Contents

1 Blinking Task **3**

2 Serial Communication **6**

List of Figures

1 Task 1 . . . 3
2 Task 2 . . . 4
3 Task 4 . . . 5
4 Task 4 Serial Output . . . 6
5 Task 5 . . . 8
6 Task 5 Output . . . 8
7 Task 6 Output . . . 9
8 Task 8 Output . . . 10
9 Task 9 Identical result . . . 10
10 Task 9 different Result . . . 10
11 Task 10 and 11 Output . . . 12
12 Task 12 Output . . . 14

List of Tables

1 LED blinking frequency and delay based on button state . . . 8

1 Blinking Task

Task 1 – Simple LED Blink

Objective: Blink an external LED using `digitalWrite()` and `delay()`.

```
#include <Arduino.h>

const uint8_t kLedPin = 13;

void setup() {
    pinMode(kLedPin, OUTPUT);
}

void loop() {
    digitalWrite(kLedPin, HIGH);
    delay(500);
    digitalWrite(kLedPin, LOW);
    delay(500);
}
```

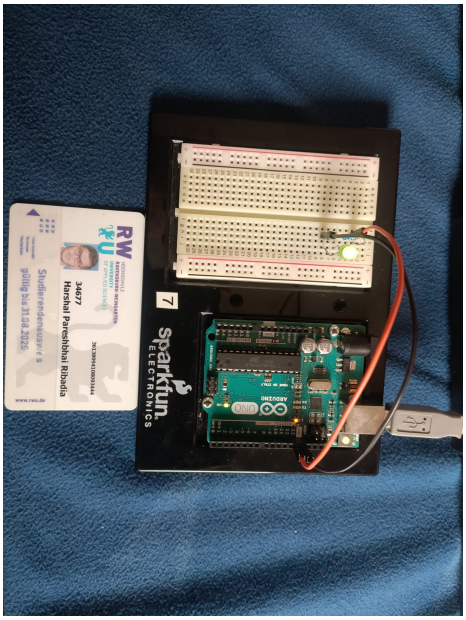

Figure 1: Task 1

Figure Description: An external LED is connected to digital pin 13 via a 220Ω resistor. The cathode of the LED is connected to GND. This setup ensures safe operation and prevents the LED from being damaged by excessive current.

Task 2 – Two LEDs Blinking with Boolean Variables

Objective: Blink two LEDs at 5 Hz using boolean state toggling.

```
#include <Arduino.h>

const uint8_t kLed1Pin = 12;
const uint8_t kLed2Pin = 13;

bool led1State = false;
bool led2State = false;

void setup() {
    pinMode(kLed1Pin, OUTPUT);
    pinMode(kLed2Pin, OUTPUT);
}

void loop() {
    led1State = !led1State;
    led2State = !led2State;
    digitalWrite(kLed1Pin, led1State);
    digitalWrite(kLed2Pin, led2State);
    delay(100); // 5 Hz
}
```

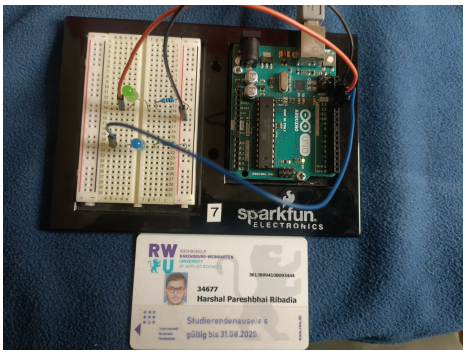

Figure 2: Task 2

Figure Description: Two LEDs are connected to pins 12 and 13 respectively. Each LED has a 220Ω resistor in series and is grounded.

Observation: LEDs blink at the same frequency as Task 1. Code readability improves with boolean state caching.

Task 3 – PWM LED Brightness Control

Objective: Control brightness of three LEDs using PWM.

```arduino
#include <Arduino.h>

const int ledPin1 = 10;
const int ledPin2 = 9;
const int ledPin3 = 6;

void setup() {
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);

  analogWrite(ledPin1, 64);   // ~1.25V
  analogWrite(ledPin2, 128);  // ~2.5V
  analogWrite(ledPin3, 192);  // ~3.75V
}

void loop() {
  // Nothing needed here
}
```

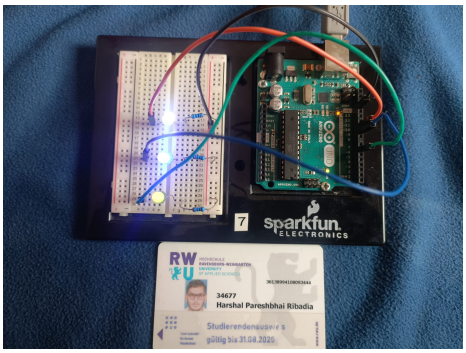

Figure 3: Task 4

Figure Description: Three LEDs are connected to PWM-capable pins 10, 9, and 6 with resistors. Brightness levels are set using PWM values.

2 Serial Communication

Task 4 – Serial Debug Output for LED States

Objective: Print LED states to the serial monitor.

```
#include <Arduino.h>

const uint8_t kLed1Pin = 12;
const uint8_t kLed2Pin = 13;

bool led1State = false;
bool led2State = false;

void setup() {
    pinMode(kLed1Pin, OUTPUT);
    pinMode(kLed2Pin, OUTPUT);
    Serial.begin(115200);
}

void loop() {
    led1State = !led1State;
    led2State = !led2State;

    digitalWrite(kLed1Pin, led1State);
    digitalWrite(kLed2Pin, led2State);

    Serial.print("LED1: ");
    Serial.print(led1State ? "HIGH" : "LOW");
    Serial.print(" | LED2: ");
    Serial.println(led2State ? "HIGH" : "LOW");

    delay(100);
}
```

Serial Output:

```
Connect to serial port COM3 at 115200
LED1: HIGH | LED2: HIGH
LED1: LOW | LED2: LOW
LED1: HIGH | LED2: HIGH
LED1: LOW | LED2: LOW
LED1: HIGH | LED2: HIGH
LED1: LOW | LED2: LOW
LED1: HIGH | LED2: HIGH
LED1: LOW | LED2: LOW
LED1: HIGH | LED2: HIGH
LED1: LOW | LED2: LOW
LED1: HIGH | LED2: HIGH
LED1: LOW | LED2: LOW
LED1: HIGH | LED2: HIGH
LED1: LOW | LED2: LOW
LED1: HIGH | LED2: HIGH
LED1: LOW | LED2: LOW
LED1: HIGH | LED2: HIGH
```

Figure 4: Task 4 Serial Output

Task 5 – Frequency Control with Pushbuttons

Objective: Use pushbuttons to adjust LED blinking frequency.

```
#include <Arduino.h>

const uint8_t kLedPin = 13;
const uint8_t kButton1Pin = 2;
const uint8_t kButton2Pin = 3;

bool ledState = false;
unsigned long lastToggleTime = 0;
unsigned long delayInterval = 1000;

void setup() {
    pinMode(kLedPin, OUTPUT);
    pinMode(kButton1Pin, INPUT);
    pinMode(kButton2Pin, INPUT);
    Serial.begin(115200);
}

void loop() {
    bool btn1 = digitalRead(kButton1Pin);
    bool btn2 = digitalRead(kButton2Pin);

    if (btn1 && btn2) delayInterval = 50;
    else if (btn1) delayInterval = 250;
    else if (btn2) delayInterval = 125;
    else delayInterval = 500;

    unsigned long currentTime = millis();
    if (currentTime - lastToggleTime >= delayInterval) {
        lastToggleTime = currentTime;
        ledState = !ledState;
        digitalWrite(kLedPin, ledState);
        Serial.print("LED State: ");
        Serial.print(ledState ? "HIGH" : "LOW");
        Serial.print(" | Delay Interval: ");
        Serial.println(delayInterval);
    }
}
```

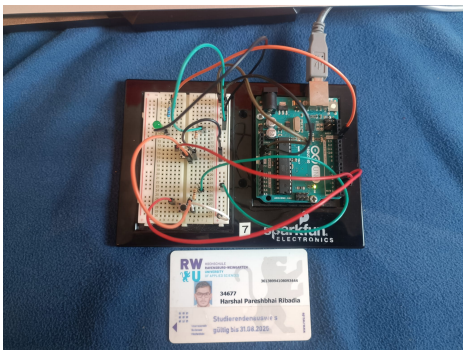

Figure 5: Task 5

Figure Description: Two pushbuttons with 10kΩ pull-down resistors adjust LED blinking frequency dynamically.

Serial output

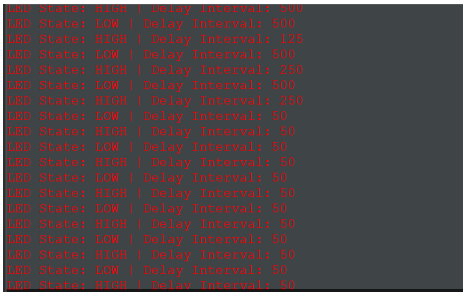

Figure 6: Task 5 Output

Condition	**Frequency (Hz)**	**Period (ms)**	**Delay (ms = Period / 2)**
No button pressed	1 Hz	1000 ms	500 ms
Button 1 (pin 2) pressed	2 Hz	500 ms	250 ms
Button 2 (pin 3) pressed	4 Hz	250 ms	125 ms
Both buttons pressed	10 Hz	100 ms	50 ms

Table 1: LED blinking frequency and delay based on button state

Casts and Operators

Task 7 – Matriculation Number Breakdown

Objective: Use modulo and integer division to extract values from the matriculation number.

```
Matriculation number: 34677
a (mat % 10) = 7
b (mat / 100) = 346
```

Figure 7: Task 6 Output

What is the purpose of the modulo operator ?
It returns the remainder of a division between two integers.

Task 8 – Float vs Integer Calculation

```
t_1 = 20 * a / b;
t_2 = 20.0 * a / b;
```

Explanation:
`t_1` performs integer division: $(20 \cdot 3)/4 = 60/4 = 15$, result is implicitly cast to float.
`t_2` performs floating-point division: $20.0 \cdot 3/4 = 60.0/4 = 15.0$
Console Output:

```
=== Type Casting and Division ===
a = 3
b = 4
t_1 = 15.00
t_2 = 15.00
```

Figure 8: Task 8 Output

Task 9 – Bitwise AND vs Modulo

Console Output: The above figure shows the calculations when the number

```
== Task 9 Output ==
Matriculation number: 34677
a (mat % 10) = 7
c (a & 15) = 7
d (a % 16) = 7
 The results are IDENTICAL.
```

Figure 9: Task 9 Identical result

is converted to the base 2 and then calculated. In this case the result are same.

```
Value of a: -30859
Using AND operator (a & 15): c = 5
Using modulo operator (a % 16): d = -11
The results are NOT equal.
```

Figure 10: Task 9 different Result

The above figure shows the calculations when the number is not converted to base 2 and used directly. In this case the result is not same.

Task 10 & 11 – Structs and Switch-Case

```
/*
 * Title       : BeveragesTodayStruct.cpp
 * Description : Defines beverages served today using a structure
    and prints the contents.
 * Author      : Harshal Ribadia
 */

#include <Arduino.h>

// Define a structure for beverage orders
struct BeverageOrder {
  int water;   // in units served
  int cola;    // in units served
  int juice;   // in units served
};

// Declare and initialize the structure
BeverageOrder todayOrders = {10, 5, 8};  // Example quantities

// Function to print beverages served today
void beverages_today() {
  Serial.println("=== Beverages Served Today ===");
  Serial.print("Water: "); Serial.print(todayOrders.water); Serial.
    println(" glasses");
  Serial.print("Cola: "); Serial.print(todayOrders.cola); Serial.
    println(" glasses");
  Serial.print("Juice: "); Serial.print(todayOrders.juice); Serial.
    println(" glasses");
}

// Function to analyze cola servings using switch-case
void check_cola() {
  int colaAmount = todayOrders.cola;

  switch (colaAmount) {
    case 1:
      Serial.println("Only one cola served.");
      break;
    case 5:
      Serial.println("Moderate cola demand today.");
      break;
    case 10:
      Serial.println("High cola consumption!");
      break;
    default:
      Serial.println("Cola quantity unusual. Check with staff.");
      break;
  }
}

void setup() {
  Serial.begin(115200);
  delay(1000); // Give time to open Serial Monitor
  beverages_today();  // Task 10: Struct and print
  check_cola();       // Task 11: Switch-case on cola
```

```
}

void loop() {
  // Nothing here
}
```

Serial Monitor Output:

```
=== Bread Recipe ===
Flour: 500.00 g
Water: 300.00 g
Salt: 10.00 g
Large batch. Ideal for multiple loaves.
```

Figure 11: Task 10 and 11 Output

Task 12 – Leap Year Function with Modules

LeapYear.h

```
/*
 * LeapYear.h
 *
 *  Created on: Apr 14, 2025
 *      Author: Harshal Ribadia
 */

#ifndef LEAPYEAR_H
#define LEAPYEAR_H

#include <Arduino.h>

// Declare shared variable from LeapYear.cpp
extern int sharedYear;

// Declare function to check leap year
bool leapYear(int year);

#endif
```

LeapYear.cpp

```
#include "LeapYear.h"

/*
 * LeapYear.cpp
 *
 *  Created on: Apr 14, 2025
 *      Author: Harshal Ribadia
 */

#include "LeapYear.h"

// Define the shared variable
int sharedYear = 2024;
```

```

// Define the function to check leap year
bool leapYear(int year) {
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
        Serial.print("Checked year ");
        Serial.print(year);
        Serial.println(": Leap year.");
        return true;
    } else {
        Serial.print("Checked year ");
        Serial.print(year);
        Serial.println(": Not a leap year.");
        return false;
    }
}
```

Base.h

```
/*
 * Base.h
 *
 *  Created on: Apr 14, 2025
 *      Author: Harshal Ribadia
 */

#ifndef BASE_H
#define BASE_H

#include <Arduino.h>

// Function to check and print leap year status
void checkLeapYearStatus();

#endif
```

Base.cpp

```
/*
 * Base.cpp
 *
 *  Created on: Apr 14, 2025
 *      Author: Harshal Ribadia
 */

#include <Arduino.h>
#include "LeapYear.h"
#include "Base.h"

// Use leapYear() and sharedYear from LeapYear module
void checkLeapYearStatus() {
    if (leapYear(sharedYear)) {
        Serial.println("It IS a leap year.");
    } else {
        Serial.println("It is NOT a leap year.");
    }
}

void setup() {
    Serial.begin(115200);
```

```
    delay(500); // Wait for serial monitor to open
    Serial.println("Leap Year Checker");
    checkLeapYearStatus();
}

void loop() {

}
```

```
Leap Year Checker
Checked year 2024: Leap year.
âœ… It IS a leap year.
```

Figure 12: Task 12 Output