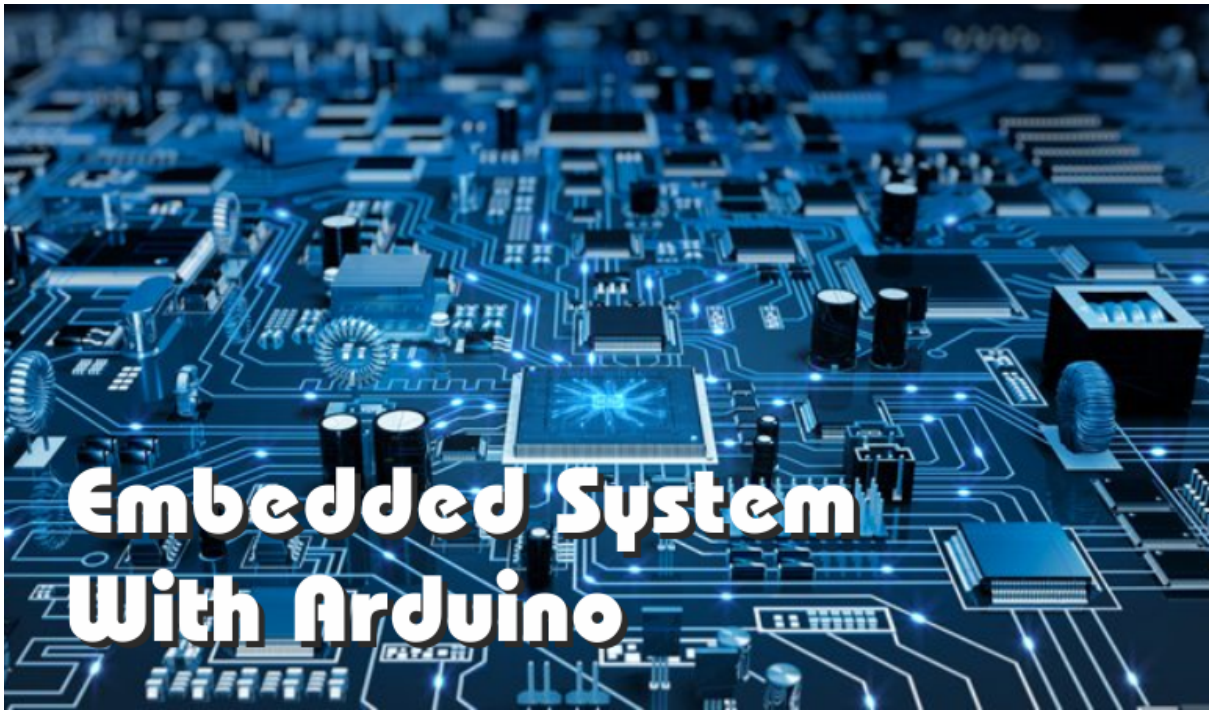
GPIO Challenge

By Jelle Schroijen (3578771) and Alexandru Malgras (3733041)



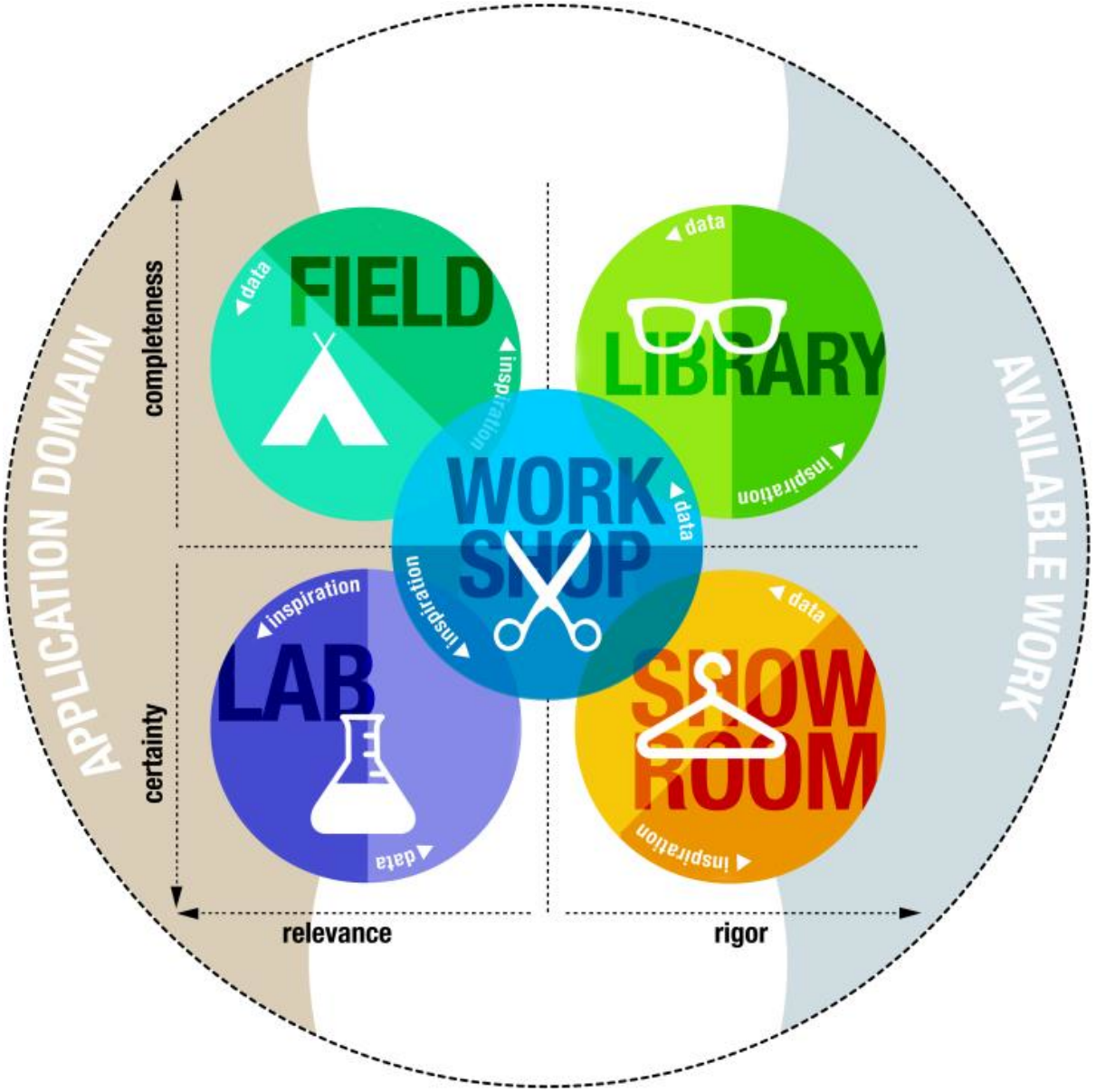
13/09/2021

Introduction

We were challenged to come up with a solution for an Arduino that makes use of multiple buttons, registering different functionalities on long or short presses. The Arduino uses multiple lights as an output. The design should be expandable, but we will use two buttons and two LEDs to test out our solution. The goal of the project is to make the solution as efficient as possible both when it comes to power, but also processing speed.

This was initially a project not meant to be done in teams, however, because Alexandru was unable to find a place to stay near Eindhoven, we decided to work together. Jelle would still be able to go to classes on location, and then the information would be relayed to Alexandru.

Dot Research Framework



Library

a. Expert Interview

From the teacher we learned that a better way of timing and delaying operations in the arduino is by using the millis function instead of delay. They also explained how to do basic interrupts using the registers of the Atmega328p.

b. Community Research

We used the available arduino documentation where we got insight on different topics, such as external interrupts and General Purpose Input/Output (GPIO), though it was explained in examples using the Arduino library.

We used the datasheet for the Atmega328p to find which registers to use to set up the GPIO pins for turning on an LED or responding to a button press.

c. Benchmark Test

We tested the program with different solutions to see which will bring the most efficiency. There were several factors we considered to be part of our 'efficiency grading'. These include memory required, general program speed or responsiveness and power usage. Although we only show the final result here, there have been many iterations before we reached the current program.

d. Code Review

Since we worked as a group we reviewed each other's code to find possible errors.

e. System Test

We performed tests on each part of the code individually to make sure every requirement specified in the assignment is performed correctly.

Components

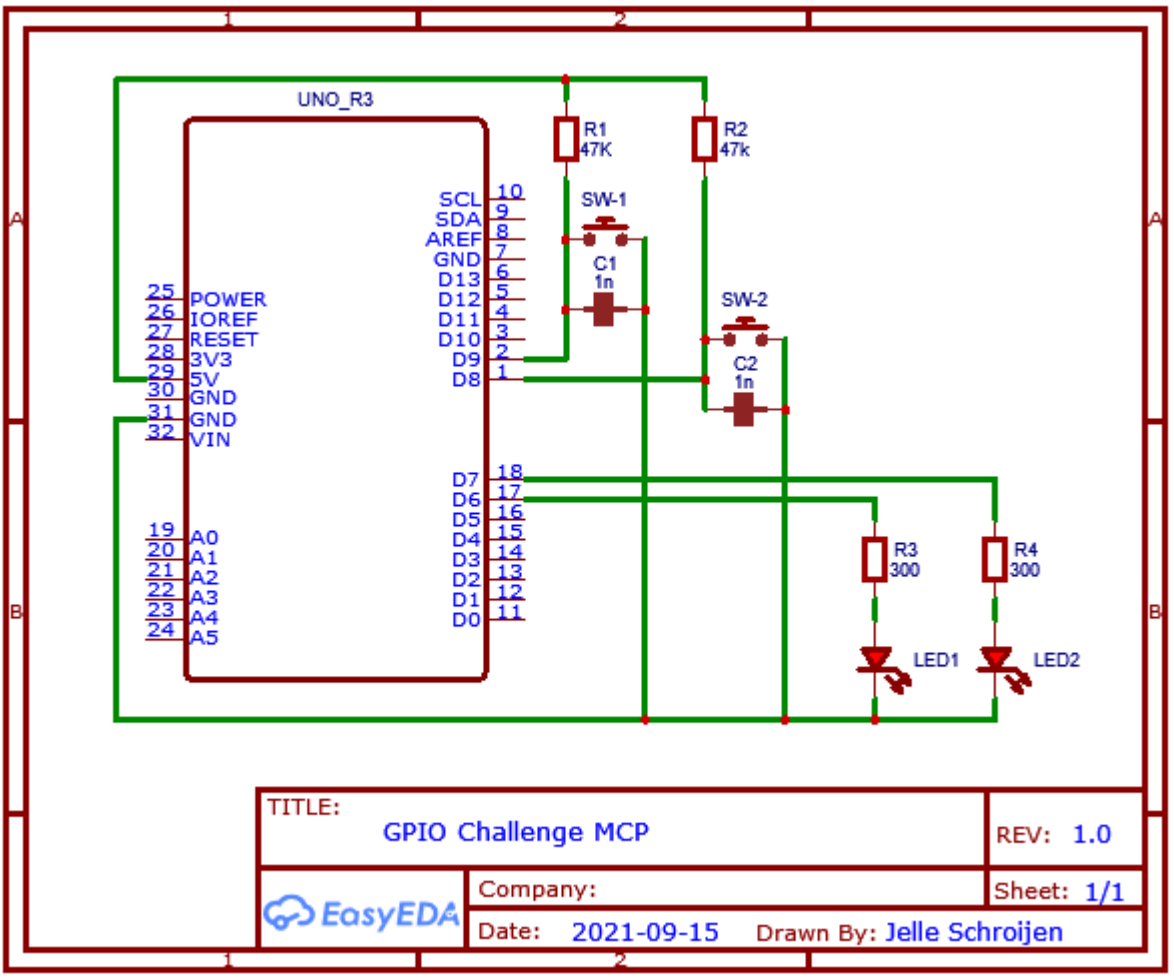
Bill of materials (BOM)

- 1x Arduino UNO R3
- 2x Push button switch
- 2x Red LED
- 2x 47KΩ .5W resistor [1]
- 2x 300Ω .5W resistor [2]
- 2x 1nF Ceramic capacitor [3]

1. We chose a 47,000Ω resistor because it is a common resistor value and it should limit the current flow through the button and to the Arduino.
2. We chose a 300Ω resistor because it is a common resistor value and it limited the current through the LEDs to roughly 11mA at 1.6V. We didn't go for the more typical 220Ω resistor because we found the LEDs to be quite bright. This should also help with the lifespan of the LEDs and the overall power consumption.
3. We decided to use some small ceramic capacitors in parallel with the push buttons, because they would act kind of unpredictable otherwise. We suspect this is because of the sudden power-draw whenever you press the button, which would make the falling and rising edges on the input pins less pronounced.

We used the following rule (Ohm's law) for the calculation of the current: $R = \frac{U}{I}$

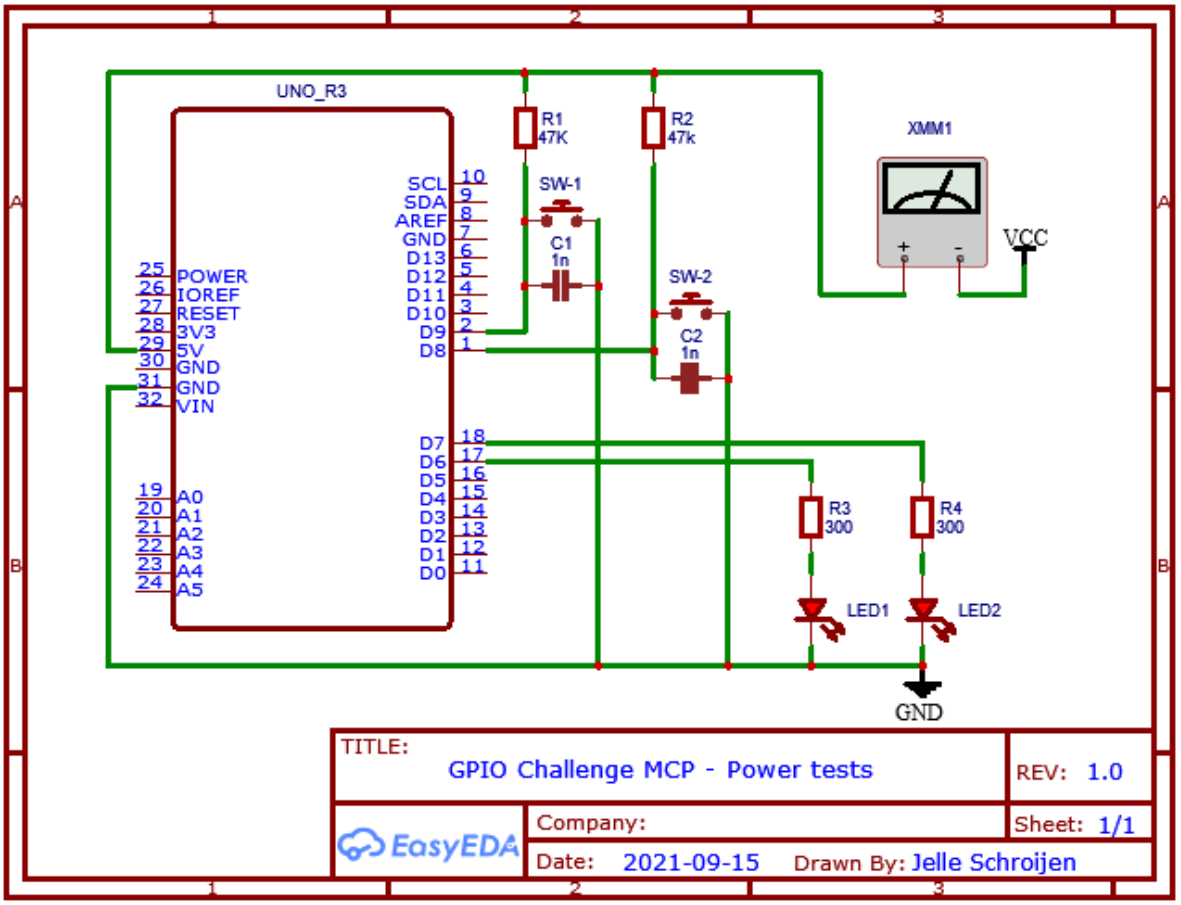
Circuit diagram



The software used to make the circuit diagram was [EasyEDA]

Testing setup

Setup explanation



The software used to make the circuit diagram was [EasyEDA]

In the diagram shown above, you are able to read the current drawn by the arduino on the multimeter denoted as XMM1. With this setup, you cannot use the input USB of the arduino as the power source, but we used a connector to manually supply 5V to the VCC input and GND. We knew that the input voltage was 5V, so we were also able to calculate the power usage after the tests.

Test results:
We tested two versions of the program on the power testing setup. The first version is the final code shown below. In the second version we made use of the LowPower library. We did not use it in the final result because we were unsure if it was allowed to use external libraries.

Final code:

Idle current draw:

25mA = 2.5e-2A

1.25e-1W at 5.00V

Current draw with both buttons pressed and both LEDs on:

36mA = 3.6e-2A
1.8e-1W at 5.00V

Code using the LowPower library:

Idle current draw:

9mA = 9e-3A
4.5e-2W at 5.00V

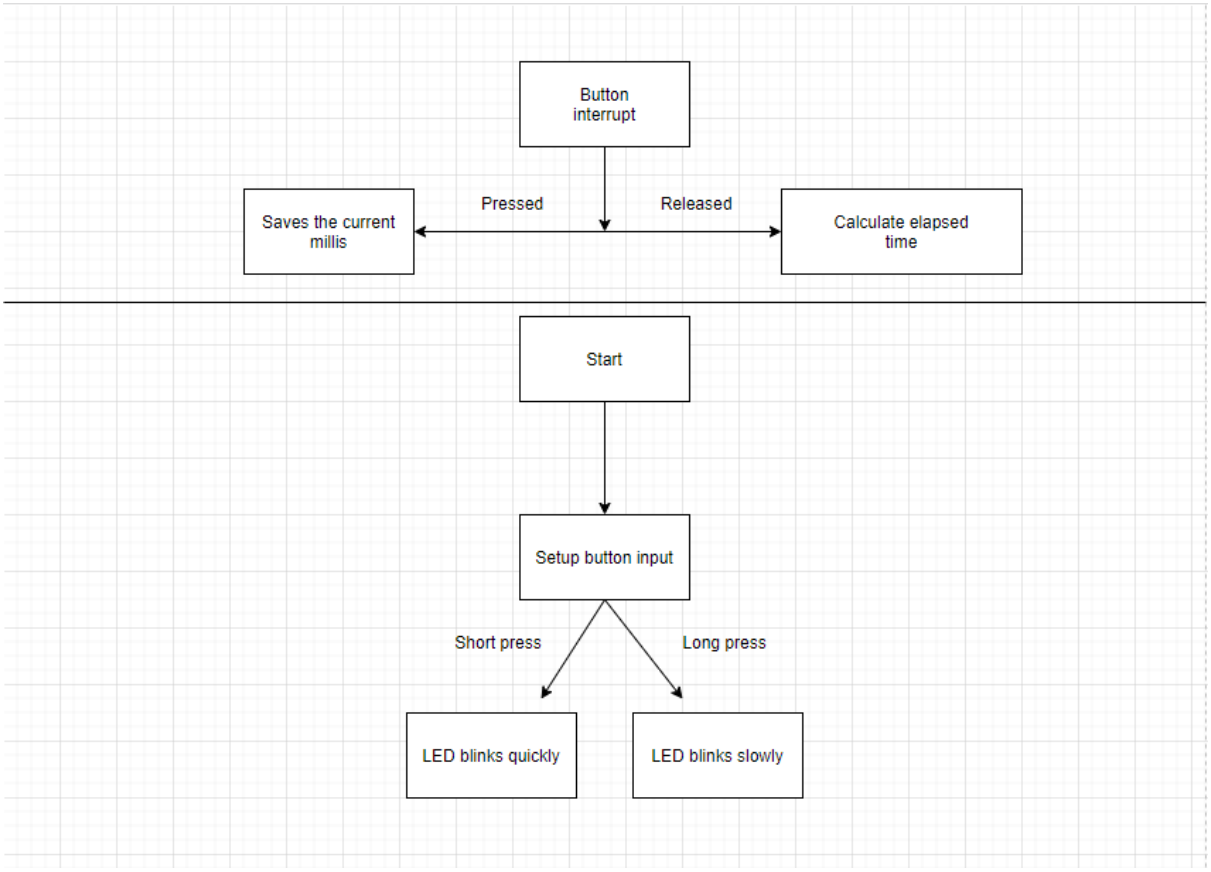
Current draw with both buttons pressed and both LEDs on:

19mA = 1.9e-2A
9.5e-2W at 5.00V

From these tests, we can conclude that it is definitely a good idea to use a library such as the one we used if you care about power consumption. Situations where you have the Arduino powered by a battery for example can yield much better results, with less than half the power draw in an idle mode.

Final code

State diagram



The software used to make the state diagram was Draw.io

The code

The software used to format the code was Hilite

```
#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:
  Serial.begin(9600);

  // 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);
}

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);
      }
    }
  }
}
```

Conclusion

To conclude this project, we would like to say that we learned a lot about the way a processor works. Not only that, we actually got to control said processor in a much more precise way than we had ever done before. This came with its own challenges, but once we were able to overcome those, we felt like we could make some very fine-tuned code.

The biggest difference between the programming style we were used to (High-level languages like C# and Python) and the one we had to apply for this project was the very close relation with the processor itself and the hardware you attach to it. Normally we would have never looked at things like datasheets and circuit diagrams, but having to do this made the project much broader than we had first imagined.

Reflection

If we were to re-do this project from the start, we would have started by doing more research related to the necessary diagrams and how to test things like power-consumption. We were able to do this just fine in the end, but the project-flow would have been much smoother if we didn’t continually have to update the diagrams because we discovered something during programming.

References

Research

Arduino reference

The official Arduino language reference is a place where you can find all about the Arduino library.

Link: https://www.arduino.cc/reference/en/

Atmel Atmega328p datasheet

The Atmel Atmega328p is the processor used in the well known Arduino Uno. The datasheet for this processor specifies most of what goes on inside, including the names of the registers you will need if you do not want to use external functions provided by the Arduino library.

Link:
https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

Tools

Ohm's law

Ohm's law is one of the laws commonly used to calculate things like Voltage (U), Current (I), Resistance (R) and Power (P).

Link: https://en.wikipedia.org/wiki/Ohm's_law

EasyEDA designer

EasyEDA is a free tool that allows you to create simple or advanced circuit diagrams. It also allows for the placement on a PCB and can even give you a 3D representation, however, these functionalities were not used for this project.

Link: https://easyeda.com/editor

Draw.io

Draw.io is a free tool that allows you to make many types of diagrams, including the UML diagrams that were used in this project.

Link: https://app.diagrams.net/

Hilite

Hilite is a free tool that allows you to convert raw source code into a stylized HTML & CSS document.

Link: http://hilite.me

Components

Arduino Uno R3

The Arduino Uno R3 is a development board based on the Atmel Atmega328p processor. It is one of the most popular go-to development boards for DIY electronic projects.

Link: https://www.arduino.cc/en/Guide/ArduinoUno/

Button

The button is an electrical component meant to alter the flow of electricity. There can be two states for a button; the open and closed state. The button will switch between these states whenever it is pressed.

Link: https://en.wikipedia.org/wiki/Push_switch

Resistor

The resistor is an electrical component meant to limit the current flow between other electrical components

Link: https://en.wikipedia.org/wiki/Resistor

Capacitor

The capacitor is an electrical component that is able to store small amounts of electrical energy. Unlike a battery, the electrical energy can be retrieved very quickly. It is often used in electrical circuits to smooth out the voltage that might not be perfectly steady.

Link: https://en.wikipedia.org/wiki/Capacitor

Light emitting diode (LED)

The Light emitting diode, usually called the LED, is an electrical component that only allows current to flow in one direction (under normal circumstances). It creates light in doing so. The LED comes in many colors. It is also one of the most efficient ways of creating light using electricity.

Link: https://en.wikipedia.org/wiki/Light-emitting_diode