# AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Engineering

## Lab Report

**Experiment # 04**

**Title:** Study of a Digital Timer using millis() function of Arduino to avoid problems associated with the delay() function.

| | | | |
|---|---|---|---|
| **Date of Perform** | 12 February 2023 | **Date of Submission** | 19 February 2023 |
| **Course Title** | MICROPROCESSOR AND EMBEDDED SYSTEMS | | |
| **Course Code** | COE3104 | **Section** | C |
| **Semester** | Spring 2022-23 | **Degree Program** | BSc in CSE |
| **Course Teacher** | **RICHARD VICTOR BISWAS** | | |

**Group # 01**

| Sl No | Name | ID | Contribution |
|---|---|---|---|
| 1 | Zaid Amin **Rawfin** | 20-42459-1 | Simulation & Code |
| 2 | Abzana Sultan **Ira** | 20-41973-1 | Abstract & Exp Procedure |
| 3 | MD. Sanjid Bin Karim **Sezan** | 19-41702-3 | Theory & Methodology |
| 4 | Abdur Rahman **Swapnil** | 18-38950-3 | Discussion & Conclusion |
| | | | |

***Faculty use only***

| | | |
|---|---|---|
| FACULTY COMMENTS | **Marks Obtained** | |
| | **Total Marks** | |

# Table of Contents

Experiment Title ........................................................................3
Abstract........................................................................................3
Introduction.................................................................................3
Theory and Methodology.........................................................3
Experimental Procedure...........................................................4
Circuit Diagram .........................................................................4
Apparatus ....................................................................................4
Hardware Output Results.........................................................5
Simulation Output Results .......................................................6
Code/Program ............................................................................7
Code Explanation.......................................................................8
Discussion & Conclusion..........................................................8
References....................................................................................8

## Experiment Title

Study of a Digital Timer using millis() function of Arduino to avoid problems associated with the delay() function.

## Abstract

This experiment aims to build a digital timer using an Arduino board and its built-in Timer library. The timer will be set to turn on an LED every minute, with the additional function of keeping track of the time spent on a particular task. The experiment involves connecting the LED to the Arduino board, coding the timer functionality using the millis() function, and incorporating a tilt switch for user input. The experiment demonstrates the usefulness and practical application of Arduino technology for time management and task tracking, while also providing hands-on experience in electronics and coding. By the end of the experiment, the digital timer will enable better time management and planning by measuring the duration of work on different projects. This experiment offers an engaging way to learn about electronics and coding, while also providing a practical tool for daily use.

## Introduction

In this experiment, a digital timer has been built that turns on an LED every minute. In addition, Arduino's built-in Timer has been used to keep track of the time spent on a particular task.

## Theory and Methodology

Up to now, when you've wanted something to happen at a specific time interval with the Arduino, you've used delay(). This is handy, but a little confining. When the Arduino calls the delay() function, it freezes its current state for the entire duration of the delay. That means, there can be no other input or output while it is waiting. Delays are also not very helpful for keeping track of time. If you wanted to do something every 10 seconds, having a 10-second delay would be cumbersome.

The millis() function helps to solve these problems. It keeps track of the time your Arduino has been running in milliseconds.

So far, you've been declaring variables as int. An int (integer) is a 16-bit number, it holds values between -32,768 and 32,767 (since the MSB is used for a signed bit, so number ranges are between – 215 and 215 – 1). Those may be some large numbers, but if the Arduino is counting 1000 times a second with millis(), you would run out of space in less than a minute. The long data type holds a 32-bit number (between -2,147,483,648 and 2,147,483,647). Since you can't run time backward, to get negative numbers, the variable to store millis() time is called an unsigned long. When a datatype is called unsigned, it is only positive. This allows you to count even higher numbers. An unsigned long number can count up to 4,294,967,295. That is enough space for milis() function to store time for almost 50 days. By comparing the current millis() function to a specific value, you can see if a certain amount of time has passed.

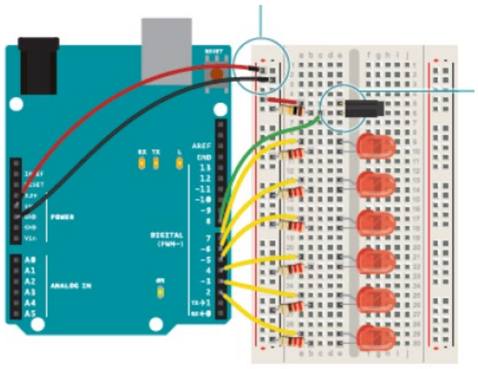
When you turn your Digital Timer over, a tilt switch will change its state, and that will set off another cycle of LEDs turning on.

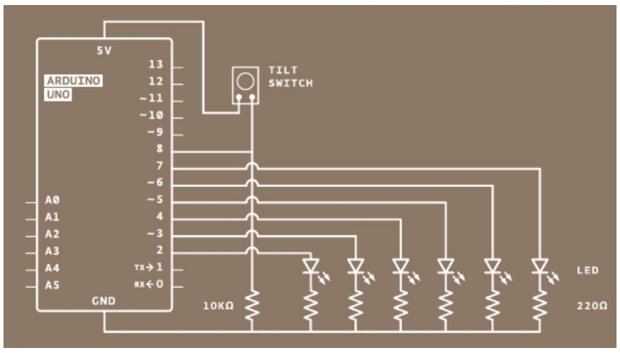
The tilt switch works just like a regular switch in that it is an on/off sensor. You'll use it here as a digital input. What makes tilt switches unique is that they detect orientation. Typically, they have a small cavity inside the housing that has a metal ball. When tilted in the proper way, the ball rolls to one side of the cavity and connects the two leads that are in your breadboard, closing the switch. With six LEDs, your timer will run for six minutes..

## Experimental Procedure

1. Power and ground were connected to the breadboard.
2. Necessary materials, including an Arduino board, an LED, a tilt switch, a breadboard, and jumper wires, were gathered.
3. The anode (longer leg) of six LEDs were connected to digital pins 2-7 and the LEDs were connected to the ground through 100 Ω resistors.
4. The tilt switch was connected to the breadboard and one lead was connected to +5V, the other lead was connected to a 10kΩ resistor to the ground, and the junction where they met was connected to digital pin 8.
5. The code for the digital timer was written using the millis() function, setting the LED to turn on every minute and incorporating the tilt switch input.
6. The code was uploaded to the Arduino board using the Arduino IDE.
7. The Arduino board was powered on and the digital timer's functionality was tested, ensuring that the LED turned on every minute and that the tilt switch input reset the timer.

## Circuit Diagram

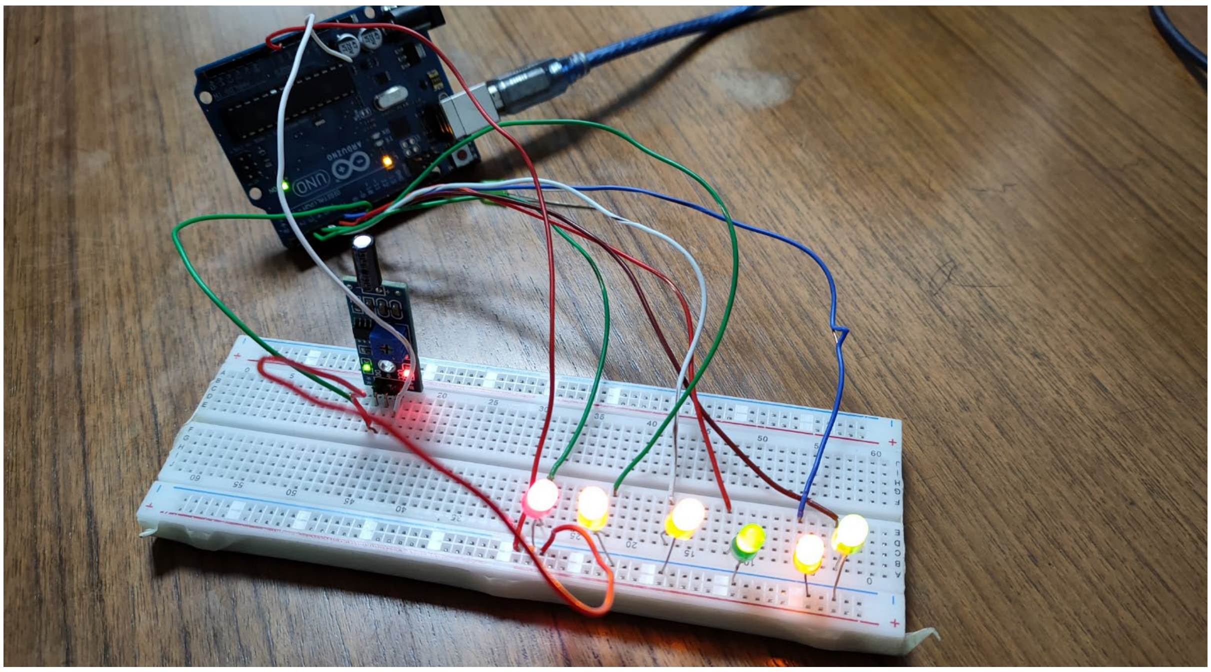




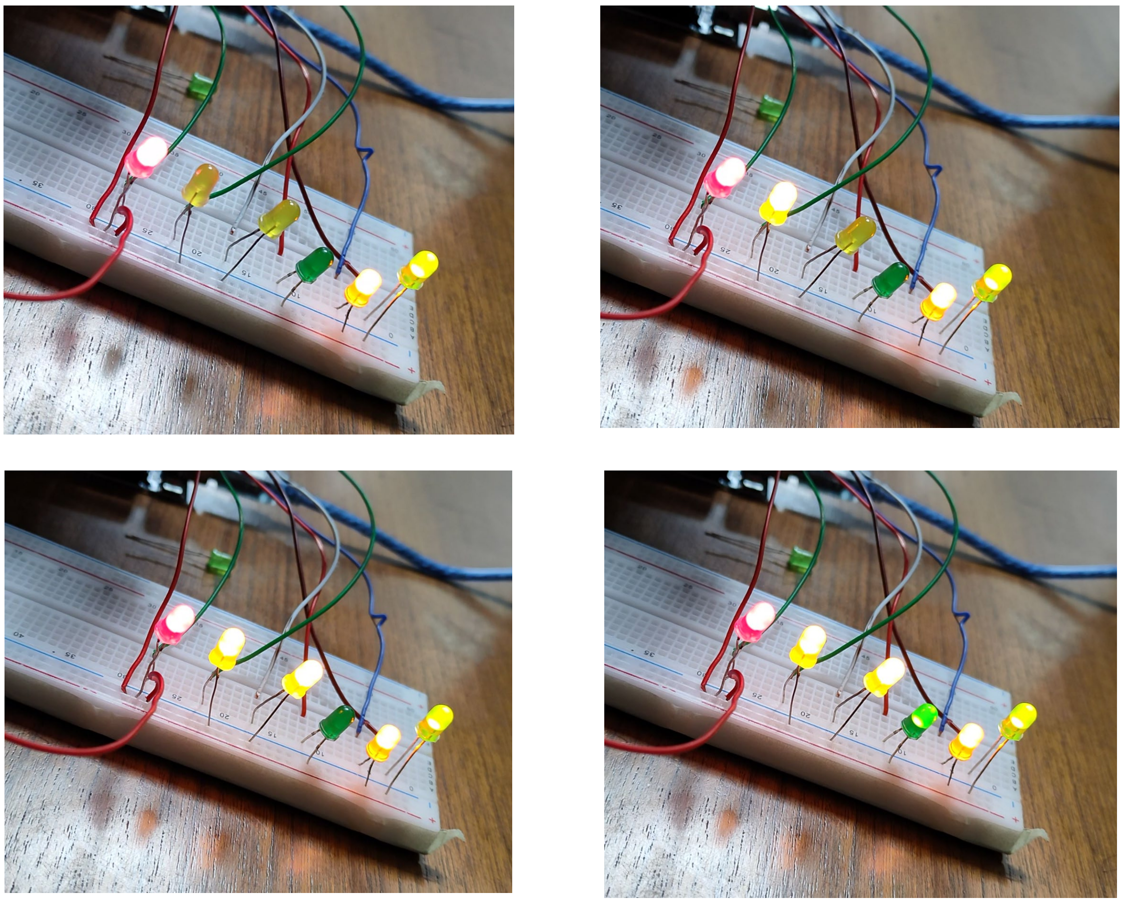
## Apparatus

- Arduino Uno/Arduino Mega Microcontroller Board
- Tilt sensor (One)
- LEDs (Six)
- Resistors (One 10 kΩ and Six 100Ω)

## Hardware Output Results

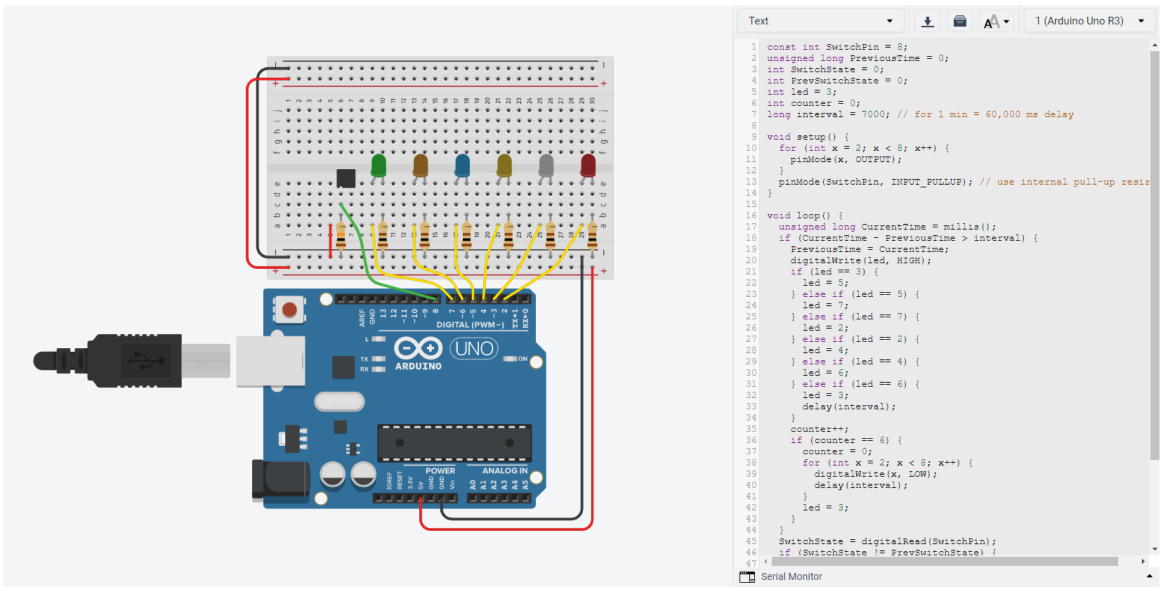


Hardware Setup



Lights turning on one by one

## Simulation Output Results

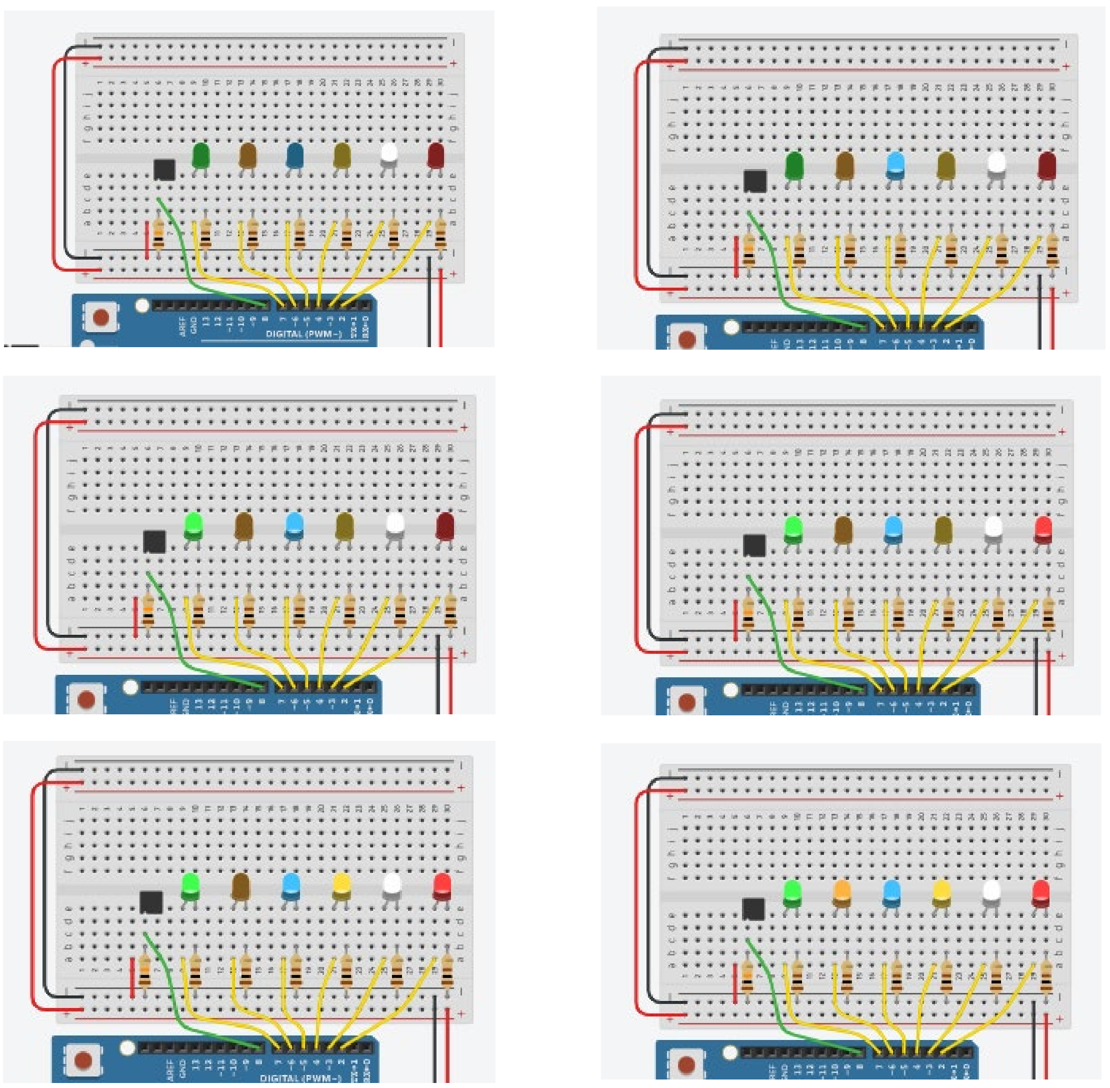


```
const int SwitchPin = 8;
unsigned long PreviousTime = 0;
int SwitchState = 0;
int PrevSwitchState = 0;
int led = 3;
int counter = 0;
long interval = 7000; // for 1 min = 60,000 ms delay

void setup() {
  for (int x = 2; x < 8; x++) {
    pinMode(x, OUTPUT);
  }
  pinMode(SwitchPin, INPUT_PULLUP); // use internal pull-up resis
}

void loop() {
  unsigned long CurrentTime = millis();
  if (CurrentTime - PreviousTime > interval) {
    PreviousTime = CurrentTime;
    digitalWrite(led, HIGH);
    if (led == 3) {
      led = 5;
    } else if (led == 5) {
      led = 7;
    } else if (led == 7) {
      led = 2;
    } else if (led == 2) {
      led = 4;
    } else if (led == 4) {
      led = 6;
    } else if (led == 6) {
      led = 3;
      delay(interval);
    }
    counter++;
    if (counter == 6) {
      counter = 0;
      for (int x = 2; x < 8; x++) {
        digitalWrite(x, LOW);
        delay(interval);
      }
      led = 3;
    }
  }
  SwitchState = digitalRead(SwitchPin);
  if (SwitchState != PrevSwitchState) {
```

Simulation Setup



Lights turning on in the sequence: 2, 4, 6, 1, 3, 5

## Code/Program

```
const int SwitchPin = 8;
unsigned long PreviousTime = 0;
int SwitchState = 0;
int PrevSwitchState = 0;
int led = 3;
int counter = 0;
long interval = 1000; // 1,000 ms = 1 second

void setup() {
  for (int x = 2; x < 8; x++) {
    pinMode(x, OUTPUT);
  }
  pinMode(SwitchPin, INPUT_PULLUP); // use internal pull-up resistor
}

void loop() {
  unsigned long CurrentTime = millis();
  if (CurrentTime - PreviousTime > interval) {
    PreviousTime = CurrentTime;
    digitalWrite(led, HIGH);
    if (led == 3) {
      led = 5;
    } else if (led == 5) {
      led = 7;
    } else if (led == 7) {
      led = 2;
    } else if (led == 2) {
      led = 4;
    } else if (led == 4) {
      led = 6;
    } else if (led == 6) {
      led = 3;
      delay(interval);
    }
    counter++;
    if (counter == 6) {
      counter = 0;
      for (int x = 2; x < 8; x++) {
        digitalWrite(x, LOW);
        delay(interval);
      }
      led = 3;
    }
  }
  SwitchState = digitalRead(SwitchPin);
  if (SwitchState != PrevSwitchState) {
    counter = 0;
    for (int x = 2; x < 8; x++) {
      digitalWrite(x, LOW);
    }
    led = 3;
    PreviousTime = CurrentTime;
  }
  PrevSwitchState = SwitchState;
}
```

## Code Explanation

This is a program for an Arduino board that controls a sequence of six LEDs. The LEDs are connected to pins 2 through 7, and a button is connected to pin 8. When the button is pressed, the sequence starts over. The program turns on the LEDs in the sequence **2, 4, 6, 1, 3, 5,** and then turns them off and starts over again.

The program uses the **millis()** function to keep track of time, so it can control the timing of the LED sequence. The variable **PreviousTime** stores the last time the LEDs were changed, and the **interval** variable specifies how long to wait between changes.

The **led** variable keeps track of which LED should be turned on next in the sequence. The program uses a series of **if** and **else if** statements to update the value of **led** in the correct order. Once all six LEDs have been turned on, the program resets **led** to 3 (the first LED in the sequence), turns off all the LEDs, and starts over.

The **counter** variable keeps track of how many times the LED sequence has been run. Once the sequence has been run six times (once for each LED), the program resets the counter, turns off all the LEDs, and starts over.

The program also checks the state of the button connected to pin 8. If the button is pressed, the program resets the **counter** and **led** variables, turns off all the LEDs, and starts over. The **PrevSwitchState** variable is used to keep track of whether the button was pressed on the previous loop iteration.

## Discussion & Conclusion

The digital timer was built using an Arduino board and its built-in Timer library. The timer turns on an LED every minute and also keeps track of the time spent on a particular task. The millis() function was used instead of the delay() function to avoid freezing the Arduino's state during the delay and to allow for more precise time tracking. The tilt switch was incorporated for user input, and the experiment offers a practical tool for time management and task tracking. The digital timer offers an engaging way to learn about electronics and coding while also providing hands-on experience in both areas. The functionality of the digital timer was successfully tested, and the experiment demonstrated the practical application of Arduino technology for time management and planning.

## References

- https://www.arduino.cc/.
- ATMega328 manual
- https://www.avrfreaks.net/forum/tut-c-newbies-guide-avr-timers
- http://maxembedded.com/2011/06/avr-timers-timer0/