Hazael Magino
9/25/2025

Document Name: Project#2 Asynchronous Execution Lab Report

Document Reference: projectTaskMgmtCE.pdf

Date of publication: November 10, 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 #2 (PROJECT-CE) of the CMPE311 class. It is an update to the previous Project#1 Asynchronous Execution Lab Report (Ref: kiddKid.cmpe311.fall25.project1).

This document contains the original customer, technical, and testing requirements, as well as the new design, educational requirements, and results of the validation testing for the Cyclic Executive implementation.

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 using a polled and round-robin Cyclic Executive task manager based upon a function pointer array. 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, utilizing a cyclic executive task manager, met all customer, technical, and testing requirements, performing identically to the original PROJECT-ASYNC implementation.

REFERENCES AND GLOSSARY

References:

- **Project-CE Lab Definition** – The definition of this document addresses.
- **PROJECT-ASYNC Report (ProjectAsynTasking_draft)** - The previous project upon which this project is based (Ref: kiddKid.cmpe311.fall25.project1).
- **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

“PROJECT-ASYNC” – The previous project with this is based on.

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:

Hazael Magino
9/25/2025

The customer requirements are identical to that of PROJECT-ASYNC. (Ref: kiddKid.cmpe311.fall25.project1).

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:

The high-level technical requirements are identical to that of PROJECT-ASYNC.

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)

Educational Requirements:

1.0 Implement a cyclic executive task manager to dispatch the asynchronous tasks you created in the previous project, PROJECT-ASYNC.

1.1 The system testing and validation requirements must contain those defined in PROJECT-ASYNC.

1.1.1 The testing and validation requirements must contain any additional tests necessary to properly evaluate/demonstrate the function of the system.

1.2 The final report must be a formal design document as in PROJECT-ASYNC.

Hazael Magino
9/25/2025

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

Design Implementations

This design fulfills the educational requirement by refactoring the code from PROJECT-ASYNC into a Cyclic Executive (CE) task manager.

1. Task Encapsulation: The three main asynchronous processes from the original project were encapsulated into their own dedicated functions:
    - void task_LED1(): Manages the state and timing for LED 1.
    - void task_LED2(): Manages the state and timing for LED 2.
    - void task_SerialCheck(): Manages the serial input state machine for receiving user commands. (See Project_CE_HM.c)
2. Function Pointer Array: A function pointer array, taskList[], is defined to act as the task manager's list of tasks to be executed. Each element in the array points to one of the encapsulated task functions.

    C

    ```
    void (*taskList[])() = {
     task_LED1,
     task_LED2,
     task_SerialCheck
    };
    ```

3. Cyclic Executive Scheduler: The main Arduino loop() function is now implemented as a simple, round-robin scheduler. It continuously iterates through the taskList and executes each task function one by one.

    C

```
void loop() {
  // Store current time once per cycle
  currentTime = millis();

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

This CE design maintains the non-blocking, asynchronous behavior of the original project. Each task function still relies on the millis() timer and interval checking to manage its own state, ensuring no task "blocks" the others.

TESTING AND VALIDATION

Description:

The tests performed are identical to those in PROJECT-ASYNC, as the functional requirements have not changed. The tests were performed on the test bed shown in Figure 1. Table 1 shows the required dialog that must be successfully performed.

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

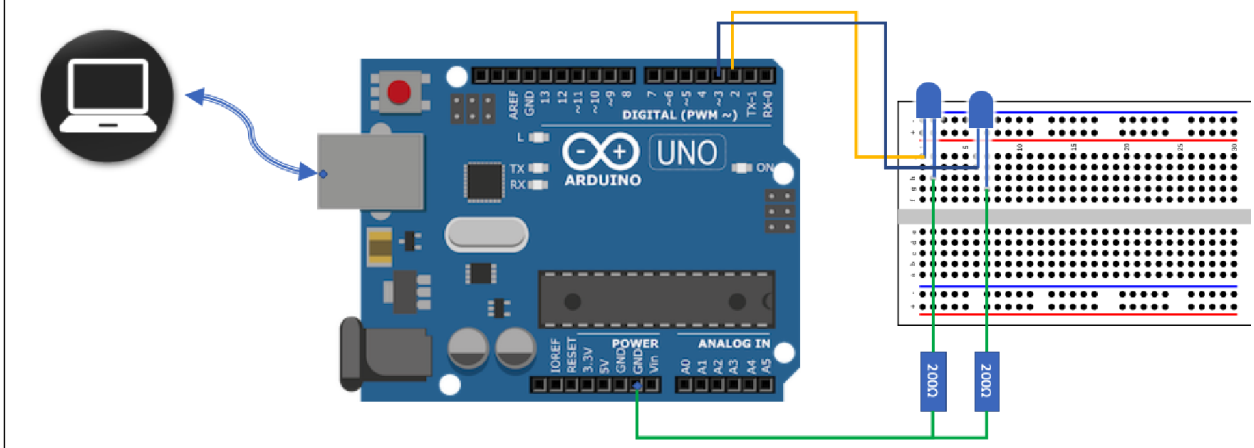

Table 1. Required Test Dialog (blue are the user inputs)

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

The testing requirements are identical to PROJECT-ASYNC, per educational requirement 1.1

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:

Hazael Magino
9/25/2025

Table 3

```
#define PIN_LED1 2  // LED #1
#define PIN_LED2 3  // LED #2
#ifndef LOW
#define LOW  0
#endif
#ifndef HIGH
#define HIGH 1
#endif

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

// function prototyping
void task_LED1();
void task_LED2();
void task_SerialCheck();

// User Input Variables
int ledSelection = 0; // Waiting for which LED to change
unsigned long currentTime = 0; // Just declare it here

// 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;

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 task_LED1(){

  // 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
 }

}

void task_LED2(){

 //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
 }

}

void task_SerialCheck(){

 // 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(" nterval 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;
    }
   }
  }

}

void (*taskList[])() = {
 task_LED1,
 task_LED2,
 task_SerialCheck
};
```

Hazael Magino
9/25/2025

```
void loop() {

 // Get the current time at the start of the loop
 currentTime = millis();

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

}
```

End of Document