TITLE AND DATE

Document name: Project#2 Round-Robing Execution Lab Report
Document reference: PROJECT-CE
Date of publication: Nov. 11th, 2025

LEAD ENGINEER: Rachel Pallipamula-Niranjeen

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 #1 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 of the program to independently specify an LED number and its blink interval. The blinking of the LEDs must continue asynchronously with any input from the user. 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:

- ProjectAsynTasking_draft – The definition of the project this document addresses
- CMPE310 Project #5 – A similar project assigned in CMPE310 in Spring 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: *<list the customer requirements which I gave you>*

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.
C5. The blink rate of an LED must be expressed in terms of milliseconds

HIGH-LEVEL TECHNICAL REQUIREMENTS: *<list the derived high-level technical requirements>*

HL.1. The User through the IDE serial monitor must be able to set the blink rate of two different LEDs. (*From requirement C1)*
HL.1.1. The blink rate of each LED must be able to be set independent of the other LED. (*From requirement C2)*
HL.1.2. The setting of an LED blink rate must not interfere with the blinking of the LEDs until the new LED and blink rate are specified. (*From requirements C2 and C3)*
HL.2. The user through the IDE serial monitor must set the blink rate in terms of once every N milliseconds (*From requirement C5)*
HL.3. The System must run upon an Arduino Uno R3 compatible development board. (*From requirement C4)*

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

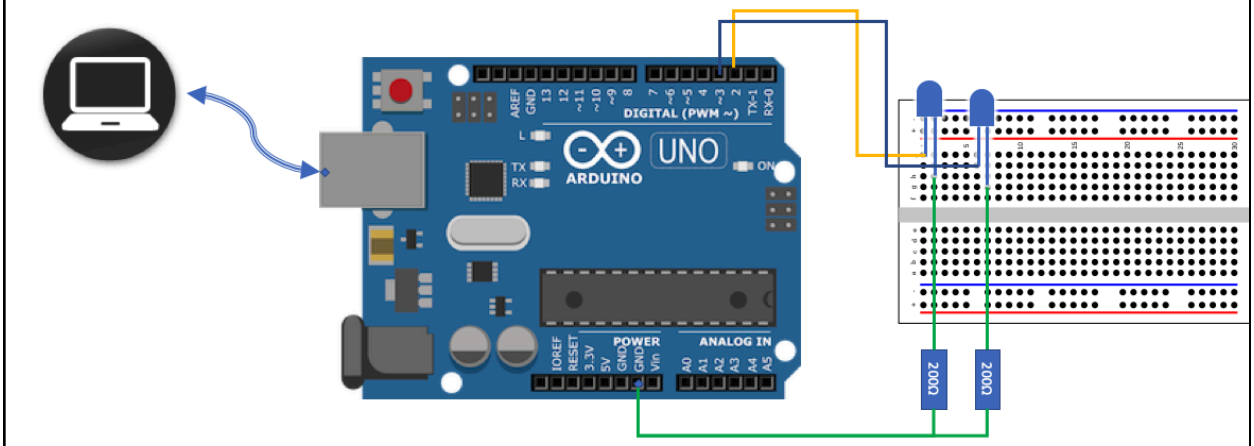
See Appendix A for the logical flow of the project program.
See Appendix B for the code executing on the Arduino Uno R3.

TESTING AND VALIDATION

<list requirements derived and any scenarios used. Including any testbed and other information need to perform the testing>

DESCRIPTION: The tests that have to be performed and validated are shown in Table xxx. 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. When necessary, a video of the test is provided along with this report.

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



TESTING PLATFORM: *<include any equipment, SW, etc necessary to reproduce your design environment>*

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 (***blue*** are the user inputs).

Serial port I/O	**Notes**
What LED? (8 or 2)	
What interval (in msec)? ***300***	LED2 starts blinking at an interval of 300ms on and 300ms off. LED1 is unaffected.
What LED? (1 or 2) ***1***	
What interval (in msec)? ***150***	LED1 starts blinking at an interval of 150ms on and 150ms off. LED2 is unaffected.
What LED? (1 or 2)	Waiting for next LED# interval pair

TESTING AND VALIDATION REQUIREMENTS: *<include all tests using appropriate language, e.g. must, so that there is no ambiguity>*

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:

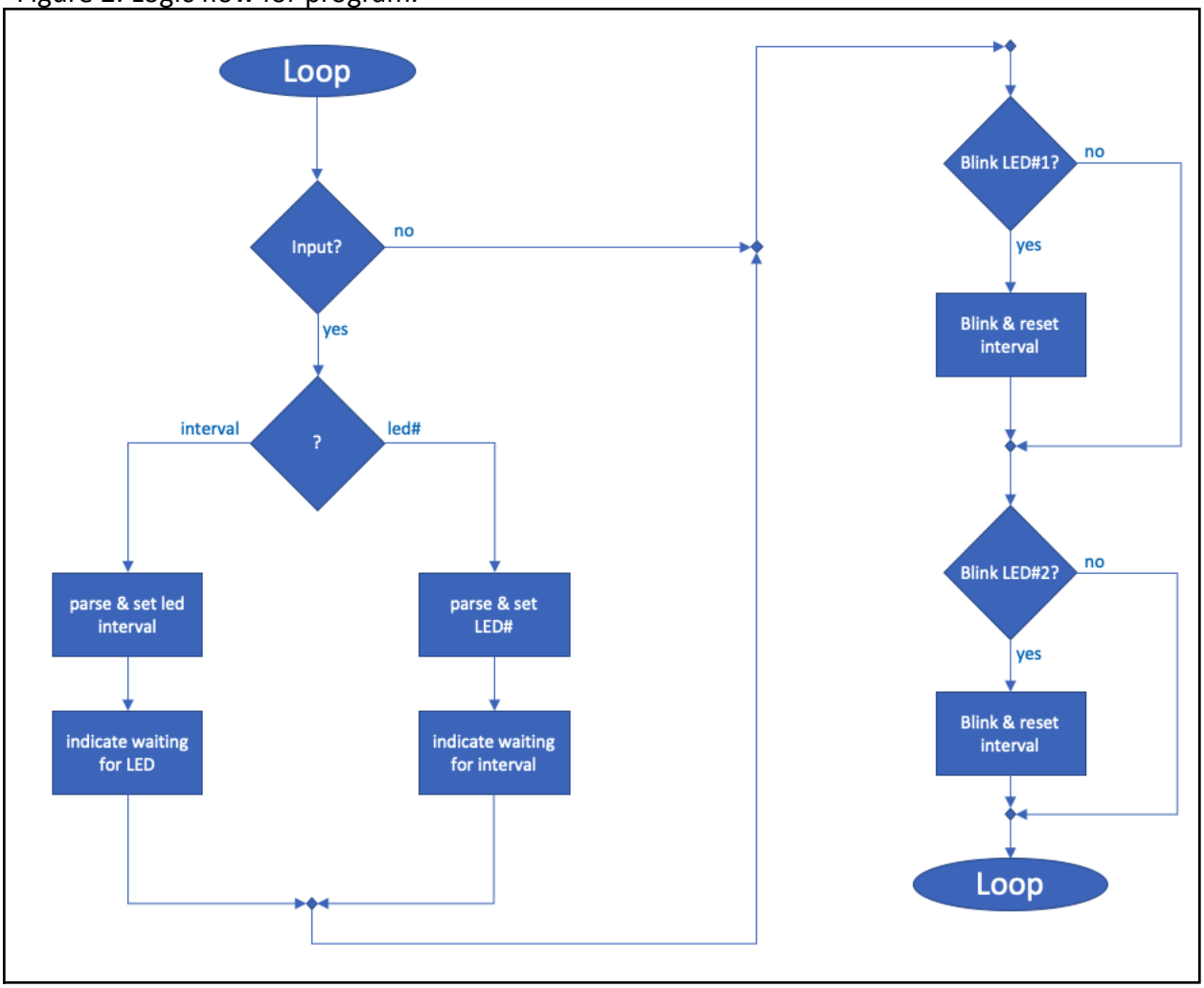
Link to the video of reults: 311Lab1.mp4

Table 3. Results 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

C/C++

```
#define LED_1_PIN 12
#define LED_2_PIN 11

typedef void (*TaskFunction)();
TaskFunction taskList[3];

unsigned long previousTimeLED1 = 0;
unsigned long previousTimeLED2 = 0;
unsigned long LED1Interval = 500;
unsigned long LED2Interval = 1000;

byte LED1State = LOW;
byte LED2State = LOW;

String inputBuffer = "";
int targetLED = 0;
int newInterval = 0;
bool inputReady = false;

void setup() {
  Serial.begin(9600);
  pinMode(LED_1_PIN, OUTPUT);
  pinMode(LED_2_PIN, OUTPUT);

  taskList[0] = taskReadSerial;
  taskList[1] = taskBlinkLED1;
  taskList[2] = taskBlinkLED2;

  Serial.println("Enter: <LED number> <interval>");
  Serial.println("Example: 1 750");
}

void loop() {
  for (int i = 0; i < 3; i++) {
    taskList[i]();
  }
}

// Task 1: Read and parse serial input
void taskReadSerial() {
  while (Serial.available()) {
    char c = Serial.read();
    if (c == '\n') {
      inputReady = true;
```

```
    } else if (isPrintable(c)) {
      inputBuffer += c;
    }
  }

  if (inputReady) {
    inputReady = false;
    inputBuffer.trim();
    int spaceIndex = inputBuffer.indexOf(' ');
    if (spaceIndex > 0) {
      targetLED = inputBuffer.substring(0, spaceIndex).toInt();
      newInterval = inputBuffer.substring(spaceIndex + 1).toInt();

      if (targetLED == 1) {
        LED1Interval = newInterval;
        Serial.print("LED 1 interval set to ");
        Serial.println(LED1Interval);
      } else if (targetLED == 2) {
        LED2Interval = newInterval;
        Serial.print("LED 2 interval set to ");
        Serial.println(LED2Interval);
      } else {
        Serial.println("Invalid LED number.");
      }
    } else {
      Serial.println("Invalid input. Use format: 1 500");
    }
    inputBuffer = "";
  }
}

// Task 2: Blink LED 1
void taskBlinkLED1() {
  unsigned long now = millis();
  if (now - previousTimeLED1 >= LED1Interval) {
    previousTimeLED1 = now;
    LED1State = !LED1State;
    digitalWrite(LED_1_PIN, LED1State);
  }
}

// Task 3: Blink LED 2
void taskBlinkLED2() {
  unsigned long now = millis();
  if (now - previousTimeLED2 >= LED2Interval) {
    previousTimeLED2 = now;
    LED2State = !LED2State;
    digitalWrite(LED_2_PIN, LED2State);
  }
}
```

END OF DOCUMENT