TITLE AND DATE

Document name: Project#2 Cyclic Executive Project Report
Document reference: ChrisMiddleton.cmpe311.fall25.project#2
Date of publication: November 11th, 2025

LEAD ENGINEER: Chris Middleton, 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 documentation for Project#2 of the CMPE311 class. It contains customer, technical, and testing requirements and details the design and results of the validation testing.

DESCRIPTION: The specific design of the project is to create a circuit on an Arduino Uno R3 development platform allowing the user of the program to separately specify an LED and set the blink interval in milliseconds. The difference between this project and Project#1 is not in functionality, but how the task is executed. As before, the blinking must be un-interrupted by user input and must continue blinking indefinitely after interval specified, but is different in that it uses a cyclic executive tasking system. This document contains customer requirements, the high-level technical requirements derived from those customer requirements, the design of the project, and the testing scenarios, requirements, and results. This document also contains the code executed on the Arduino Uno R3.

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

REFERENCES AND GLOSSARY

REFERENCES:

- projectTaskMgmtCE_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
- CMPE311 Project #1 – The project that defines the functionality of project #2, done on 9/25/2025

***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 upon which this project is based

The definitions, "The User" through "PROJECT-ASYNC", are identical to that of PROJECT-ASYNC. Ref: ChrisMiddleton.cmpe311.fall25.project#1, DEFINITIONS section.

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

The acronyms, “Arduino” through “LED”, are identical to that of PROJECT-ASYNC.
Ref: ChrisMiddleton.cmpe311.fall25.project#1, ACRONYMS section.

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

The customer requirements, C1 through C5, are identical to that of PROJECT-ASYNC.
Ref: ChrisMiddleton.cmpe311.fall25.project#1, CUSTOMER REQUIREMENTS section.

EDUCATIONAL REQUIREMENTS:
E1. Implement a cyclic executive task manager to dispatch the asynchronous tasks created in the previous project, PROJECT-ASYNC.
E1.1 The system testing and validation requirements must contain those defined in PROJECT-ASYNC.
E1.1.1 The testing and validation requirements must contain any additional tests necessary to properly evaluate/demonstrate the function of the system.
E1.2 The final report must be a formal design document as in PROJECT-ASYNC.

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)*

HL.4. The System must be implemented such that each task is cyclically executed (*From requirement E1*)

The high-level technical requirements, HL.1 through HL.3, are identical to that of PROJECT-ASYNC.

Ref: ChrisMiddleton.cmpe311.fall25.project#1, HIGH-LEVEL TECHNICAL REQUIREMENTS section.

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:

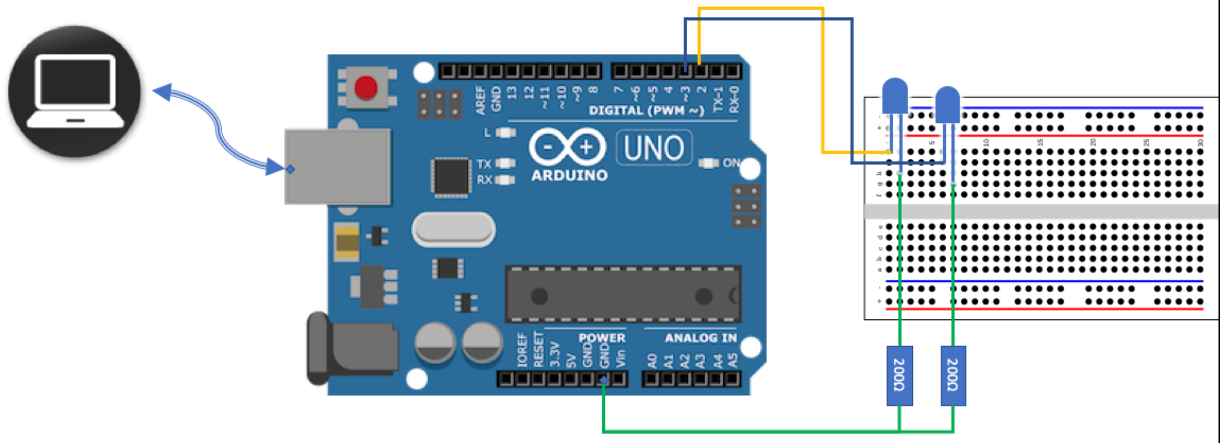
1. None

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

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:

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

The testing platform, 1 through 2, and design of the circuit are identical to that of PROJECT-ASYNC.
Ref: ChrisMiddleton.cmpe311.fall25.project#1, TESTING PLATFORM and DESIGN sections.

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

Serial port I/O	**Notes**
What LED? (1 or 2) ***2***	
What interval (in msec)? ***600***	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)? ***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

The testing and validation requirements, T.0 through T.5.2, are identical to that of PROJECT-ASYNC.
Ref: ChrisMiddleton.cmpe311.fall25.project#1, TESTING PLATFORM and DESIGN sections.

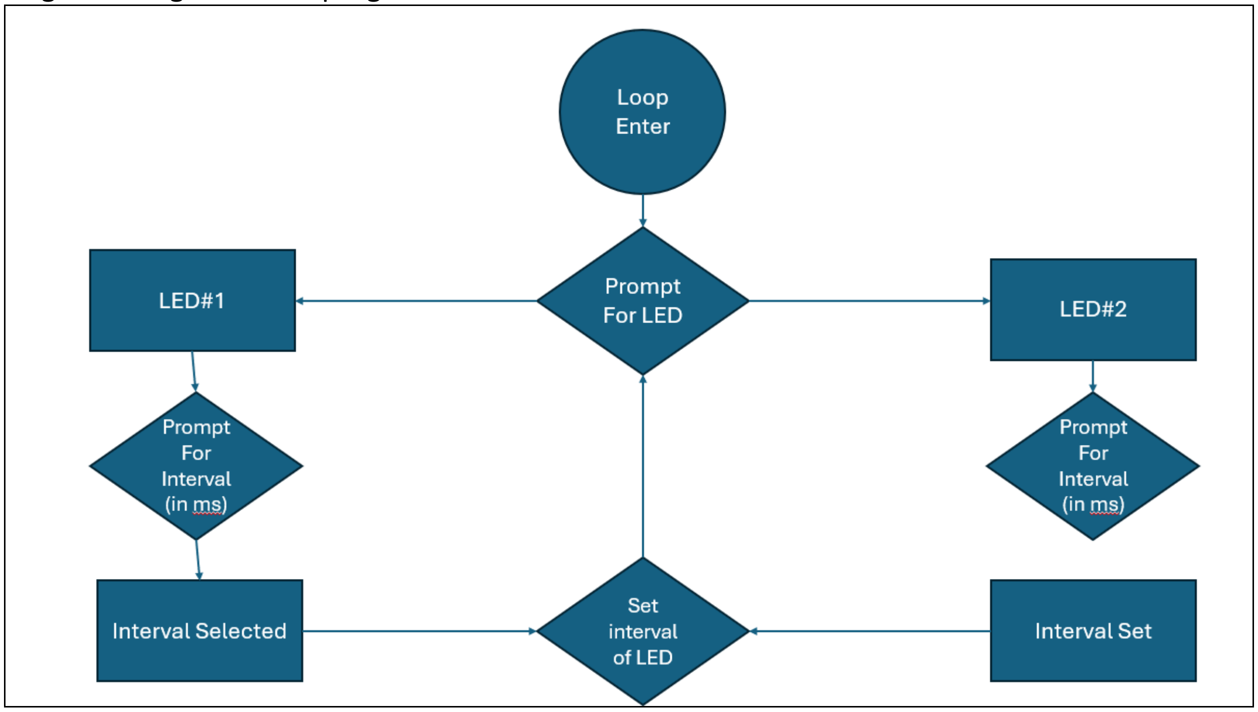
TESTING RESULTS:

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.



The logic flow chart is identical to that of PROJECT-ASYNC.
Ref: ChrisMiddleton.cmpe311.fall25.project#1, Appendix A..

APPENDIX B. DESIGN CODE

```
//led1
const int LED_1_PIN = 2;
unsigned long previousTimeLED1Blink = 0;
unsigned long LED1BlinkDelay = 0;
byte LED1State = LOW;

//led2
const int LED_2_PIN = 3;
unsigned long previousTimeLED2Blink = 0;
unsigned long LED2BlinkDelay = 0;
byte LED2State = LOW;

bool LED1Start = false;
bool LED2Start = false;

int serialState = 1;
int currentLED = 0;

int readFix = 0;

//functions to be executed cyclicly
```

```
//led1
const int LED_1_PIN = 2;
unsigned long previousTimeLED1Blink = 0;
unsigned long LED1BlinkDelay = 0;
byte LED1State = LOW;

//led2
const int LED_2_PIN = 3;
unsigned long previousTimeLED2Blink = 0;
unsigned long LED2BlinkDelay = 0;
byte LED2State = LOW;

bool LED1Start = false;
bool LED2Start = false;

int serialState = 1;
int currentLED = 0;

int readFix = 0;

//functions to be executed cyclicly
void led1();
void led2();
void userPrompt();

//define array for functions to be stored in
typedef void (*function)(void);

//define functions to be put in array
function func[3] = {&led1, &led2, &userPrompt};

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(LED_1_PIN,OUTPUT);
  pinMode(LED_2_PIN,OUTPUT);
  //needs to print this once to start rest of prompts
  Serial.println("What LED (1 or 2)?");
}

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

void userPrompt(){
  // put your main code here, to run repeatedly:
  if(Serial.available() > 0){
    //had to do this to stop serial from executing twice
    readFix++;
    if(readFix%2 == 0){
      readFix = 0;
    }

    //Takes user input for LED
    bool run = false;
    int userInput = Serial.parseInt();
    //checks if input in Serial
    //prevents accidental trigger
    if(serialState == 1 && !run && readFix%2 == 1){
      if(userInput == 1){
        currentLED = 1;
      }
      else if(userInput == 2){
        currentLED = 2;
      }
      serialState = 2;

      Serial.println("What interval (in msec?");
      run = true;
    }

    //checks if input in Serial
    //prevents accidental trigger
    //Takes user input for time delay
    else if(serialState == 2 && !run && readFix %2 == 1){
      if(currentLED == 1){
        LED1Start = true;
        LED1BlinkDelay = userInput;
      }
      else if(currentLED == 2){
        LED2Start = true;
        LED2BlinkDelay = userInput;
      }
      serialState = 1;
      Serial.println("What LED (1 or 2)?");
      run = true;
    }
  }
}
void led1(){
  //current time using millis
  unsigned long timeNow = millis();

  //LED 1
  if(LED1Start && (timeNow - previousTimeLED1Blink > LED1BlinkDelay)){
    previousTimeLED1Blink += LED1BlinkDelay;
    //checks condition of LED, sets accordingly
    if(LED1State == HIGH){
      LED1State = LOW;
    }
    else{
      LED1State = HIGH;
    }
    digitalWrite(LED_1_PIN,LED1State);
  }
}

void led2(){
  //current time using millis
  unsigned long timeNow = millis();
  //LED 2
  if(LED2Start && (timeNow - previousTimeLED2Blink > LED2BlinkDelay)){
    previousTimeLED2Blink += LED2BlinkDelay;
    //checks condition of LED, sets accordingly
    if(LED2State == HIGH){
      LED2State = LOW;
    }
    else{
      LED2State = HIGH;
    }
    digitalWrite(LED_2_PIN,LED2State);
  }
}
```

APPENDIX C. VIDEO OF TEST

https://drive.google.com/file/d/1L9NerrgwUu1fe7bw9wQayz0EQZW3BWul/view?usp=sharing

END OF DOCUMENT