Hazael Magino
9/25/2025

Document Name: Project#1 Asynchronous Execution Lab Report

Document Reference: kiddKid.cmpe311.fall25.project1

Date of publication: September 23, 2025

Lead Engineer: Hazael Magino, UMBC

Stakeholders:

Prof Kidd, Instructor, University of Maryland Baltimore County, Baltimore, Maryland, USA

MD Safwan Zaman, TA, University of Maryland Baltimore County, Baltimore, Maryland, US

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 link to 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.

Hazael Magino
9/25/2025

REFERENCES AND GLOSSARY

References:

- ProjectAsynTasking_draft – The definition of this document addresses
- Arduino UNO R3 Product Reference Manual SKU A000066, 9/23/2025

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

Hazael Magino
9/25/2025

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 ar5e 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.

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

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 Prerequisites:

1. ELEGOO Arduino Uno R3 clone
2. Arduino IDE 2.3.3 or better
3. LED Lights (2x)
4. 200 Ohm R (2x)

Development Platform:

1. See DESIGN PRE-REQUISITES above

Additional Design Considerations

See Appendix A for the code on the Arduino Uno R3

TESTING AND VALIDATION

Description:

The tests that must be performed and validated are shown in Table 2. These tests were performed on the test bed shown in 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.

Hazael Magino
9/25/2025

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

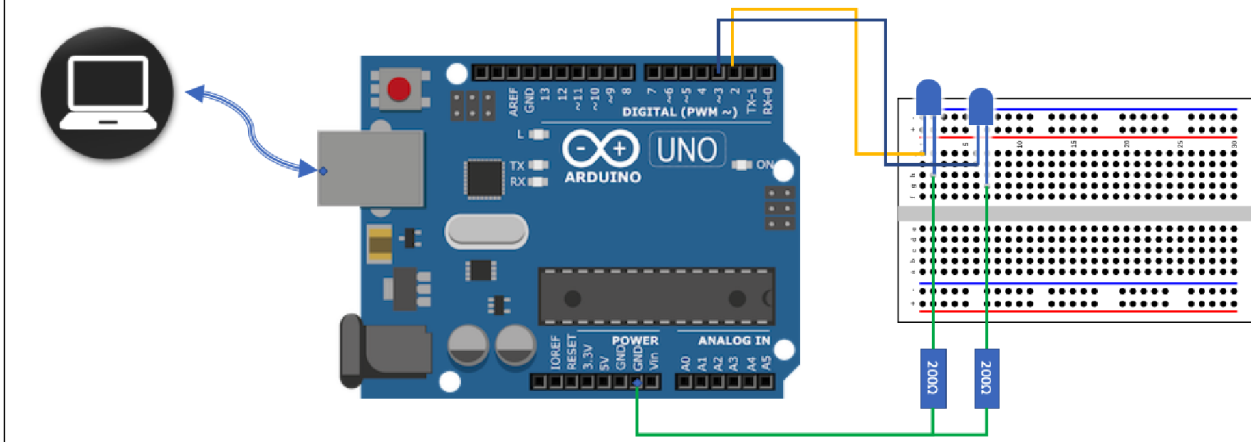

Table 1. Required Test Dialog (blue are the user inputs)

Serial port I/O	Notes
What LED? (1 or 2) 2	LED2 starts blinking at an interval of 300ms on and 300ms off. LED1 is unaffected.
What interval (in msec)? 1000	
What LED? (1 or 2) 1	
What interval (in msec)? 1600	LED1 starts blinking at an interval of 800ms on and 800ms off. LED2 is unaffected
What LED? (1 or 2)	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 2

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. Design Code:

Table 3

```
#define PIN_LED1 2  // LED #1
#define PIN_LED2 3  // LED #2

// unsigned long for time
unsigned long prevLed1Ms = 0;
unsigned long prevLed2Ms = 0;

// User Input Variables
int ledSelection = 0; // Waiting for which LED to change

// 0 = waiting for LED number
// 1 = waiting for interval
int inputState = 0;

// Store Led states
int led1State = LOW;
int led2State = LOW;

// Default intervals
long led1Interval = 1000;
long led2Interval = 500;
```

Hazael Magino
9/25/2025

```
void setup() {
 Serial.begin(9600); //set baud rate
 pinMode(PIN_LED1, OUTPUT); //define pins
 pinMode(PIN_LED2, OUTPUT);

 digitalWrite(PIN_LED1, LOW); //define starting states
 digitalWrite(PIN_LED2, LOW);

 Serial.println("Which LED do you want to change (1 or 2)?");
}

void loop() {
 unsigned long currentTime = millis(); //store times

 // Asynchronously handle serial input
 if (Serial.available() > 0) {
  switch (inputState) {
   case 0: { // Waiting for user to select an LED
    int ledNum = Serial.parseInt();

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

    if (ledNum == 1 || ledNum == 2) {
     ledSelection = ledNum; // Store the chosen LED
     Serial.println("What interval (in ms)?");
     inputState = 1; // Move to the next state
    }
    break;
   }
   case 1: { // Wait for the interval
    long newInterval = Serial.parseInt();

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

    if (newInterval > 0) {
     if (ledSelection == 1) {
      led1Interval = newInterval;
     } else if (ledSelection == 2) {
      led2Interval = newInterval;
```

```
        }
        Serial.println("New interval has been set.");
        Serial.println("-----------------------------------");
        Serial.println("Which LED do you want to change (1 or 2)?");
        inputState = 0; // Reset for new led selection
       }
       break;
     }
   }
  }

  // LED 1 control
  if (currentTime - prevLed1Ms >= led1Interval) {
   prevLed1Ms = currentTime;


   if (led1State == LOW) {
    led1State = HIGH;
   } else {
    led1State = LOW;
   }

   digitalWrite(PIN_LED1, led1State); //change the states for led 1
  }

  //LED 2 control
  if (currentTime - prevLed2Ms >= led2Interval) {
   prevLed2Ms = currentTime;

   if (led2State == LOW) {
    led2State = HIGH;
   } else {
    led2State = LOW;
   }

   digitalWrite(PIN_LED2, led2State); //change the states for led 2
  }
}
```

End of Document