Project 1 - September 18, 2025
> Document Name: "CMPE 311 Project 1"
> Document Reference: Evers_Siena.cmpe311.fall25.proj1
> Date of Publication: 9/23/2025

Lead Engineer:
Siena Evers, Undergraduate Student, 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 Descriptor:
This product will allow The User to control the blinking of two LED lights independently from each other. They will select and control the speed of the LED blinking with the serial monitor of the Arduino Uno R3. The timing between HIGH and LOW states of the LEDs will be in terms of milliseconds. The LEDs must keep blinking at the timing they are set to until The User changes their timing.

Description:
The Customer requests an embedded system that can independently control the timing of two LEDs. The User will be able to interact with the LEDs through an Arduino Uno R3 serial monitor. The LEDs must not be interrupted in their timing unless The User changes the timing.

Definitions And Abbreviations:
- "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
- "serial-monitor" – The serial port used by the Arduino IDE to communicate with The User.
- Arduino – an Italian open-source hardware and software company; also refers to a development board created by the company
- LED – Light Emitting Diode

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.#"

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 (PART 1):
- HL.1 The System must have two operating LEDs
- HL.2 The LEDs must be wired independently of each other
- HL.3 The System must use an Arduino Uno R3 serial monitor to control the blinking of the LEDs
- HL.4 The LEDs must blink at the time interval inputted by The User in milliseconds
- HL.5 The blinking LEDs must not be interrupted unless changed by The User

(These High-Level Technical Requirements are my own, for Part 1 of this project)

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
- 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:
- E.1 The Developer must use the version of gcc used by Arduino IDE 2.3.3 or later for the AVR ATmega328p to cross compile any program used by The System
- E.2 The Developer must have any software used in The System hosted on a github repository. This software does not need to include libraries, macros or other code not developed by The Developer.
    - E.2.1 The Evaluator must be able to (a) download from the repository any software, documentation and other material needed for their evaluation, (b) cross compile any software, and (c) run that software on The Reference Platform.
- E.3 LEDs must be connected to Arduino digital Pins 2 & 3.
- E.4 The Evaluator must be able to successfully have The System perform the dialog in Table 5

Testing and Validation Requirements:
- T.1 The blinking of the LED matches the time in milliseconds imputed
- T.2 The User cannot interrupt or change the blinking of the LED through any means other than proper inputs through the serial-monitor
- T.3 The program outputs the proper information for The User to understand and interface

Design:

Equipment used:
- Arduino Uno R3
- 330 Ohm resistor (x2)
- Red LED (x2)
- Wires

Before starting the code, I first did the wiring on the breadboard. Requirement E.3 required the pins 2 and 3 to be used for the LEDs. Using those pins I connected them to the board with resistors connecting to the LEDs.
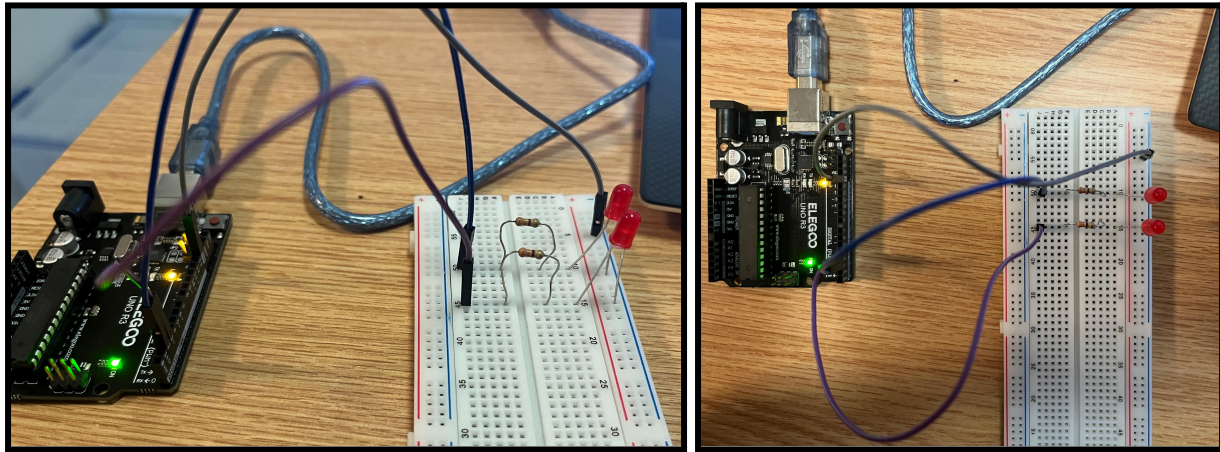
After the wiring, I started coding. I created the serial monitor and set up the variables for the timing. Because of the serial monitor you cannot use the delay function so all of the code must be able to run at the same time. The floor flag booleans are used for outputting text and getting the user input. The LEDs update based on real time at the bottom of the code. The code has been copied and pasted at the bottom of this report.

Testing:

For testing I record a video to show The System working as specified. While recording I was unable to show the stopwatch timing the blinks, but that was because I was recording with my phone so I could not use it as a stopwatch and one of my hands had to hold my phone to record. In the video I show the LEDs blinking at the correct time intervals and also correctly updating the time intervals while the program is running.

Video Link

Code:

```
// E.3 states that pins 2 and 3 are the outputs
#define LED1 2
```

```cpp
#define LED2 3


//varibles created for timing, HL.5 the time is in msec
int ledSelect = 0;
int timing = 0;
unsigned long prevTime1 = millis();
unsigned long prevTime2 = millis();
long timeInt1 = 0;
long timeInt2 = 0;

int ledState1 = LOW;
int ledState2 = LOW;

//these boolean exist to control the serial monitor
  //since the loop keeps going constantly for the blinking
  //the booleans must keep the print statements from
  //looping forever so the user can input their values
bool flagPrint1 = true;
bool flagPrint2 = false;
bool flag3Check = true;
bool flag4Check = false;


void setup() {
  //sets the pinMode to output
  pinMode(LED1,OUTPUT);
  pinMode(LED2,OUTPUT);

  //makes the serial monitor
  Serial.begin(9600);


}

void loop() {

  unsigned long currentTime = millis();
  timing = 0;

  //prints message to ask what LED
```

```cpp
if(flagPrint1){
  Serial.println("What LED? (1 or 2)?");
  flag3Check = true;
  flagPrint1 = false;
}

//takes user input (integer)
if (flag3Check && Serial.available()) {
  ledSelect = Serial.parseInt();
  if(ledSelect != 0) {
    flagPrint2 = true;
    flag3Check = false;
  }
}

//prints message to ask what time interval
if(flagPrint2){
  Serial.println("What interval(in msec)?");
  flagPrint2 = false;
}

//takes the user input (integer)
if (!flag3Check && Serial.available()) {
  timing = Serial.parseInt();
  if(timing != 0){
    flag4Check = true;
  }
}

//sets timeing Interval 1 to timing
if(flag4Check && ledSelect == 1){
  timeInt1 = timing;

  if(timeInt1 < 0){
    ledState1 = LOW;
  }

  flagPrint1 = true;
  flagPrint2 = false;
  flag3Check = true;
```

```arduino
    flag4Check = false;

    ledSelect = 0;
    timing = 0;

  }

  //sets timeing Interval 2 to timing
  else if(flag4Check && ledSelect == 2){
    timeInt2 = timing;

    if(timeInt2 < 0){
      ledState2 = LOW;
    }

    flagPrint1 = true;
    flagPrint2 = false;
    flag3Check = true;
    flag4Check = false;

    ledSelect = 0;
    timing = 0;
  }

  //if the user did not type the right number, the print statements reset
  and the LEDs are not impacted
  else if(flag4Check){
    //user did not type the right LED
    flagPrint1 = true;
    flagPrint2 = false;
    flag3Check = true;
    flag4Check = false;

    ledSelect = 0;
    timing = 0;
  }


  //this controls the blinking of the LED
    //if the time interval is zero there should be no blinking
```

```
    //if the time interval is not zero the LED will switch states when the
time interval passes

  if(timeInt1 != 0 && currentTime - prevTime1 > timeInt1){
    prevTime1 = currentTime;

    if(ledState1 == HIGH){
      ledState1 = LOW;
    }
    else {
      ledState1 = HIGH;
    }
    digitalWrite(LED1, ledState1); //updates the LED to on or off
  }

  //same logic as above but with the second LED
  if(timeInt2 != 0 && currentTime - prevTime2 > timeInt2){
    prevTime2 = currentTime;

    if(ledState2 == HIGH){
      ledState2 = LOW;
    }
    else {
      ledState2 = HIGH;
    }
    digitalWrite(LED2, ledState2);
  }


}
```

END OF DOCUMENT