Make things work together (optional)

Make different task work together

- You might find that the last example in Add an external LED and, another button has some problem.
- When you press one button, then one LED is on for 2 seconds, but during this 2 seconds, no matter what you do, the system will not be responsive.

Pre-requisite:

1. Have 2 button, each controls an LED to switch on for 2 seconds and then turn off via Add an external LED and, another button.

Objectives:

1. Learn the concept of "block", poll.
2. If possible learn the concept of interrupt.

Descriptions:

1. In the last programme of Add an external LED and, another button
   In line : 35 and 44, there are 2 "delay" functions, the delay function "Block" the whole process that means when the delay is functioning, the system is basically frozen and cannot process any input or change any output.

   The blocking part

```
  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (button0State == HIGH) {
    // turn LED on:
    digitalWrite(led0Pin, HIGH);
    // wait for 2 seconds;
    delay(2000);
    // turn LED off:
    digitalWrite(led0Pin, LOW);
  }

  if (button1State == HIGH) {
    // turn LED on:
    digitalWrite(led1Pin, HIGH);
    // wait for 2 seconds;
    delay(2000);
    // turn LED off:
    digitalWrite(led1Pin, LOW);
  }
}
```

2. What is "poll" (please refer to http://en.wikipedia.org/wiki/Polling_(computer_science))
   Basically, polling is constantly checking a status of something. In this case, we need to constantly check how much time has passed after the LED is on.
   So we need to introduce a new function of Arduino SDK, Millis/Micros (Please refer to : https://www.arduino.cc/reference/en/language/functions/time/millis/ and https://www.arduino.cc/reference/en/language/functions/time/micros/)
3. We don't need it to be precise as Microseconds, so we are using millis() in this case.

```
const int button0Pin = 2;     // the number of the pushbutton pin
const int button1Pin = 3;     // the number of the pushbutton pin
const int led0Pin =  13;      // the number of the LED pin
const int led1Pin =  12;      // the number of the LED pin

// variables will change:
int button0State = 0;         // variable for reading the pushbutton status
int button1State = 0;

// these variables record the time:
int led0time;
int led1time;

void setup() {
  // initialize the LED pin as an output:
  pinMode(led0Pin, OUTPUT);
  pinMode(led1Pin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(button0Pin, INPUT);
  pinMode(button0Pin, INPUT);
  // turn LED off initially:
  digitalWrite(led0Pin, LOW);
  digitalWrite(led1Pin, LOW);
}

void loop(){
  // read the state of the pushbutton value:
  button0State = digitalRead(button0Pin);
  button1State = digitalRead(button1Pin);

  // Now if button is low, we "poll" how much time is passed.
  if (button0State == HIGH) {
    // turn LED on:
    digitalWrite(led0Pin, HIGH);
    led0time = millis();  // Recorded the time that button0 pressed
  }else if(millis() - led0time > 2000){  // if 2 seconds has passed after not pressing button 0
    // turn LED off;
    digitalWrite(led0Pin, LOW);
    // this is optional, because mostly the programme won't run long enough
    // for the figure to overflow but just in case
    led0time = millis();
  }

  // Do the same thing for button 1 and LED 1
  if (button1State == HIGH) {
    // turn LED on:
    digitalWrite(led1Pin, HIGH);
    led1time = millis();
  }else if(millis() - led1time > 2000){
    digitalWrite(led1Pin, LOW);
    led1time = millis();
  }
}
```

4. In this tutorial, we won't try to implement interrupt based I/O handling.
    a. However, it will involve using something like a hardware timer (please refer to http://playground.arduino.cc/Code/Timer#.Uzo7wsQW3ks)
    b. Interrupt means that if the device wants the processor's attention, it will inform the processor that it will suspend whatever task it's processing and turns to process the request.
       (Please refer to http://en.wikipedia.org/wiki/Interrupt)
    c. If you are interested in using interrupt, you can try to use the library above. Whenever you press a button, you will set a timer that will inform the processor when 2 seconds has past.
5. In conclusion, take example that you are the processor, and you are cooking something on an oven and it has to cook for 5 minutes.
    a. Blocking using "delay" is like you do nothing, but staring at the watch for 5 minutes and then turn off the oven.
       (Indicates this is most precise, but the whole processing power is wasted in the period)
    b. Polling is like you constantly look at your watch when doing something else, and if 5 minutes has passed, you go turn off the oven.
       (Indicates this will some how be a little bit inefficient)
    c. Interrupt is like you have an alarm clock when you set it to go off in 5 minutes, you go do whatever else you are working on, when the alarm go off, you turn off the oven.
       (Indicates this is mostly efficient, but needs a hardware that supports it)