Title and Date

Name: Project 2 Cyclic Executive Task Manager
Document Reference: kiddKid.cmpe311.fall25.project#2
Date of publication: November 11, 2025

Lead Engineer: Shawn Pourifarsi, 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 report for Project 2 which builds off of Project 1, manipulating the code to now be a cyclic executive task manager using a function pointer array to do the same identical tasks as Project 1.

Description: In this project, the goal is to create a circuit using an Arduino Uno R3 development platform that allows the user of the program to independently select 1 of 2 LEDs to control and its blink interval in milliseconds. The blinking of the LED continues with no regard to user input unless specifically changing that LEDs blink interval. Included in this document are the high level technical requirements derived from the customer, the design, testing scenarios and requirements. Videos and results from testing are attached to this document as code is executed within the video documentation. The code implementation requires the use of a round robin cyclic executive task manager based upon a function pointer array.

Result Summary: The project was a success with the embedded system design meeting all testing and high level requirements and was successfully built on a function pointer array

References and Glossary

REFERENCES:
• ProjectTaskMgmtCE– The definition of the project this document addresses
• CMPE310 Project #5 – A similar project assigned in CMPE310 in Spring 2025
• CMPE311 Project #1 – What this project was based off of and built on
• 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 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

LANGUAGE:
Must – this requirement must be satisfied
May – this requirement is optional
Inform/Informative – the following is description only and contains no requirements

Requirements

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 Requirements
HL.1 The System must use at least 2 LEDs
HL.2 The System must use a standard Arduino compatible development board (e.g. the

provided ELEGOO Uno R3)
HL.3 The System must communicate with The User only via the Arduino IDE serial-monitor port
HL.4 Any use of the serial-monitor in HL.3 must be asynchronous and not affect the blinking of the LEDs.
HL.5 The User must be able to set the blink interval of the LEDs in msec
HL.6 The blink rate of each LED must be constant unless changed by The User

Educational Constraints
E.1 Implement a cyclic executive task manager to dispatch the asynchronous tasks you created in the previous project, PROJECT-ASYNC.
E.2 The system testing and validation requirements must contain those defined in PROJECT-ASYNC.
E.2.1 The testing and validation requirements must contain any additional tests necessary to properly evaluate/demonstrate the function of the system.
E.3 The final report must be a formal design document as in PROJECT-ASYNC

Table 1. Example Dialogue

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, nothing happens or changes with LED1**
What LED? (1 or 2) ***1***	
What interval in msec? **1600**	**LED1 starts blinking at an interval of 800ms on and 800ms off. LED2 keeps blinking with an interval of 600ms**

Testing and Validation Requirements
T.1 The dialogue must similarly match Table 1 within the Arduino IDE Serial-Monitor
T.2 LED selection should correspond with the proper LED, when selecting LED1 within the IDE, the corresponding LED on the arduino should change everytime and not switch
T.3 The blink rate of one LED must be able to be set without affecting the blink rate of the second LED
T.4 The blink rate of the selected LED that is about to change should not change until the user input is complete
T.5 User input for blink rate must be in milliseconds and blink rate on the test bed should correspond with user input
T.6 The setting of an LED's blink rate must be repeatable at least 5 times

Testing and Validation

Design

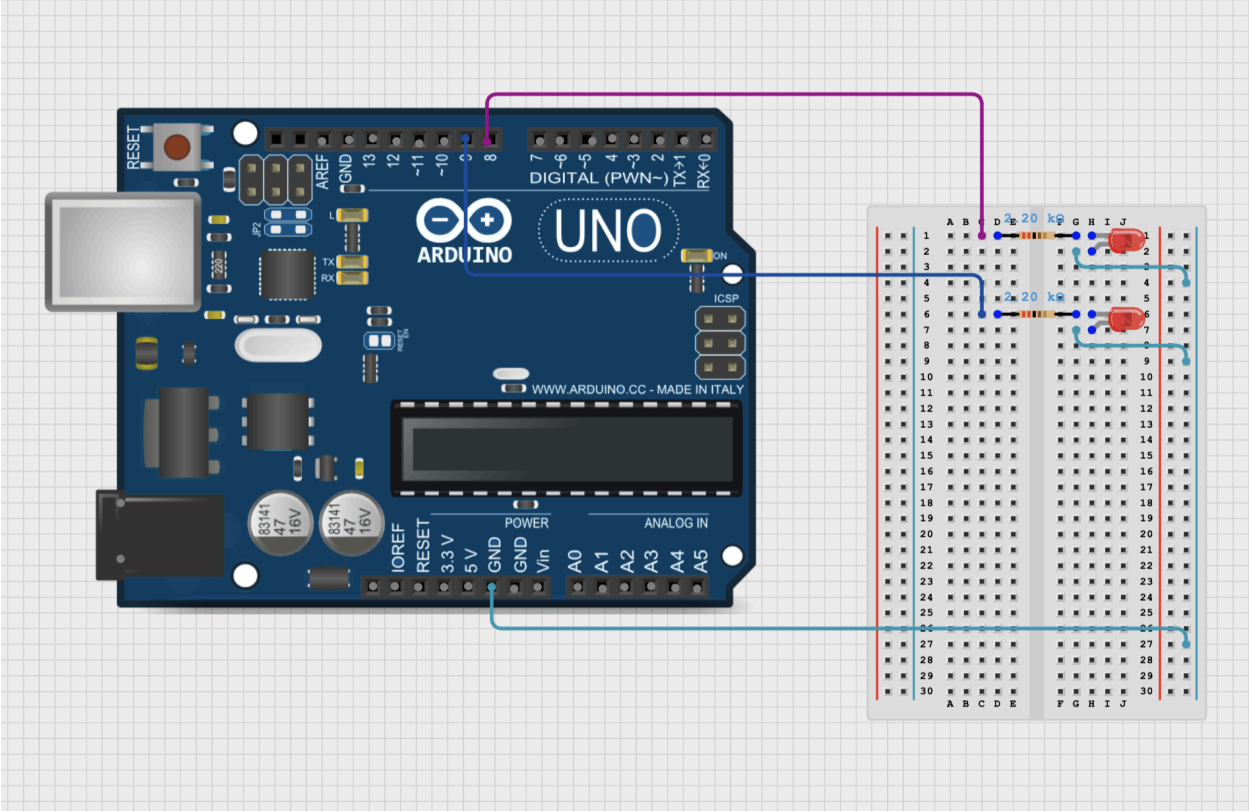

Figure 1: Testbed Setup, LEDs connected to digital pins 8 and 9

Testing Platform

1. **ELEGOO UNO R3**
2. **Jumper Cables**
3. **Arduino IDE 2.3.5 or higher**
4. **Arduino Power from USB cable, LEDs connected to arduino digital pins with 2.2kΩ resistor in series**

Testing Results

Table 2: Results of tests

Test Performed	Results

T.1	Satisfied | Check Video
T.2	Satisfied | Check Video
T.3	Satisfied | Check Video
T.4	Satisfied | Check Video
T.5	Satisfied
T.6	Satisfied

Appendix A. Design Code

```
const int led1 = 8;
const int led2 = 9;

unsigned long interval1 = 0; // interval for light1
unsigned long interval2 = 0; // interval for light 2
unsigned long prevMillis1 = 0; // prev interval light1
unsigned long prevMillis2 = 0; // prev interval light2
// light states
bool ledState1 = LOW;
bool ledState2 = LOW;

int step = 0; // 0 = ask for LED, 1 = ask for interval
int selectedLED = 0; // led input

void ledInterval(unsigned long*, unsigned long*);
void checkInterval(unsigned long*, unsigned long, bool*, int);

typedef void(*task)();

void setInterval() { ledInterval(&interval1, &interval2); }
void checkLED1() { checkInterval(&prevMillis1, interval1, &ledState1, led1); }
void checkLED2() { checkInterval(&prevMillis2, interval2, &ledState2, led2); }
task task_arr[] = {setInterval, checkLED1, checkLED2};


void setup() {
```

```
 // setup LEDs
 pinMode(led1, OUTPUT);
 pinMode(led2, OUTPUT);
 // initiate serial
 Serial.begin(9600);
 Serial.println("Arduino: What LED? (1 or 2)");



}

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



void ledInterval(unsigned long* interval1, unsigned long* interval2){
 if (Serial.available()) {
 String input = Serial.readStringUntil('\n'); // once user hits enter, thats the input

 if (step == 0){
   selectedLED = input.toInt();
   Serial.println("Arduino: What interval (in msec)?"); // print out to ask for
interval
   step = 1;
 }
 else if (step == 1){
   unsigned long newInterval = input.toInt();
   if (selectedLED == 1){
     *interval1 = newInterval;
   }
   else if (selectedLED == 2){
     *interval2 = newInterval;
   }
   step = 0;
   Serial.println("Arduino: What LED? (1 or 2)");
 }
```

```
 }
}



void checkInterval(unsigned long* prevMillis, unsigned long interval, bool* ledState,
int lednum){
 if(millis() - *prevMillis >= interval){
   *prevMillis = millis();
   *ledState = !(*ledState);
   digitalWrite(lednum, *ledState);
 }
}
```