Assessment 3. Software project Python

Demonstration (+ regular repository submissions); week 18;

30% - individual submission

Students will be writing code related to an engineering project, and submit regularly to a dedicated repository. The final version of their software will be demonstrated via video submission to the instructors at week 18.

Students will have to write Python code for creating an original light animation on 6 LED bars. They will be provided with:

- interfacing library `opc.py` which allows wireless access to the LED device
- simulator software which allows to test the code locally before connecting to the LED device
- example scripts showing how to use the library and produce some basic animations

Students will have to create a repository, named as their student number, on www.github.com , and invite the tutors so they can have access to it. They are expected to submit regularly the code they develop, since repository activity will affect the final grade.

Originality in the design and implementation of animations is strongly encouraged, as is the use of programming tools beyond those explained in class. Additional grades are awarded for employing external interfacing methods (e.g. Arduino based).

A video demonstration of your work must be submitted. Upload to onedrive or Youtube **as 'Unlisted**' and submit a link. The video must include a short intro, a demonstration of your animations and any gui elements, a description of the code in abstract terms. You may wish to speak about further development, experiments performed, etc. It should be no longer then 5 minutes.

Summarising, project 3 will be assessed based on repository activity, quality of final submission, and knowledge demonstrated during the video demonstration, according to the rubric below.

PDE 2400

Project 3 evaluation rubric

Item	1	2.1	2.2	3	Fail
Repository & documentation 20 %	Consistent activity (more than twice a week) and advanced use of repository tools (branching, etc.). Advanced comments and documentation. Included comprehensive readme file.	Consistent activity in and out of class (twice a week), structured and commented commits. Satisfactory comments and documentation. Included basic readme file.	Inconsistent activity in and out class (once a week). Basic but correct documentation and comments.	Some activity overall (less than once a week), only late evidence of out of class work. Poor documentation and comments.	Minimum (<2 commits) or no activity. Missing or minimal comments or documentation
Code 40 %	High quality code exhibiting deep knowledge and creativity. Good use of functions and repetitive constructs, code reusability/ modularity demonstrated. Console or graphical UI implemented with input handling and error reporting to the user.	Code shows structure, understanding of programming tools and personal initiative. Some functions defined, demonstrated good use of constructs. UI implemented with input handling.	Code shows some personal initiative and good understanding of basic concepts - sequences and basic use of loops. Basic UI implemented.	Code works, but is mostly based on given examples with little personal contribution. No UI present.	Code does not work, and shows little understanding of language/algorithms. No UI present.
Animations 30 %	More than five highly creative and varied animations using a range of techniques.	Four or five animations that show novel thinking with clear personal ideas.	A small number of animations. Some new ideas incorporated into simple animations.	Less than three animations. Mostly variations on given examples, with few personal ideas.	Same as in examples, with no demonstration of any personal contribution
Video demonstration 10 %	Very good and organised demonstration	Careful demonstration of code and user guide	Demonstration is clear but not very organised	Basic explanation of code and animations	No clear explanation, lack of understanding

Technical information

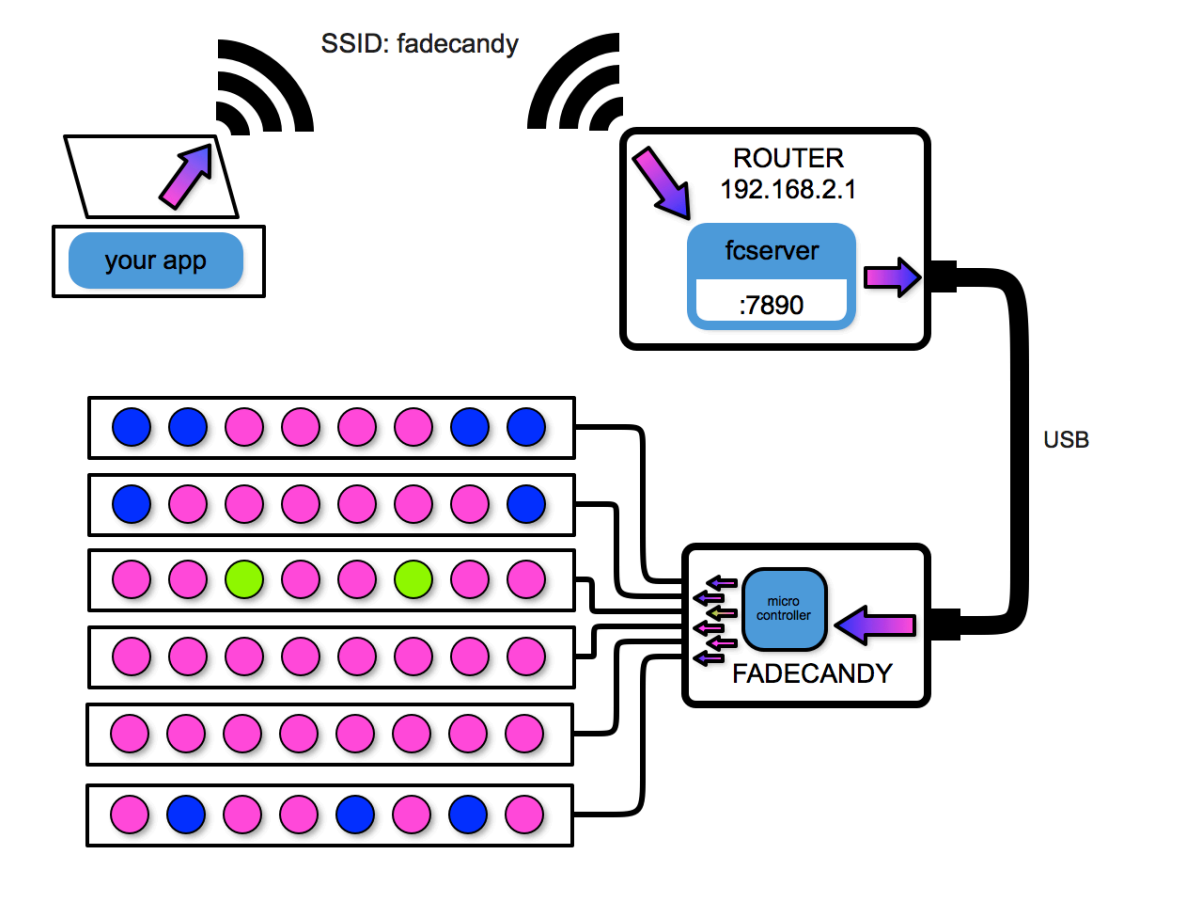
There exists a physical installation of 6 led strips suspended in the Ritterman builiding. The simulator you are provided with replicates them exactly, including addressing – you will only be required to work on the simulator.

Each strip has 60 RGB LEDS. When all the strips are installed they will appear to be connected in a long line. The first strip will be addressable as the first 60 LEDs, (0-59), the next strip will be LEDs 60-119, going up all the way to 359. Values are sent over via OPC, which will be provided at the start of the project with the simulator. If you send to many values the additional ones will have no effect, but will not generate an error. If you send less values then only the ones that you send data for will be updated.

When you are running the simulator you need to send your data to localhost:7890 (the local loopback address which will resolve to your machine) as both applications will be running on the same machine.

Physical connection:

The strands are attatched to a fadecandy controller which enables them to be controlled over USB. The fadecandy controller is plugged into a wifi router with a USB socket which is running the fcserver application. This application enables us to send data to it using the OPC format by connecting to the wifi network the router creates. This data is then used to control the LED strips. The fcserver will be running at the address 192.168.2.1:7890 once you connect to the wifi network, you will need to set this address in your application.


SSID: fadecandy
your app
ROUTER
192.168.2.1
fcserver
:7890
USB
micro
controller
FADECANDY