

TITLE AND DATE

Document name: CMPE 311 Project 2 Report

Document reference: Das, Abhinna.cmpe311.fall25.Project 2

Date of publication: 11/11/2025

LEAD ENGINEER: Abhinna Das, CMPE student

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: Project 1 for CMPE 311 shall use Arduino Uno R3 to implement multitasking on the Arduino Uno R3. The task for which multitasking will be implemented is controlling the blink rate of at least 2 different LEDs. Each LED's blink rate will be independent of each other LED. These blink rates shall be determined by the user at any point of operation of the device.

DESCRIPTION: The problem that is to be solved is to design a circuit and use the Arduino R3 board to allow a user to set the blink intervals of two different functions. The LEDs will blink at the rate defined by the user until changed. These defined blink rate will blink asynchronously while waiting for User input.

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 are started with "T.#"

Part 1 GIVEN CUSTOMER REQUIREMENTS, DERIVE HIGH-LEVEL ENGINEERING 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.

DERIVED HIGH-LEVEL TECHNICAL REQUIREMENTS:

- C1. The person operating the blinking system shall be able to set the blink rate of each LED independently through some serial interface.
- C2. The person operating the blinking system shall be able to update the blink rate of each LED independently through some serial interface.
- C3. The LED shall blink at a constant rate until the person operating the blinking system changes the blink rate.
- C4. The blinking system must be able to run on an Arduino Uno R3 or equivalent 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

EDUCATION CONSTRAINTS:

- 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 PROJECTASYNC.
- 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.

TESTING AND VALIDATION REQUIREMENTS:

Test requirements list

T.1 Verify TECHNICAL REQUIREMENTS

- T.1.1 Verify that at least 2 LEDs are present in the submitted project design.
- T.1.2 Verify that the code submitted works on a different R3 Uno than the one used for development.
- T.1.3 Verify that the LEDs can be interacted with using the serial monitor port.
- T.1.4 Verify that use of the serial monitor does not affect the ongoing blink rate.
- T.1.5 Verify the blink rate is set in milliseconds
- T.1.6 Verify that the blink rate does not change unless changed by the User.

T.2 Verify EDUCATION CONSTRAINTS

- T.2.1 Verify that the program can be cross compiled on Arduino IDE 2.3.3 or later on an AVR ATmega328p.
- T.2.2 Verify that all programs developed by the developer and used in the project are available on GitHub to be publicly downloaded
- T.2.3 Verify that the program uses digital pins 2 and 3
- T.2.4 Verify that the program is able to complete the basic task given in the lab document.

Example Dialog

Serial Port	Expected output
None	To get started LED number(1,2) and blink interval(0-n)What LED? (1 or 2)
Pick invalid LED not (0,1)	LED Picked is not available
Pick valid LED 1 or 2	Picked LED: 2
What interval (in msec) for LED? 500	Blink Rate: 500
Pick valid LED 1 or 2	Picked LED: 1
What interval (in msec) for LED? 500	Blink Rate : 1000

The result of this test is we should see LED 2 blink 2 times before LED 1 blinks once.
As shown in this video.

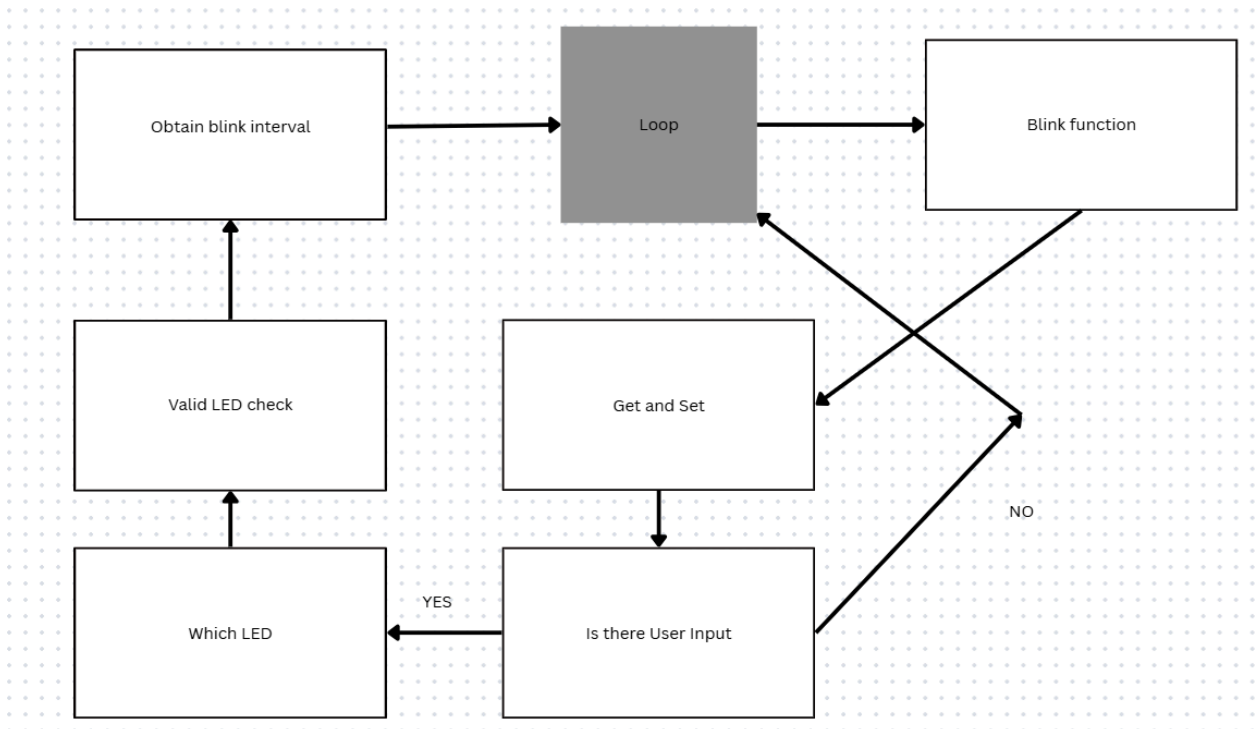
Testing results:

Test performed	Results
T.1.1	Satisfied. See video.
T.1.2	Satisfied
T.1.3	Satisfied. See video.
T.1.4	Satisfied. See video.
T.1.5	Satisfied. See video.
T.1.6	Satisfied. See video.

Video link:

<https://drive.google.com/file/d/12sMAJme1Kx2tdebAXBkl-5TXvQdbk02C/view?usp=sharing>

DESIGN:



Code:

```
// define statments should help a bit with speed
#define LED_1 2
#define LED_2 3

// Default LED states
byte LED1State = LOW;
byte LED2State = LOW;
// Default blink interval
unsigned long prevTimeLED1 = millis(), LED1BlinkDelay = 0;
unsigned long prevTimeLED2 = millis(), LED2BlinkDelay = 0;
// Variables imporant to getting data
int chosenLED = 0;
int chosenDelay = 0;

typedef void (*cycleCode)();
void GetAndSet();
void blink();

cycleCode tasks[] = {GetAndSet,blink};

int count = 0;
int task_count = sizeof(tasks)/sizeof(tasks[0]);

void setup(){
    pinMode(LED_1, OUTPUT);
    pinMode(LED_2, OUTPUT);
    Serial.begin(9600);
}
// This loop runs to activate and deactiave the LED at a set rate.
void loop(){
    tasks[count % task_count]();
    count++;
}

// The function that runs the loop to obtain the User Input to change the blink rate of LEDs
void GetAndSet(){
    if (chosenLED == 0 && chosenDelay == 0){
```

```

Serial.print('\n');
Serial.print("To get started LED number(1,2) and blink interval(0-n)");
chosenLED = 1;
}
if (Serial.available() == 0){
    chosenLED = pickvalue("What LED? (1 or 2)");
    Serial.print("Picked LED: ");
    Serial.print(chosenLED);
    Serial.print('\n');
    // Print newline to make output look nicer
    if (chosenLED == 1 || chosenLED == 2){
    } else {
        Serial.print("LED Picked is not available");
        Serial.print('\n');
        return;
    }
    Serial.print('\n');
    chosenDelay = pickvalue("What interval (in msec) for LED?");
    Serial.print("Blink Rate: ");
    Serial.print(chosenDelay);
    // Print newline to make output look nicer
    Serial.print('\n');
}
// Assign the correct LED only if the choice is 1 or 2
if (chosenLED == 1){
    LED1BlinkDelay = chosenDelay;
}
else if (chosenLED == 2){
    LED2BlinkDelay = chosenDelay;
}
}
// This the function that blinks the LED's at the specified declared rates.
void blink(){
    // gets the time from the internal clock this is imporant to allow for proper intervals
    unsigned long timeNow = millis();
    // check the delay of of each LED. As long as time now will be less then the Delay time then
    we should be fine.
    if (timeNow - prevTimeLED1 > LED1BlinkDelay){
        prevTimeLED1 += LED1BlinkDelay;
        // The LED switches states if the time is correct.
        if (LED1State == HIGH)
            LED1State = LOW;
        else

```

```

    LED1State = HIGH;
    // Write out the state (turn on or off the LED);
    digitalWrite(LED_1, LED1State);
}
// We do the same here as the code above but for LED 2
if (timeNow - prevTimeLED2 > LED2BlinkDelay){
    prevTimeLED2 += LED2BlinkDelay;
    if (LED2State == HIGH)
        LED2State = LOW;
    else
        LED2State = HIGH;
    digitalWrite(LED_2, LED2State);
}
}
// The function that gets the user input to change the blink intervals of the LEDs
int pickvalue(const char* printout) {
    Serial.println(printout);
    // Wait for user input but still run the blink cycle
    while (Serial.available() == 0) {
        blink();
    }
    // Read the integer input
    int number = Serial.parseInt();

    // Clear any remaining input for the next run
    while (Serial.available() != 0) {
        Serial.read();
    }
    return number;
}

```

END OF DOCUMENT