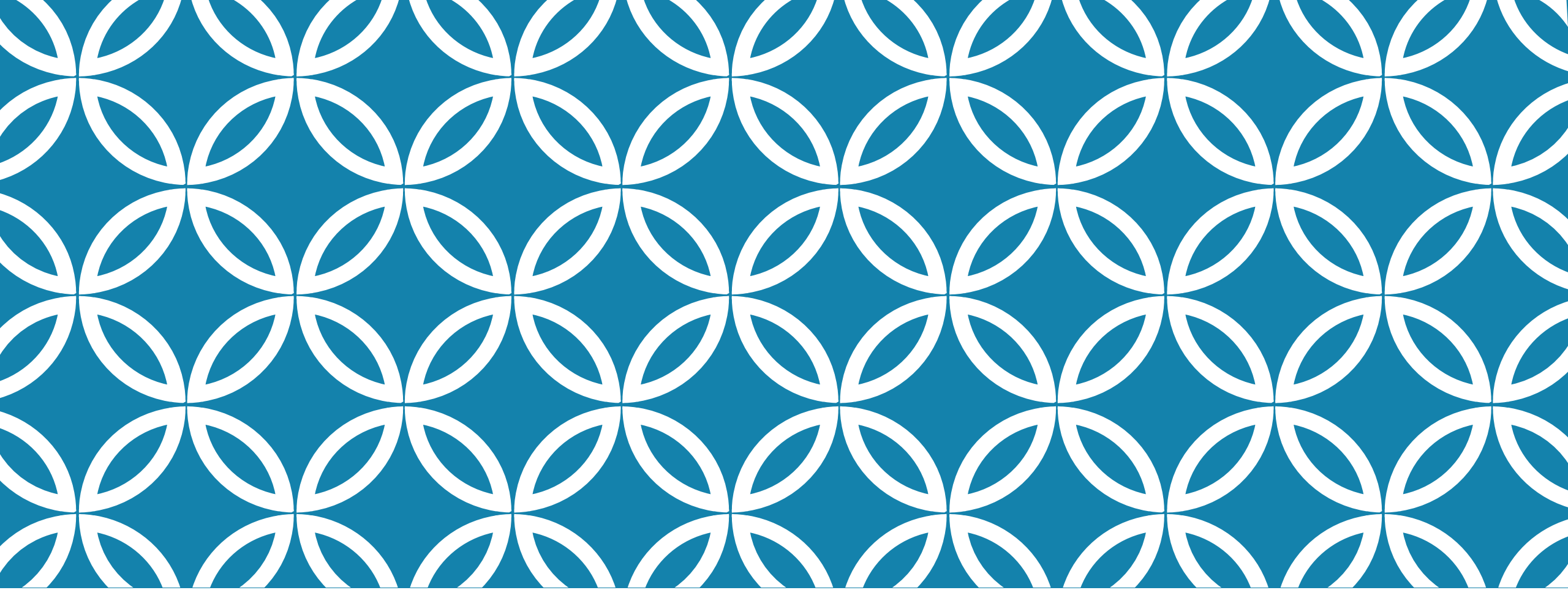


INTERRUPT USING ARDUINO

EMBEDDED SYSTEMS
CS427

INTRO

Interrupts are useful for making things happen automatically in microcontroller programs and can help solve timing problems.

WHAT IS INTERRUPTS?

An interrupt is a signal that is generated by hardware or software when a process or an event needs immediate attention.

- Sources of Interrupts.

Hardware:

- Types:
- It is an electronic signal sent from an external device, or from an internal peripheral (Timer, ADC, ...).
- Maskable: These interrupts can be enabled or disabled by software.
- Non-Maskable: These interrupts can not be disabled.

Software:

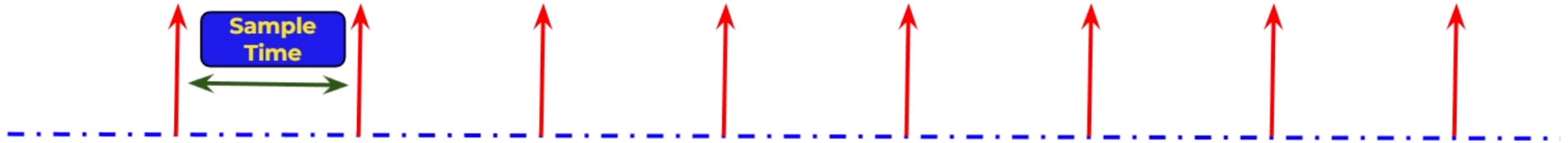
- Types:
- It is caused either by an exceptional condition or a special instruction in the instruction set.
- Exceptions.

POLLING

It refers to actively sampling the status of an external device by a program as a synchronous activity.

Example

- Keyboard polling rate 125Hz.
- This means that the keyboard is being scanned 125 times per second.



ISR – INTERRUPT SERVICE ROUTINE

ISRs are special kinds of functions that have some unique limitations most other functions do not have. An ISR cannot have any parameters, and they shouldn’t return anything.

- It is a function that takes void and returns void.
- It must contain small logic.
- It is not called by the software.
- It mustn't be optimized by the compiler.

ISR – INTERRUPT SERVICE ROUTINE

global variables are used to pass data between an ISR and the main program. To make sure variables shared between an ISR and the main program are updated correctly, declare them as volatile.

INTERRUPT SYNTAX

```
Declare interrupt pin
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)
```

Parameters

interrupt: the number of the interrupt. Allowed data types: int.

pin: the Arduino pin number.

ISR: the ISR to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.

mode: defines when the interrupt should be triggered.

INTERRUPT SYNTAX

Four constants are predefined as valid values:

LOW to trigger the interrupt whenever the pin is low,

CHANGE to trigger the interrupt whenever the pin changes value

RISING to trigger when the pin goes from low to high,

FALLING for when the pin goes from high to low.

INTERRUPT SYNTAX

`digitalPinToInterrupt(pin)`

translate the actual digital pin to the specific interrupt number

BOARD	DIGITAL PINS USABLE FOR INTERRUPTS
Uno, Nano, Mini, other 328-based	2, 3
Uno WiFi Rev.2, Nano Every	all digital pins
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21 (**pins 20 & 21** are not available to use for interrupts while they are used for I2C communication)
Micro, Leonardo, other 32u4-based	0, 1, 2, 3, 7
Zero	all digital pins, except 4
MKR Family boards	0, 1, 4, 5, 6, 7, 8, 9, A1, A2
Nano 33 IoT	2, 3, 9, 10, 11, 13, A1, A5, A7
Nano 33 BLE, Nano 33 BLE Sense	all pins
Due	all digital pins
101	all digital pins (Only pins 2, 5, 7, 8, 10, 11, 12, 13 work with **CHANGE**)

INTERRUPT SYNTAX

Example Code

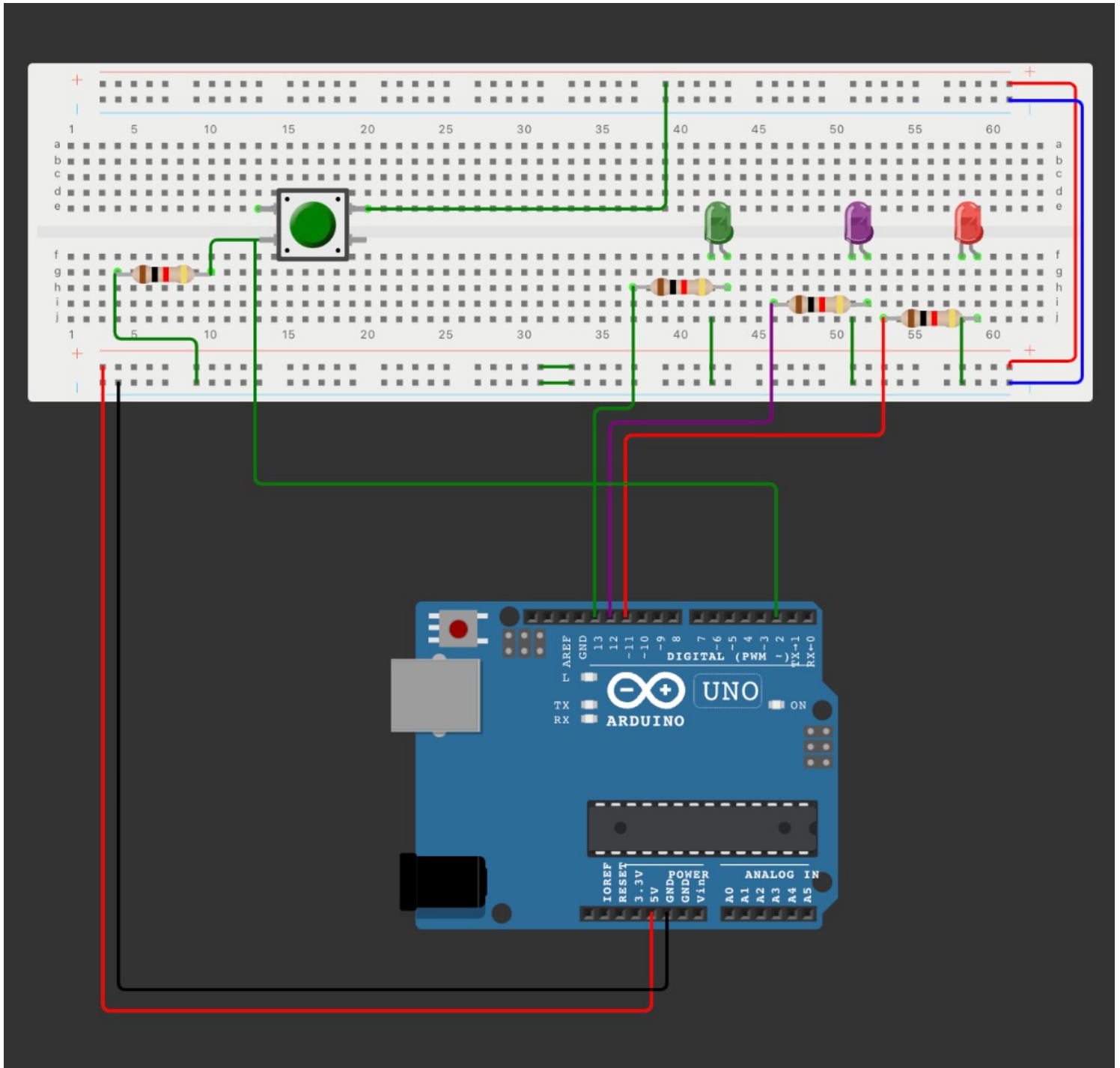
```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```

EXAMPLE 2

1. Hardware Requirements
    1. Three LEDs (LED1, LED2, LED3)
    2. One button (BUTTON0)
2. Software Requirements
    1. Initially, all LEDs are OFF
    2. Once BUTTON0 is pressed, LED0 will be ON
    3. Each press further will make another LED is ON
    4. At the fifth press, LED0 will changed to be OFF
    5. Each press further will make only one LED is OFF
    6. This will be repeated forever
    7. The sequence is described below
    1. Initially (OFF, OFF, OFF)
    2. Press 1 (ON, OFF, OFF)
    3. Press 2 (ON, ON, OFF)
    4. Press 3 (ON, ON, ON)
    5. Press 4 (OFF, ON, ON)
    6. Press 5 (OFF, OFF, ON)
    7. Press 6 (OFF, OFF, OFF)
    8. Press 7 (OFF, OFF, OFF)

EXAMPLE 2

CONNECTION



EXAMPLE 3

1. Hardware Requirements
    1. Three LEDs (LED1, LED2, LED3)
    2. two button (BUTTON0, BUTTON0)
2. Software Requirements
    1. Initially, all LEDs are OFF
    2. Once BUTTON0 is pressed, LED0 will be ON
    3. Each press on BUTTON0 further will make another LED is ON
    4. When BUTTON1 is pressed then the LEDs prigntess with change (3 modes – 100%, 50%,10%)
    4. At the fifth press, LED0 will changed to be OFF
    5. Each press further will make only one LED is OFF
    6. This will be repeated forever
    7. The sequence is described below
        1. Initially (OFF, OFF, OFF)
        2. Press 1 (ON, OFF, OFF)
        3. Press 2 (ON, ON, OFF)
        4. Press 3 (ON, ON, ON)
        5. Press 4 (OFF, ON, ON)
        6. Press 5 (OFF, OFF, ON)
        7. Press 6 (OFF, OFF, OFF)
        8. Press 7 (OFF, OFF, OFF)

EXAMPLE 3

CONNECTION