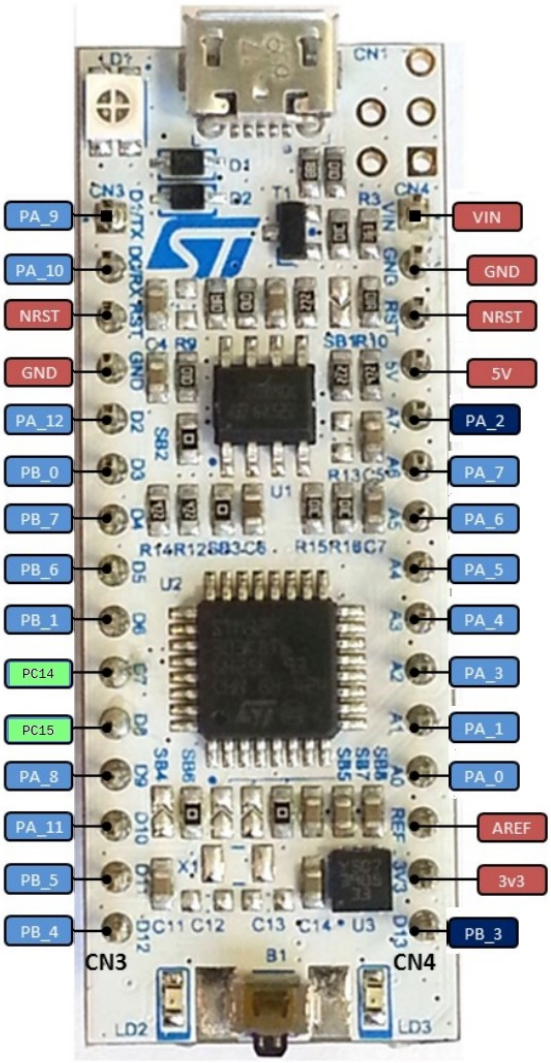
Lab 1: Traffic lights.

Introduction

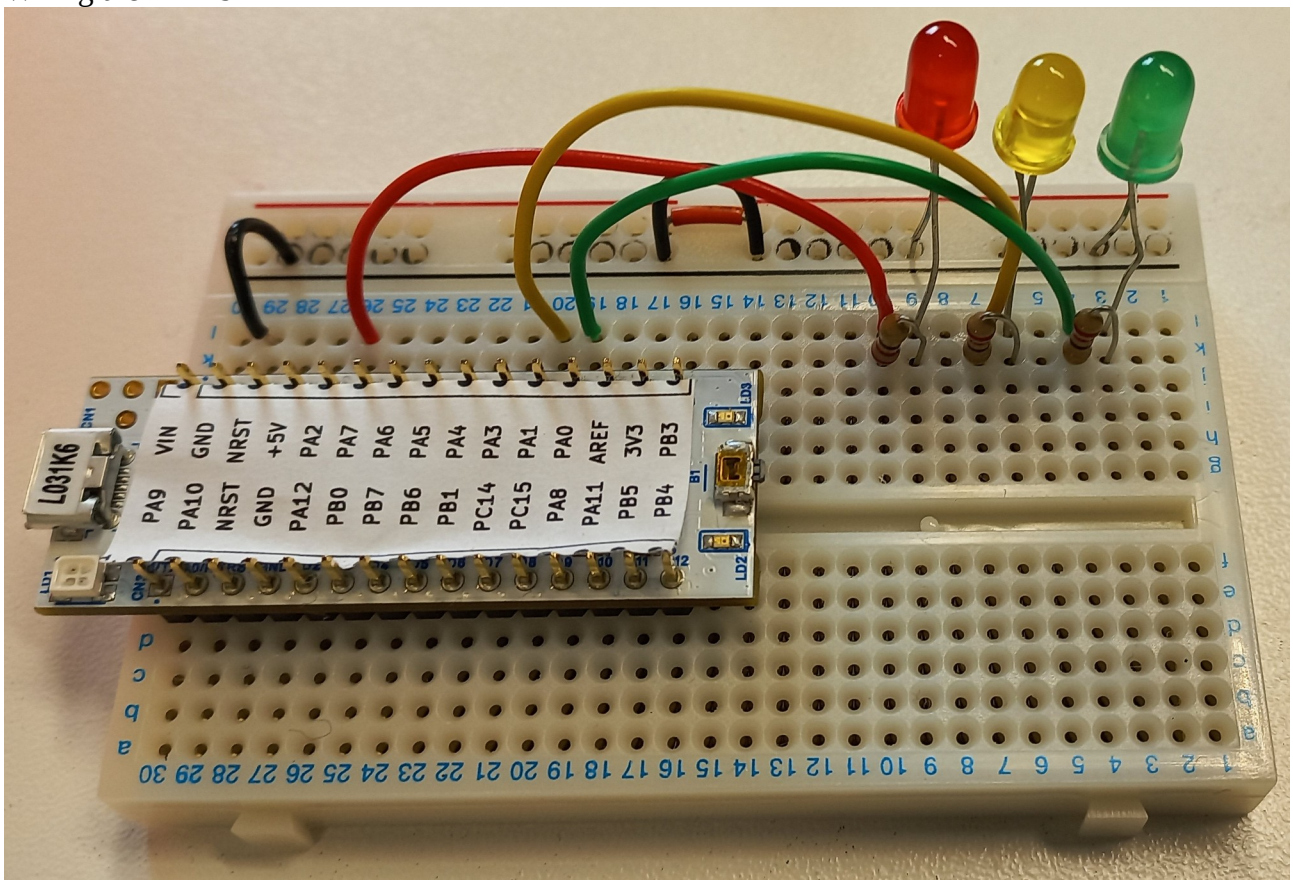
In this lab you will learn about digital outputs and inputs. You will learn to set (to 1) and clear (to 0) individual digital outputs on a parallel input/output port. You will also learn to examine individual inputs and so control the flow of execution of a program.

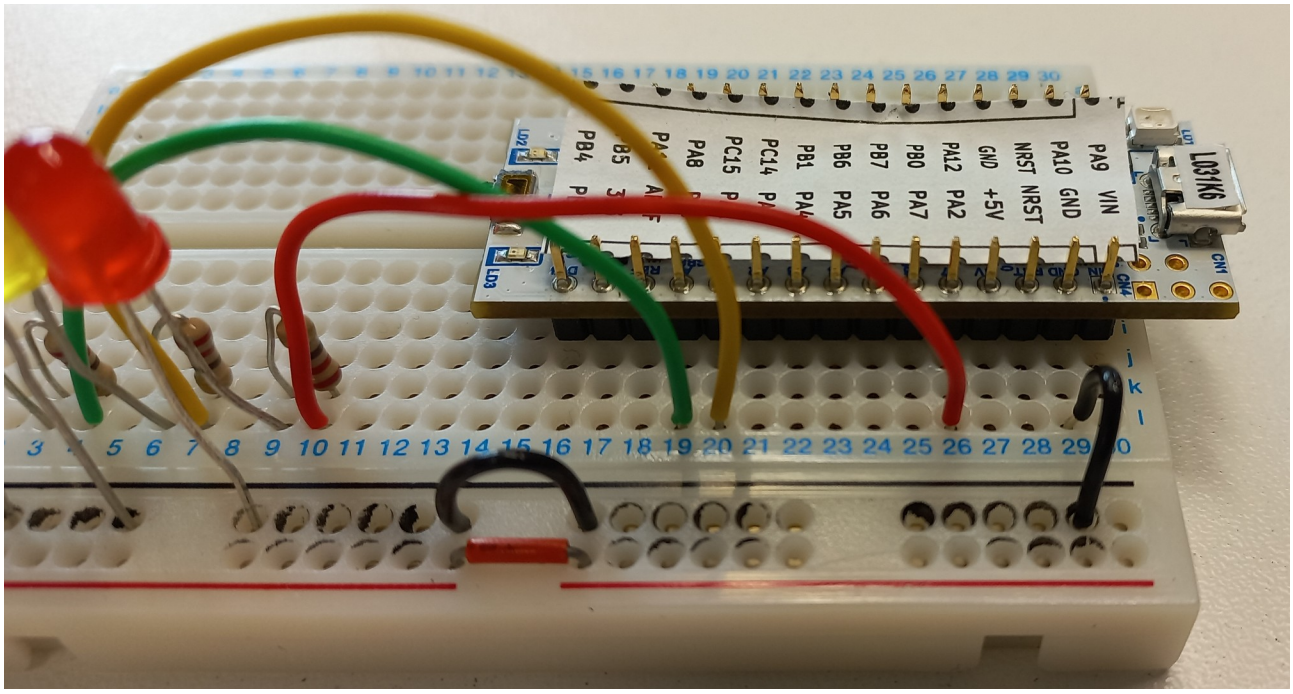


The STM32L031 Nucleo is shown above. The port pins are labelled in a manner that reflects their connection to the internal I/O ports on the chip. This is different to the labels on the PCB. For example, the pin labelled PA3 refers bit 3 to General Purpose Input/Output Port A (we abbreviate this to GPIOA). Bit numbering starts at 0. There are three ports available on the pins of this chip: GPIOA, GPIOB and GPIOC. Not all bits of all ports are available (there can be up to 16 bits per port) and they may seem to be arranged in a haphazard way. The ordering of pins is as a result of routing PCB tracks and connections within the STM32L031 chip itself.

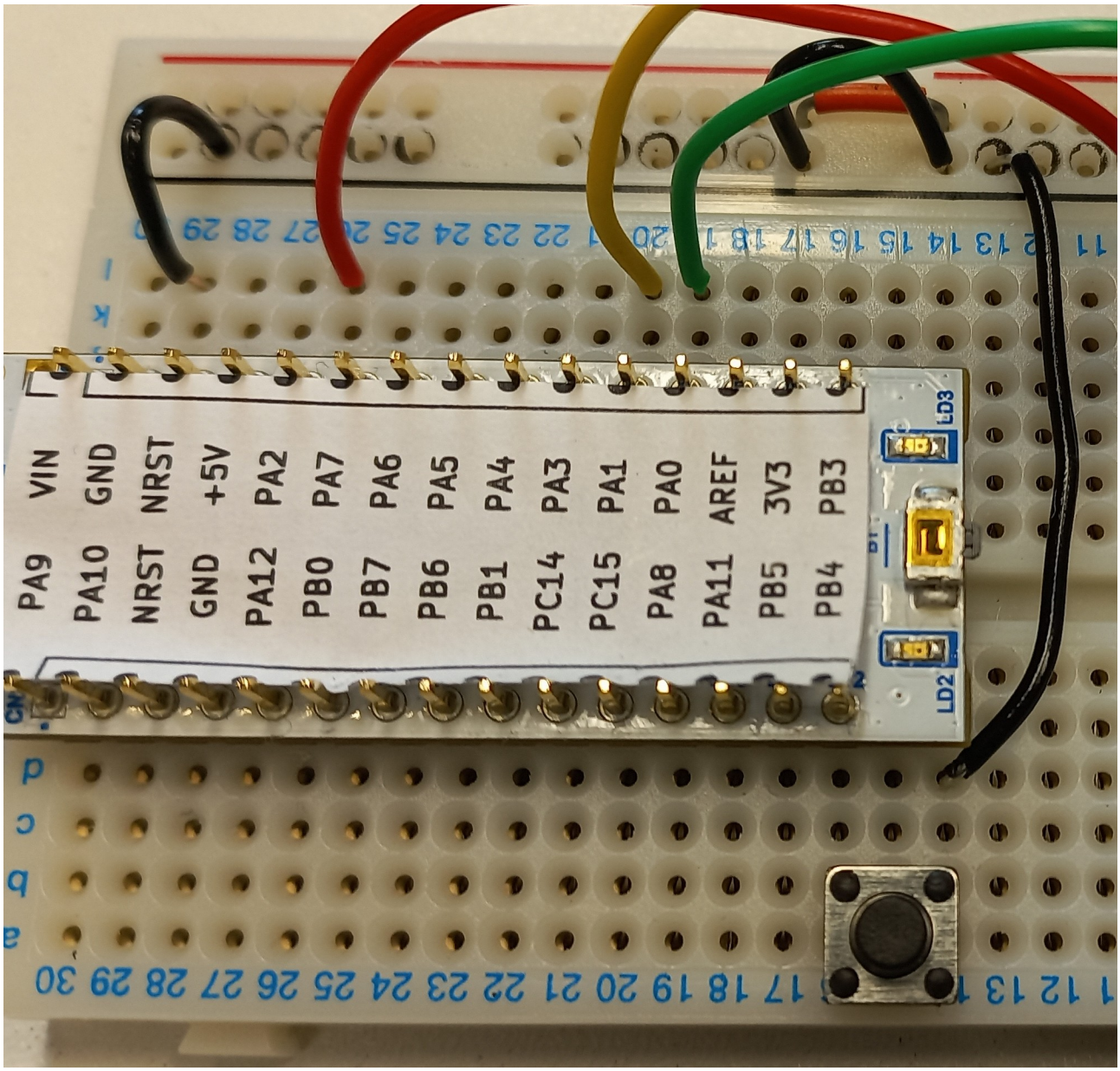
Procedure

Wiring the LED’s





Adding the button



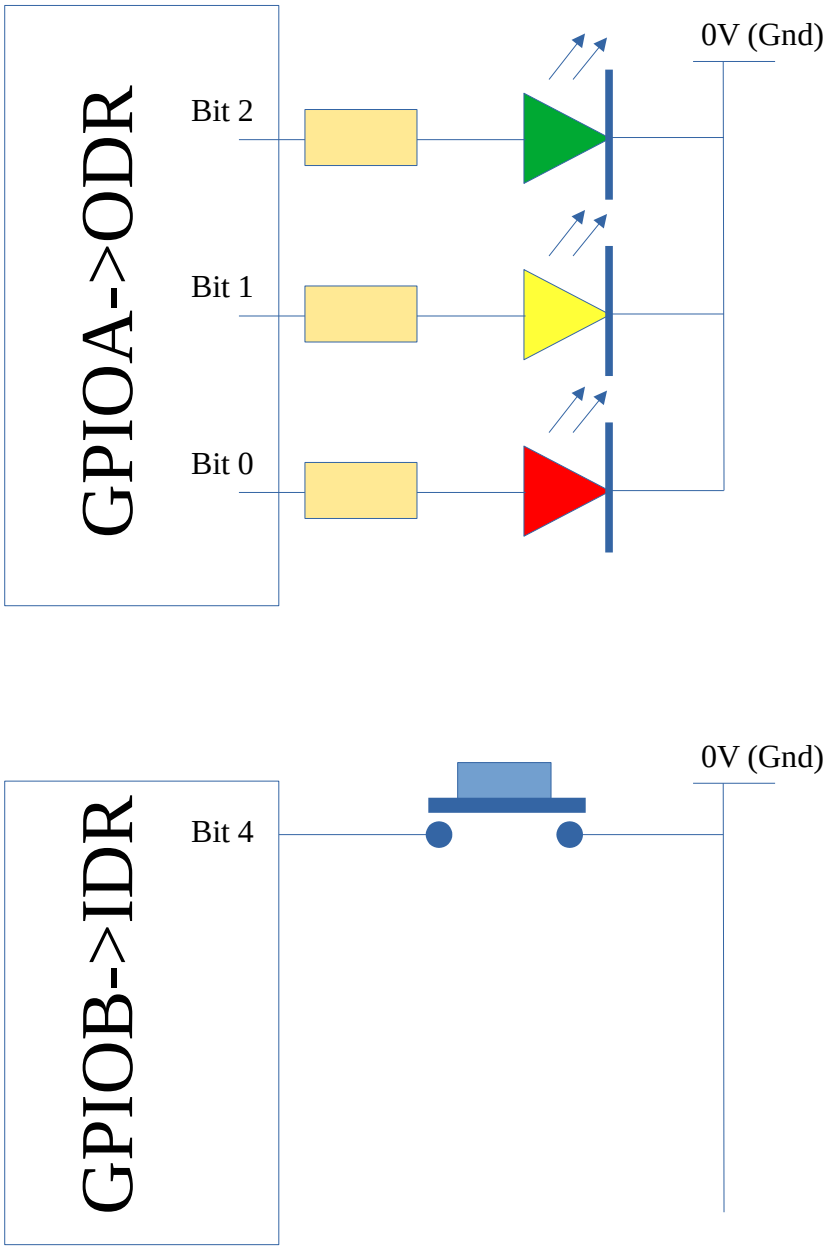
There is a video explaining how a breadboard works in the “Getting Started” section.

Begin by plugging the STM32L031 Nucleo board into the breadboard as shown. Please note the orientation:
The USB socket is on the left.
The left-most pin is in column 30.

If you plug it in incorrectly please raise your hand and someone will help you correct it. It is easy to damage the pins on the device by pulling it out of the breadboard.

Three LED’s are connected as shown. The shorter leg goes into a hole along the black rail of the breadboard.

Connect the remaining components and wires and please use the colour code shown in the picture as it simplifies debugging.



Electrical view of the circuit

Procedure (code)

The starting point for the code is shown below.

Execute this code on the stm32L031. You should note that the LED's count up in 3 bit binary.

Add functions to do the following:

`void RedOn(void);` // turns on the red LED without changing the others

`void RedOff(void);` // turns off the red LED without changing the others

`void YellowOn(void);` // turns on the yellow LED without changing the others

`void YellowOff(void);` // turns off the yellow LED without changing the others

`void GreenOn(void);` // turns on the green LED without changing the others

`void GreenOff(void);` // turns off the green LED without changing the others

`int ButtonPressed(void);` // returns if the button is pressed. 0 otherwise

```c
#include <stdint.h>
#include <stm32l031xx.h>

void pinMode(GPIO_TypeDef *Port, uint32_t BitNumber, uint32_t Mode);
void enablePullUp(GPIO_TypeDef *Port,uint32_t BitNumber);
void delay(volatile uint32_t dly);
int main()
{
	uint32_t count = 0;
	RCC->IOPENR |= (1 << 0) + (1 << 1);
	pinMode(GPIOA,0,1); // Make GPIOA bit 0 an output
	pinMode(GPIOA,1,1); // Make GPIOA bit 1 an output
	pinMode(GPIOA,2,1); // Make GPIOA bit 2 an output

	pinMode(GPIOB,3,1); // Make GPIOB bit 3 an output
	pinMode(GPIOB,4,0); // Make GPIOB bit 4 an input

	enablePullUp(GPIOB,4); // Turn on pull-up resistor for bit 4
	while(1)
	{
	GPIOA->ODR = count;
	count++;
	delay(200000);
	}
}
void enablePullUp(GPIO_TypeDef *Port, uint32_t BitNumber)
{
	Port->PUPDR = Port->PUPDR &~(3u << BitNumber*2); // clear pull-up resistor
bits
	Port->PUPDR = Port->PUPDR | (1u << BitNumber*2); // set pull-up bit
}
void pinMode(GPIO_TypeDef *Port, uint32_t BitNumber, uint32_t Mode)
{
	/*
	*/
	uint32_t mode_value = Port->MODER;
	Mode = Mode << (2 * BitNumber);
	mode_value = mode_value & ~(3u << (BitNumber * 2));
	mode_value = mode_value | Mode;
	Port->MODER = mode_value;
}
void delay(volatile uint32_t dly)
{
	while(dly--);

}
```

Modify your code so that it operates as a pedestrian crossing:

Default : Green on.

Button pressed (momentary).

Delay

Yellow On, Green Off

Delay

Red On, Yellow Off

Delay

Green on.

(repeat)