TITLE AND DATE

Document name: ProjectReportTaskMgmt
Document reference: *Davenport, Robert*.cmpe311.fall25.*Lab-Project-TaskMgmt*
Date of publication: *11/11/2025*

LEAD ENGINEER: *Robert Davenport, UMBC CSEE, Catonsville, 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 outlines requirements for project #2. These include customer, technical, educational and testing requirements

DESCRIPTION: *Design and implement an embedded system using an Arduino UNO compatible development board. System must use a cyclic executive task manager to dispatch the following tasks. Task 1, System must prompt users over serial monitor for LED number and blinking interval in msec. Task 2, the system reads and stores user input. Task 3 takes the input and must blink the correct LED and the specified interval. Each LED will work independently of each other. The functionality performs identical to that of PROJECT-ASYNC.*

RESULT SUMMARY: The project was successful, and the design proposed meets all requirements outlined

REFERENCES AND GLOSSARY

REFERENCES:

- projectTaskMgmtCE.pdf – *Original project outline*
- Davenport, Robert.cmpe311.fall25.Lab-Project-Async – *PROJECT-ASYNC doc by lead engineer*
- Example_Project_Report.docx – *Further clarification on report outline*
- https://www.w3schools.com/c/c_structs.php - *How to use structs in C*
- https://docs.arduino.cc/language-reference/en/functions/time/millis - *How to use millis() built in function*
- Arduino UNO R3 Product Reference Manual SKU A000066 12/03/2024

DEFINITIONS AND ABBREVIATIONS:
"PROJECT-ASYNC" – The previous project upon which this project is based
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
Msec – Millisecond time interval used by IDE

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.
C3. The System must run upon an Arduino Uno R3 compatible development board

HIGH-LEVEL TECHNICAL 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 REQUIREMENTS:
ER.1.0 Implement a cyclic executive task manager to dispatch the asynchronous tasks you created
in the previous project, PROJECT-ASYNC.
ER.1.1 The system testing and validation requirements must contain those defined in PROJECT-ASYNC.
ER.1.1.1 The testing and validation requirements must contain any additional tests necessary to properly evaluate/demonstrate the function of the system.

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

TESTING AND VALIDATION REQUIREMENTS:

T.1 System dialogue prompts the user to choose LED# and Blink Rate

T.1.1 Dialogue prompts on 2 separate lines (i.e. 1 line to ask LED # and next to ask Blink Rate)

T.1.2 Setting LED# and Blink Rate must be through serial monitor

T.2 Each LED is independent and setting the blink rate of one LED should not affect the other

T.3 The user specifies the blink rate in msec per blink

T.4 Blink rate specified is correctly displayed on LED's

T.4.1 Blink rate is on correct LED specified by the user

T.5 The setting of an LED should have the LED blink at least 5 time

T.5.1 Updating a blinking LED should not affect the other LED

DESIGN AND TESTING

DESIGN

Materials:

- Arduino Uno R3 or similar development board
- Arduino IDE 2.3.3 or newer
- Cables & Breadboard
- 2 LED's
- 2 resistors

Schematic: Located under Appendix A.

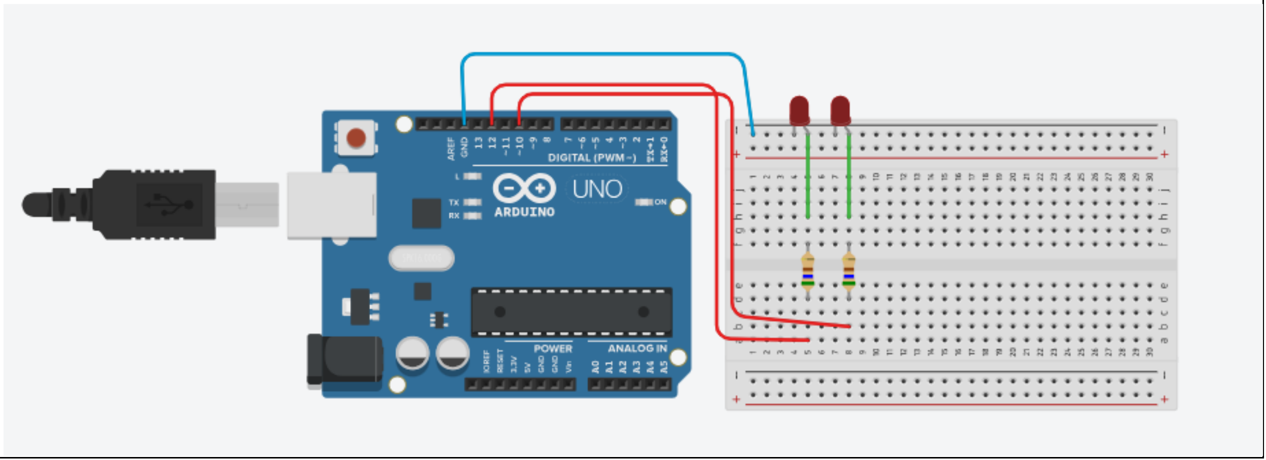
Design code: Located under Appendix B.

TESTING RESULTS:

Test # (See Testing and Validation requirements)	Result (See video of test)
T.1	Passed
T.1.1	Passed
T.1.2	Passed
T.2	Passed
T.3	Passed
T.4	Passed
T.4.1	Passed
T.5	Passed
T.5.1	Passed

Appendix A. Schematic

Testbed Schematic. LEDs connected to digital pins 10 & 12. -*Same as PROJECT-ASYNC*



Appendix B. Design Code

```
// Pin configuration
const int led1 = 12;
const int led2 = 10;

// LED structure
struct LED_STATE {
  int led;
  int interval;
  bool on;
  unsigned long prevMillis;
  int currState;
};

// Instances
LED_STATE LED1 = {led1, 0, false, 0, LOW};
LED_STATE LED2 = {led2, 0, false, 0, LOW};

//TASK DEFINITIONS
void task_blink_LED1() { blink(LED1); }
void task_blink_LED2() { blink(LED2); }
void task_serial_input();

// Task function pointer type
typedef void (*TaskFunction)();

// Function pointer array (Cyclic Executive Task List)
TaskFunction taskList[] = {
  task_blink_LED1,
  task_blink_LED2,
 task_serial_input
};

const int NUM_TASKS = sizeof(taskList) / sizeof(TaskFunction); //3
tasks

//LED blink function
void blink(LED_STATE &ledState) {
  if (ledState.on && ledState.interval > 0) {
    unsigned long currMillis = millis();
    if (currMillis - ledState.prevMillis >= (unsigned
long)ledState.interval) {
      ledState.prevMillis = currMillis;
      ledState.currState = !ledState.currState;
      digitalWrite(ledState.led, ledState.currState);
    }
  }
}

//Input States
enum InputState { READ_LED, READ_INTERVAL };
InputState inputState = READ_LED;
int led_choice = 0;
int interval_choice = 0;

void task_serial_input() {
  if (Serial.available() > 0) {
    String input = Serial.readStringUntil('\n');
    input.trim();

    if (input.length() > 0) {
      switch (inputState) {
        case READ_LED:
          led_choice = input.toInt();
          Serial.println("What is the blink rate (in msec)?");
          inputState = READ_INTERVAL;
          break;

        case READ_INTERVAL:
          interval_choice = input.toInt() / 2;

          if (led_choice == 1) {
            LED1.on = true;
            LED1.interval = interval_choice;
          } else if (led_choice == 2) {
            LED2.on = true;
            LED2.interval = interval_choice;
          }

          Serial.println("What LED? (1 or 2)");
          inputState = READ_LED;
          break;
      }
    }
 }
}
```

```
void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  Serial.begin(9600);
  Serial.println("What LED? (1 or 2)");
}

void loop() {
  for (int i = 0; i < NUM_TASKS; i++) {
    taskList[i](); // Call each task (task manager)
  }

  delay(1); // Minor cycle time
}
```

END OF DOCUMENT