TITLE AND DATE

Document name: CMPE311Project2_CyclicExecutive
Document reference: kiddKid.cmpe311.fall25.project#2
Date of publication: November 11, 2025

LEAD ENGINEER: Tyler Martin, University of Maryland Baltimore County, Baltimore, Maryland, USA

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 details the information regarding the Cyclic Executive Task Manager Project, an expansion on the Asynchronous Tasking project now using a cyclic executive. Below are the customer, technical, and testing requirements as well as the design and results of the validation testing.

DESCRIPTION: The purpose of the project is to create a design using resistors, LEDs, and the Arduino Uno R3, along with its development platform, that is able to blink LEDs independently and cycle through blinking LED tasks and duration setting tasks through a cyclic executive. That is, the design features LEDs with configurable blink intervals that do not affect one another; the user chooses which LED, and how long their blink intervals will be. The blinking of the LEDs must continue asynchronously with any input from the user. These tasks must be functionized and referenced through a task manager that goes through them, a cyclic executive. 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. A video demonstrating the process is submitted through a google drive link and included in the appendix.

RESULT SUMMARY: The LEDs successfully blinked asynchronously with input. Their intervals were modifiable without stopping the actual blinking and changing the interval for one did not affect the other. Additionally, the implementation of the cyclic executive task manager succeeded with no problems.

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 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.
HL.1.1. The blink rate of each LED must be able to be set independent of the other LED.
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.
HL.2. The user through the IDE serial monitor must set the blink rate in terms of once every N milliseconds.
HL.3. The System must run upon an Arduino Uno R3 compatible development board.
HL.4. Must use a round-robin cyclic executive task manager based upon a function pointer array.

DESIGN:

DESIGN PRE-REQUISITES:

1. ELEGOO Arduino Uno R3 clone
2. Arduino IDE 2.3.3 or better
3. LEDs
4. BreadBoard
5. Resistors

DEVELOPMENT PLATFORM:

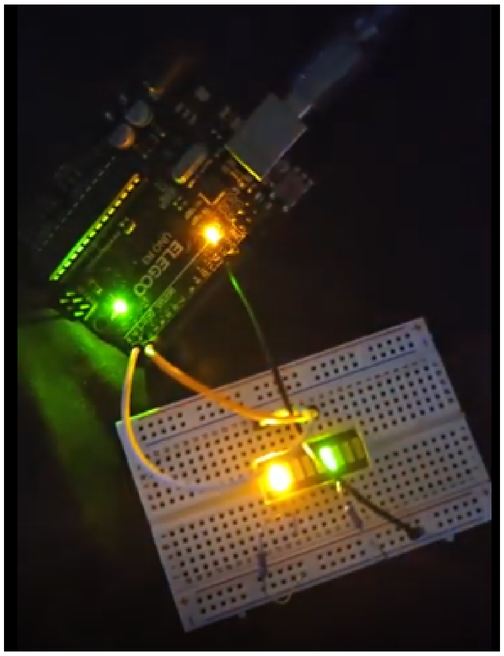
1. Arduino IDE 2.3.3 or better

ANY ADDITIONAL DESIGN CONSIDERATIONS:

There are no additional design considerations

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

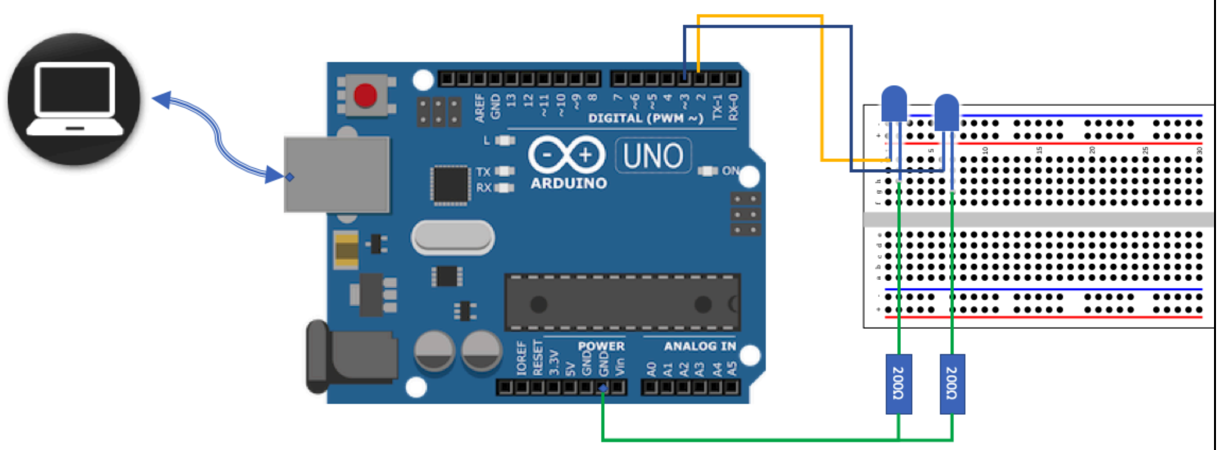
Figure 1. Physical Design of the Circuit. LEDs are connected to pins 2 and 3



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 2. 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
4. Resistors
5. Breadboard

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

Serial port I/O	**Notes**
What LED? (1 or 2) ***2***	Waiting for next LED# interval pair
What interval (in msec)? ***600***	LED2 starts blinking at an interval of 600ms on and 600ms off. LED1 is unaffected.
What LED? (1 or 2) ***1***	Waiting for next LED# interval pair
What interval (in msec)? ***1600***	LED1 starts blinking at an interval of 1600ms on and 1600ms 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

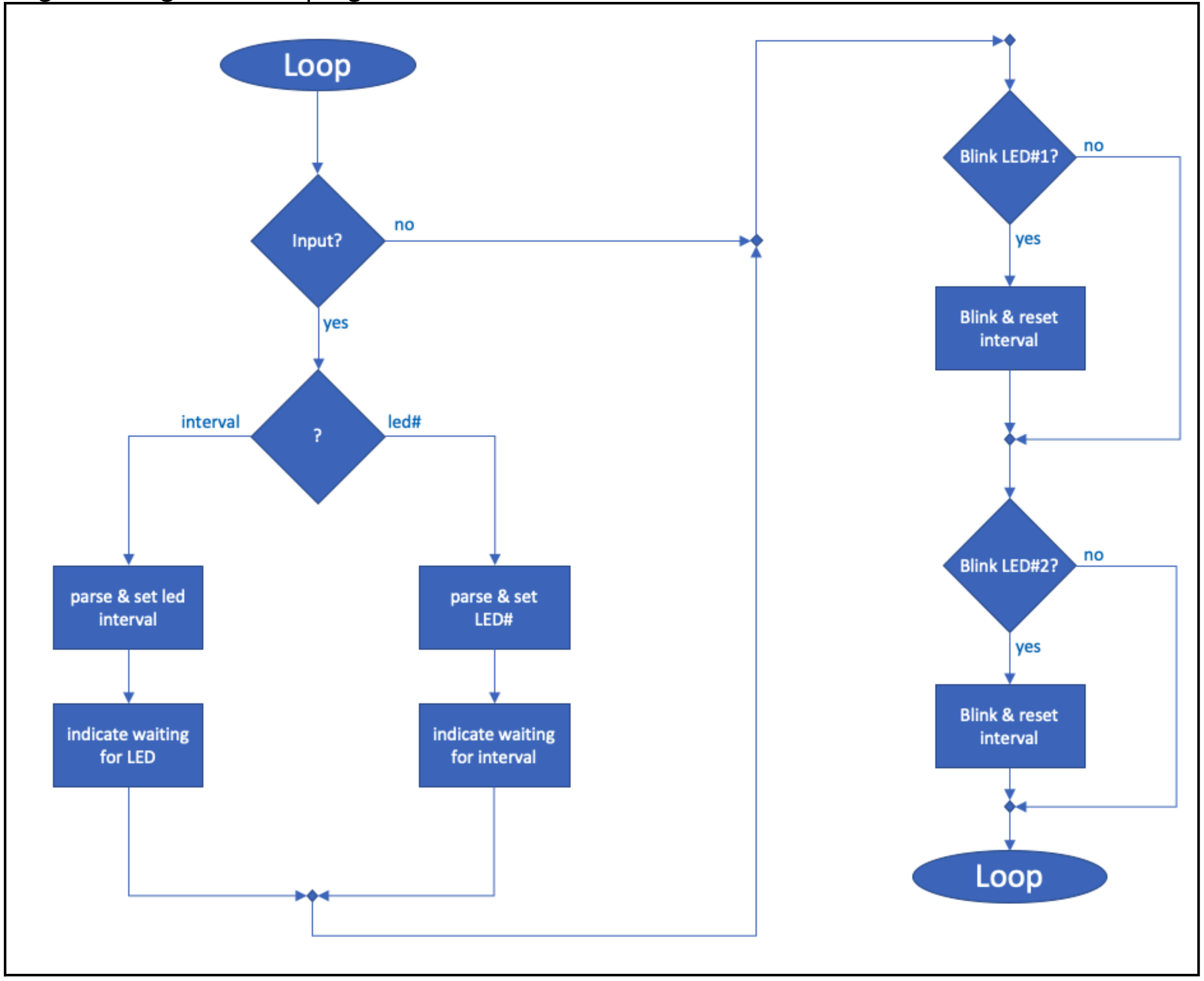
TESTING RESULTS:

Table 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.

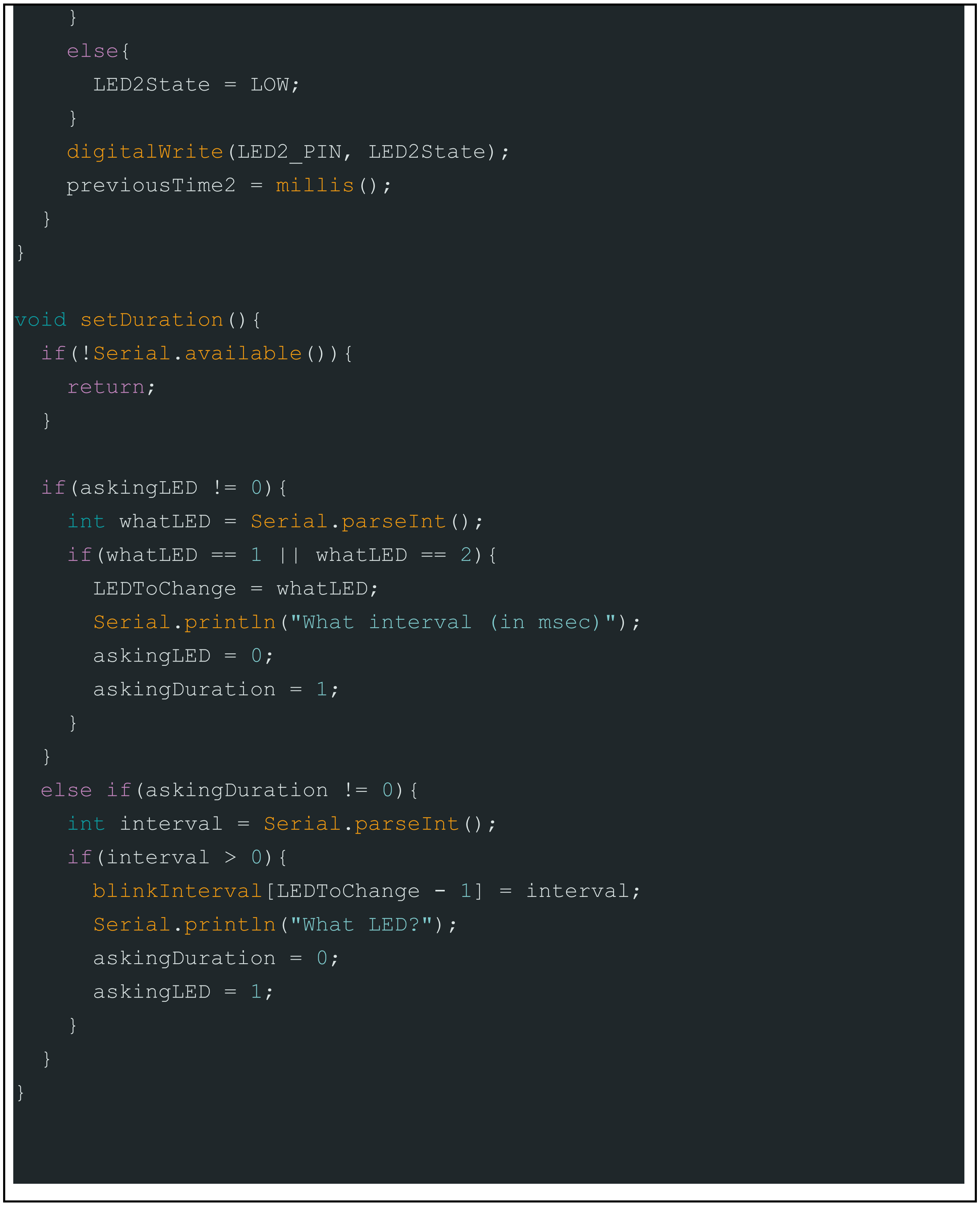
Appendix A. Logic Flow Chart

Figure 3. Logic flow for program.



APPENDIX B. DESIGN CODE

```
#define LED1_PIN 2
#define LED2_PIN 3
#define NUM_TASKS 3
#define EXECUTIVE_CYCLE_TIME 50

unsigned long blinkInterval[2] = {500, 1000};

int LED1State = LOW;
int LED2State = LOW;


unsigned long currentTime = millis();
unsigned long previousTime1 = millis();
unsigned long previousTime2 = millis();
unsigned long previousCycleTime = 0;

int askingLED = 1;
int askingDuration = 0;


int LEDToChange = 0;
unsigned long changeBlinkInterval = 0;

int LED2Time = millis();

typedef void(*CyclicTasks)();

void blinkLED1();
void blinkLED2();
void setDuration();

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(LED1_PIN, OUTPUT);
  pinMode(LED2_PIN, OUTPUT);
  Serial.println("What LED?");
  Serial.flush();
}

CyclicTasks tasks [NUM_TASKS] = {setDuration, blinkLED1, blinkLED2};


void loop() {
  // put your main code here, to run repeatedly:
  currentTime = millis();

  if(currentTime - previousCycleTime > EXECUTIVE_CYCLE_TIME){
    for(int i = 0; i < NUM_TASKS; i++){
      (*tasks[i])();
    }
    previousCycleTime = currentTime;
  }

}


void blinkLED1(){
  if(millis() - previousTime1 > blinkInterval[0]){
    if(LED1State == LOW){
      LED1State = HIGH;
    }
    else{
      LED1State = LOW;
    }
    digitalWrite(LED1_PIN, LED1State);
    previousTime1 = millis();
  }

}

void blinkLED2(){
  if(millis() - previousTime2 > blinkInterval[1]){

    if(LED2State == LOW){
      LED2State = HIGH;
```



```
    }
    else{
      LED2State = LOW;
    }
    digitalWrite(LED2_PIN, LED2State);
    previousTime2 = millis();
  }
}

void setDuration(){
  if(!Serial.available()){
    return;
  }

  if(askingLED != 0){
    int whatLED = Serial.parseInt();
    if(whatLED == 1 || whatLED == 2){
      LEDToChange = whatLED;
      Serial.println("What interval (in msec)");
      askingLED = 0;
      askingDuration = 1;
    }
  }
  else if(askingDuration != 0){
    int interval = Serial.parseInt();
    if(interval > 0){
      blinkInterval[LEDToChange - 1] = interval;
      Serial.println("What LED?");
      askingDuration = 0;
      askingLED = 1;
    }
  }
}
```

APPENDIX C. Video Link

https://drive.google.com/file/d/1Kqp1p9MTmt06FzGe1aaqvAk1tlHza3KM/view?usp=drive_link