LOW-POWER SLEEP OPTIMISATION

OVERVIEW

This project implements an **ultra-low-power, interrupt-driven microcontroller system** using the ATmega328P, powered by a coin cell. The main goal is to maximize battery life by keeping the MCU in **Power-Down sleep mode** whenever it is idle. A button connected to the external interrupt pin (PD2/INT0) allows the MCU to wake up only when user input is detected, ensuring that energy is consumed only during meaningful activity.

The hardware consists of a single LED connected to PB5, a push-button with the internal pull-up resistor enabled, and the MCU itself. The software is minimal and efficient: the LED pin is configured as output, the button as input, and the external interrupt is set to trigger on the falling edge. An **Interrupt Service Routine (ISR)** toggles the LED state whenever the button is pressed, while the main loop repeatedly executes the sleep instruction to maintain Power-Down mode.

By leveraging Power-Down mode, the MCU stops its CPU and most peripherals, reducing current draw to **sub-microamp levels**. This approach makes it possible to extend the battery life of a standard coin cell from a few weeks to **up to six months or more**, depending on usage. The project demonstrates key embedded systems concepts such as **interrupt-driven design, register-level programming, and low-power optimization**, making it a practical example for energy-efficient IoT devices or portable electronics.

PROCESS

I tried to only do this using inbuild avr registers for everything from pin allocation,sleep mode,interupts everything for thus I build complete low power optimisation of a coin cell

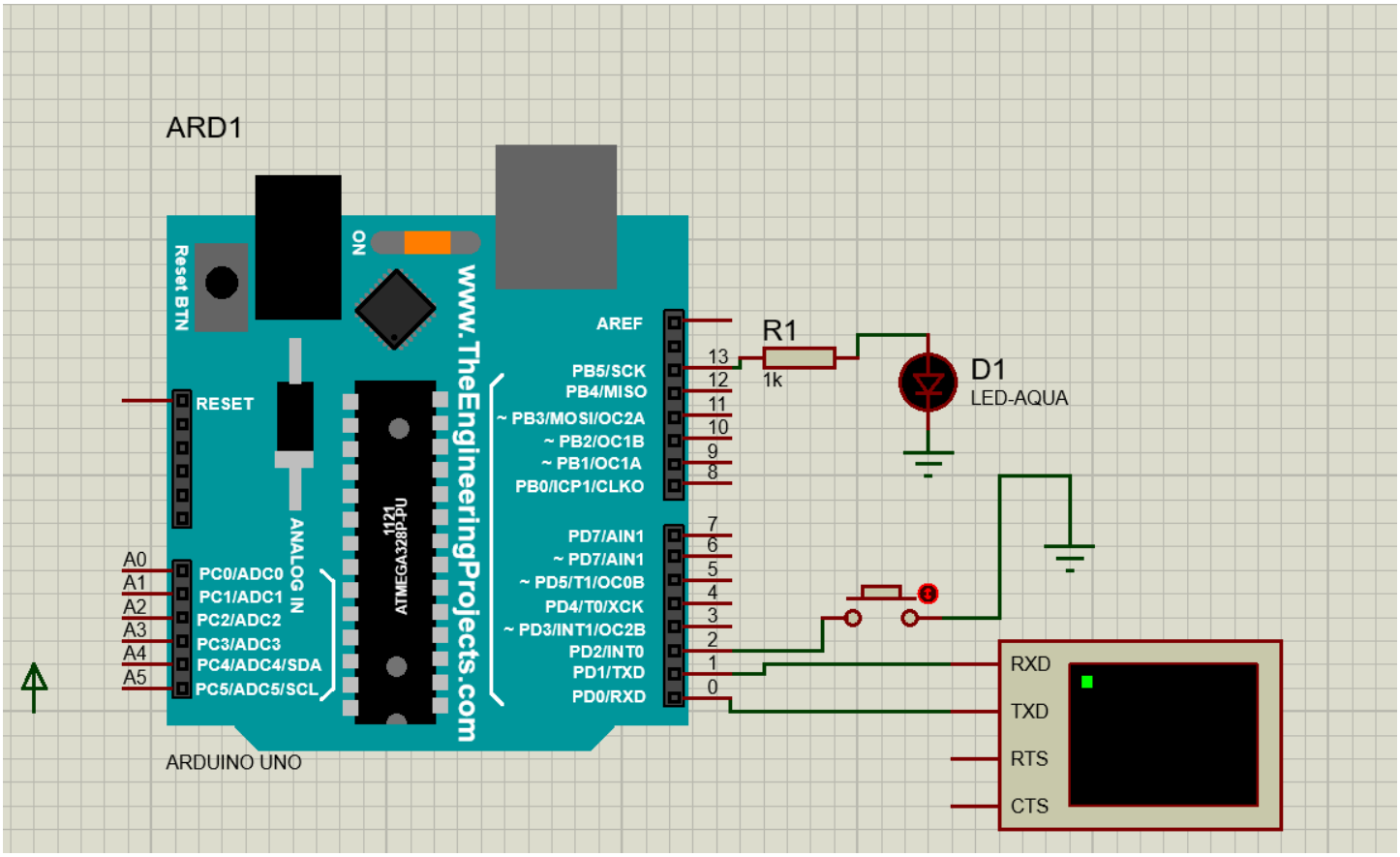
now this only uses an average of less than 1 microampere of current which will drastically increase the battery to about an year or so

only upon clicking the button will mcu awake and the led will toggle to on and off correspondingly

source code

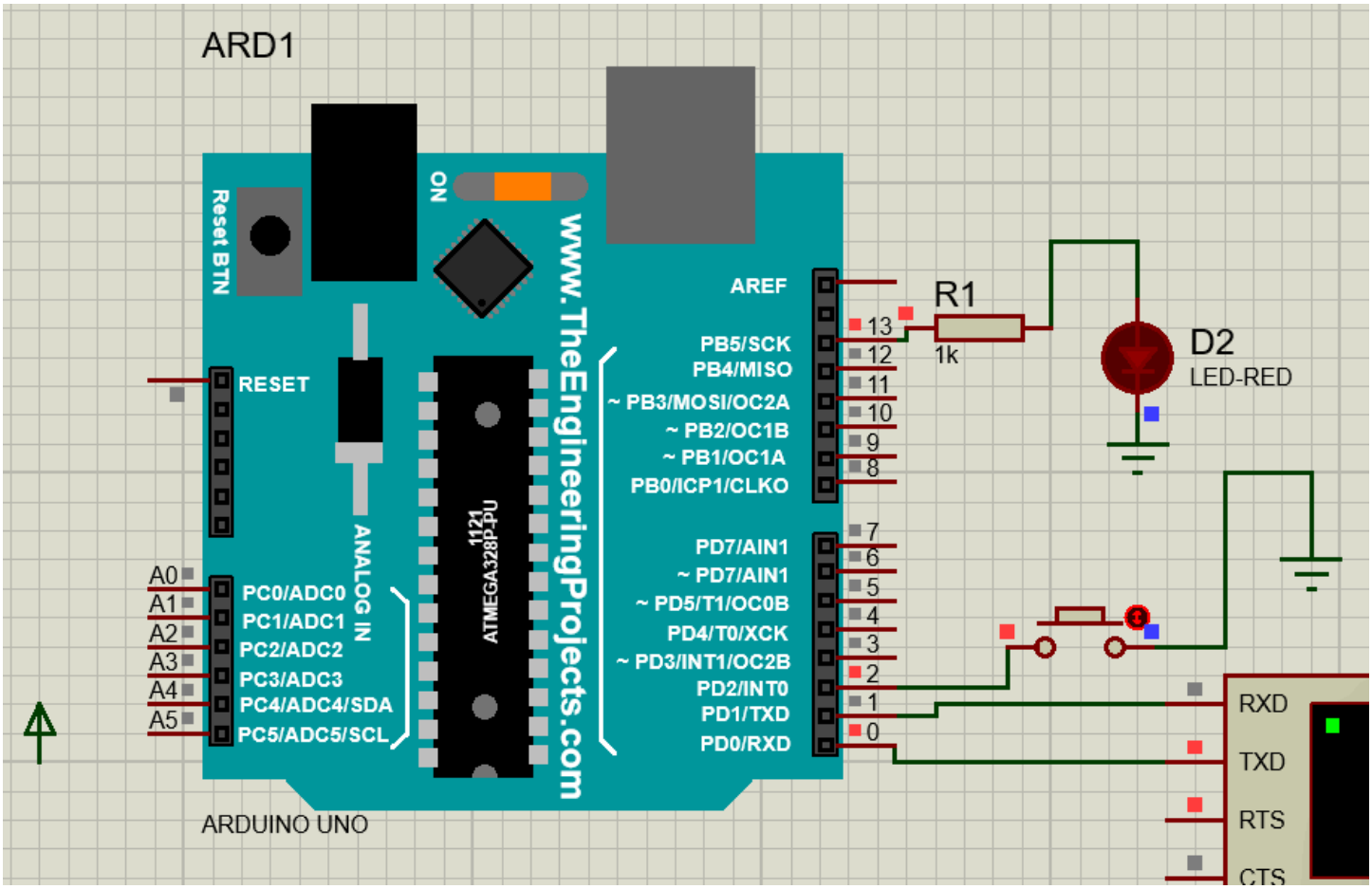
```
#include <avr/io.h>

#include <avr/interrupt.h>


ISR(INT0_vect) {

   PORTB ^= (1 << PB5);

}


int main(void) {

   DDRB |= (1 << PB5);

   PORTB &= ~(1 << PB5);


   DDRD &= ~(1 << PD2);

   PORTD |= (1 << PD2);


   EICRA |= (1 << ISC01);

   EIMSK |= (1 << INT0);


   sei();


   SMCR = (1 << SE) | (1 << SM2);


   while (1) {

      __asm__ __volatile__("sleep");

   }

}
```

output



this is the basic circuit with a spst switch and a red led

led toggle



after toggling the led will glow and thus the circuit awakes