San Jose State University

Department of Electrical Engineering

EE104, Spring 2023, Pham

Laboratory Assignment #8

# Objectives

For the software part, you will practice to create application with OpenAI. Specifically, you will create a web scrapper to download the web files to your local drive, then use OpenAI to respond to questions related to that website. You will also create a client-server environment to answer questions related to files existing inside the intranet across an organization.

For the hardware part, you will interface the KRIA board with the Traffic Controller circuit that you already design in an earlier lab. Your KRIA board now will be the brain of the Finite State Machine to control the traffic lights on your breadboard that you built earlier.

# Grading

Refer to the section **Python Programming** for grading criteria.

# References

<https://platform.openai.com/docs/quickstart>

<https://platform.openai.com/docs/api-reference/introduction>

<https://colab.research.google.com/github/pinecone-io/examples/blob/master/integrations/openai/semantic_search_openai.ipynb>

<https://github.com/openai/openai-cookbook>

<https://github.com/openai/openai-cookbook/tree/main/apps/file-q-and-a/nextjs-with-flask-server>

<https://platform.openai.com/docs/tutorials/web-qa-embeddings>

<https://github.com/openai/openai-cookbook/tree/main/apps/web-crawl-q-and-a>

# Download, Installation, and Licensing

For the OpenAI part, you will follow the instructions from these files that you can download from Canvas:

1. openai-quickstart.txt
2. FileQ&A\_Instructions.txt
3. WebCrawlQ&A.txt
4. You will also need to download the sample python source code web-qa.py

For the KRIA interface, you will create your own program.

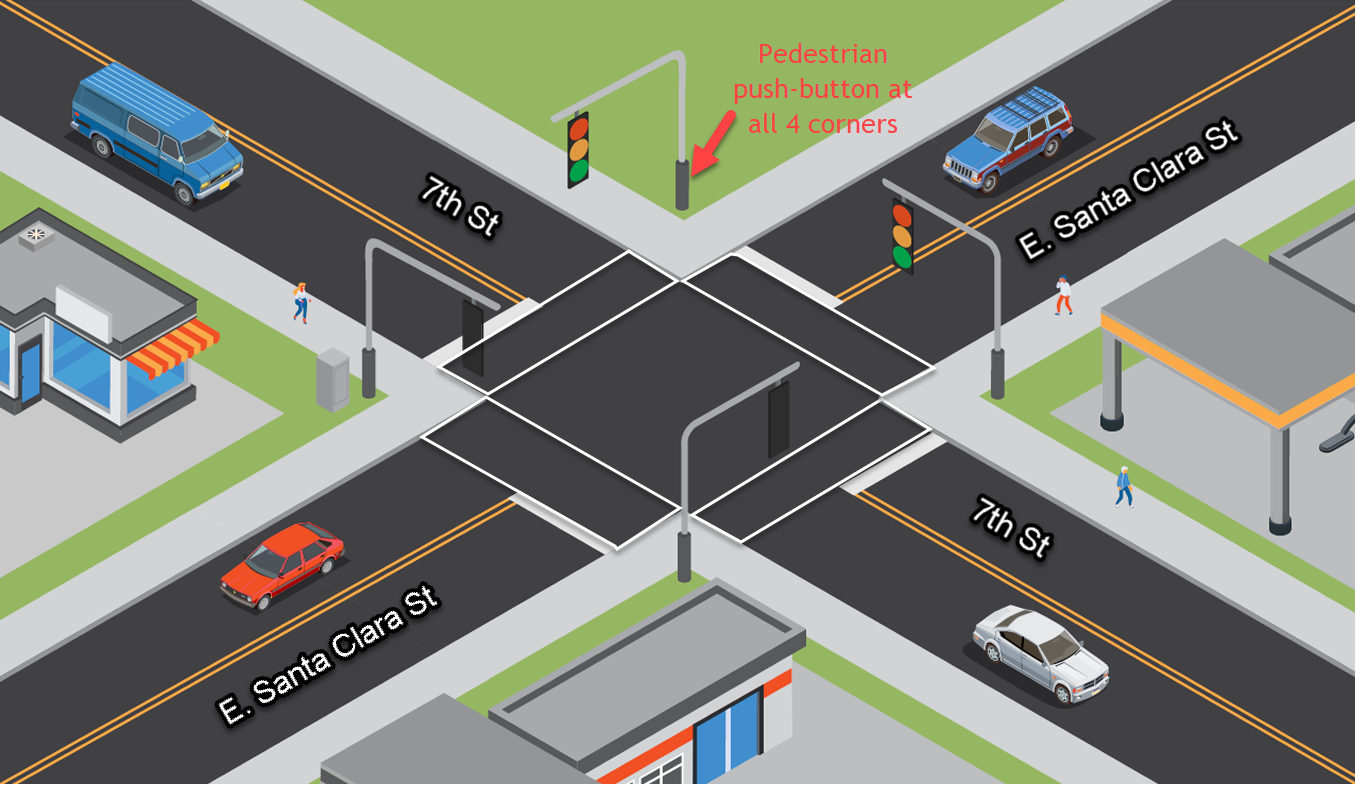
# Deliveries

#### Lab Submission Everything must be submitted to GitHub because Canvas may not take your larget Zip file.

|  |  |  |
| --- | --- | --- |
| **Program or Requirement** | **Use Case** | **Earned Score / Max Score** |
| README file and Developer Note | README: This is a brief user guide so that the user can use your programs.   * Client Server solution * Web scrapper solution * Traffic controller   Because you will submit files to GitHub this time, submit this part to GitHub as well. | \_\_\_\_\_ / 10 |
| OpenAI File Q&A | OpenAI chat bot will use the information from the files you store in the directory to answer questions related to those files. This is a client-server front end application.  Because the zip file will be very large, you will need to upload to GitHub because Canvas may not take it. | \_\_\_\_\_ / 30 |
| OpenAI Web Crawl Q&A | OpenAI chat bot will use the information from the files that your program scrapes from a website to answer questions related to those this website. This is an example of a virtual chat bot to guide users to the correct information.  Because the zip file will be very large, you will need to upload to GitHub because Canvas may not take it. | \_\_\_\_\_ / 30 |
| Simple Traffic Controller:  Part 2, Hardware | Simple Traffic Controller: Part 2 – replace the Finite State Machine that you used in Part1 (that was implemented with Flip Flops) with software Finite State Machine that you will write for this lab.  Because you will submit files to GitHub this time, submit this part to GitHub as well. | \_\_\_\_\_ / 30 |
|  | **TOTAL** | **\_\_\_\_\_/ 100** |

Specifically, you will create a web scrapper to download the web files to your local drive, then use OpenAI to respond to questions related to that website. You will also create a client-server environment to answer questions related to files existing inside the intranet across an organization.

# Simple Traffic Controller: Part 1, Hardware



East Santa Clara Street has the priority, i.e. it always has green light and pedestrian walkways along E. Santa Clara St. is always allowed, except for when there is a request for pedestrian crossing from the 7th Street or a car coming from 7th Street.

You will be given the following hardware to implement your hardware controller portion:

LED For traffic signals.

7-Segment For a count-down pedestrian signal along the 7th Street

Push-button To accept pedestrian’s input request

74LS47 BCD to 7-Segment Decoder

74LS193 Binary Up/Down Counter with Clear

74LS90 Decade Counter

555 Timer For use as clock

Resistors For connecting LEDs, 7-Segments, and other open-collector signals

Capacitors

You will supply your own:

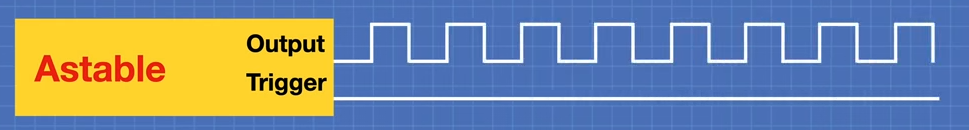
Breadboard You will reuse your own breadboard from your previous courses

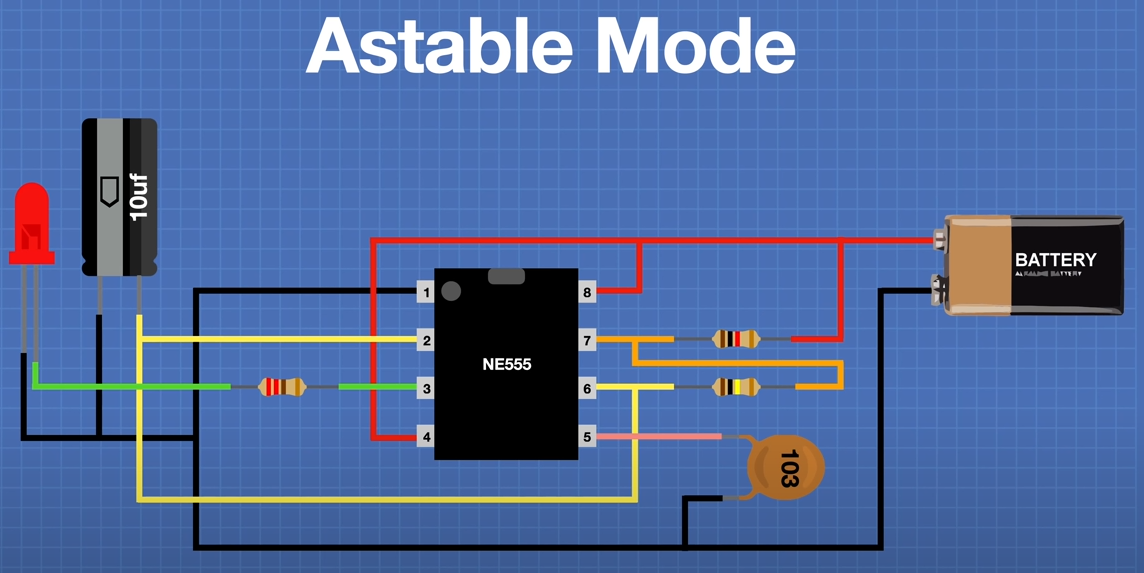
Power/Ground Power/Ground from a battery or a power source. Use 3 x 1.5V = 4.5V

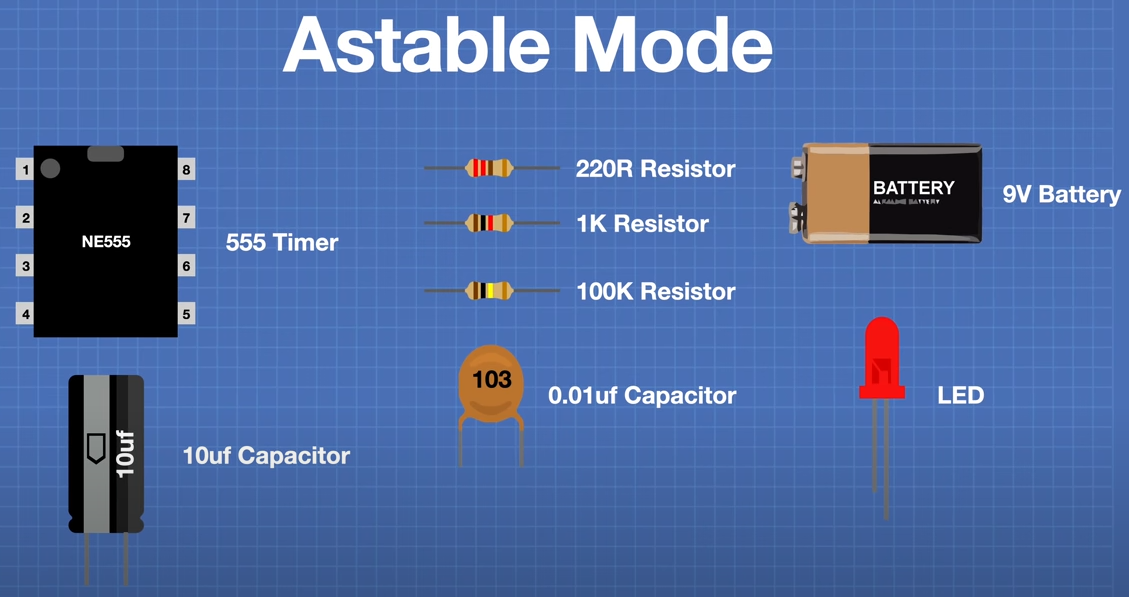
**74-Series datasheet**: <https://www.futurlec.com/IC74LS00Series.shtml>

**Clock generation using 555 chipset using Astable Mode:**

Ref: <https://www.youtube.com/watch?v=ABWU7FlM1T0&ab_channel=DroneBotWorkshop> (time=10:20 )



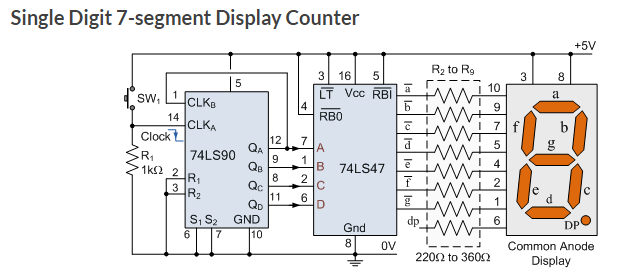
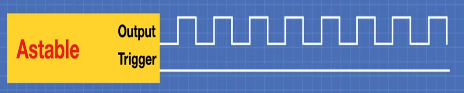


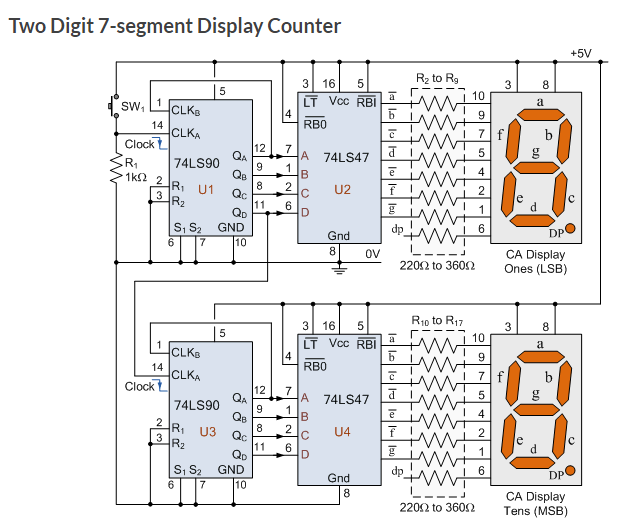
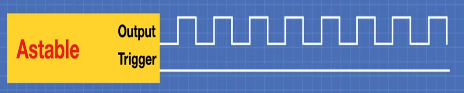


**7-Segment circuit using clock, decade counter, BCD-to-7segment Decoder, and the 7-Segment Display**:

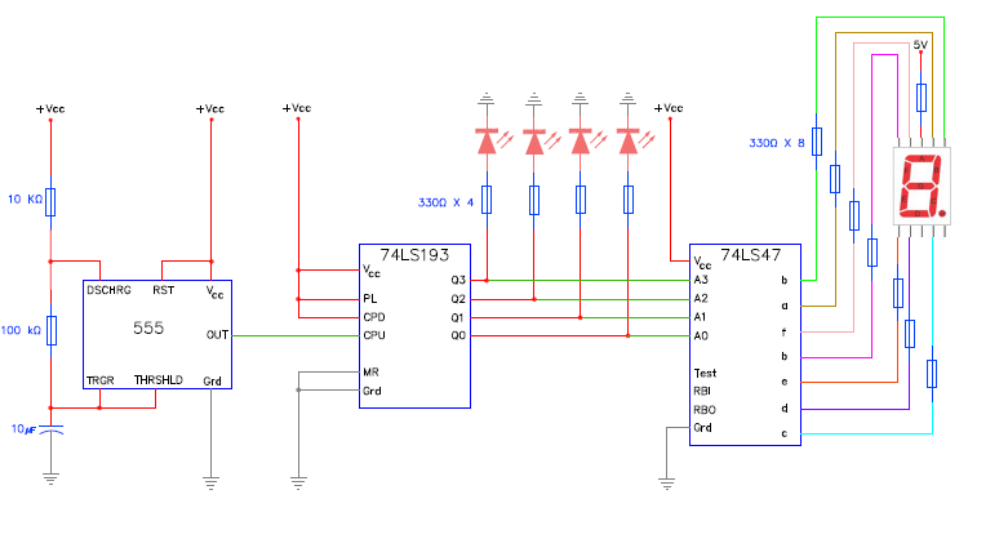
One-digit Ref: <https://www.youtube.com/watch?v=XCJqoae4hgY&ab_channel=element14presents>

One and Two-digit Ref: <https://www.electronics-tutorials.ws/counter/7-segment-display.html>





Alternatively, using 74LS193 up/down counter: <https://www.fwdskillzone.com/digital_logic_design.html>

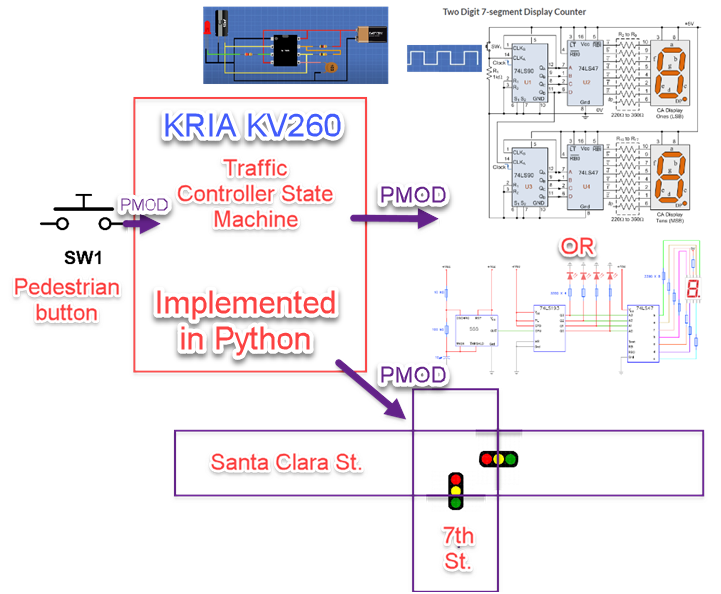


**State Machine Design:**

This time you will implement your Finite State Maching using Python programming language embedded in the KRIA board. The KRIA board will interface to your existing breadboard via the PMOD interface.

You can refresh your state machine circuit design using your old notes from previous classes or from this site showing a 4-state Mealy FSM implementation: <https://www.cpsc.ucalgary.ca/custom/321_challenge/04%20Finite%20State%20Machines.html> for from this video clip: <https://www.youtube.com/watch?v=Z4Zz7n-Lj0g>. Instead of using flip flops to implement the FSM, your Python program will implement the FSM.

Note: Your design may be a combination of Mealy & Moore FSM.



That’s all for this lab. Hopefully you found it useful and increase your interest in the Python world! See you in the next lab.

# Laboratory Hand-In Requirements

Once you have completed a working design, prepare for the submission process. You are required to upload YouTube videos to demonstrate your working solutions. You are also required to submit an archive of your project in the form of a ZIP file. Use 7-Zip option to create the ZIP file. Name the archive lab#\_yourlastname\_yourfirstname.zip. Refer to Lab 1 for detail instructions.

You will submit your zip file to the instructor through Canvas by the due date and time. If your program is not completely functional by the due date, you should demonstrate and turn in what you have accomplished to receive partial credit. See the syllabus for the late penalty guideline