Lab Worksheet 1

CSE360: Computer Interfacing

Department of Computer Science and Engineering

Lab 01: Introduction to STM32 development board, using the user button on the board to control a LED.

I. Topic Overview

The lab is designed to introduce the basics of a microcontroller using STM32F446RE development board along with the Keil µVision5 IDE. The students will get introduced to the pins of the board, their functions and lastly, learn how to use the GPIO - General Purpose Input Output pins for interfacing simple I/O devices like LED and user button on the board.

II. Learning Outcome

After this lab, students will be able to:

1. Know the various pins of the STM32 board and their functions.
2. Build a simple circuit with the board and I/O devices.
3. Do register level programming of the GPIO pins to control different circuits for different usages.

III. Materials

- STM32 development board (e.g., STM32F446RE)
- LED (Light Emitting Diode)
- Resistor (appropriate value for current limiting, typically around 220 ohms)
- Jumper wires
- Breadboard

IV. Keil µVision5 IDE

Installation Guideline can be found in this link: Keil µVision5 Installation Guideline

V. STM32F446RE Overview

The STM32F446xx devices use the high-performance Arm® Cortex®-M4 32-bit RISC core, which can operate at up to 180 MHz. The Cortex-M4 core has a floating point unit (FPU) that supports all Arm® single-precision data-processing instructions and types. The board also comes with a debugger.
The STM32F446RE devices include high-speed embedded memories (Flash memory up to 512 Kbytes and SRAM up to 128 Kbytes), backup SRAM up to 4 Kbytes, and a wide range of enhanced I/Os and peripherals that are connected to two APB buses, two AHB buses, and a 32-bit multi-AHB bus matrix. The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously.

All devices include three 12-bit ADCs, two DACs, a low-power RTC, twelve general-purpose 16-bit timers including two PWM timers for motor control, and two general-purpose 32-bit timers. There are 8 GPIO ports (A-H) connected to an AHB bus. Each port consists of 16 pins.

There are upto 20 communication interfaces which include 4 I^2C for serial communication (synchronized by common clock), 4 USART and 2 UART to communicate with other devices each having a single dataline to send and receive data, 4 SPI each having 2 datalines for sending and receiving data, SDIO for sd card, 2 SAI (serial audio interface). There is a USB 2.0, an 8-14 bit parallel camera interface, a CRC calculation unit among many other features.

The layout of the board used in the lab is illustrated in the following diagrams.

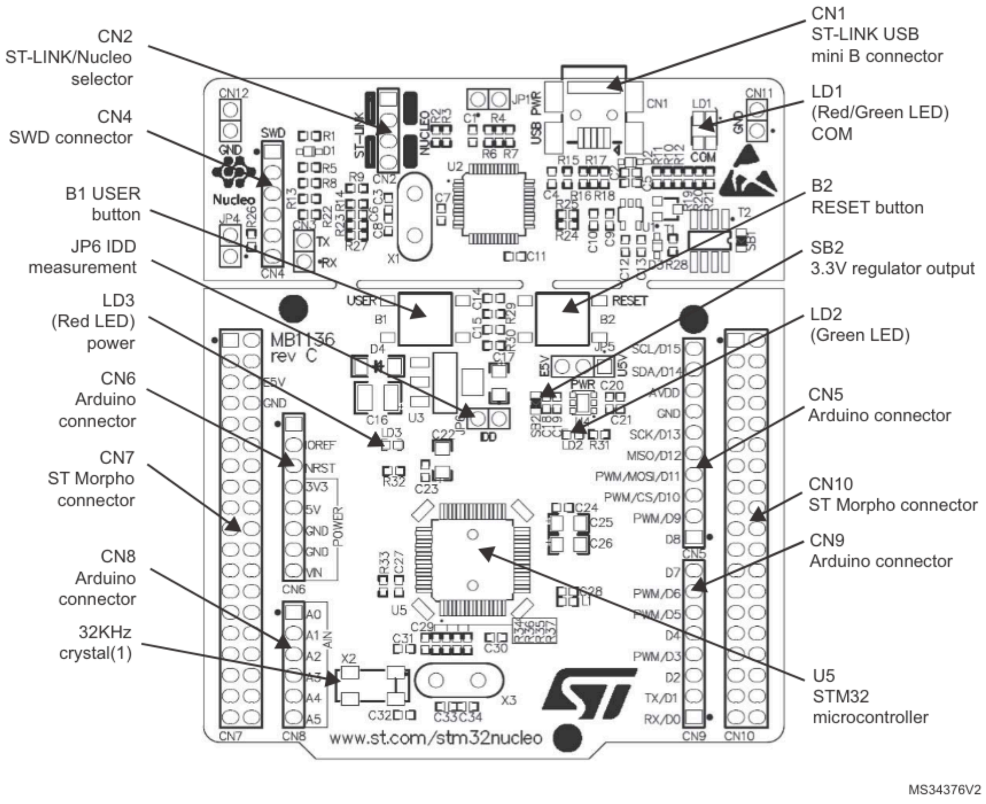


Figure 1: Top layout of STM32 development board

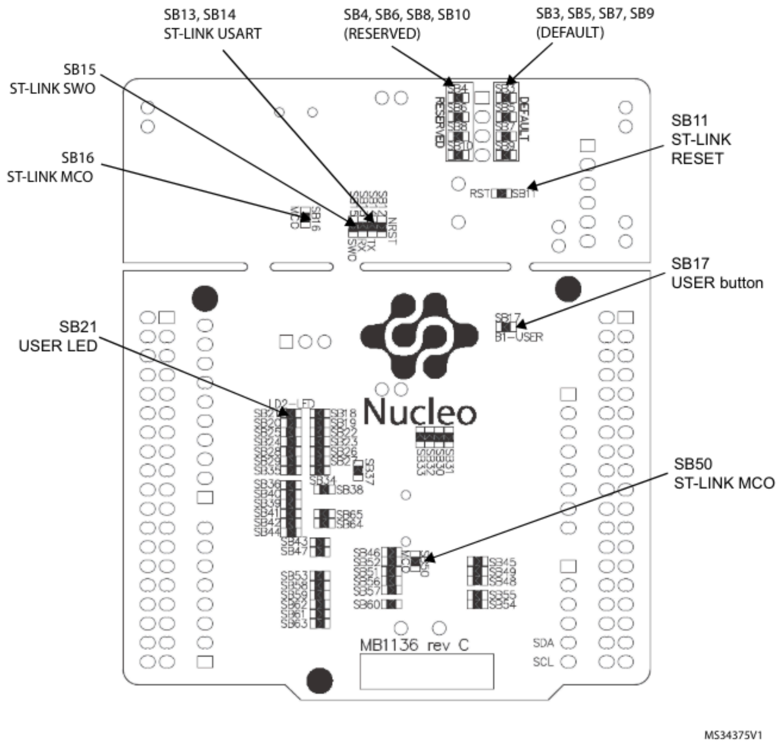


Figure 2: Bottom layout of STM32 development board

VI. STM32F446RE Pins

The board consists of 64 pins out of which 38 pins are used and depicted in the following diagram. The relevant ports, pins and registers for today's lab are described later.

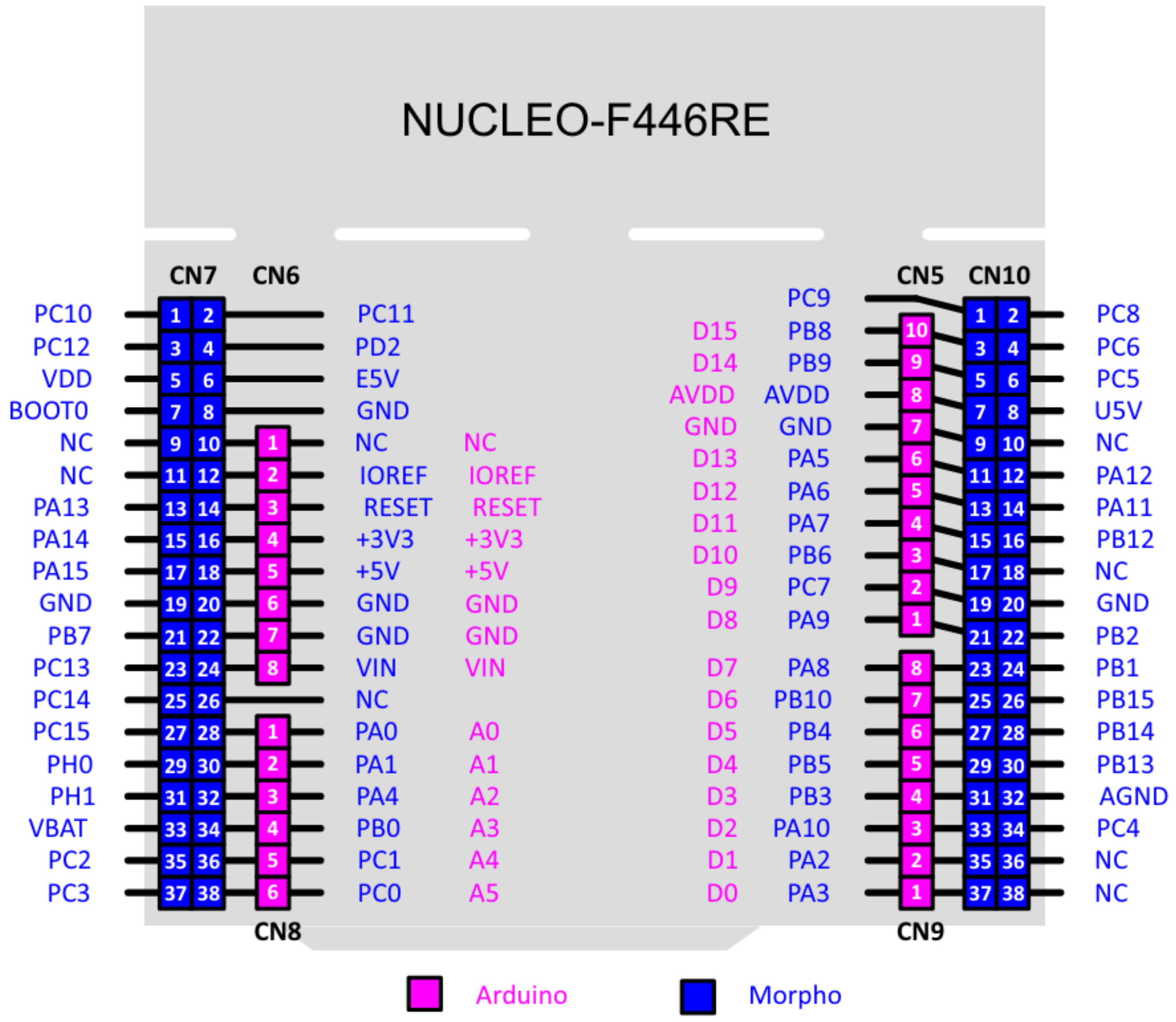


Figure 3: Pin diagram of STM32 development board

Power: 5 watt through the CN1 ST-Link USB mini B connector demonstrated in Fig. 1.

Bus: APB1 and APB2 peripheral bus, AHB port bus.

Ports: 8 GPIO ports (A-H) each with 16 pins. We will be using the ones depicted in Fig. 3 annotated as PortNamePinNumber. For example, port A pin 0 is annotated as PA0. These GPIO pins can be used for all types of signals.

Ground and Voltage Source: one 3.3V, one 5V and six GND.

Registers:

1. **32-bit AHB1 Enable register** to enable the ports. The last 8 bits are used to enable or disable the 8 ports discussed previously.

   0: Port disabled

   1: Port enabled

   For example, to enable port A, set 1 in AHB1ENR register bit 0.

7	6	5	4	3	2	1	0
GPIOH EN	GPIOG EN	GPIOF EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
rw	rw	rw	rw	rw	rw	rw	rw

left shift

Figure 4: Bits 7 - 0 of AHB1ENR register

$1 << 2$

2. Each GPIO Port consists of a **32-bit MODER register** to configure the mode of the ports' pins. Each mode uses 2 bits for configuration. So, to configure 16 pins of a port each taking 2 bits for configuration, the 32-bit MODER register is used.

   00: Input (reset state)

   01: General purpose output mode

   10: Alternate function mode

   11: Analog mode(for manual control by pressing the BLUE button)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 5: 32-bit GPIOx_MODER register (x = A, … H)

Bit 1-0 to configure pin 1, bit 3-2 for pin 2 and so on. For example, to configure port A pin 1 in input mode, set 00 in GPIOA MODER register bit 1-0.

3. Each GPIO port includes a **32-bit OTYPER register** which defines the output type of the pins. Bit 15-0 is used to configure the output type of 16 pins.

   [✓ 0: Output push-pull (reset state)]

   1: Output open-drain (won't use)

   We will use the default type which is the push-pull for this lab.

4. A **32-bit GPIOx_OSPEEDR register** is used to set the output speed of each ports' pins.

   00: Low speed

   01: Medium speed

   10: Fast speed

   11: High speed ✓ Use most of the time

   We will use high speed (11) for this lab.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OSPEEDR7 [1:0]		OSPEEDR6 [1:0]		OSPEEDR5 [1:0]		OSPEEDR4 [1:0]		OSPEEDR3[1:0]		OSPEEDR2 [1:0]		OSPEEDR1 [1:0]		OSPEEDR0 1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Figure 6: 32-bit GPIOx_OSPEEDR register (x = A, … H)

5. Each port has one **32-bit GPIOx_IDR register (read only)**, input data register where bits 15-0 are used to read the digital inputs from 16 pins.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Figure 7: 32-bit GPIOx_IDR register (x = A, … H) ambiguous reg

6. Similar to the IDR register, each port has one **32-bit GPIOx_ODR register (read and write)**, output data register where bits 15-0 are used to read and write the digital outputs of 16 pins. But we won't be writing directly in the ODR register. We will be using the GPIOx_BSRR register for atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..H).

So, we will not use this as this pin is ambiguous.

Insted, we will use this pin

7. Each port has a **32-bit GPIOx_BSRR register**, bit set/reset register for digital write. We can only write in this register, to read we use the ODR register.

   Bit 31:16 are reset bits where,

   0: No action

   1: Resets corresponding ODRx pin

   Bit 15:0 are set bits where,

   0: No action

   1: Sets corresponding ODRx pin

Reset Pins (0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Figure 8: 32-bit GPIOx_BSRR register (x = A, … H)

set pins (1)

1<< 5 << 16 → easy to remember for reset

VII. Experiment: Controlling a LED using the user button on the board.

1. **Description the components:**

a. Breadboard

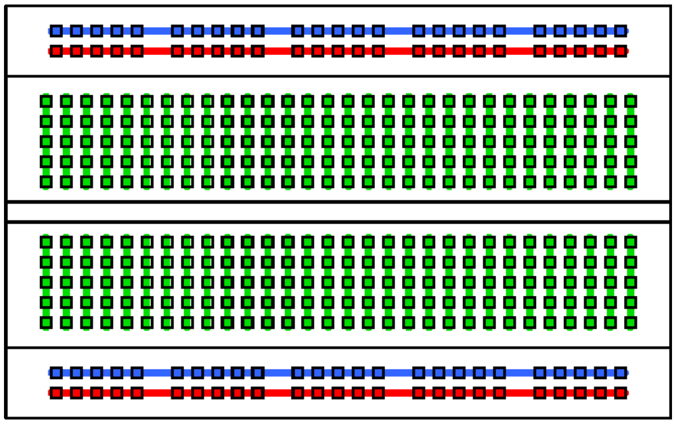


Figure 9: Breadboard internal connection

b. Jumper wires

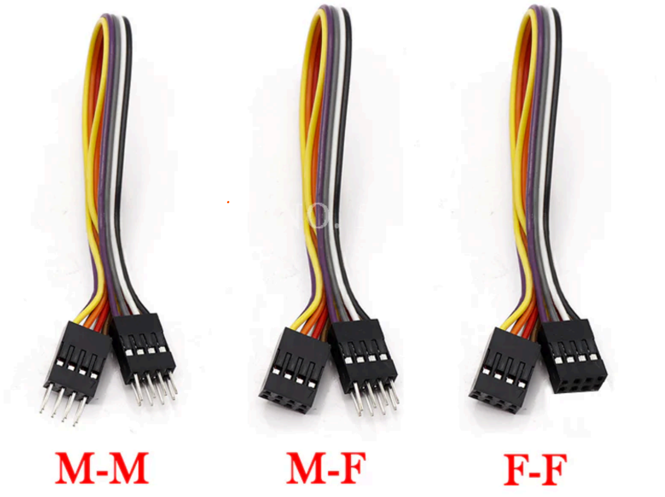
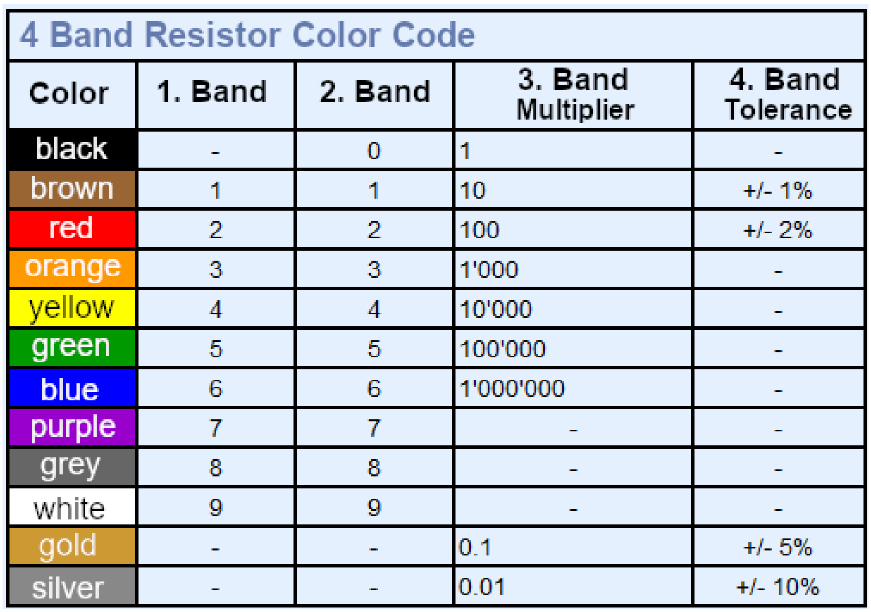


Figure 10: 3 types of jumper wires

c. Resistors



4 Band Resistor Color Code				
Color	1. Band	2. Band	3. Band Multiplier	4. Band Tolerance
black	-	0	1	-
brown	1	1	10	+/- 1%
red	2	2	100	+/- 2%
orange	3	3	1'000	-
yellow	4	4	10'000	-
green	5	5	100'000	-
blue	6	6	1'000'000	-
purple	7	7	-	-
grey	8	8	-	-
white	9	9	-	-
gold	-	-	0.1	+/- 5%
silver	-	-	0.01	+/- 10%

Table 1: 4 band resistor color codes

d. LED

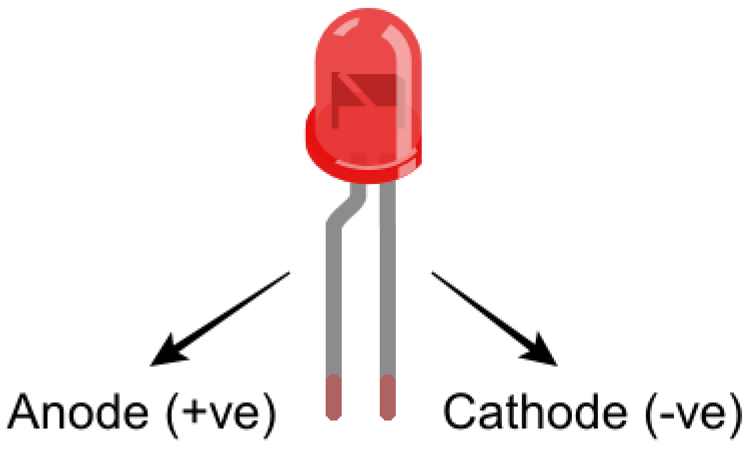


Figure 11: LED anode (longer leg) - cathode (shorter leg) identification

2. **Setting up the circuit:**
    a. Insert the LED into the breadboard. Make sure to note the longer leg (anode) and the shorter leg (cathode).
    b. Connect one end of the resistor to the anode (longer leg) of the LED.
    c. Connect the other end of the resistor to any GPIO pins on the STM32 board. e.g., Port A pin 1 which is denoted as PA1 or A1 in Fig. 3. and A1 on the STM32 board.
    d. Connect the cathode (shorter leg) of the LED to any of the ground (GND) pins of the board.
    e. Connect the STM32 board using the USB mini B connector with the computer.

3. **Circuit diagram:**

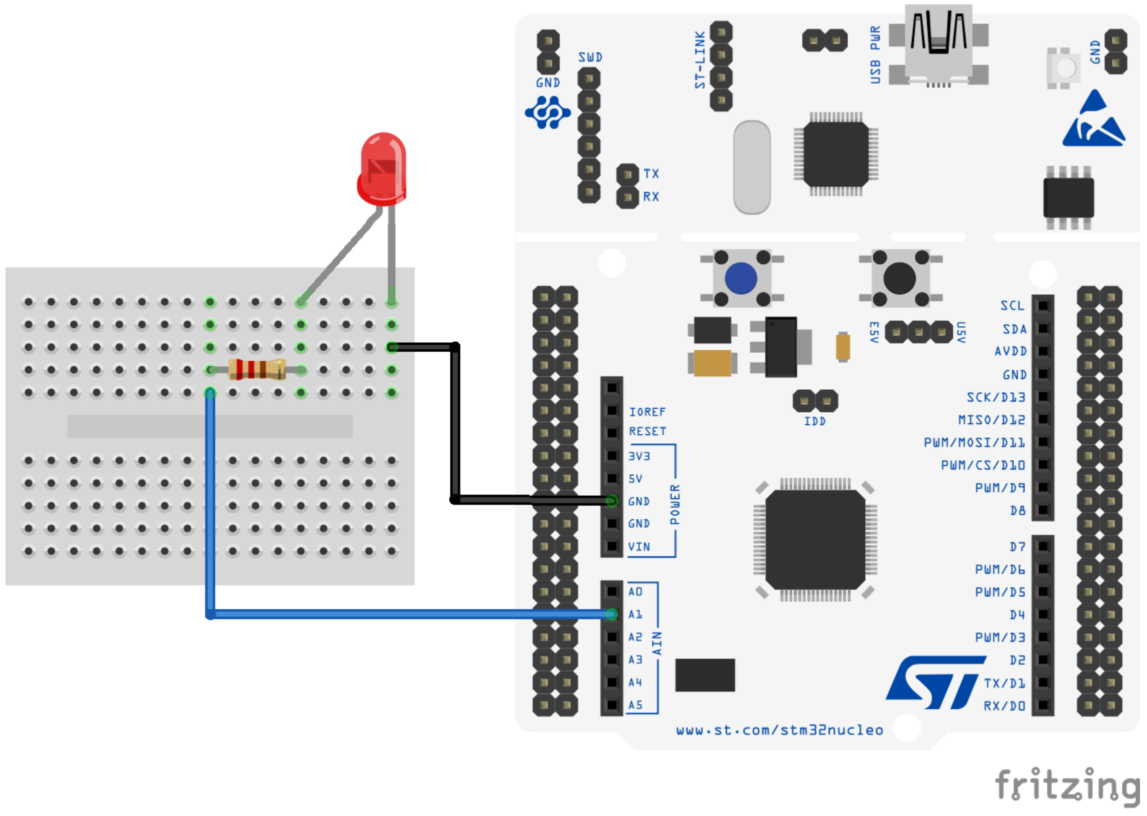


Figure 12: Circuit diagram of a LED connected to PA1/A1 pin of the STM32 board

4. **Code:** Open the Keil µVision5 IDE and follow the steps.

   a. From the project tab in the menu bar, open a new project.

   b. In the select device for target window, select STM32F446RETx under STMicroelectronics and click ok.

   c. In the manage run-time environment window, select Core under CMSIS and Startup under Devices and click ok.

   d. To create files, right click on the Source Group 1 tab under Target 1 and select Add new item. A pop up window will appear for file type. Create the following files with the mentioned type and name. Paste the following codes in their respective files.

   e. File name: gpio, file type: header file (.h)

```
#ifndef __GPIO_H
#define __GPIO_H
#include "stdint.h"
void init_gpio(void);
void write_pin(int);
int read_pin(void);
#endif
```

   → main file

   f. File name: test1, file type: C file (.c)

```
#include "gpio.h"
int main()
{
      init_gpio();
      while(1)
      {
            write_pin(read_pin());
            for(int i = 0; i<100; i++);
      }

}
```

   g. File name: gpio, file type: C file (.c)

```
#include "gpio.h"
#include "stm32f446xx.h"

void init_gpio(void)
{
      //enable PORT A for LED output
      RCC->AHB1ENR |= 1<<0;  1 << 1
                                       Input/output
      //Pin 1 output mode and speed for LED 1
      GPIOA->MODER |= 1<<2;    0<<4
      GPIOA-> OSPEEDR |= 3<<2;       Doing OR and 2 bit left shift
                              3<<4
          speed control                OR

      //enable PORT C for button input
      RCC->AHB1ENR |= 1<<2;

      //Pin 13 input mode for the mounted button on PORT C Pin
13
      GPIOC->MODER &= ~(3<<26);

}
//reading the input from button, default = 1, pressed = 0
int read_pin(void){

      int value = 1;
      if (GPIOC->IDR & 1<<13){
      value = 0;
      }
      return value;
}

//turn led on based on button
void write_pin(int value)
{
      if (value == 1){
      //if pressed, LED 1 on
      GPIOA->BSRR |= 1<<1; 1<<2
      }

      if (value == 0){
      //if not pressed, LED 1 off
      GPIOA->BSRR |= 1<<1<<16;  1<<2<<16
      }
   }
}
```

h. After creating all the files, save all files > batch build > download. The LED should turn on the press of the user button.

VIII. Lab evaluation: Connect the LED to Port B Pin 0 instead of Port A Pin 1 and modify the code accordingly so that the LED turns on after pressing the user button. [5]

IX. References:

1. https://www.st.com/en/microcontrollers-microprocessors/stm32f446re.html
2. https://www.electronicsplanet.ch/en/electronics/resistor-color-code/resistor-color-code.php
3. https://pmdway.com/products/2-54mm-0-1-pitch-dual-row-jumper-cables-various-types-5-pack
4. http://designbuildcode.weebly.com/breadboard-circuits.html