyde and Dr Padget however once I had decided I would undertake this project I found a lot of information on similar projects and the project was deliberately chosen to be more about programming rather than the hardware, I knew I was a confident programmer, therefore I picked a project which played to my strengths.

I also had to adapt my program as I found it would be too difficult to get live data directly from the Arduino Ethernet shield, with more time and resources I would have liked to do this as it would mean the project would work with any power source a battery or power from USB, however with

the current project it needs to be connected via USB to a computer running the accompanying Python program with Python 3.4 and the 2 extra libraries installed, however this is still better than the program only running test data which was my another alternative. This project could be improved by using a more complex sentiment analysis tool to get more accurate data. This could be achieved my using a more efficient programming language than Python. It could also be improved by using a Wifi Arduino Shield so it could work wirelessly with battery power.

The outcome of the project has surprised me slightly as Twitter is always happy in the tests I have done with a light hint of sadness. This can lead to the project seeming like a plain green LED however you can spot subtle changes. Additionally I would be interested to see how my project reacted to major catastrophes and if Twitter reacts strongly to this. However in its current state I am pleasantly surprised to see that in general my project shows Twitter is happy.