https://github.com/kbala42/Tinkercad

# 2.Dijital Inputs (part 3)

## 2.10.1. Timer with LED on-off

https://www.tinkercad.com/things/jSoSg7l2Osi-2101-timer-with-led-on-off

```
/*********************************************************************************
**
Program Name: Timer with LED on-off

Program Description: This code is designed to flash an LED at a specific interval
using a timer (TIMER1) on Arduino.

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
*********************************************************************************
***/


#include <avr/io.h>
#include <avr/interrupt.h>

const int LED_pin = 13;
unsigned int reload = 0xF424;
volatile int count;

ISR(TIMER1_COMPA_vect)
{
  count++;
  flash();
}
void setup()
{
  Serial.begin(115200);
  pinMode(LED_pin, OUTPUT);
  digitalWrite(LED_pin, LOW);
  cli();
  TCCR1A = 0;
  TCCR1B = 0;
  OCR1A = reload;
  TCCR1B = (1<<WGM12) | (1<<CS12);
  TIMSK1 = (1<<OCIE1A);
  sei();
  Serial.println("TIMER1 Setup Finished.");
}
void loop()
{
  Serial.println(count); // do anything
  delay(200);
}
```

```
void flash()
{
  static boolean output = HIGH;
  digitalWrite(LED_pin, output);
  output = !output;
}
```

## 2.10.2.Timer ile 2 led on-off

https://www.tinkercad.com/things/1v9V23YE2G7-2102timer-ile-2-led-on-off/

```
/*********************************************************************************
**
Program Name: Timer with 2xLED on-off

Program Description: In the design, we will perform a flip-flop
function by sequentially activating and deactivating between
two timers. When the first timer ends, the second timer starts,
and this cycle continues.

Written by: Kamil Bala

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
*********************************************************************************
***/

#include <avr/io.h>
#include <avr/interrupt.h>

const int LED1_pin = 12; // Pin for the first LED
const int LED2_pin = 13; // Pin for the second LED

volatile bool toggle = false;

ISR(TIMER1_COMPA_vect)
{
 if (toggle)
 {
  digitalWrite(LED1_pin, HIGH);
  digitalWrite(LED2_pin, LOW);
  OCR1A = 0xF424; // Reload value for the first timer
 }
 else
 {
  digitalWrite(LED1_pin, LOW);
  digitalWrite(LED2_pin, HIGH);
  OCR1A = 0xB71B; // Reload value for the second timer
 }
 toggle = !toggle;
}

void setup()
{
 pinMode(LED1_pin, OUTPUT);
```

```
  pinMode(LED2_pin, OUTPUT);
  digitalWrite(LED1_pin, LOW);
  digitalWrite(LED2_pin, LOW);

  cli(); // Disable all interrupts
  TCCR1A = 0;
  TCCR1B = 0;
  OCR1A = 0xF424; // Reload value for the first timer
  TCCR1B |= (1 << WGM12); // CTC mode
  TCCR1B |= (1 << CS12); // 256 prescaler
  TIMSK1 |= (1 << OCIE1A); // Enable compare match interrupt
  sei(); // Enable all interrupts
}

void loop()
{
  // Other codes can be executed here
}
```

## 2.10.3.Two LEDs controlled by two buttons

https://www.tinkercad.com/things/2sl8YQKXHGj-2103two-leds-controlled-by-two-buttons

```
/*********************************************************************************
**
Program Name: Two LEDs controlled by two buttons

Program Description: In this code, two buttons (Button1_pin and
Button2_pin) control two different LEDs (LED1_pin and LED2_pin).
Each time a button is pressed, the corresponding LED lights up
and a timer starts. When the timer duration is over, the ISR is
called and the corresponding LED is turned off. Also, the status
of the LEDs is displayed on the serial monitor.

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
*********************************************************************************
***/

#include <avr/io.h>
#include <avr/interrupt.h>

const int LED1_pin = 12; // Pin for the first LED
const int LED2_pin = 13; // Pin for the second LED
const int Button1_pin = 2; // Pin for the first button
const int Button2_pin = 3; // Pin for the second button

volatile bool LED1_status = false;
volatile bool LED2_status = false;

ISR(TIMER1_COMPA_vect)
{
 if (LED1_status)
 {
   digitalWrite(LED1_pin, LOW);
   LED1_status = false;
   Serial.println("LED1 turned off");
 }
}

ISR(TIMER2_COMPA_vect)
{
 if (LED2_status)
 {
   digitalWrite(LED2_pin, LOW);
   LED2_status = false;
   Serial.println("LED2 turned off");
```

```
  }
}

void setup()
{
  pinMode(LED1_pin, OUTPUT);
  pinMode(LED2_pin, OUTPUT);
  pinMode(Button1_pin, INPUT_PULLUP);
  pinMode(Button2_pin, INPUT_PULLUP);
  Serial.begin(9600);

  // TIMER1 Settings
  cli(); // Disable all interrupts
  TCCR1A = 0;
  TCCR1B = 0;
  OCR1A = 0xF424; // Reload value for TIMER1
  TCCR1B |= (1 << WGM12); // CTC mode
  TCCR1B |= (1 << CS12); // 256 prescaler
  TIMSK1 |= (1 << OCIE1A); // Enable compare match interrupt

  // TIMER2 Settings
  TCCR2A = 0;
  TCCR2B = 0;
  OCR2A = 0xF4; // Reload value for TIMER2
  TCCR2B |= (1 << WGM21); // CTC mode
  TCCR2B |= (1 << CS22); // 64 prescaler
  TIMSK2 |= (1 << OCIE2A); // Enable compare match interrupt

  sei(); // Enable all interrupts
}

void loop()
{
  if (digitalRead(Button1_pin) == LOW)
  {
    digitalWrite(LED1_pin, HIGH);
    LED1_status = true;
    Serial.println("LED1 is on");
    delay(200); // Simple debouncing
  }

  if (digitalRead(Button2_pin) == LOW)
  {
    digitalWrite(LED2_pin, HIGH);
    LED2_status = true;
    Serial.println("LED2 is on");
    delay(200); // Simple debouncing
  }
}
```

## 2.10.4.Two timers and temperature control

https://www.tinkercad.com/things/8R3A6DQeVEk-2104two-timers-and-temperature-control

```
/**************************************************************************
**
Program Name: Two timers and temperature control

Program Description: This code uses two different timers (TIMER1
and TIMER2) to perform two different tasks. TIMER1 is used to
automatically turn off an LED after a certain period of time.
TIMER2 takes readings from a temperature sensor at regular
intervals and prints these values to the serial monitor. Such a
structure can be used in many different applications, such as
greenhouse automation.


Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
**************************************************************************
***/

#include <avr/io.h>
#include <avr/interrupt.h>

// Define LED and sensor pins
const int LED_pin = 12;
const int tempSensor_pin = A0;

// Define reload values for timers
unsigned int lightTimerReload = 0xF424; // For example, 1 second
unsigned int tempTimerReload = 0xF424; // For example, 5 seconds

volatile bool lightStatus = false;

ISR(TIMER1_COMPA_vect)
{
  // Light timer ISR
  digitalWrite(LED_pin, LOW);
  lightStatus = false;
}

ISR(TIMER2_COMPA_vect)
{
  // Temperature timer ISR
  int tempReading = analogRead(tempSensor_pin);
  float voltage = tempReading * 5.0;
  voltage /= 1024.0;
```

```
  float temperatureC = (voltage - 0.5) * 100; // For LM35 temperature sensor
  Serial.print("Temperature: ");
  Serial.print(temperatureC);
  Serial.println(" C");
}

void setup()
{
  pinMode(LED_pin, OUTPUT);
  pinMode(tempSensor_pin, INPUT);
  Serial.begin(9600);

  // Light timer settings (TIMER1)
  cli();
  TCCR1A = 0;
  TCCR1B = 0;
  OCR1A = lightTimerReload;
  TCCR1B |= (1 << WGM12);
  TCCR1B |= (1 << CS12);
  TIMSK1 |= (1 << OCIE1A);

  // Temperature timer settings (TIMER2)
  TCCR2A = 0;
  TCCR2B = 0;
  OCR2A = tempTimerReload;
  TCCR2B |= (1 << WGM21);
  TCCR2B |= (1 << CS22);
  TIMSK2 |= (1 << OCIE2A);

  sei();
}

void loop()
{
  if (!lightStatus)
  {
    digitalWrite(LED_pin, HIGH);
    lightStatus = true;
    // Restart the light timer
    TCNT1 = 0;
  }

  // Other functionalities can be placed here
}
```

## 2.10.5.Temperature Automation with Timer

https://www.tinkercad.com/things/6M8dEEPZL7O-2105temperature-automation-with-timer

```
/**************************************************************************
**
Program Name: Temperature Automation with Timer

Program Description: This code controls the green and red LEDs according to a
predetermined set point. If the temperature rises above 25 degrees, the green LED turns off
after 5 seconds. If the temperature falls below 25 degrees, the red LED lights up after 5
seconds. These durations are controlled by the greenLEDOffTime and redLEDOnTime
variables and are tracked using the millis() function.

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
**************************************************************************
***/

#include <avr/io.h>
#include <avr/interrupt.h>

// Pin definitions
const int tempSensor_pin = A0; // Analog pin for the temperature sensor
const int ledGreen = 3; // Digital pin for the green LED
const int ledRed = 4; // Digital pin for the red LED

// Timer and status variables
unsigned long greenLEDOffTime = 0;
unsigned long redLEDOnTime = 0;
bool greenLEDStatus = false;
bool redLEDStatus = false;

// Temperature set point
const float tempSetPoint = 25.0;

void setup() {
pinMode(ledGreen, OUTPUT);
pinMode(ledRed, OUTPUT);
pinMode(tempSensor_pin, INPUT);
Serial.begin(9600);
}

void loop() {
  // Read the temperature
  int tempReading = analogRead(tempSensor_pin);
  float voltage = tempReading * 5.0;
```

```
  voltage /= 1024.0;
  float temperatureC = (voltage - 0.5) * 100;

  // Write the temperature value to the serial port
  Serial.print("Temperature: ");
  Serial.print(temperatureC);
  Serial.println(" C");

  // Temperature control
  if (temperatureC > tempSetPoint) {
    if (!greenLEDStatus) {
      greenLEDStatus = true;
      redLEDStatus = false; // Turn off the red LED
      greenLEDOffTime = millis() + 5000; // Turn off the green LED after 5 seconds
      digitalWrite(ledGreen, HIGH);
      digitalWrite(ledRed, LOW);
    }
    } else if (temperatureC < tempSetPoint) {
    if (!redLEDStatus) {
      redLEDStatus = true;
      greenLEDStatus = false; // Turn off the green LED
      redLEDOnTime = millis() + 5000; // Light up the red LED after 5 seconds
      digitalWrite(ledRed, HIGH);
      digitalWrite(ledGreen, LOW);
    }
  }

  // LED timer control
  if (greenLEDStatus && millis() > greenLEDOffTime) {
    greenLEDStatus = false;
    digitalWrite(ledGreen, LOW);
  }

  if (redLEDStatus && millis() > redLEDOnTime) {
    redLEDStatus = false;
    digitalWrite(ledRed, LOW);
  }

  delay(1000); // Wait for 1 second
}
```

## 2.10.6.TTemperature Automation with Timer (watchdog)

https://www.tinkercad.com/things/cSDMxH5snf4-2106ttemperature-automation-with-timer-watchdog

```
/**************************************************************************
**
Program Name: Temperature Automation with Timer

Program Description: This updated code includes the Watchdog Timer and a reset button. If
the reset button is not pressed and a wdt_reset() call is not made, the system will
automatically reset after 4 seconds. This will serve as a safety feature in case the system
becomes stuck.

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
**************************************************************************
***/

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/wdt.h>

// Pin definitions
const int tempSensor_pin = A0; // Analog pin for the temperature sensor
const int ledGreen = 3; // Digital pin for the green LED
const int ledRed = 4; // Digital pin for the red LED
const int resetButtonPin = 2; // Digital pin for the reset button

// Timer and status variables
unsigned long greenLEDOffTime = 0;
unsigned long redLEDOnTime = 0;
bool greenLEDStatus = false;
bool redLEDStatus = false;

// Temperature set point
const float tempSetPoint = 25.0;

void setup() {
pinMode(ledGreen, OUTPUT);
pinMode(ledRed, OUTPUT);
pinMode(tempSensor_pin, INPUT);
pinMode(resetButtonPin, INPUT_PULLUP); // Enable the internal pull-up resistor
Serial.begin(9600);

wdt_disable(); // Disable WDT
wdt_enable(WDTO_4S); // Enable WDT with a 4-second timeout
}
```

```
void loop() {
// Check the reset button
if (digitalRead(resetButtonPin) == LOW) {
Serial.println("Reset button pressed. Watchdog Timer reset.");
wdt_reset(); // Reset the WDT
}

// Read the temperature
int tempReading = analogRead(tempSensor_pin);
float voltage = tempReading * 5.0;
voltage /= 1024.0;
float temperatureC = (voltage - 0.5) * 100;

// Write the temperature value to the serial port
Serial.print("Temperature: ");
Serial.print(temperatureC);
Serial.println(" C");

// Temperature control
if (temperatureC > tempSetPoint) {
if (!greenLEDStatus) {
greenLEDStatus = true;
redLEDStatus = false; // Turn off the red LED
greenLEDOffTime = millis() + 5000; // Turn off the green LED after 5 seconds
digitalWrite(ledGreen, HIGH);
digitalWrite(ledRed, LOW);
}
} else if (temperatureC < tempSetPoint) {
if (!redLEDStatus) {
redLEDStatus = true;
greenLEDStatus = false; // Turn off the green LED
redLEDOnTime = millis() + 5000; // Light up the red LED after 5 seconds
digitalWrite(ledRed, HIGH);
digitalWrite(ledGreen, LOW);
}
}

// LED timer control
if (greenLEDStatus && millis() > greenLEDOffTime) {
greenLEDStatus = false;
digitalWrite(ledGreen, LOW);
}

if (redLEDStatus && millis() > redLEDOnTime) {
redLEDStatus = false;
digitalWrite(ledRed, LOW);
}

// Reset the WDT. If you remove this code, the Arduino will reset after 4 seconds.
```

```
wdt_reset();

delay(1000); // Wait for 1 second
}
```

## 2.10.7.Animal Repellent Circuit with Timer

https://www.tinkercad.com/things/6m93AyQSaBV-2107animal-repellent-circuit-with-timer

```
/*********************************************************************************
Program Name: Animal Repellent Circuit with Timer

Program Description: This setup is designed to make the buzzer
sound at random times and for random durations when the value
read from the LDR falls below 70. This can be quite effective
in a device used to scare off animals.

In this code, when the value read from the LDR falls below 70,
a structure is established that will cause the buzzer to sound
at random times (between 5 and 10 seconds) and for random
durations (between 100 and 1000 milliseconds). The Timer1
interrupt is used to track when the buzzer will stop.

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
*********************************************************************************/

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdlib.h>

const int buzzerPin = 3; // Pin connected to the buzzer
const int ldrPin = A0; // Pin connected to the LDR
const int ledPin = 9; // Pin connected to the LED

volatile bool buzzerActive = false;
volatile unsigned long buzzerOffTime = 0;

void setup() {
pinMode(buzzerPin, OUTPUT);
pinMode(ldrPin, INPUT);
pinMode(ledPin, OUTPUT);
Serial.begin(9600);

// Set up Timer1 (16-bit timer)
noInterrupts(); // Disable interrupts
TCCR1A = 0; // Clear TCCR1A register
TCCR1B = 0; // Clear TCCR1B register
TCNT1 = 0; // Reset the timer
OCR1A = 15624; // Generate an interrupt every 1 second (16MHz / 1024 prescaler)
TCCR1B |= (1 << WGM12); // Enable CTC mode
```

```
TCCR1B |= (1 << CS12) | (1 << CS10); // 1024 prescaler
TIMSK1 |= (1 << OCIE1A); // Enable timer interrupt
interrupts(); // Enable interrupts
}

ISR(TIMER1_COMPA_vect) {
unsigned long currentMillis = millis();

if (buzzerActive && currentMillis >= buzzerOffTime) {
noTone(buzzerPin);
buzzerActive = false;
}
}

void loop() {
int ldrValue = analogRead(ldrPin);
int brightness = map(ldrValue, 0, 1023, 255, 0);
analogWrite(ledPin, brightness);

// Write the LDR value to the serial port
Serial.print("LDR Value: ");
Serial.println(ldrValue);

// If the LDR value is below 70
if (ldrValue < 70 && !buzzerActive) {
unsigned long currentMillis = millis();
unsigned long randomDuration = random(100, 1000); // Random duration (100ms - 1000ms)
unsigned long randomDelay = random(5000, 10000); // Random delay (5s - 10s)

if (currentMillis >= randomDelay) {
  tone(buzzerPin, 1000, randomDuration); // Activate the buzzer for a random duration
  buzzerOffTime = currentMillis + randomDuration;
  buzzerActive = true;
}

  }

  delay(100); // Wait for 0.1 second
}
```

## 2.10.8.Animal Repellent Circuit with Timer

https://www.tinkercad.com/things/6a4UFQfGm2N-2108animal-repellent-circuit-with-timer

```
/**************************************************************************
**
Program Name: Animal Repellent Circuit with Timer

Program Description: This setup is arranged to make the buzzer sound at random times and
durations when the value read from the LDR drops below 70. This can be quite effective in a
device used for scaring animals away.

In this code, when the value read from the LDR drops below 70, a structure is set up to make
the buzzer sound at random times (between 5 and 10 seconds) and for random durations
(between 100 and 1000 milliseconds). The Timer1 interrupt is used to track when the buzzer
will stop.

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
**************************************************************************
***/

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdlib.h>

const int buzzerPin = 3; // Pin connected to the buzzer
const int ldrPin = A0; // Pin connected to the LDR
const int ledPin = 9; // Pin connected to the LED
const int resetButtonPin = 8; // Pin for the reset button

volatile bool buzzerActive = false;
volatile unsigned long buzzerOffTime = 0;

void setup() {
pinMode(buzzerPin, OUTPUT);
pinMode(ldrPin, INPUT);
pinMode(ledPin, OUTPUT);
pinMode(resetButtonPin, INPUT_PULLUP); // Enable the internal pull-up resistor
Serial.begin(9600);

// Set up Timer1 (16-bit timer)
noInterrupts(); // Disable interrupts
TCCR1A = 0; // Clear TCCR1A register
TCCR1B = 0; // Clear TCCR1B register
TCNT1 = 0; // Reset the timer
OCR1A = 15624; // Generate an interrupt every 1 second (16MHz / 1024 prescaler)
TCCR1B |= (1 << WGM12); // Enable CTC mode
```

```
TCCR1B |= (1 << CS12) | (1 << CS10); // 1024 prescaler
TIMSK1 |= (1 << OCIE1A); // Enable timer interrupt
interrupts(); // Enable interrupts
}

ISR(TIMER1_COMPA_vect) {
unsigned long currentMillis = millis();

if (buzzerActive && currentMillis >= buzzerOffTime) {
noTone(buzzerPin);
buzzerActive = false;
}
}

void loop() {
int ldrValue = analogRead(ldrPin);
int brightness = map(ldrValue, 0, 1023, 255, 0);
analogWrite(ledPin, brightness);

// Write the LDR value to the serial port
Serial.print("LDR Value: ");
Serial.println(ldrValue);

// If the LDR value is below 70
if (ldrValue < 70 && !buzzerActive) {
unsigned long currentMillis = millis();
unsigned long randomDuration = random(100, 1000); // Random duration (100ms - 1000ms)
unsigned long randomDelay = random(5000, 10000); // Random delay (5s - 10s)
if (currentMillis >= randomDelay) {
  tone(buzzerPin, 1000, randomDuration); // Activate the buzzer for a random duration
  buzzerOffTime = currentMillis + randomDuration;
  buzzerActive = true;
}

}
// Reset button check
if (digitalRead(resetButtonPin) == LOW) {
asm volatile (" jmp 0"); // Manually reset the Arduino
}

delay(100); // Wait for 0.1 second
}
```

## 2.11.1.intro to multithreading

https://www.tinkercad.com/things/cRbS8gKA0HX-2111intro-to-multithreading-/editel

```
/**********************************************************************
**
Program Name: Intro to Multithreading

Program Description: This program will flash an LED at certain time intervals. In addition,
using another timer, we will print a message to the serial monitor every second. This
demonstrates that the two "tasks" operate independently of each other.

This code uses Arduino's millis() function to perform two different tasks at specific time
intervals. The millis() function returns the time in milliseconds that has passed since the
Arduino program started. This allows us to trigger tasks at specific intervals.

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
**********************************************************************
***/

// Pin definitions
const int ledPin = 13;

// Variables for timers
unsigned long previousMillisLed = 0;
unsigned long previousMillisSerial = 0;
const long intervalLed = 1000; // Interval for LED (in milliseconds)
const long intervalSerial = 1000; // Interval for serial printing (in milliseconds)

void setup() {
pinMode(ledPin, OUTPUT);
Serial.begin(9600);
}

void loop() {
unsigned long currentMillis = millis();

// Timer for LED
if (currentMillis - previousMillisLed >= intervalLed) {
previousMillisLed = currentMillis;
digitalWrite(ledPin, !digitalRead(ledPin)); // Change LED state
}
// Timer for serial monitor
if (currentMillis - previousMillisSerial >= intervalSerial) {
previousMillisSerial = currentMillis;
Serial.println("Print this message every second!");
}}
```

## 2.11.2.a.Triple Chain Execution

https://www.tinkercad.com/things/4ij8WAABWcc-2112atriple-chain-execution

```
/*****************************************************************************
**
Program Name: Triple Chain Execution

Program Description:
Here we define three tasks.
1. Flashing an LED at certain intervals.
2. Printing a message to the serial monitor every second.
3. Flashing another LED every 2 seconds.
This code uses the millis() function to run three different
tasks independently. Separate timers and intervals have been
defined for each task.

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
*****************************************************************************
***/

// Pin definitions
const int ledPin1 = 13; // First LED pin
const int ledPin2 = 12; // Second LED pin

// Variables for timers
unsigned long previousMillisLed1 = 0;
unsigned long previousMillisLed2 = 0;
unsigned long previousMillisSerial = 0;
const long intervalLed1 = 1000; // Interval for the first LED (in milliseconds)
const long intervalLed2 = 2000; // Interval for the second LED (in milliseconds)
const long intervalSerial = 1000; // Interval for serial printing (in milliseconds)

void setup() {
pinMode(ledPin1, OUTPUT);
pinMode(ledPin2, OUTPUT);
Serial.begin(9600);
}

void loop() {
unsigned long currentMillis = millis();

// Timer for the first LED
if (currentMillis - previousMillisLed1 >= intervalLed1) {
previousMillisLed1 = currentMillis;
digitalWrite(ledPin1, !digitalRead(ledPin1)); // Change the state of the first LED
}
```

```
// Timer for the second LED
if (currentMillis - previousMillisLed2 >= intervalLed2) {
previousMillisLed2 = currentMillis;
digitalWrite(ledPin2, !digitalRead(ledPin2)); // Change the state of the second LED
}

// Timer for the serial monitor
if (currentMillis - previousMillisSerial >= intervalSerial) {
previousMillisSerial = currentMillis;
Serial.println("Print this message every second!");
}
}
```

## 2.11.2.b.Quadruple Chain Execution

https://www.tinkercad.com/things/gM2TjfjKOHi-2112bquadruple-chain-execution

```
/*********************************************************************************
**
Program Name: Quadruple Chain Execution

Program Description: Four independent tasks operate at different time intervals, each using its
own timer.

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
*********************************************************************************
***/

// Pin definitions
const int ledPin1 = 13; // First LED pin
const int ledPin2 = 12; // Second LED pin
const int ledPin3 = 11; // Third LED pin

// Variables for timers
unsigned long previousMillisLed1 = 0;
unsigned long previousMillisLed2 = 0;
unsigned long previousMillisLed3 = 0;
unsigned long previousMillisSerial = 0;
const long intervalLed1 = 1000; // Interval for the first LED (in milliseconds)
const long intervalLed2 = 2000; // Interval for the second LED (in milliseconds)
const long intervalLed3 = 2500; // Interval for the third LED (in milliseconds)
const long intervalSerial = 1000; // Interval for serial printing (in milliseconds)

void setup() {
pinMode(ledPin1, OUTPUT);
pinMode(ledPin2, OUTPUT);
pinMode(ledPin3, OUTPUT);
Serial.begin(9600);
}

void loop() {
unsigned long currentMillis = millis();

// Timer for the first LED
if (currentMillis - previousMillisLed1 >= intervalLed1) {
previousMillisLed1 = currentMillis;
digitalWrite(ledPin1, !digitalRead(ledPin1)); // Change the state of the first LED
}

// Timer for the second LED
```

```
if (currentMillis - previousMillisLed2 >= intervalLed2) {
previousMillisLed2 = currentMillis;
digitalWrite(ledPin2, !digitalRead(ledPin2)); // Change the state of the second LED
}

// Timer for the third LED
if (currentMillis - previousMillisLed3 >= intervalLed3) {
previousMillisLed3 = currentMillis;
digitalWrite(ledPin3, !digitalRead(ledPin3)); // Change the state of the third LED
}

// Timer for the serial monitor
if (currentMillis - previousMillisSerial >= intervalSerial) {
previousMillisSerial = currentMillis;
Serial.println("Print this message every second!");
}
}
```

## 2.11.2.c.Quintuple Chain Execution (button control)

https://www.tinkercad.com/things/4ubTeUL81dm-2112cquintuple-chain-execution-button-control

```
/*****************************************************************************
**
Program Name: Quintuple Chain Execution

Program Description:

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
*****************************************************************************
***/

// Pin definitions
const int ledPin1 = 13;
const int ledPin2 = 12;
const int ledPin3 = 11;
const int buttonPin = 2; // Button pin

// Variables for timers
unsigned long previousMillisLed1 = 0;
unsigned long previousMillisLed2 = 0;
unsigned long previousMillisLed3 = 0;
unsigned long previousMillisSerial = 0;
const long intervalLed1 = 1000;
const long intervalLed2 = 2000;
const long intervalLed3 = 2500;
const long intervalSerial = 1000;

// Variables for button state
int buttonState = 0; // Current state of the button
int lastButtonState = 0; // Last read state of the button
bool isButtonPressed = false; // Flag to check if the button is pressed

void setup() {
pinMode(ledPin1, OUTPUT);
pinMode(ledPin2, OUTPUT);
pinMode(ledPin3, OUTPUT);
pinMode(buttonPin, INPUT);
Serial.begin(9600);
}

void loop() {
unsigned long currentMillis = millis();
```

```
// Read the button state
buttonState = digitalRead(buttonPin);

// Check if the button state has changed
if (buttonState != lastButtonState) {
// Check for rising edge (if the button is pressed)
if (buttonState == HIGH) {
isButtonPressed = !isButtonPressed; // Change the button state
}
// Update the button state
lastButtonState = buttonState;
}

// Timer for the first LED (Change state when button is pressed)
if (currentMillis - previousMillisLed1 >= intervalLed1 && isButtonPressed) {
previousMillisLed1 = currentMillis;
digitalWrite(ledPin1, !digitalRead(ledPin1)); // Change the state of the first LED
}

// Update timers for other LEDs and serial monitor
// Timer for the second LED
if (currentMillis - previousMillisLed2 >= intervalLed2) {
previousMillisLed2 = currentMillis;
digitalWrite(ledPin2, !digitalRead(ledPin2)); // Change the state of the second LED
}

// Timer for the third LED
if (currentMillis - previousMillisLed3 >= intervalLed3) {
previousMillisLed3 = currentMillis;
digitalWrite(ledPin3, !digitalRead(ledPin3)); // Change the state of the third LED
}

// Timer for the serial monitor
if (currentMillis - previousMillisSerial >= intervalSerial) {
previousMillisSerial = currentMillis;
Serial.println("Print this message every second!");
}
}
```

## 2.11.2.d.6-Link Analog Input Chain

https://www.tinkercad.com/things/56Ybv6jO3hK-2112d6-link-analog-input-chain

```
/**************************************************************************
**
Program Name: 6-Link Analog Input Chain

Program Description: In this code, the readTemperature function reads the analog signal from
the LM35 sensor and returns the temperature value in Celsius degrees. A separate timer is
used for temperature reading, and the read temperature value is printed to the serial port every
5 seconds.

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
**************************************************************************
***/

// Pin definitions
const int ledPin1 = 13;
const int ledPin2 = 12;
const int ledPin3 = 11;
const int buttonPin = 2; // Button pin
const int lm35Pin = A0; // LM35 temperature sensor pin (Analog input)

// Variables for timers
unsigned long previousMillisLed1 = 0;
unsigned long previousMillisLed2 = 0;
unsigned long previousMillisLed3 = 0;
unsigned long previousMillisSerial = 0;
unsigned long previousMillisTemperature = 0; // Timer for temperature reading
const long intervalLed1 = 1000;
const long intervalLed2 = 2000;
const long intervalLed3 = 2500;
const long intervalSerial = 1000;
const long intervalTemperature = 5000; // Temperature reading interval (5 seconds)

// Variables for button state
int buttonState = 0; // Current state of the button
int lastButtonState = 0; // Last read state of the button
bool isButtonPressed = false; // Flag to check if the button is pressed

void setup() {
pinMode(ledPin1, OUTPUT);
pinMode(ledPin2, OUTPUT);
pinMode(ledPin3, OUTPUT);
pinMode(buttonPin, INPUT);
Serial.begin(9600);
```

```
}

void loop() {
unsigned long currentMillis = millis();

// Read the button state
buttonState = digitalRead(buttonPin);

// Check if the button state has changed
if (buttonState != lastButtonState) {
// Check for rising edge (if the button is pressed)
if (buttonState == HIGH) {
isButtonPressed = !isButtonPressed; // Change the button state
}
// Update the button state
lastButtonState = buttonState;
}

// Timer for the first LED (Change state when button is pressed)
if (currentMillis - previousMillisLed1 >= intervalLed1 && isButtonPressed) {
previousMillisLed1 = currentMillis;
digitalWrite(ledPin1, !digitalRead(ledPin1)); // Change the state of the first LED
}

// Update timers for other LEDs and serial monitor
// ... [Update timers for other LEDs and serial monitor]

// Timer for reading temperature
if (currentMillis - previousMillisTemperature >= intervalTemperature) {
previousMillisTemperature = currentMillis;
float temperature = readTemperature(lm35Pin);
Serial.print("Temperature: ");
Serial.print(temperature);
Serial.println(" °C");
}
}

float readTemperature(int pin) {
int reading = analogRead(pin);
float voltage = reading * 5.0;
voltage /= 1024.0;
float temperatureC = (voltage - 0.5) * 100 ; // Convert the voltage from the LM35 sensor to
Celsius degrees
return temperatureC;
}
```

## 2.11.3.Using the protothread library

https://www.tinkercad.com/things/atae6lgAgtW-2113using-the-protothread-library

```
/*********************************************************************
**
Program Name: Using the protothread library (citation)

Program Description:Protothreads is a programming technique for
cooperative multitasking that requires very low memory usage.
Cooperative multitasking is a multitasking method where threads
control the order of execution manually. This means that threads
work cooperatively rather than competing with each other.

The Protothreads library is particularly effective when managing
time-based tasks and enabling multiple tasks to run simultaneously.
However, this technique may not be suitable for more complex
multitasking requirements, such as resource sharing and
synchronization between tasks.

Written by: Kamil Bala
kamilbala42@gmail.com
tw: @tek_elo
Yalova / 2023
*********************************************************************
***/

#if 0
//////////////////////////////////////////////////////
// protothread를 적용하지 않은 버전
//////////////////////////////////////////////////////

const int red_led = 13;
const int green_led = 9;
const int blue_led = 5;

void setup() {
  pinMode(red_led, OUTPUT);
  pinMode(green_led, OUTPUT);
  pinMode(blue_led, OUTPUT);
}

void play_red_led()
{
  digitalWrite(red_led, HIGH);
  delay(500);
  digitalWrite(red_led, LOW);
  delay(500);
}
```

```
void play_green_led()
{
  digitalWrite(green_led, HIGH);
  delay(200);
  digitalWrite(green_led, LOW);
  delay(800);
}

void play_blue_led()
{
  int level;  // 자동변수: 코드블록이 시작할 때 자동으로 생성되었다가
              //         코드블록이 종료될 때 자동으로 소멸되는 변수
  for (level = 0; level <= 255; level += 5) {
          analogWrite(blue_led, level);
          delay(30);
  }
  for (level = 255; level >= 0; level -= 5) {
          analogWrite(blue_led, level);
          delay(30);
  }
}

void loop() {
  play_red_led();
  play_green_led();
  play_blue_led();
}


#else

////////////////////////////////////////////////////////////
// protothread 버전
// https://www.arduino.cc/reference/en/libraries/protothreads/
////////////////////////////////////////////////////////////

// #define USE_PROTOTHREAD_LIBRARY

#if defined(USE_PROTOTHREAD_LIBRARY)

#include "protothreads.h"

#else // #if defined(USE_PROTOTHREAD_LIBRARY)

typedef void * lc_t;

#define LC_INIT(s) s = NULL

#define LC_RESUME(s)                                \
```

```
  do {                                      \
    if(s != NULL) {                         \
      goto *s;                              \
    }                                       \
  } while(0)

#define LC_CONCAT2(s1, s2) s1##s2
#define LC_CONCAT(s1, s2) LC_CONCAT2(s1, s2)

#define LC_SET(s)                           \
  do {                                      \
    LC_CONCAT(LC_LABEL, __LINE__):              \
    (s) = &&LC_CONCAT(LC_LABEL, __LINE__);      \
  } while(0)

#define LC_END(s)

struct pt {
  lc_t lc;
};

#define PT_WAITING 0
#define PT_YIELDED 1
#define PT_EXITED  2
#define PT_ENDED   3

#define PT_INIT(pt)   LC_INIT((pt)->lc)
#define PT_THREAD(name_args) char name_args
#define PT_BEGIN(pt) { char PT_YIELD_FLAG = 1; LC_RESUME((pt)->lc)
#define PT_END(pt) LC_END((pt)->lc); PT_YIELD_FLAG = 0; \
                   PT_INIT(pt); return PT_ENDED; }
#define PT_WAIT_UNTIL(pt, condition)        \
  do {                                      \
    LC_SET((pt)->lc);                       \
    if(!(condition)) {                      \
      return PT_WAITING;                    \
    }                                       \
  } while(0)

#define PT_WAIT_WHILE(pt, cond)  PT_WAIT_UNTIL((pt), !(cond))

#define PT_WAIT_THREAD(pt, thread) PT_WAIT_WHILE((pt), PT_SCHEDULE(thread))

#define PT_EXIT(pt)                         \
  do {                                      \
    PT_INIT(pt);                            \
    return PT_EXITED;                       \
  } while(0)

#define PT_SCHEDULE(f) ((f) < PT_EXITED)
```

```
#define PT_YIELD(pt)                                    \
  do {                                          \
    PT_YIELD_FLAG = 0;                                  \
    LC_SET((pt)->lc);                           \
    if(PT_YIELD_FLAG == 0) {                            \
      return PT_YIELDED;                        \
    }                                           \
  } while(0)

#define PT_YIELD_UNTIL(pt, cond)                        \
  do {                                          \
    PT_YIELD_FLAG = 0;                                  \
    LC_SET((pt)->lc);                           \
    if((PT_YIELD_FLAG == 0) || !(cond)) { \
      return PT_YIELDED;                        \
    }                                           \
  } while(0)

#define PT_SLEEP(pt, delay) \
{ \
  do { \
    static unsigned long protothreads_sleep; \
    protothreads_sleep = millis(); \
    PT_WAIT_UNTIL(pt, millis() - protothreads_sleep > delay); \
  } while(false); \
}
// end of "protothread.h"
//////////////////////////////////////////////////////

#endif // #if defined(USE_PROTOTHREAD_LIBRARY)

const int red_led = 13;
const int green_led = 9;
const int blue_led = 5;

struct pt ptx_play_red_led;  // 쓰레드 실행 정보 변수

PT_THREAD(play_red_led( struct pt *pt))
{
  PT_BEGIN(pt);       // protothread 시작 선언
  for (;;) {          // 무한반복 작업하는 쓰레드인 경우

          digitalWrite(red_led, HIGH);

    // PT_SLEEP은 500ms 대기 기능
    // 타임아웃되지 않으면 바로 return 하고, --> YIELD(양보)
```

```
   // 다음 쓰레드 호출 때 이 곳에서부터 실행됨
        PT_SLEEP(pt, 500);    // delay(500);

        digitalWrite(red_led, LOW);
        PT_SLEEP(pt, 500);    // delay(500);
  }

  PT_END(pt);          // protothread 종료 선언
}

struct pt ptx_play_green_led;

PT_THREAD(play_green_led( struct pt *pt))
{
  PT_BEGIN(pt);
  for (;;) {
    digitalWrite(green_led, HIGH);
    PT_SLEEP(pt, 200);
    digitalWrite(green_led, LOW);
    PT_SLEEP(pt, 800);
  }
  PT_END(pt);
}

struct pt ptx_play_blue_led;  // 정적변수

PT_THREAD(play_blue_led( struct pt *pt))
{
  PT_BEGIN(pt);
  for (;;) {
    // protothread 함수 내에서 변수는 static 변수로 선언해야함
    // 자동변수의 값은 보존되지 않을 수 있음
    //   --> 쓰레드가 임시 멈추었다(return)가 다시 실행될 때
    //       이 전 자동변수의 값은 지워질 수 있음
        static int level; // static (정적변수)
                // 아두이노 프로그램이 시작할 때 생성되어
                // 아두이노 프로그램이 종료될 때 소멸됨
                // 함수(코드블록)이 종로되어도 계속 남아있음
    for (level = 0; level <= 255; level += 5) {
      analogWrite(blue_led, level);
      PT_SLEEP(pt, 30);
    }
    for (level = 255; level >= 0; level -= 5) {
      analogWrite(blue_led, level);
      PT_SLEEP(pt, 30);
```

```
    }
  }
  PT_END(pt);
}

void setup() {
  pinMode(red_led, OUTPUT);
  pinMode(green_led, OUTPUT);
  pinMode(blue_led, OUTPUT);

  // protothread 실행정보 변수 초기화(쓰레드 시작 상태로)
  PT_INIT(&ptx_play_red_led);
  PT_INIT(&ptx_play_green_led);
  PT_INIT(&ptx_play_blue_led);
}

void loop() {
  // 쓰레드 실행, PT_SLEEP()할 때마다 다음 쓰레드가 돌아가며 실행됨
  PT_SCHEDULE(play_red_led(&ptx_play_red_led));

  play_green_led(&ptx_play_green_led);  // PT_SCHEDULE 생략 가능
  play_blue_led(&ptx_play_blue_led);

  // loop() 내에서 긴 delay()를 사용하면 안됨
  //   --> 빠른 쓰레드 순환을 방해하지 않도록
}

#endif  // protothread 버전
```