# Sturen en lezen van de pinnen van de Raspberry Pi

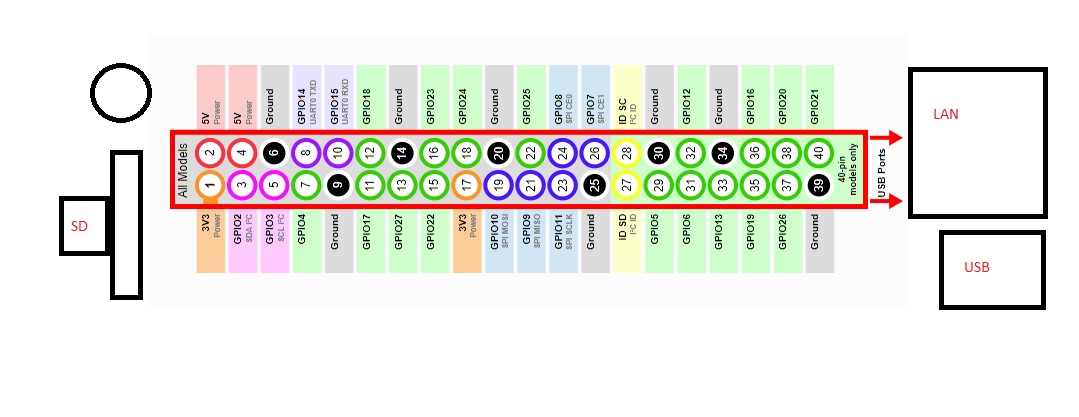
# De RPi.GPIO module is eenvoudig en vergelijkbaar met het sturen van pinnen in Arduino.

# Importeren van de module in je Python programma:

**import** **RPi.GPIO**

**of**

**import** **RPi.GPIO** **as** **GPIO**

**Pin nummering van de 40 pins mannelijke header.**

**Afbeelding met tekst

Automatisch gegenereerde beschrijvingAansluiten van 3 leds en 2 drukknoppen**

**Keuze van de pinnen op 2 manieren :**

**GPIO.setmode(GPIO.BOARD)** dit refereert naar de pinnen 1-40 van de header

of

**GPIO.setmode(GPIO.BCM)** dit refereert naar de GPIO nummers

Opm : channel parameter hieronder afhankelijk van bovenstaande keuze

**Waarschuwingen indien de pin in een ander programma werd geconfigureerd.**

**GPIO.setwarnings(False)** dit disabelt de waarschuwingen

**Een pin als input of output definiëren**

**GPIO.setup(channel, GPIO.OUT)** definieer de pin als output

**GPIO.setup(channel, GPIO.IN)** definieer de pin als input

Om zweven van de pin tegen te gaan kan je een uitwendige of inwendige pullup pulldown weerstand aanbrengen of configureren.

**GPIO.setup(channel, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)**

of

**GPIO.setup(channel, GPIO.IN, pull\_up\_down=GPIO.PUD\_DOWN)**

Lezen van de toestand van een pin die als input geconfigureerd werd

**GPIO.input(channel)**

De toestand van de pin sturen

**GPIO.output(channel, state)**

state => 0 / GPIO.LOW / False of 1 / GPIO.HIGH / True

channel is de pin vd header indien setmode= BOARD en channel is GPIOnr indien setmode =BCM

Channel kan een list zijn ( zie later)

Vrijgeven pinnen aan het einde vh programma

**GPIO.cleanup()**

**Events met callbacks toevoegen**

*# add rising edge detection on a channel, ignoring further edges for 200ms for switch bounce handling*

GPIO.add\_event\_detect(channel, GPIO.RISING, callback=my\_callback, bouncetime=200)