Python Project (Arduino based) light animation using opc software and potentiometer

GROUP WORK Project

Course Name: Computer Architecture

Course ID: CCE1000

Student Names:

* **Rohini Paligadu**

Student Number: M00646489

* **Perrine Julia Estelle**

Student Number: M00682697

SUBMISSION DATE: 29 March 2019

Contents

[INTRODUCTION 2](#_Toc4751904)

[A summary of the project and an overview of the work. 2](#_Toc4751905)

[Implementation 3](#_Toc4751906)

[\*terminology 5](#_Toc4751907)

[Installation instructions on how to build the connections on the arduino board circuit 6](#_Toc4751908)

[The code ( Basic explanation) 7](#_Toc4751909)

[How to run and interact with the system(Details) 11](#_Toc4751910)

[Limitations of the work 16](#_Toc4751911)

[REFERENCES 17](#_Toc4751912)

### INTRODUCTION

## A summary of the project and an overview of the work.

The project consists of a simulator software (OPC) where various light animations, written in python code is demonstrated.

User must interact with the system by the use of keyboard and potentiometer input to vary animations.

User is prompted with a list of options where he/she can choose from in order to animate LEDs either by single key press or use of Potentiometer.

An Arduino board is incorporated into the project to allow user to interact with potentiometer

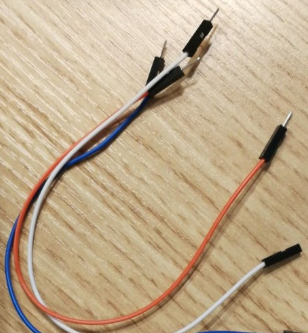
### Implementation

##### Materials needed

1. A potentiometer(sensor)



1. Jumper wires



1. ARDUINO UNO



1. USB cable



5) Open pixel control software simulator

A simulator that lets you test and visualize your program before wiring it to real lights.

OPC describes the format of a stream of bytes, typically sent over a TCP connection, to control an array of RGB lights (pixels). The pixels are assumed to be arranged in strands, where each pixel has a fixed index in its strand.

Open Pixel Control (OPC) is a simple protocol for controlling arrays of RGB lights, such as Total Control Lighting LEDs.

## \*terminology

**Potentiometer (sensor):**  variable resistor with a rotating contact that controls voltage input (vary voltage values)

**Arduino uno board:** Microcontroller board developed by arduino.cc, an open source electronics

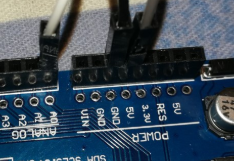
### Installation instructions on how to build the connections on the arduino board circuit

* The potentiometer is connected to the Arduino uno board (5V, A0 and ground respectively) .

Potentiometer to arduino connection

We connect three wires to the Arduino board. The first goes to ground from one of the outer pins of the potentiometer. The second goes from 5 volts to the other outer pin of the potentiometer. The third goes from analog input A0 to the middle pin of the potentiometer.

By turning the shaft of the potentiometer, we change the amount of resistence on either side of the wiper which is connected to the center pin of the potentiometer. This changes the relative "closeness" of that pin to 5 volts and ground, giving us a different analog input.



### The code ( Basic explanation)

Python 3.72 software version is used.

Program is written in python.

Firmata is a library used to connect python with Arduino, it is integrated in the arduino itself

* Import random

from random import randint

The Random module contains some very useful functions

randint is a function that is part of the random module.

* Use of randint function for random colours

Function should be imported in python IDE in order to use it

* random.shuffle(led\_colour)# changes position of item in a list

client.put\_pixels(led\_colour)

* \*import opc

Start by making the OPC library available to your application. This imports the OPC.py module file located in the lib/site-packages/ directory.

>>> import OPC

2. Create OpenOPC instance (Open mode)

In Open mode a connection is made to the OpenOPC Gateway Service running on the specified node. This mode is available to both Windows and non-Windows clients.

>>> opc = OpenOPC.open\_client ('localhost')

e.g #creating a client object

client = opc.Client('localhost: 7890’) was used

* import time

Import time imports the method time. sleep() from the module, on the other hand, imports the module time. Import time and use time. time () instead

* import sys

For program to work properly with both

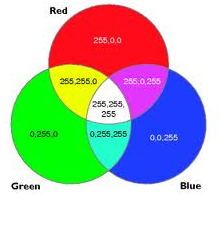
Sys.stdout and sys.exit().

* Functions

The program consists of a series of functions that have been defined and called in the Menu

* **def** followed by the function name and parentheses ( ( ) ).
* Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
* The code block within every function starts with a colon (:) and is indented.
* RGB colors

R-red,G-green,B-blue



Colours are assigned to variables in different ways using different combination of RGB varying numbers

e.g variables

1)Led\_colour =[(255,255,255)]\*360

Print(Led\_colour) , colour being printed on simulator

client.put\_pixels(led\_colour) , colour being printed on simulator

2) blue = [(0,0,255)]\*numLEDs

black = [ (0,0,0) ] \* numLEDs

white = [ (255,255,255) ] \* numLEDs

WHERE numLEDS, is a variable to which 360 was assigned to.

360 used because maximum number of LEDS that can be displayed on the OPC software does not exceed 360

* Python library /built-in function

Reverse()

Used in program to reverse the range sequence

**reversed**(*seq*)

Return a reverse [iterator](https://docs.python.org/2/glossary.html#term-iterator). *seq* must be an object which has a [**\_\_reversed\_\_()**](https://docs.python.org/2/reference/datamodel.html#object.__reversed__) method or supports the sequence protocol (the [**\_\_len\_\_()**](https://docs.python.org/2/reference/datamodel.html#object.__len__) method and the [**\_\_getitem\_\_()**](https://docs.python.org/2/reference/datamodel.html#object.__getitem__) method with integer arguments starting at 0).

e.g

#( instead of 0 to 360 --🡪 360 to 0)

for x in reversed(range(0,360)) :

* LOOPS

A while-loop checks its termination condition before each entry into the loop body, and terminates if the condition has gone False. Thus, the loop body will normally iterate zero, one or more *complete* times.

e.g while loop

while True:

to repeat Menu

A for-loop iterates its body once for each value returned from the iterator expression. Again, each iteration is normally of the *complete* loop body.

e.g for loop

for x in range (93,99) :

# for each number / position between 93 and 99

### How to run and interact with the system(Details)

Program is being run in python IDLE (3.7)

Program opens up in python shell and user can read prompts and enter necessary keyboard inputs there.

There are six animations in all

Program consists of a menu, all options shows animated led colours

User chooses among a list of functions by simple keyboard press (numbers),

* **SWITCH COLOURS**

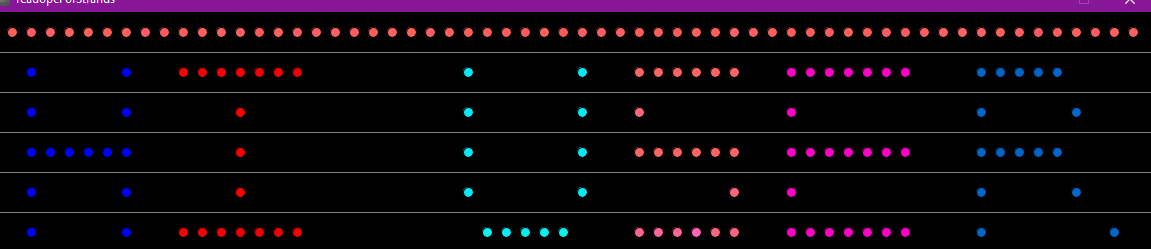
Colour switches from a colour to another during a certain amount of seconds

<https://youtu.be/WtpIliYq9sM>

The link above demonstrates how the leds switch different colours.

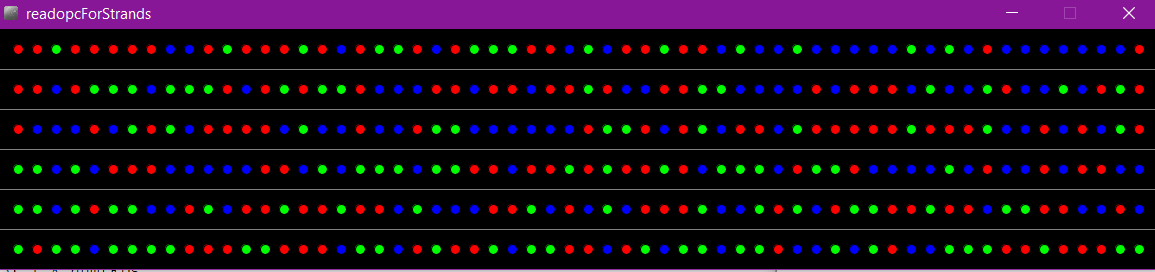
* **PRINT "HI USER"**





* **SHUFFLE COLOURS**

Colour shuffles between red, blue and green.





* **Diagonal/horizontal animation**





* **Random color animation**

The link below demonstrates the random animation of colours.

<https://youtu.be/JOTzRJyfgto>

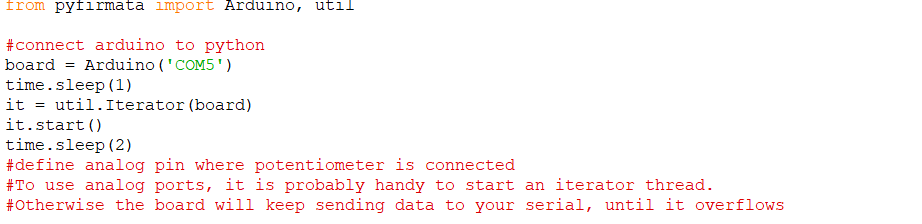
The colours change randomly and varies between a certain period of time



* **Use of potentiometer to vary animation…and play around with the system**

To be able to use the potentiometer and Arduino together with our python shell, we connected it using the pyfirmata library.

COM5 is used in the system here, it can vary on other system



POTENTIOMETER ANIMATION DEMONSTRATION YOUTUBE LINK

<https://youtu.be/1KjH-bzpGCg>

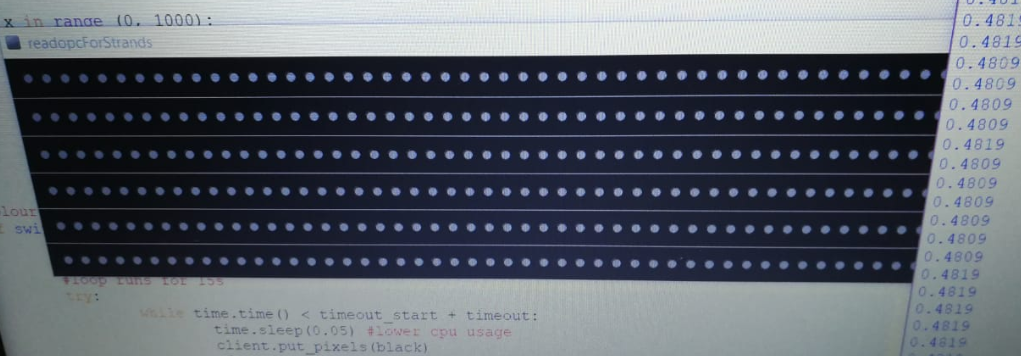
When user rotates the potentiometer, the voltage value varies. Colour fades when voltage decreases and colour intensity increases with increasing voltage value.

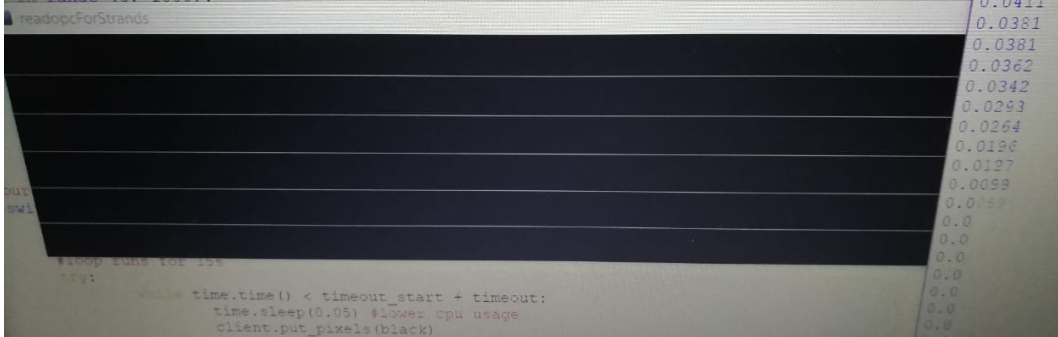
Colour vary from white to black

User animate LEDs using keyboard and potentiometer (user interface)

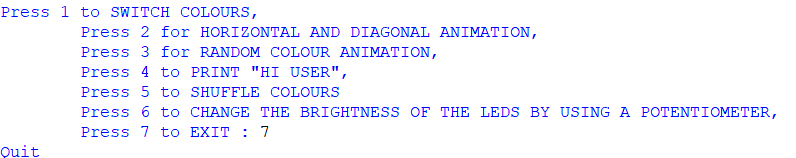
User uses potentiometer (sensor) to input and vary voltage values.







USER TERMINATE PROGRAM EXECUTION BY PRESSING 7 IN THE PROMPTED MENU CHOICE



### Limitations of the work

**1.Lack of / Broken materials**

The project works as expected with no errors, indeed, there were lack of wires and sensors .

Some sensors were indeed damaged and got burnt due to improper circuit connection, potentiometer got burnt we had to change it .

Due to loose wire connection, we had problems to display colours

More sensors and jumper wires could have been used for a better project presentation

2. **Incorporation of more sensors**

The project would have been better if a motion sensor was implemented

Sensor detects movement from instead of using only potentiometer as sensor and changes led animation

### REFERENCES

* **Opc import**

[[1]](#endnote-1)<http://openopc.sourceforge.net/api.html>

<http://openpixelcontrol.org/>

* **library/functions**

<https://docs.python.org/2/library/functions.html#reversed>

<https://www.tutorialspoint.com/python/python_functions.htm>

* **RGB colors**

<http://www.discoveryplayground.com/computer-programming-for-kids/rgb-colors/>

* **Pyfirmata**

<https://github.com/tino/pyFirmata>

We also embedded the firmata library found in the Arduino ide onto the Arduino.

1. [↑](#endnote-ref-1)