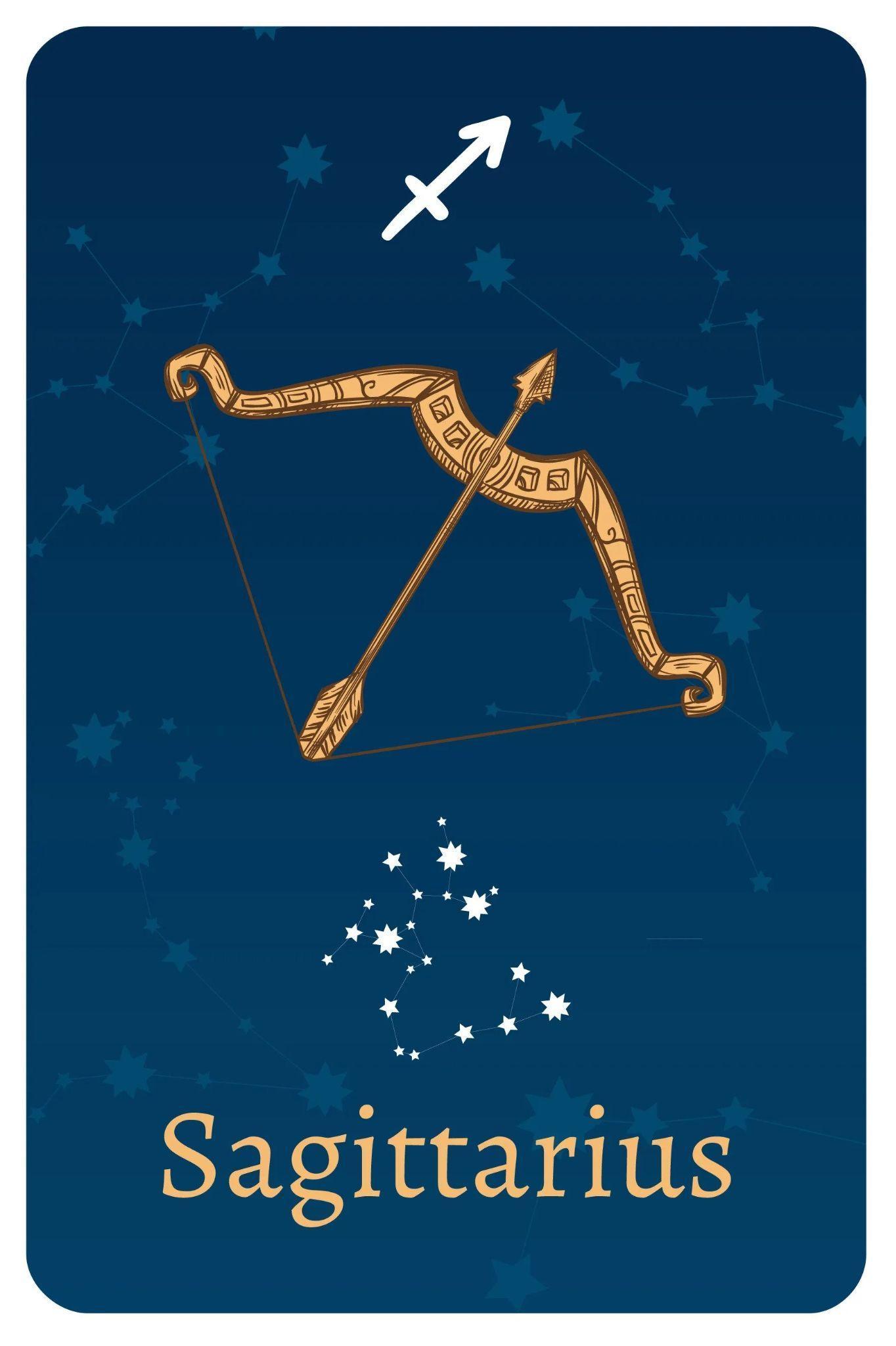
****

TEAM MEMBERS

Shivansh Shukla

Nathan Lee

Edwin Kulakkattolikel

Contents

[PURPOSES AND GOALS 3](#_heading=h.gjdgxs)

[HOW TO INSTALL THE PROGRAMS 4](#_heading=h.30j0zll)

[DEPENDENCIES 4](#_heading=h.1fob9te)

[HOW TO RUN THE PROGRAMS 4](#_heading=h.3znysh7)

[DESIGN ARCHITECTURE 5](#_heading=h.2et92p0)

[PROCESS & WORKFLOW 5](#_heading=h.tyjcwt)

[TEST DATA](#_heading=h.3dy6vkm) 7

[VIDEO RECORDINGS](#_heading=h.1t3h5sf) 10

[CONCLUSIONS](#_heading=h.4d34og8) 11

[REFERENCES](#_heading=h.2s8eyo1) 11

# PURPOSES AND GOALS

The goal of this laboratory assignment is to introduce students to the practical implementation of a traffic control system using Finite State Machines (FSM). This hands-on exercise involves both hardware and software components, emphasizing the application of theoretical concepts in a real-world scenario.

Goals:

1. Practical Application of FSM
2. Integration of Hardware and Software
3. Understanding State Transitions
4. Hands-on Learning Experience

# HOW TO INSTALL THE PROGRAMS

## DEPENDENCIES

**Dependencies: Python, KRIA Software, Jupyter Notebook**

# HOW TO RUN THE PROGRAMS

**Hardware:** Connect jumper wires to GPIO of KRIA Board **Software:** Connect KRIA Board Jupyter notebook and hit run on software

**Application:** After software is ran, circuit will light up and the button(Slider) can be moved to activate.

# DESIGN ARCHITECTURE

The design architecture of the Simple Traffic Controller combines both hardware and software components to create an integrated system for traffic management. At a high level, the hardware and software logical blocks are interrelated to ensure seamless functionality. The hardware block includes components such as LEDs, a pedestrian button, a 7-segment display, and electronic components. The software components, implemented in Python, drive the Finite State Machine (FSM) that controls the traffic lights based on inputs from the hardware elements. The logical blocks interact closely to facilitate the state transitions of the traffic lights in response to pedestrian inputs.

The hardware components act as the physical interface with the environment, capturing inputs from the pedestrian button and displaying outputs through LEDs and the 7-segment display. These hardware elements are tightly linked to the software, where the Python code interprets inputs, manages the FSM, and triggers corresponding actions on the hardware. The interaction between hardware and software is crucial for the accurate representation of traffic scenarios and the dynamic adjustment of traffic light states.

Furthermore, while the hardware elements are localized in the physical environment, the software components can be extended to cloud-based services for enhanced control, monitoring, or remote management. This potential cloud integration allows for scalability and additional features, expanding the capabilities of the Simple Traffic Controller.

# PROCESS & WORKFLOW

The process of the Simple Traffic Controller involves a sequence of actions to manage traffic and pedestrian signals effectively. The user activates the pedestrian button, initiating the process. The workflow begins with the activation of the button, signaling the FSM to transition states. The FSM logic is programmed to consider pedestrian input and adjust the traffic light states accordingly. LEDs convey these states to drivers and pedestrians, ensuring a clear and safe traffic environment.

The process can be outlined as follows: a pedestrian activates the button, signaling the FSM to transition through states that manage traffic lights. LEDs visually represent these states, guiding both drivers and pedestrians. The workflow, on the other hand, starts with users approaching the intersection. They observe the traffic lights and the pedestrian button. Upon activation of the button, the FSM adjusts the lights, allowing pedestrians to cross or stopping traffic as needed. Once the cycle is complete, the lights return to the default state, concluding the workflow until the next activation.

This systematic process and workflow ensure the reliable and predictable operation of the Simple Traffic Controller, demonstrating the effective coordination between hardware and software components to manage traffic scenarios in a controlled environment.

# TEST DATA

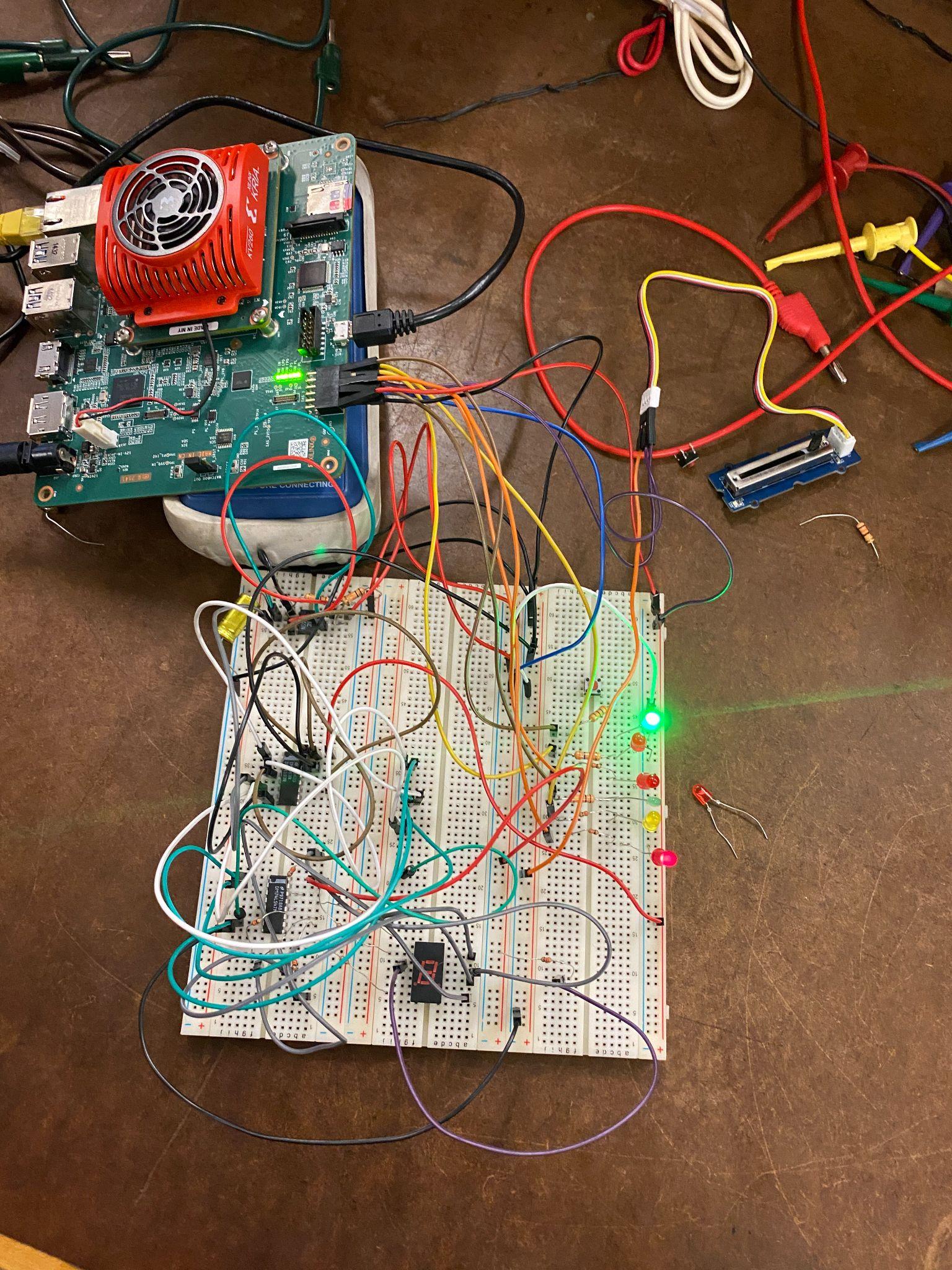


Figure 1: Testing Hardware for Traffic Controller

# VIDEO RECORDINGS

| **Recording Title** | **URL** | **Notes** |
| --- | --- | --- |
| EE104 Lab 4 Traffic Controller | <https://www.youtube.com/watch?v=wzklSMfqNEY> | Traffic Controller Software and Hardware Demo |

# 

# CONCLUSIONS

In conclusion, the Simple Traffic Controller segment of Laboratory Assignment #4 provides students with a hands-on application of Finite State Machines in digital systems design. Through a blend of hardware setup and Python programming, the exercise bridges theoretical concepts with real-world functionality. Emphasizing the integration of hardware and software, it offers a practical understanding of state transitions in FSMs. The iterative nature of the assignment encourages experimentation and troubleshooting, fostering a comprehensive grasp of the challenges inherent in constructing responsive control systems. Ultimately, this exercise not only achieves its immediate goal but also instills a broader appreciation for the interdisciplinary nature of digital systems design, preparing students for future complexities in the field.

# REFERENCES

* What Is Agile Reporting? (Definition and How To Complete) <https://www.indeed.com/career-advice/career-development/agile-reporting>
* Test Report <https://strongqa.com/qa-portal/testing-docs-templates/test-report>