UCSD CSE WES 237A

Winter 2026

Pulse Width Modulation Lab Report

(Due: 01 /21 /26)

Demonstration Date : 01 /21 /26 Group#: 7

Members: ***Benediction Bora*, *Gabriel Martinez***

INSTRUCTOR / TUTOR/

Nadir Weibel, Pushkal Mishra

Self-test Report Demo Reviewer

Name : ***Benediction Simeons***

Working / Not working Demo score Report score

Part1: _____ _____ _____: Executed Successfully

Part2: _____ _____ _____ : Executed Successfully

Part3: _____ _____ _____ : Executed Successfully

Part4: _____ _____ _____ : Executed Successfully

Part5: _____ _____ _____: Executed Successfully

Part6: _____ _____ _____: Executed Successfully

Video Demo Link: Video Demo Youtube

GitHub Repo: Asignment1 - WES237A GitHub Repo

TOTAL Score: _____

Pulse Width Modulation on RGB LED Using PYNQ-Z2

This report documents the design, implementation, and evaluation of a PWM RGB LED scheme implemented on the **PYNQ-Z2** board as part of WES 237A; Intro to Wireless Embedded Systems. The system integrates **MicroBlaze GPIO** control, Python **asyncio**-based task scheduling, push-button user interaction.

1. Objectives

The objectives of this assignment were to:

(1) generate PWM signals using C++ blocks and python in Jupyter Notebook
(2) determine an appropriate PWM frequency to avoid visible flicker,
(3) control perceived LED brightness via duty cycle adjustment, and
(4) design a responsive, asynchronous system that reacts to user button inputs in real time.

2. Design Methodology

A top-down design methodology was employed. The complete system was first divided into independent subsystems: low-level GPIO access, PWM signal generation, and user interaction. Each subsystem was implemented and validated independently within dedicated Jupyter notebook cells prior to full system integration. Asynchronous programming using asyncio was selected to enable concurrent LED blinking and button monitoring without blocking execution. A shared state dictionary was used to safely coordinate behavior between tasks.

3. Task and State Architecture

Figure below illustrates the interaction between asynchronous tasks and shared system state. The button task monitors user input and updates the system state, while the blinking task generates PWM-driven LED output based on the current state.

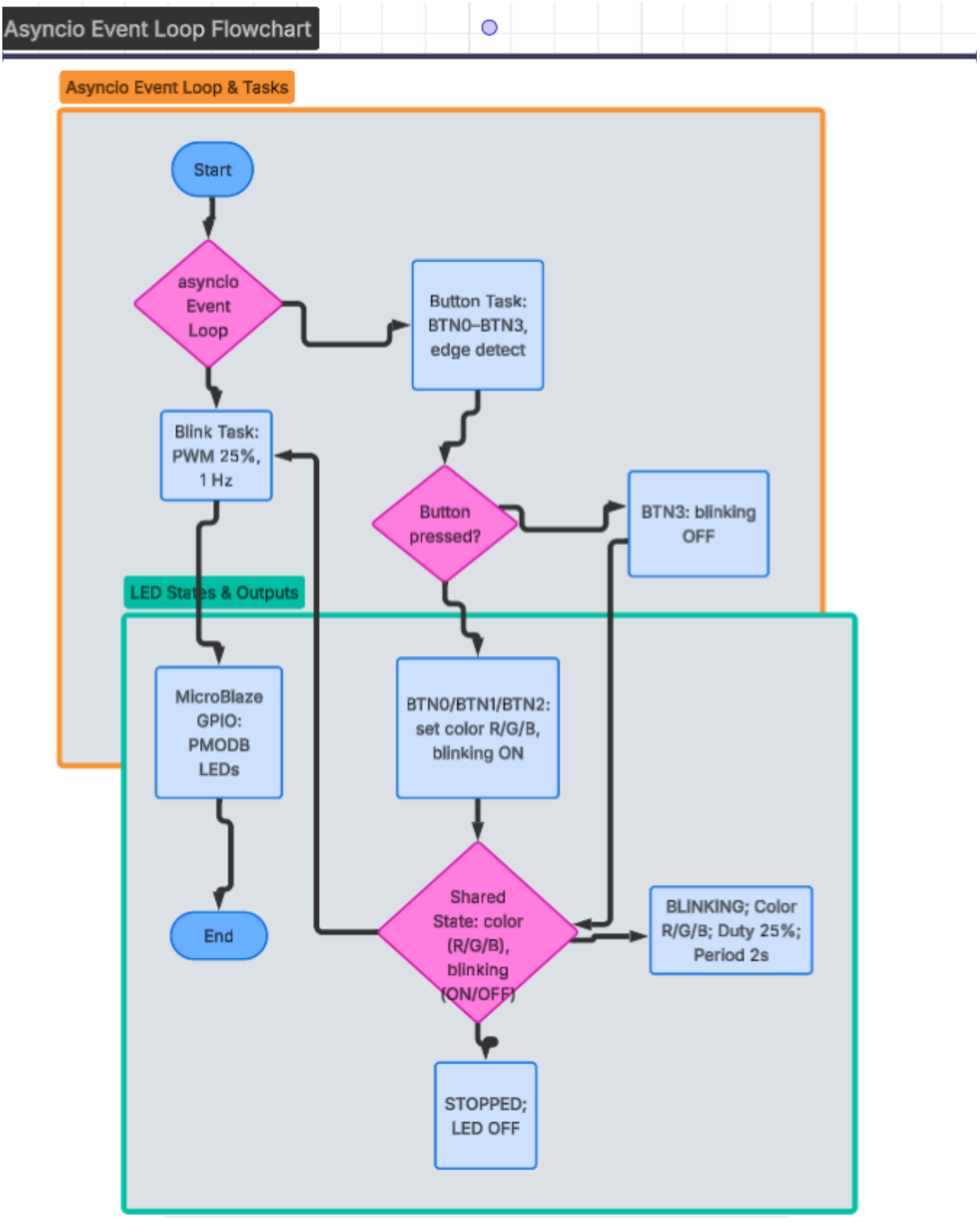

Figure (1) *System state and Asynchronous tasks*

4. Results and Observations

Experimental testing demonstrated that a PWM frequency of approximately 1 kHz eliminates visible flicker while remaining within software timing constraints. Duty cycles of 25%, 50%, 75%, and 100% produced progressively increasing brightness levels, with perceptual nonlinearity clearly observed.

```
Duty Cycle:   0%  →  Perceived Brightness:    0.0%
Duty Cycle:   5%  →  Perceived Brightness:   25.6%
Duty Cycle:  10%  →  Perceived Brightness:   35.1%
Duty Cycle:  25%  →  Perceived Brightness:   53.3%
Duty Cycle:  50%  →  Perceived Brightness:   73.0%
Duty Cycle:  75%  →  Perceived Brightness:   87.7%
Duty Cycle: 100%  →  Perceived Brightness:  100.0%
```

Figure(2) *Plot Data at a glance*

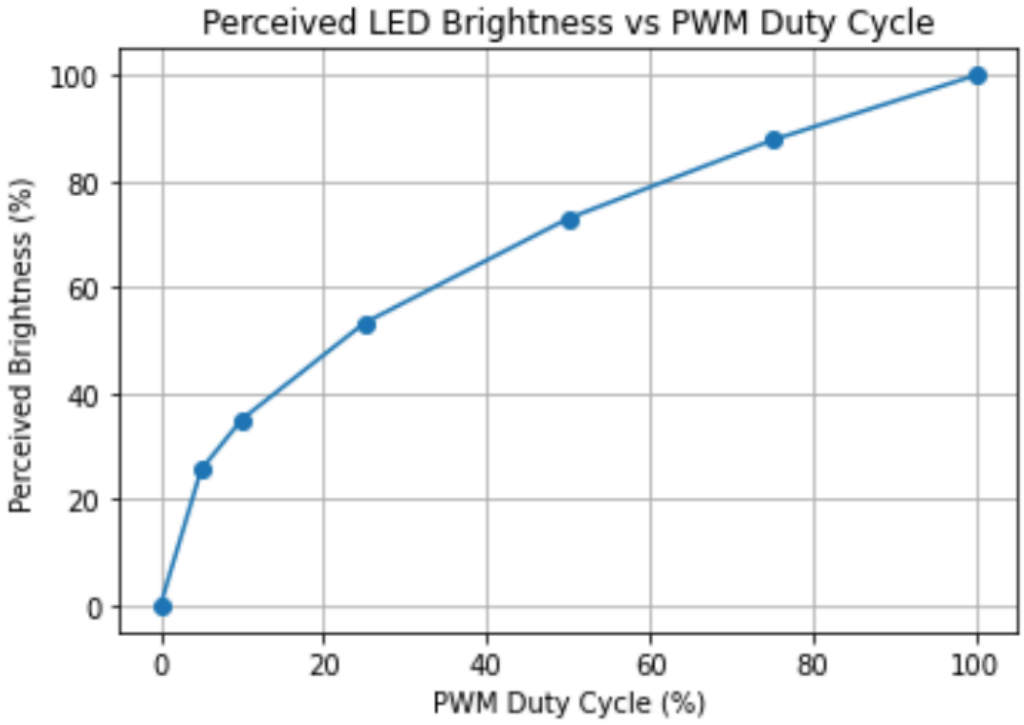

Figure (3) *Plot showing Perceived Brightness Vs PWM Duty Cycle*

The brightness curve above follows a gamma-corrected relationship **($\gamma \approx 2.2$)**, indicating that perceived brightness increases more slowly at low duty cycles and saturates at higher values.

5. Discussion and Troubleshooting

The primary implementation challenge involved improper coroutine invocation, which initially resulted in runtime warnings, stalled execution and the board consequently freezing due to being run for a long period of time. These issues were resolved by ensuring all coroutines were created and awaited or scheduled via asyncio in a main() routine within a single event loop.

6. Conclusion

This assignment successfully demonstrated software-based PWM generation, human-perceptual effects in LED brightness control, and asynchronous system design on an ZYNQ-Z2 board and the Jupyter Python3 embedded platform. The final implementation satisfies all functional requirements.