TITLE AND DATE

Document name: Project #2 Cyclic Executive Lab Report
Document reference: kiddKid.cmpe311.fall25.project#2
Date of publication: November 8, 2025

LEAD ENGINEER: Kid Kidd, UMBC

STAKEHOLDERS:

Prof Kidd, Instructor, University of Maryland Baltimore County, Baltimore, Maryland, USA
MD Safwan Zaman, TA, University of Maryland Baltimore County, Baltimore, Maryland, USA

HIGH-LEVEL DESCRIPTION: This document is the project document for Project#2 of the CMPE311 class. It contains customer, technical, and testing requirements as well as the design and results of the validation testing.

DESCRIPTION: This design is to create a circuit on an Arduino Uno R3 development platform that allows the user to independently specify an LED number and its blink interval. In this version, the blinking of the LEDs is managed through a Cyclic Executive that uses a function-pointer array to execute all system tasks—LED control and serial input—in a round-robin sequence. Each LED continues to blink at its specified interval while the program polls for new input, maintaining smooth, non-blocking operation throughout execution. Also included in this document are the customer requirements obtained from the customer, the high-level technical requirements derived from those customer requirements, the design, the testing scenarios and requirements, and the results of testing. Appended to this document is the code executed and a video of those tests that required video documentation.

RESULT SUMMARY: The project was a success, the embedded system design meeting all testing and high-level requirements.

REFERENCES AND GLOSSARY

REFERENCES:

- CMPE311 Project #1 – Asynchronous Execution Lab Report, Fall 2025
- Arduino UNO R3 Product Reference Manual SKU A000066, 12/03/2024
- Async Programming in Arduino: Unleashing the Power of Non-Blocking Code, Mahdi Valizadeh, medium.com, 4/8/2024

DEFINITIONS:

"The User" – The person operating (not programming) the embedded system
"The System" – The embedded system being operated by The User
"The Customer" – The person(s) paying for the embedded system being designed and built
"The Developer" – The person(s) designing and building the System
"The Evaluator" – The person(s) that determine whether or not The System satisfies The Customer-requirements.
"The Customer-requirements" – The requirements defined by The Customer as satisfying The Contract.
"The Requirements" – The System's high-level technical requirements derived from The Customer-requirements.
"The Educational-constraints" – Requirements imposed by the instructor unrelated to the embedded system that allow The System to be evaluated.
"The Company" – The organization The Customer has contracted with to build The System.
"The Contract" – The business document that legally binds The Company to provide some service or product to The Customer.
"serial-monitor" – The serial port used by the Arduino IDE to communicate with The User.
"The Reference-platform" – The configuration of The System used by The Developer to test and validate The System. For this class, The System is the Arduino compatible ELEGOO Uno R3 development board.

ACRONYMS AND ABBREVIATIONS:

Arduino – an Italian open-source hardware and software company; also refers to a development board created by the company
arduino.h – header for a library of convenience functions specific to the Arduino development platform
AVR – A family of microcontrollers, originally developed by Atmel, and currently owned by Microchip Technology
ELEGOO – A Chinese company that develops and markets 3D printers and accessories
IDE – Integrated Development Environment
gcc – front end for the GNU Compiler Collection
Github – A widely used distributed SVC (Software Version Control) system
LED – Light Emitting Diode

REQUIREMENTS

CONVENTIONS:

Must, shall or will – your design must satisfy the requirement
May – your design may satisfy the requirement but doesn't have to
Informative – the intent of the following description is to make the requirement more understandable
All customer requirements are started with "C.#".
All high-level requirements are started with "HL.#".
All testing/validation requirements are started with "T.#"

CUSTOMER REQUIREMENTS:

C1. The User must be able to set the blink rate of two different LEDs.
C2. The User must be able to update the blink rate of each of the LEDs independently.
C3. The LED must blink at the set rate until The User tells the LED to blink at a different rate.
C4. The System must run upon an Arduino Uno R3 compatible development board.

HIGH-LEVEL TECHNICAL REQUIREMENTS:

HL.1 The System must use at least 2 LEDs
HL.2 The System must use a standard Arduino compatible development board (e.g. the provided ELEGOO Uno R3)
HL.3 The System must communicate with The User only via the Arduino IDE serial-monitor port
HL.4 Any use of the serial-monitor in HL.3 must be asynchronous and not affect the blinking of the LEDs.
HL.5 The User must be able to set the blink interval of the LEDs in msec
HL.6 The blink rate of each LED must be constant unless changed by The User

DESIGN

DESIGN PRE-REQUISITES:

1. ELEGOO Arduino Uno R3 clone
2. Arduino IDE 2.3.3 or better

DEVELOPMENT PLATFORM:

1. See DESIGN PRE-REQUISITES above

ANY ADDITIONAL DESIGN CONSIDERATIONS:

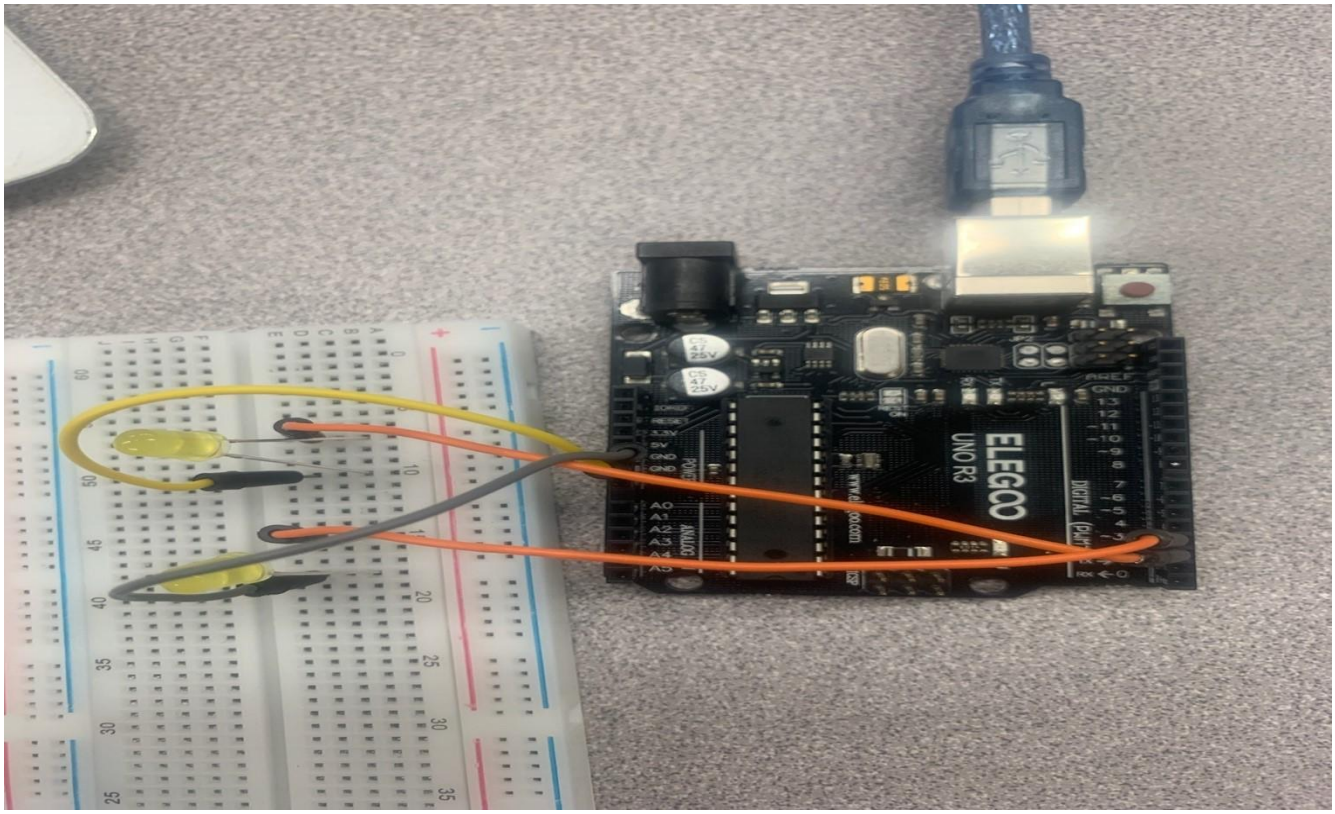
1. None

See Appendix A for the logical flow of the project program.
See Appendix B for the code executing on the Arduino Uno R3.
See Appendix C for the video of the test.

TESTING AND VALIDATION

DESCRIPTION: The tests that have to be performed and validated are shown in Table 2. These tests were performed on the testbed shown Figure 1. Table 1 shows a dialog that must be successfully performed by the embedded system. The results of testing are shown in Table 3. A video of the test is provided along with this report.

Figure 1. Testbed Setup. LEDs connected to digital pins 2 & 3



TESTING PLATFORM:

1. ELEGOO Arduino Uno R3 clone
2. Arduino IDE 2.3.3 or better
3. Power: Testing platform powered through USB cable, LEDs connected directly through Digital Outputs

Table 1. Required Test Dialog

Serial port I/O	**Notes**
Which LED? (2 or 3) **2**	
Enter interval (ms): **600**	LED2 starts blinking at an interval of 300ms on and 300ms off. LED3 is unaffected.
Which LED? (2 or 3) **3**	
Enter interval (ms): **1600**	LED3 starts blinking at an interval of 800ms on and 800ms off. LED2 is unaffected.
Which LED? (2 or 3)	Waiting for next LED# interval pair

TESTING AND VALIDATION REQUIREMENTS:

Table 2. Testing and Validation Requirements

T.0 All testing and validation must be done on the testbed illustrated in Figure 1. T.1 The dialog above (or similar) must work as shown T.1A The blink rate of an LED must be able to be set without interfering with the blinking of the other LED T.1.1 Setting of the blink rate (and LED#) must be through the IDE serial monitor T.2 The blink rate of the LED being set must not change until the input is complete T.3 The user must specify the blink rate in milliseconds per blink T.4 The blink rate specified by the user must be correctly reflected on the testbed LEDs. T.4.1 The blink rate specified by the user must be reflected on the LED specified by the user T.5. The setting of an LED's blink rate must be able to be repeated at least 5 times T.5.1. Successively for the same LED T.5.2. Alternately between the different LEDs

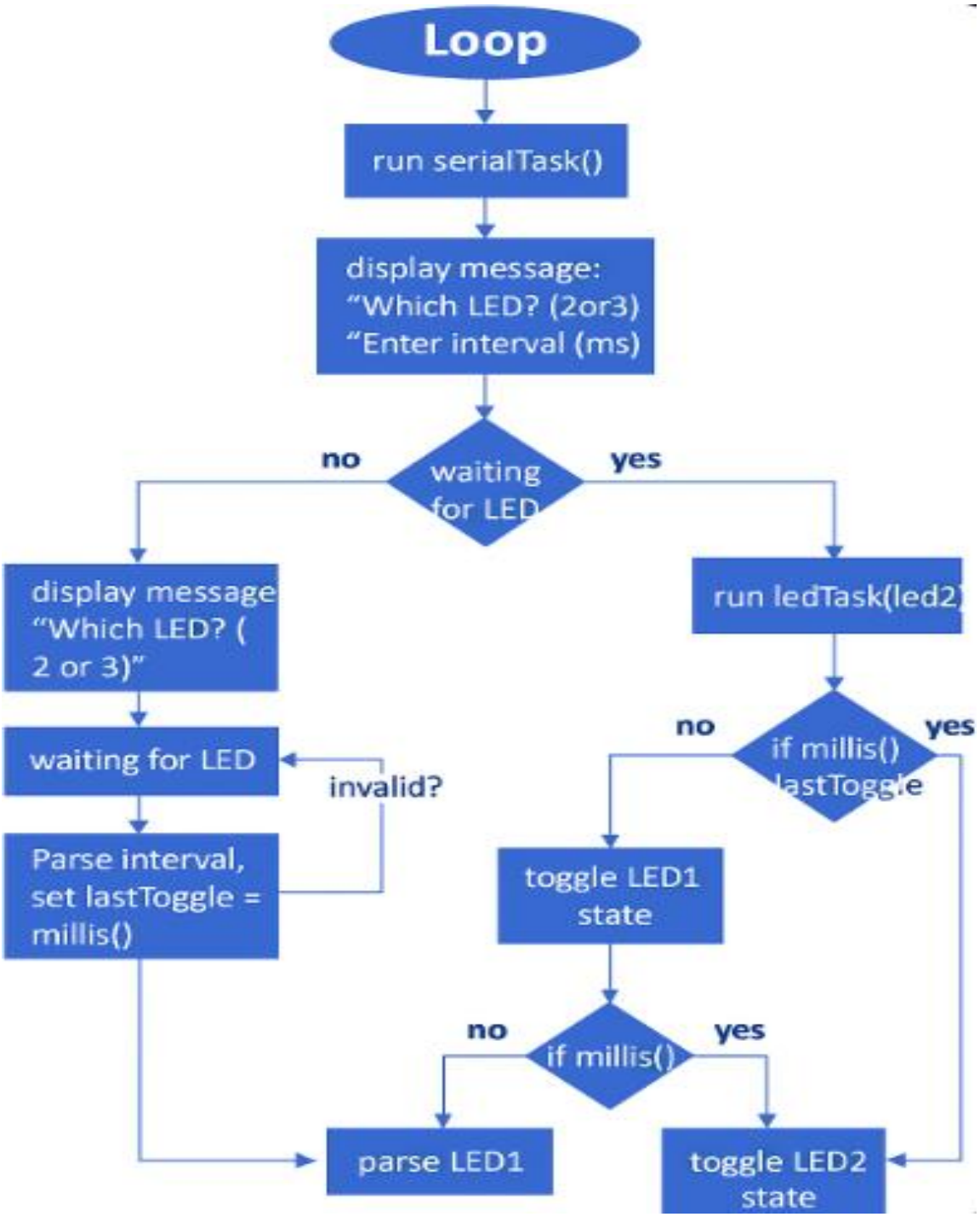
TESTING RESULTS:

Table 3. Result of tests

Test performed	Results
T.1.1	Satisfied. See video of test.
T.1	Satisfied
T.2	Satisfied. See video of test.
T.3	Satisfied
T.4	Satisfied
T.4.1	Satisfied. See video of test.
T.5.1	Satisfied. See video of test.
T.5.2	Satisfied. See video of test.

Appendix A. Logic Flow Chart

Figure 2. Logic flow for program



Appendix B. Design Code

```
#include <Arduino.h>

#define LED_PIN1 2
#define LED_PIN2 3

struct Led {
  int pin;
  unsigned long interval;
  unsigned long lastToggle;
  bool state;
};

Led led1 = { LED_PIN1, 0, 0, false };
Led led2 = { LED_PIN2, 0, 0, false };

int selectedLed = LED_PIN1;
int currentInterval = 0;
bool waitingForLed = true;
bool waitingForInterval = false;

void clearSerial() { while (Serial.available() > 0) Serial.read(); }

void ledTask(Led &l) {
  if (l.interval == 0) return;
  unsigned long now = millis();
  if (now - l.lastToggle >= l.interval) {
    l.state = !l.state;
    digitalWrite(l.pin, l.state ? HIGH : LOW);
    l.lastToggle = now;
  }
}

void serialTask() {
  static bool promptShown = false;
  if (!promptShown) {
    if (waitingForLed) Serial.println("Which LED? (2 or 3): ");
    else Serial.println("Enter interval (ms): ");
    promptShown = true;
  }
  if (Serial.available() == 0) return;

  if (waitingForLed) {
    int val = Serial.parseInt();
    clearSerial();
    if (val == LED_PIN1 || val == LED_PIN2) {
      selectedLed = val;
```

```
      waitingForLed = false;
      waitingForInterval = true;
      promptShown = false;
    } else {
      Serial.println("Invalid LED. Enter 2 or 3.");
      promptShown = false;
    }
  } else {
    int ms = Serial.parseInt();
    clearSerial();
    if (ms > 0) {
      if (selectedLed == LED_PIN1) { led1.interval = ms; led1.lastToggle = millis();
}
      if (selectedLed == LED_PIN2) { led2.interval = ms; led2.lastToggle = millis();
}
      waitingForLed = true;
      waitingForInterval = false;
      promptShown = false;
    } else {
      Serial.println("Invalid interval. Try again.");
      promptShown = false;
    }
  }
}

typedef void (*TaskFn)();
TaskFn tasks[] = { [](){ ledTask(led1); }, [](){ ledTask(led2); }, serialTask };
const int NUM_TASKS = 3;

void setup() {
  Serial.begin(9600);
  pinMode(LED_PIN1, OUTPUT);
  pinMode(LED_PIN2, OUTPUT);
  digitalWrite(LED_PIN1, LOW);
  digitalWrite(LED_PIN2, LOW);
}

void loop() {
  for (int i = 0; i < NUM_TASKS; i++) tasks[i]();
}
```

Appendix C. Video

https://www.youtube.com/watch?v=18kXo_K7TWU&t=28s