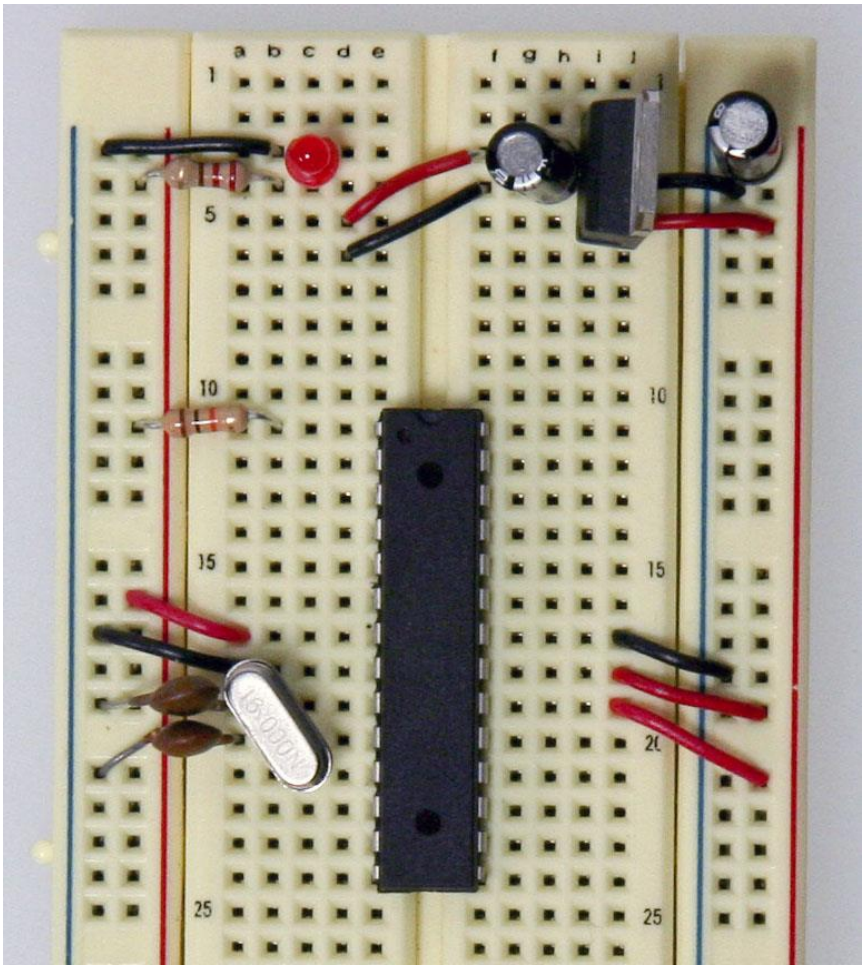
GPIO Low Power



Alexandru Malgras 10/01/2022

Introduction

This assignment is the continuation of the GPIO assignment. The main goal is to extend the program created in a way that will reduce the energy cost of running the system.

Research

Community research

Browsing the internet, I was able to find a couple tutorials explaining the process of reducing the power consumption of an Arduino.

From what I understood the most efficient way to preserve power is to put the Arduino in a deep sleep mode and wake it up only when it is required. Putting an Arduino in deep sleep stops processes that are not required from running and wakes them up when necessary. The wakeup call can be done using interrupts or timers.

Other methods used to reduce the power consumption:

- Disable the analog to digital converter.
- Disable the brownout detection.
- Set pin modes to input.

Main sources of information used to complete the assignment:

- Tutorial on Arduino low power
- Arduino Uno data sheet

Testing

Stage 1

The amperage before any power reduction code was added.

Stage 2

The amperage after the analog to digital converter and brownout detection were disabled and the pin modes were set to input.

The analog to digital converter was not needed because the analog pins were not used in the assignment and the information that was gathered from them had no purpose.

The brownout detection was disabled because there is a high confidence that the Arduino will run at 5V and even if it drops it would not be much of a problem, considering it does not drop to extreme low values.

The pin modes were set to input to enable internal pull-up resistor. The reason why I did this is to make all the pins use as low voltage as possible. This method was the most efficient way to reduce the power consumption and it saved around 1.9 mA.

Stage 3

The amperage after the Arduino was put in a deep sleep mode. However, I was not able to successfully wake up the Arduino back and the LED would not turn on. This part was left out the code and only a fully working version was submitted.

The code

```
#include <Arduino.h>


bool button_1_down = false;
bool button_2_down = false;
unsigned long button_1_down_start = 0;
unsigned long button_2_down_start = 0;
bool button_1_pressed = false;
bool button_2_pressed = false;
unsigned long button_1_pressed_millis = 0;
unsigned long button_2_pressed_millis = 0;


ISR(PCINT0_vect)
{
    // Remove interrupt flag
    PCIFR |= (1 << PCIF0);
    button_1_down = (PINB & (1 << 0)) == 0;
    button_2_down = (PINB & (1 << 1)) == 0;
    if (button_1_down)
    {
        button_1_down_start = millis();
        button_1_pressed = false;
        button_1_pressed_millis = 1;
    }
    else if (button_1_pressed_millis == 1)
    {
        button_1_pressed = true;
        button_1_pressed_millis = millis() - button_1_down_start;
```

```
    }
    if (button_2_down)
    {
        button_2_down_start = millis();
        button_2_pressed = false;
        button_2_pressed_millis = 1;
    }
    else if (button_2_pressed_millis == 1)
    {
        button_2_pressed = true;
        button_2_pressed_millis = millis() - button_2_down_start;
    }
}


unsigned long millis_target_led_1 = 0;
unsigned long millis_target_led_2 = 0;


void setup()
{
    // Put your setup code here, to run once:


    // Set all pins to input
    DDRB = 0x00;
    DDRC = 0x00;
    DDRD = 0x00;
```

```
// Enable pull-up on all pins (-1.9 mA)
PORTB = 0xFF;
PORTC = 0xFF;
PORTD = 0xFF;


// Apply pin modes
DDRD |= (1 << 6);
DDRD |= (1 << 7);
PORTD &= ~(1 << 6);
PORTD &= ~(1 << 7);
DDRB &= ~(1 << 0);
DDRB &= ~(1 << 1);
PORTB &= ~(1 << 0);
PORTB &= ~(1 << 1);


// Enable pin change interrupts for the required port
PCICR |= (1 << PCIE0);
// Add button pin to pin change mask register
PCMSK0 |= (1 << PCINT0);
PCMSK0 |= (1 << PCINT1);


// Disable ADC (-0.05 mA)
ADCSRA &= ~(1 << ADEN);


// Disable the brown-out detection (-0.05 mA)
MCUCR |= (1 << BODS) | (1 << BODSE);
```

```
    MCUCR = (MCUCR & ~(1 << BODS)) | (1 << BODSE);
}


void loop()
{
    // Put your main code here, to run repeatedly:


    if (millis() >= millis_target_led_1)
    {
        if (button_1_pressed)
        {
            if (button_1_pressed_millis > 500)
            {
                millis_target_led_1 = millis() + 1000;
            }
            else if (button_1_pressed_millis > 20)
            {
                millis_target_led_1 = millis() + 200;
            }
            if (button_1_pressed_millis > 0)
            {
                PORTD ^= (1 << 6);
            }
        }
    }
    if (millis() >= millis_target_led_2)
    {
        if (button_2_pressed)
```

```
        {
            if (button_2_pressed_millis > 500)
            {
                millis_target_led_2 = millis() + 1000;
            }
            else if (button_2_pressed_millis > 20)
            {
                millis_target_led_2 = millis() + 200;
            }
            if (button_2_pressed_millis > 0)
            {
                PORTD ^= (1 << 7);
            }
        }
    }
}
```