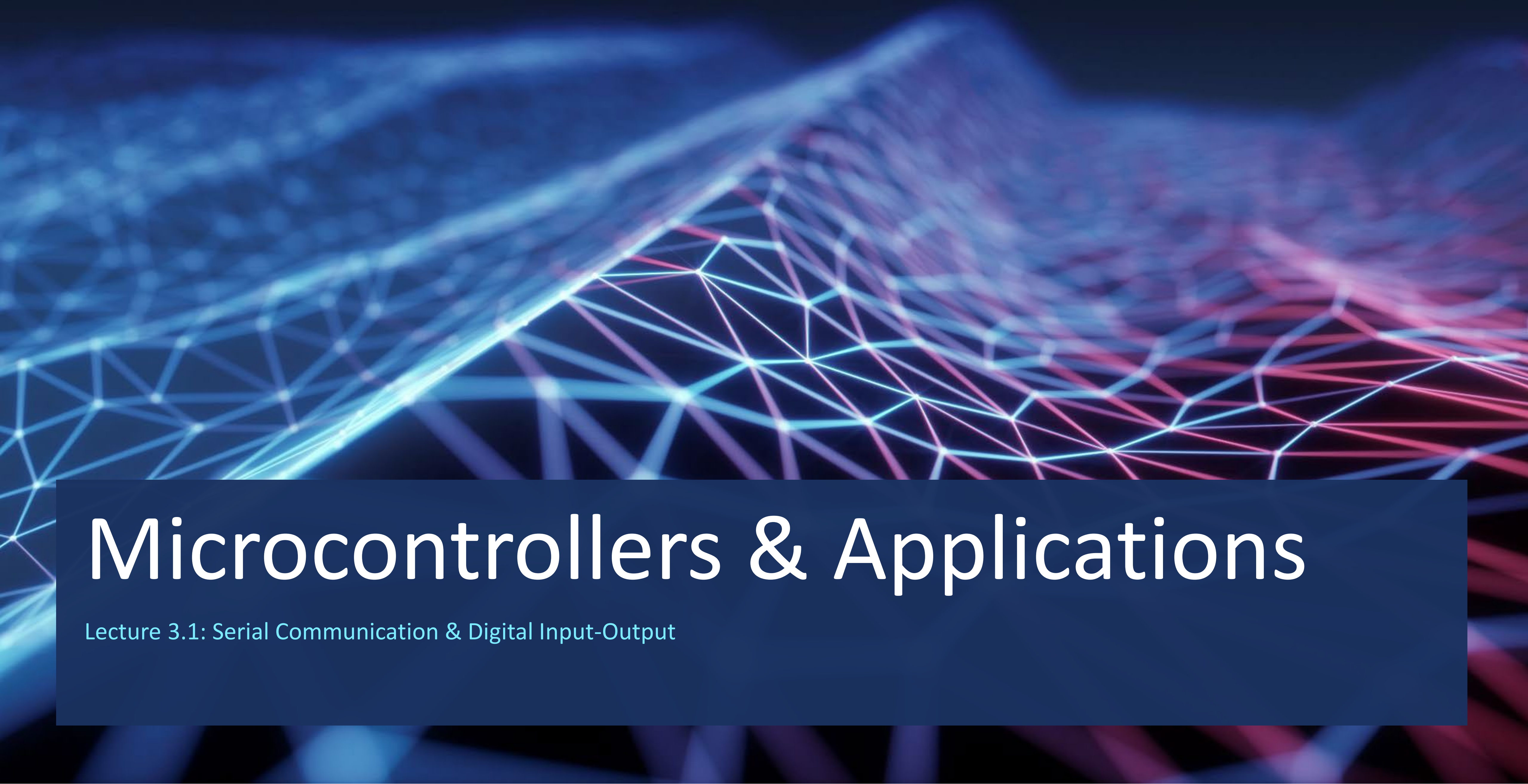


Microcontrollers & Applications

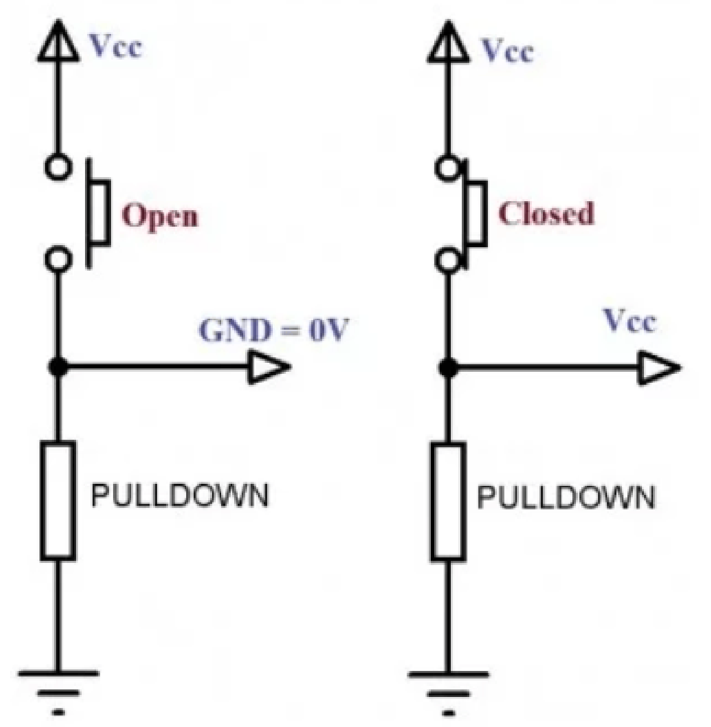
Lecture 3.1: Serial Communication & Digital Input-Output

Serial Communication

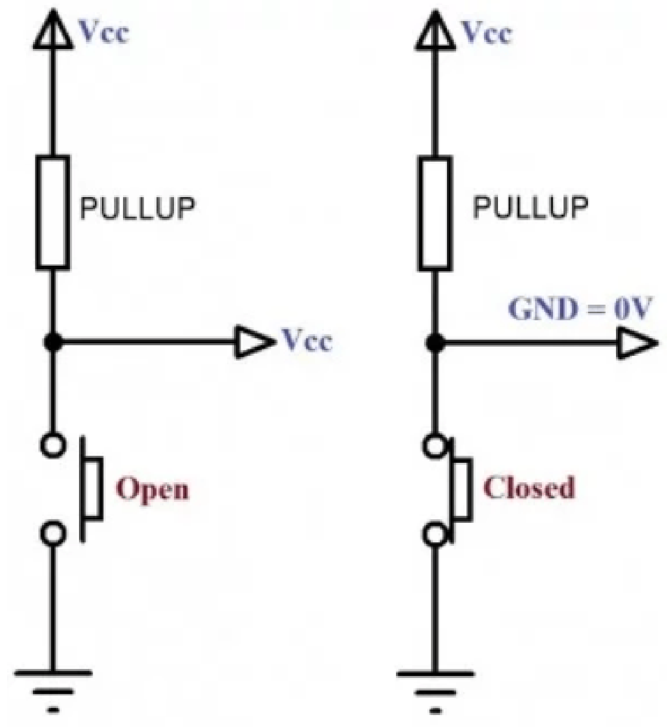
- /* Serial communication library is included by default, not necesary to declare. If different than hardware pins will be used, SoftwareSerial.h should be imported indeed */
- Serial.begin(9600)
  - Starts the serial communication with the speed of 9600 bps
- Serial.end()
  - Finalizes the serial communication
- Serial.print(value)
  - Sends the value via serial port
- Serial.println(value)
- Sends the value and the next line character (i.e., value+'\n') via serial port
- Serial.available()
  - true if there are some values received via serial port
  - false if there is no received value via serial port
- Serial.read()
  - Reads the incoming byte value from the serial port
- Serial.readString()
  - Reads the received string from the serial port

- from machine import UART
  - Import the required library
- uart = UART(id=0,baudrate=9600,bits=8,parity=None,stop=1)
- uart = UART(0, 9600)
  - Starts the serial communication via GPIO0 pin with the speed of 9600 bps
- uart.read(n)
  - Reads n characters from the serial port
- uart.read()
  - Reads all available characters from the serial port
- uart.readline()
  - Reads a line of characters (until reaching the '\n' character) from the serial port
- uart.readinto(buf)
  - Reads and stores into the given buffer
- uart.write(value)
  - Sends the value via serial port

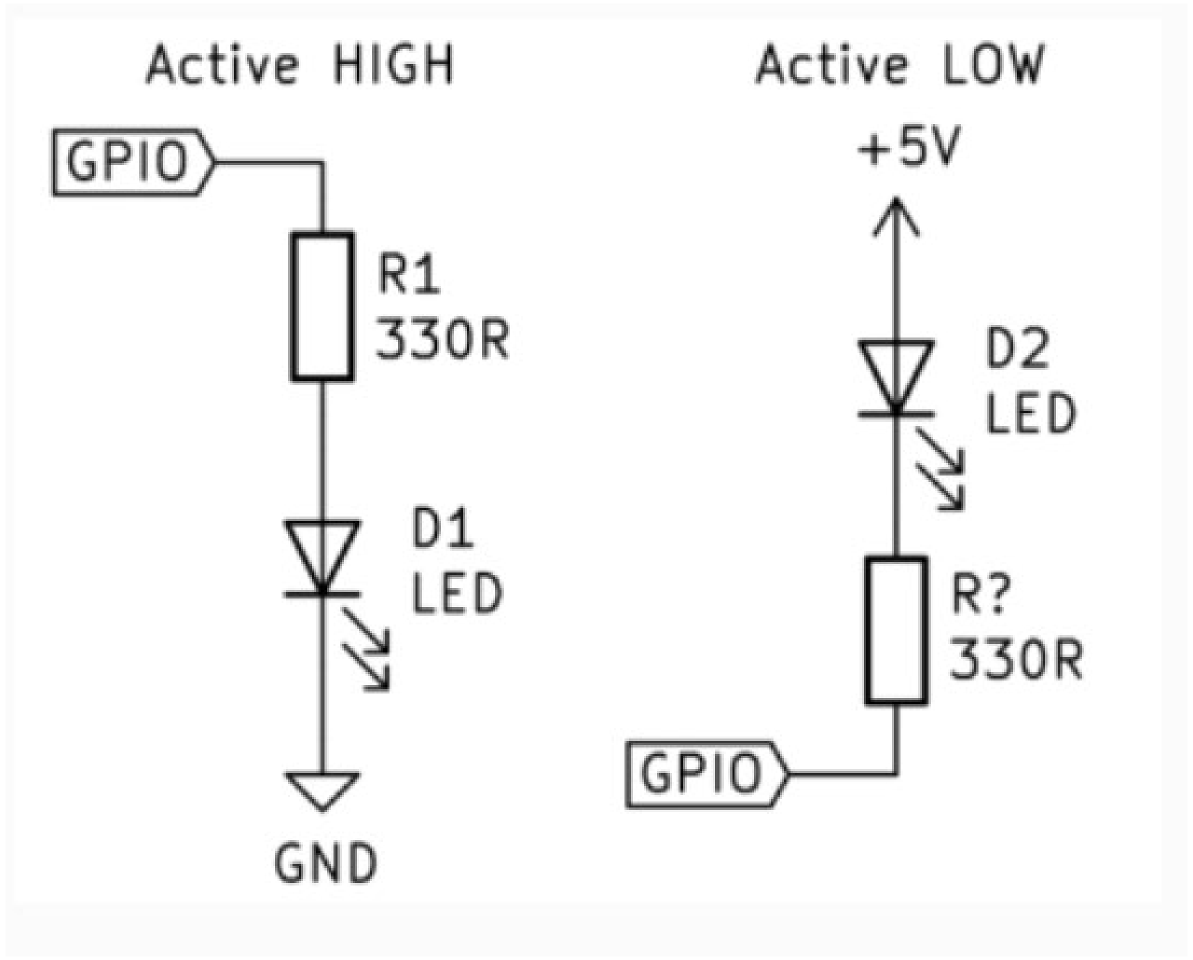
Digital Inputs



- Active HIGH (PULLDOWN):
  - When you read HIGH (+5V) from the digital input it is activated (button pressed here); otherwise, it will be in passive. The pin must be connected to the GND via a pulldown resistor and the other terminal must be directly connected to the +5V.
- Active LOW (PULLUP):
  - When you read LOW (GND = 0 V) from the digital input it is activated (button pressed here); otherwise, it will be in passive. The pin must be connected to the +5V via a pullup resistor and the other terminal must be directly connected to the GND.
- If the input device requires extra hardware components, you must use them in series.

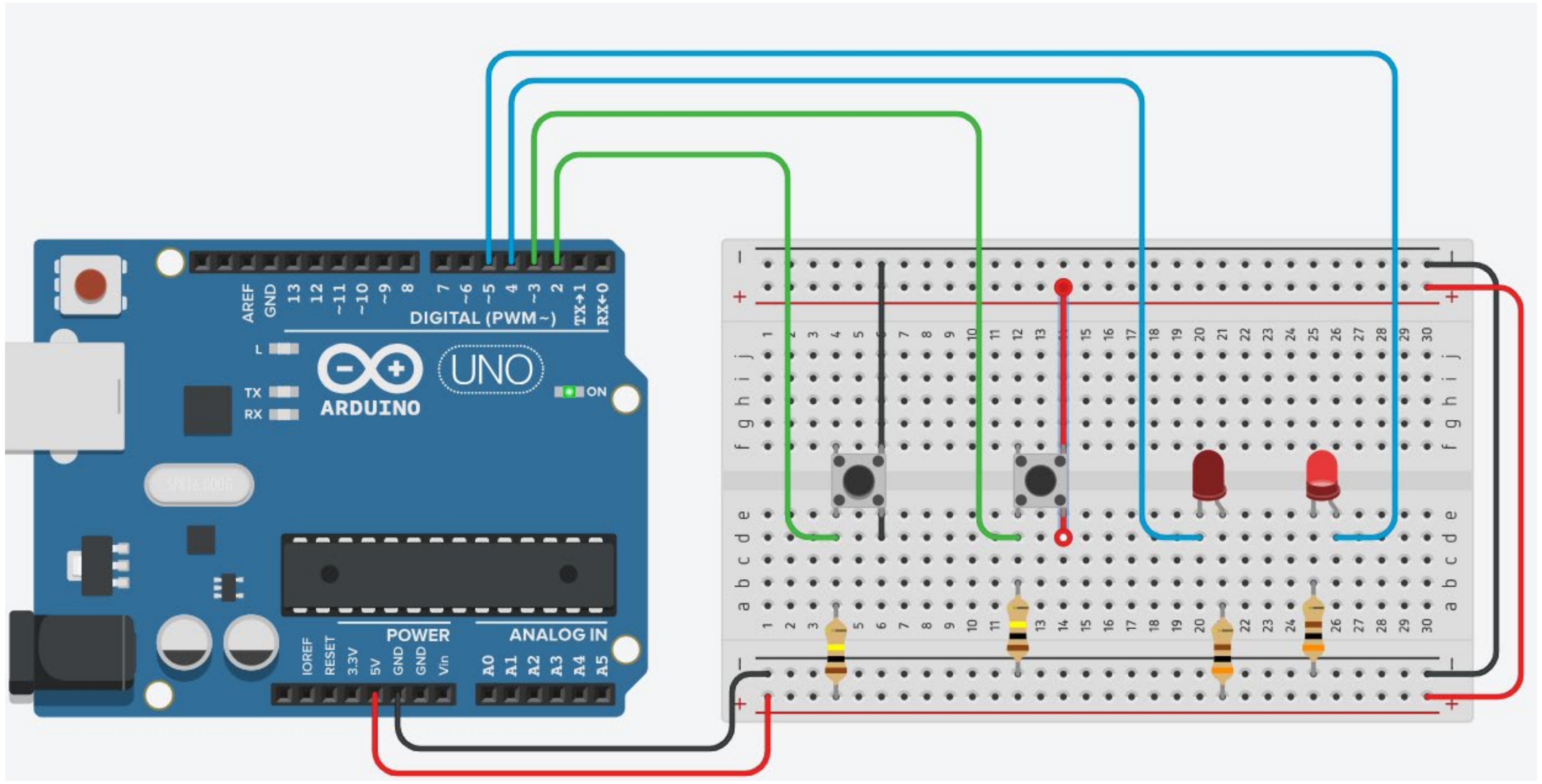


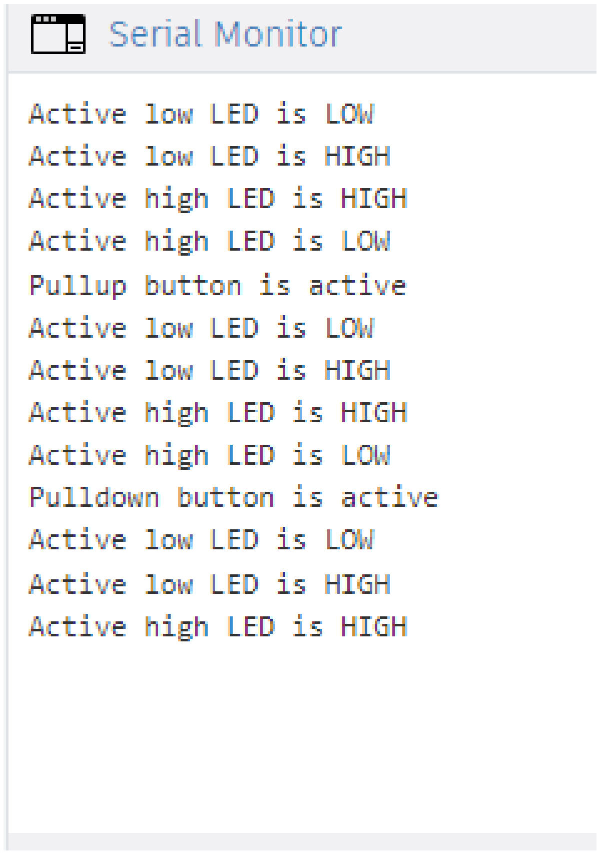
Digital Outputs



- Active HIGH:
  - When you send HIGH (+5V) to the digital output it activates (lightens here); otherwise, it will be in passive. The other terminal must be connected to the GND.
- Active LOW:
  - When you send LOW (GND = 0 V) to the digital output it activates (lightens here); otherwise, it will be in passive. The other terminal must be connected to the +5V.
- If the output device requires current limitation, you must connect a resistor in series.

Arduino Example





```
Active low LED is LOW
Active low LED is HIGH
Active high LED is HIGH
Active high LED is LOW
Pullup button is active
Active low LED is LOW
Active low LED is HIGH
Active high LED is HIGH
Active high LED is LOW
Pulldown button is active
Active low LED is LOW
Active low LED is HIGH
Active high LED is HIGH
```

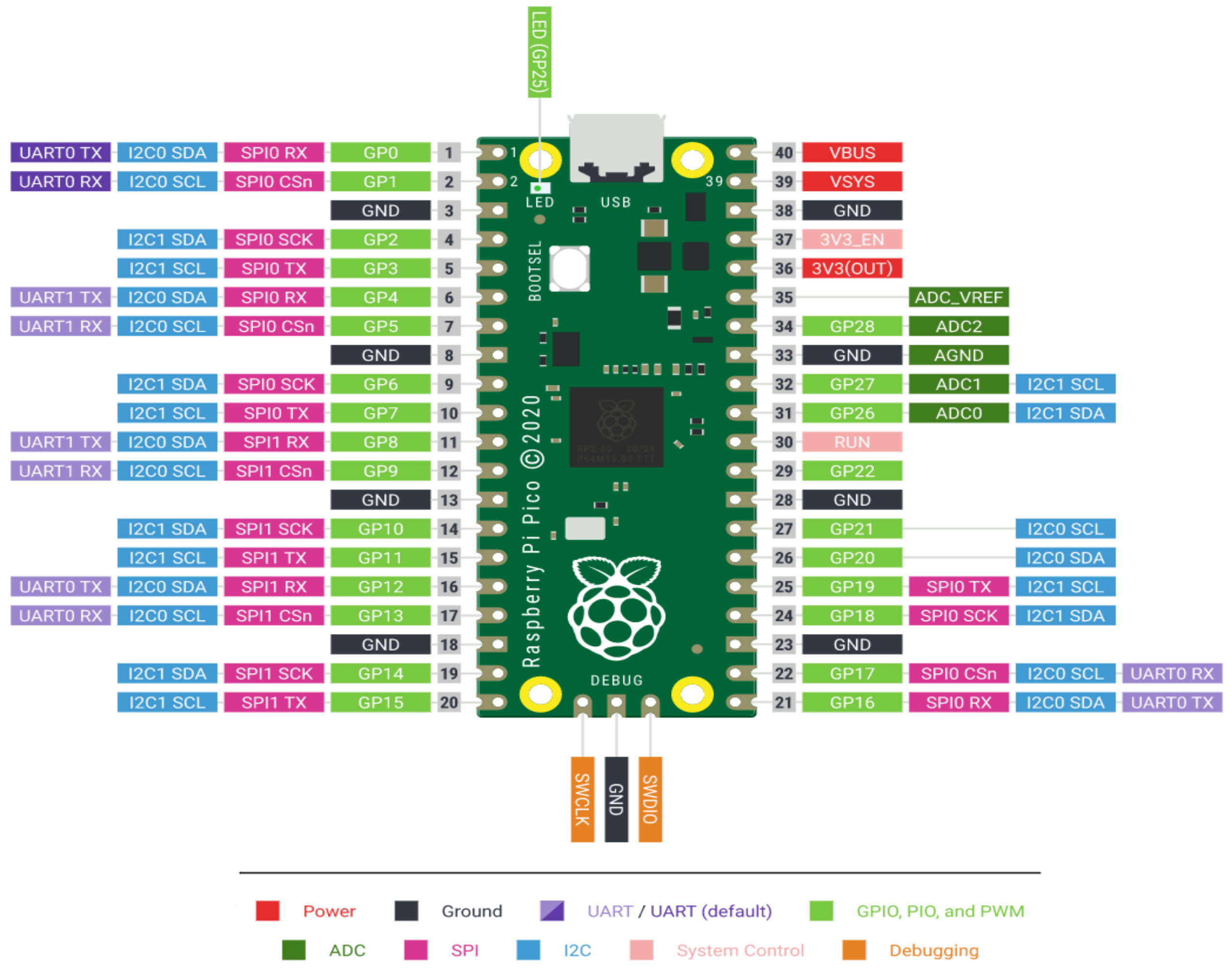
Code of Arduino Example

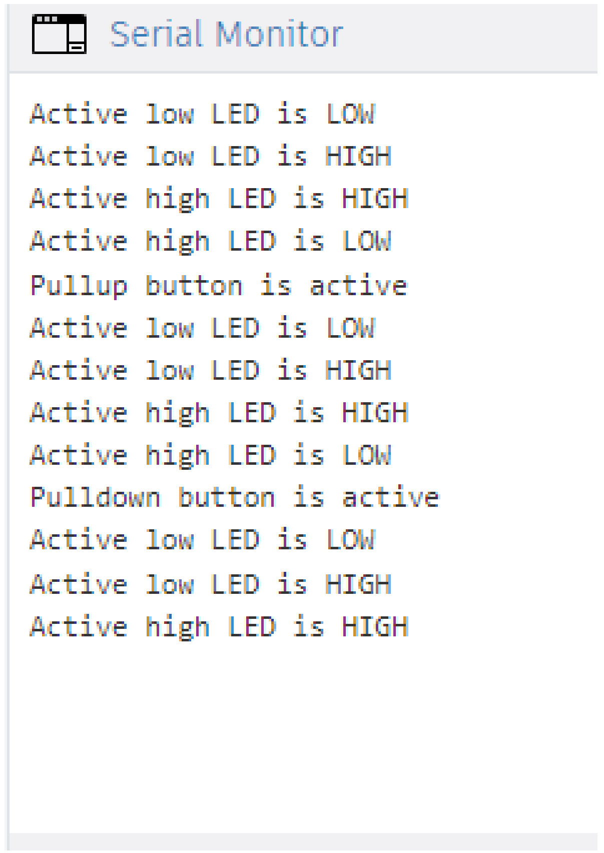
```
#define BUTTON_PULLUP   2
#define BUTTON_PULLDOWN 3
#define LED_ACTIVE_LOW  4
#define LED_ACTIVE_HIGH 5

void setup() {
  pinMode(BUTTON_PULLUP, INPUT);
  pinMode(BUTTON_PULLDOWN, INPUT);
  pinMode(LED_ACTIVE_LOW, OUTPUT);
  digitalWrite(LED_ACTIVE_LOW, HIGH);
  pinMode(LED_ACTIVE_HIGH, OUTPUT);
  digitalWrite(LED_ACTIVE_HIGH, LOW);
  Serial.begin(9600);
}
```

```
void loop()
{
  digitalWrite(LED_ACTIVE_LOW, LOW);
  Serial.println("Active low LED is LOW");
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_ACTIVE_LOW, HIGH);
  Serial.println("Active low LED is HIGH");
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_ACTIVE_HIGH, HIGH);
  Serial.println("Active high LED is HIGH");
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_ACTIVE_HIGH, LOW);
  Serial.println("Active high LED is LOW");
  delay(1000); // Wait for 1000 millisecond(s)
```

```
  if (digitalRead(BUTTON_PULLUP) == LOW) {
    Serial.println("Pullup button is active");
    delay(100); // Wait for 100 millisecond(s)
  }
  if (digitalRead(BUTTON_PULLDOWN) == HIGH)
{
    Serial.println("Pulldown button is active");
    delay(100); // Wait for 100 millisecond(s)
  }

}
```

Raspberry Pi Pico Example


LED (GP25)
UART0 TX | I2C0 SDA | SPI0 RX | GP0 | 1
UART0 RX | I2C0 SCL | SPI0 CSn | GP1 | 2
GND | 3
I2C1 SDA | SPI0 SCK | GP2 | 4
I2C1 SCL | SPI0 TX | GP3 | 5
UART1 TX | I2C0 SDA | SPI0 RX | GP4 | 6
UART1 RX | I2C0 SCL | SPI0 CSn | GP5 | 7
GND | 8
I2C1 SDA | SPI0 SCK | GP6 | 9
I2C1 SCL | SPI0 TX | GP7 | 10
UART1 TX | I2C0 SDA | SPI1 RX | GP8 | 11
UART1 RX | I2C0 SCL | SPI1 CSn | GP9 | 12
GND | 13
I2C1 SDA | SPI1 SCK | GP10 | 14
I2C1 SCL | SPI1 TX | GP11 | 15
UART0 TX | I2C0 SDA | SPI1 RX | GP12 | 16
UART0 RX | I2C0 SCL | SPI1 CSn | GP13 | 17
GND | 18
I2C1 SDA | SPI1 SCK | GP14 | 19
I2C1 SCL | SPI1 TX | GP15 | 20
40 | VBUS
39 | VSYS
38 | GND
37 | 3V3_EN
36 | 3V3(OUT)
35 | ADC_VREF
34 | GP28 | ADC2
33 | GND | AGND
32 | GP27 | ADC1 | I2C1 SCL
31 | GP26 | ADC0 | I2C1 SDA
30 | RUN
29 | GP22
28 | GND
27 | GP21 | I2C0 SCL
26 | GP20 | I2C0 SDA
25 | GP19 | SPI0 TX | I2C1 SCL
24 | GP18 | SPI0 SCK | I2C1 SDA
23 | GND
22 | GP17 | SPI0 CSn | I2C0 SCL | UART0 RX
21 | GP16 | SPI0 RX | I2C0 SDA | UART0 TX
LED
USB
BOOTSEL
Raspberry Pi Pico © 2020
DEBUG
SWCLK
GND
SWDIO
Power
Ground
UART / UART (default)
GPIO, PIO, and PWM
ADC
SPI
I2C
System Control
Debugging



```
Active low LED is LOW
Active low LED is HIGH
Active high LED is HIGH
Active high LED is LOW
Pullup button is active
Active low LED is LOW
Active low LED is HIGH
Active high LED is HIGH
Active high LED is LOW
Pulldown button is active
Active low LED is LOW
Active low LED is HIGH
Active high LED is HIGH
```

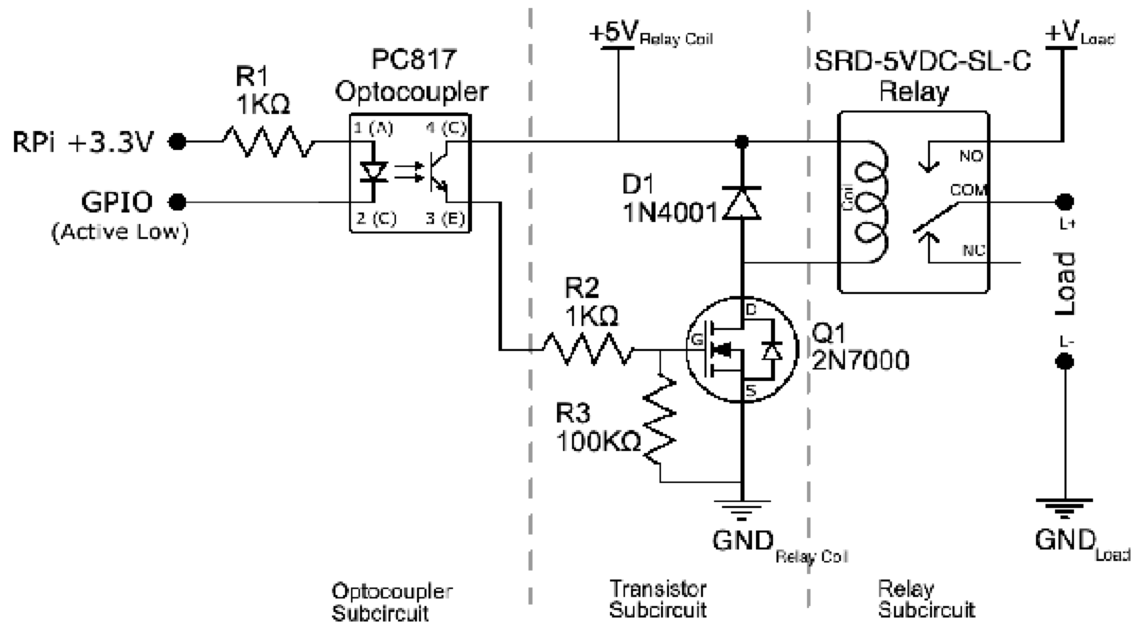
Code of Raspberry Pi Pico Example

```
import time
import board
import digitalio
import UART

BUTTON_PULLUP = digitalio.DigitalInOut(board.GP2)
BUTTON_PULLUP.direction = digitalio.Direction.INPUT
BUTTON_PULLDOWN = digitalio.DigitalInOut(board.GP3)
BUTTON_PULLDOWN.direction = digitalio.Direction.INPUT

LED_ACTIVE_LOW = digitalio.DigitalInOut(board.GP4)
LED_ACTIVE_LOW.direction = digitalio.Direction.OUTPUT
LED_ACTIVE_LOW.value = True
LED_ACTIVE_HIGH = digitalio.DigitalInOut(board.GP5)
LED_ACTIVE_HIGH.direction = digitalio.Direction.OUTPUT
LED_ACTIVE_HIGH.value = False
uart = UART(0, 9600)
```

```
while True:
    LED_ACTIVE_LOW.value = False
    uart.write("Active low LED is LOW\n")
    time.sleep(1)

    LED_ACTIVE_LOW.value = True
    uart.write("Active low LED is HIGH\n")
    time.sleep(1)

    LED_ACTIVE_HIGH.value = True
    uart.write("Active high LED is HIGH\n")
    time.sleep(1)

    LED_ACTIVE_HIGH.value = False
    uart.write("Active high LED is LOW\n")
    time.sleep(1)
```

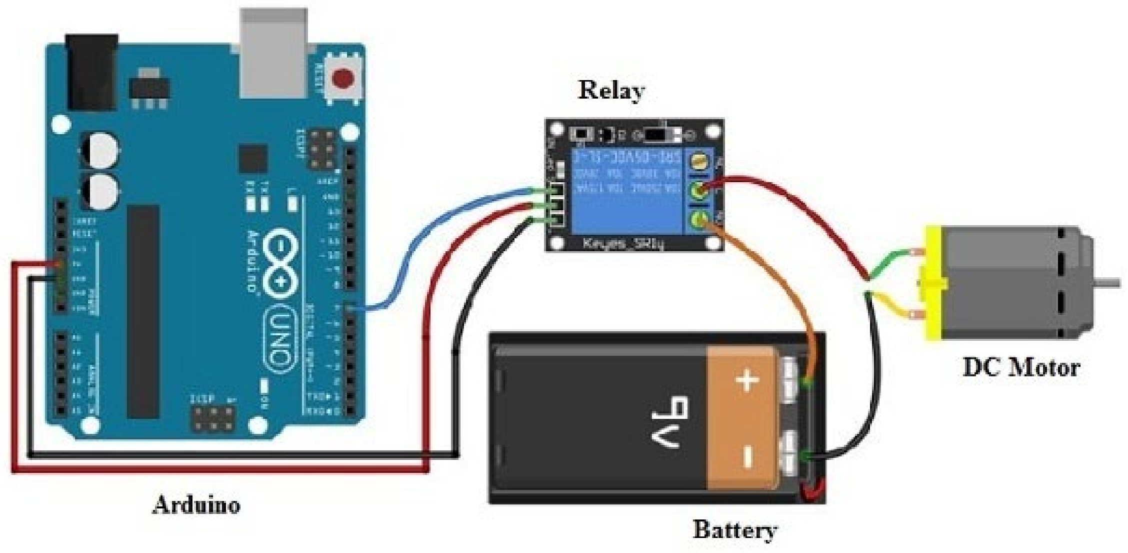
```
    if BUTTON_PULLUP == False:
      uart.write("Pullup button is active\n")
      time.sleep(0.1)
    if BUTTON_PULLDOWN == True:
      uart.write("Pulldown button is active\n")
      time.sleep(0.1)
```

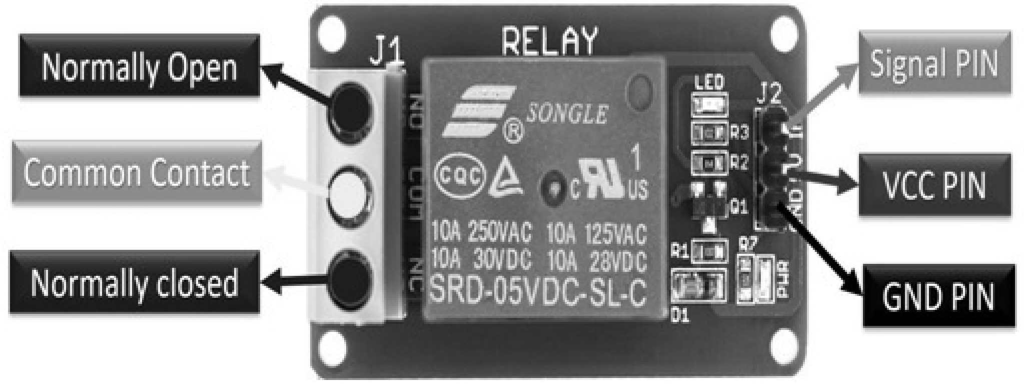
Connecting High-Power Outputs using Relays (1)



- Diode (D_1) is reversely connected to the coil of the relay via MOSFET (Q_1).
- Practical optocoupler and relay connection. Using a Raspberry Pi board (or any other microcontroller) to control very-high-voltage devices via digital input/output voltages.

Connecting High-Power Outputs using Relays (2)





- Practical simple relay module. Using an Arduino Uno R3 board (or any other microcontroller) to control high-voltage devices (upto a 220-V lamp, or similar) via digital input/output voltages.
  - A diode (1N4007) is reversely connected to the coil of the relay via NPN transistor (BC547).
  - VCC: Relay input voltage
  - GND: Relay ground voltage reference
  - Signal: Digital control signal
  - Common Contact: Power to the external device
  - Normally Closed: Common contact is connected when the signal is LOW ; otherwise, disconnected.
  - Normally Open: Common contact is connected when the signal is HIGH; otherwise, disconnected.

Thanks for listening ☺

YALÇIN İŞLER

Assoc. Prof.