

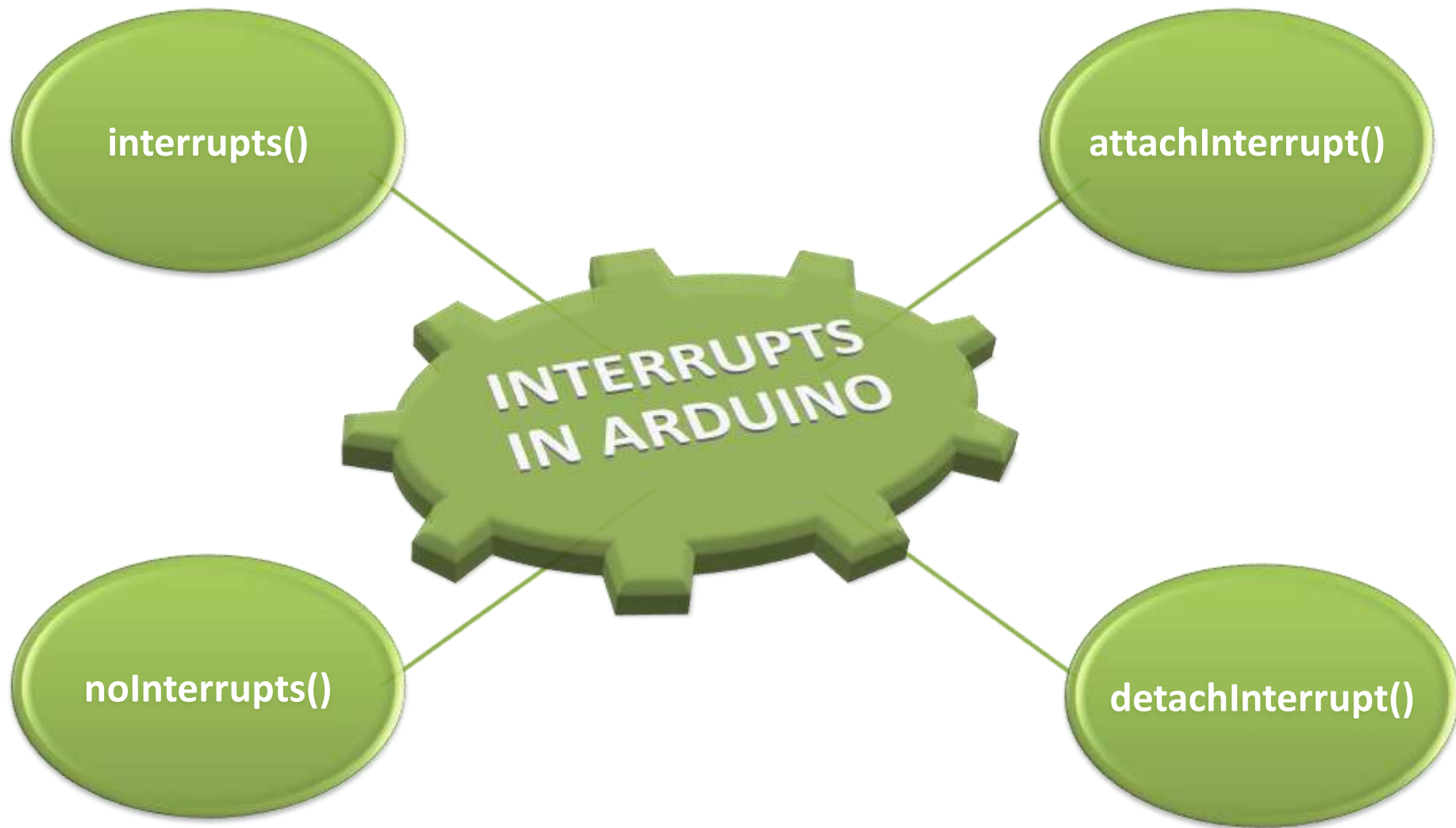
Interrupts Interfacing

Introduction to Interrupts

- Interrupts are mechanisms which enable instant response to events such as counter overflow, pin change, data received, etc.
- In normal mode, microcontroller executes the main program as long as there are no occurrences that would cause an interrupt.

Executing an Interrupt

Upon activation of interrupt in the microcontroller, it executes the instruction defined in interrupt service routine.



Interrupts()

- Interrupts allow certain important tasks to happen in the background and are enabled by default.
- Some functions will not work while interrupts are disabled, and incoming communication may be ignored.
- Interrupts can slightly disrupt the timing of code, however, and may be disabled for particularly sections of code.

noInterrupts()

It interrupts are disable then you can re-enable them with interrupts().

Example:

```
void setup()
{
}

void loop() {
  noInterrupts(); //Disables interrupt
  // critical, time-sensitive code here
  interrupts(); //Enables Interrupt
  // other code here
}
```

attachInterrupt()

- Specifies a function to call when an external interrupt occurs.
- Replaces any previous function that was attached to the interrupt.
- Most Arduino boards have two external interrupts: numbers 0 (on digital pin 2) and 1 (on digital pin 3).
- The Arduino Mega has an additional four: numbers 2 (pin 21), 3 (pin 20), 4 (pin 19), and 5 (pin 18).

Syntax:

`attachInterrupt(interrupt, function, mode);`

Parameters:

- **Interrupt** : the number of the interrupt (int).
 - **Function** : the function to call when the interrupt occurs.
- Mode defines when the interrupt should be triggered.

Four modes 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.

Example:

```
int pin = 13;
int state = LOW;
void setup() {
  pinMode(pin, OUTPUT);
  attachInterrupt(0, blink, CHANGE); }
void loop(){
  digitalWrite(pin, state);
}
void blink() {
  state = !state;
}
```


detachInterrupt()

Turns off the given interrupt.

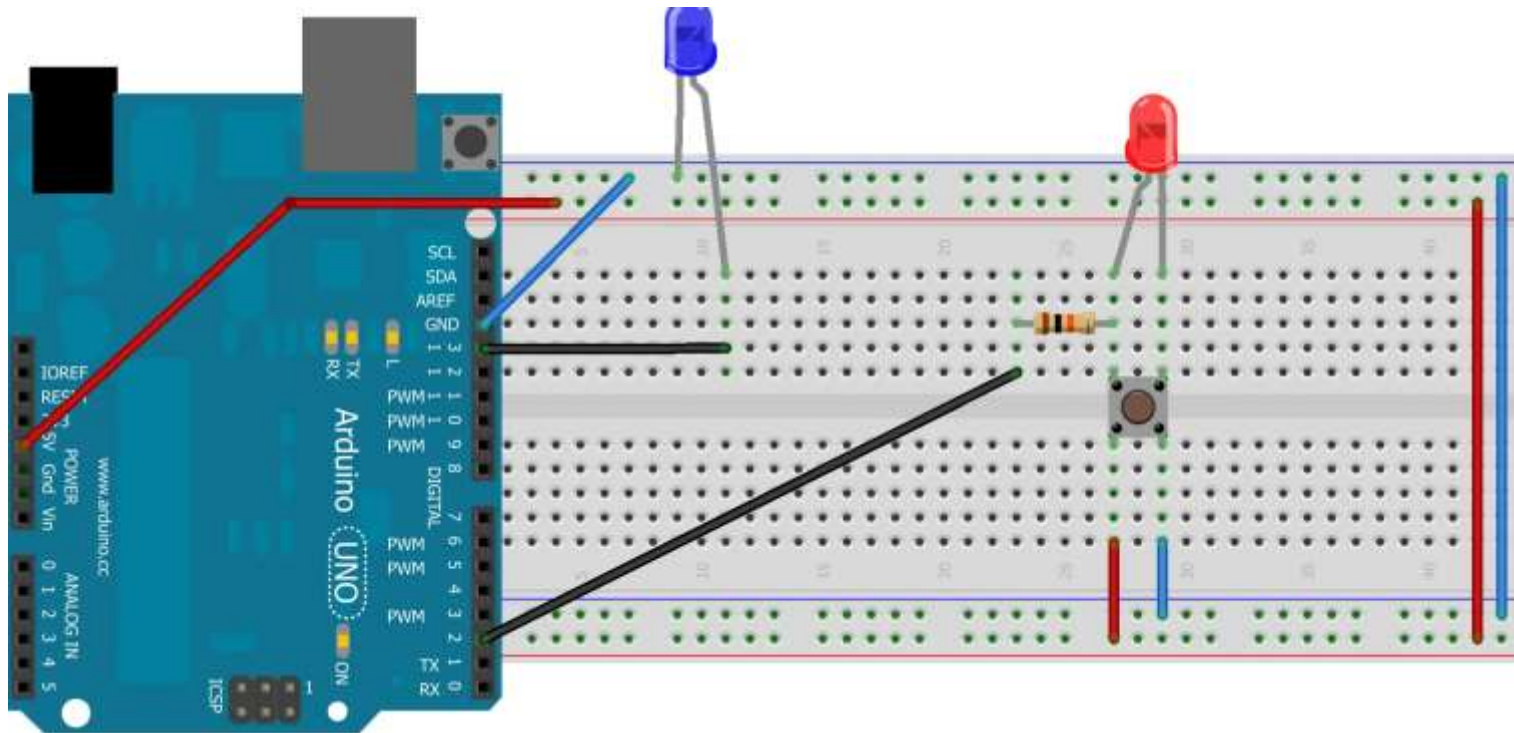
Syntax:

```
detachInterrupt(interrupt);
```

Parameters:

interrupt: the number of interrupt to disable (0 or 1).

Connection Diagram



Project Link : <https://youtu.be/Ko42rdDFtGU>