https://www.halvorsen.blog

Part 2: PWM and Push Buttons

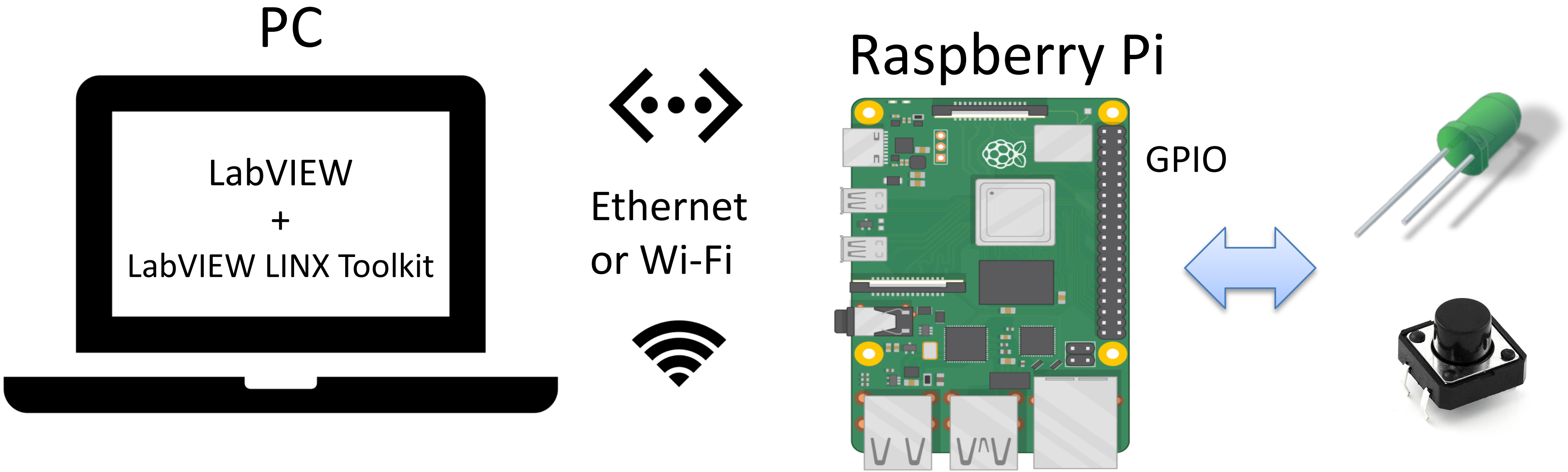
# LabVIEW LINX and Raspberry Pi

LabVIEW + LabVIEW LINX Toolkit + Raspberry Pi

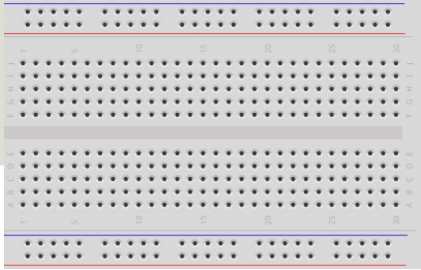
Hans-Petter Halvorsen

# Table of Contents

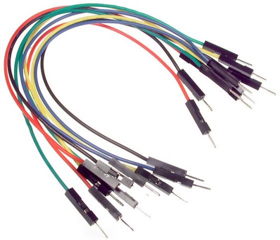
# LabVIEW + LabVIEW LINX Toolkit

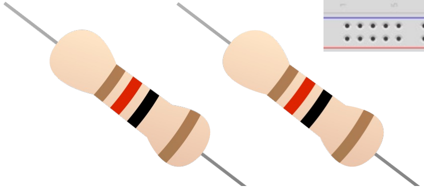


# Hardware

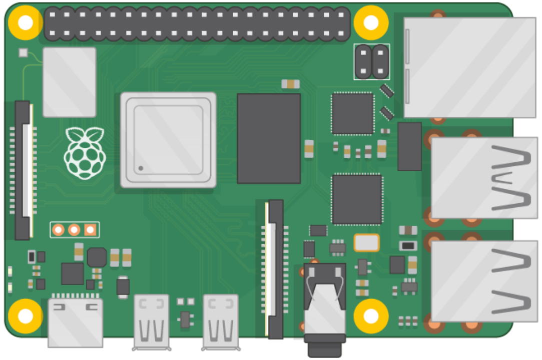


- Raspberry Pi





- Breadboard
- Wires (Jumper Wires)



- Resistors ($R = 270\Omega$)

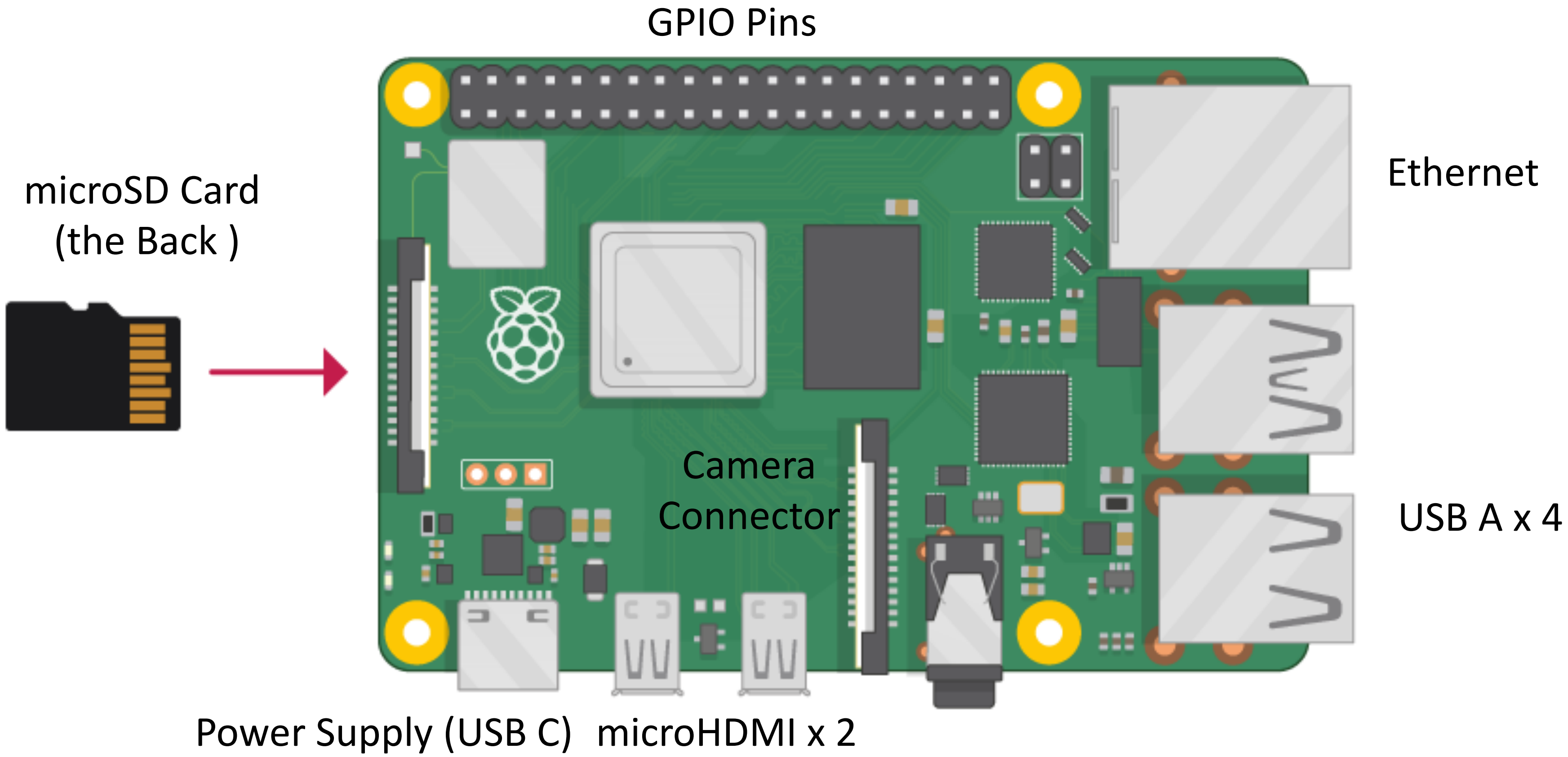
- LED, Push Button

# Raspberry Pi and LabVIEW LINX

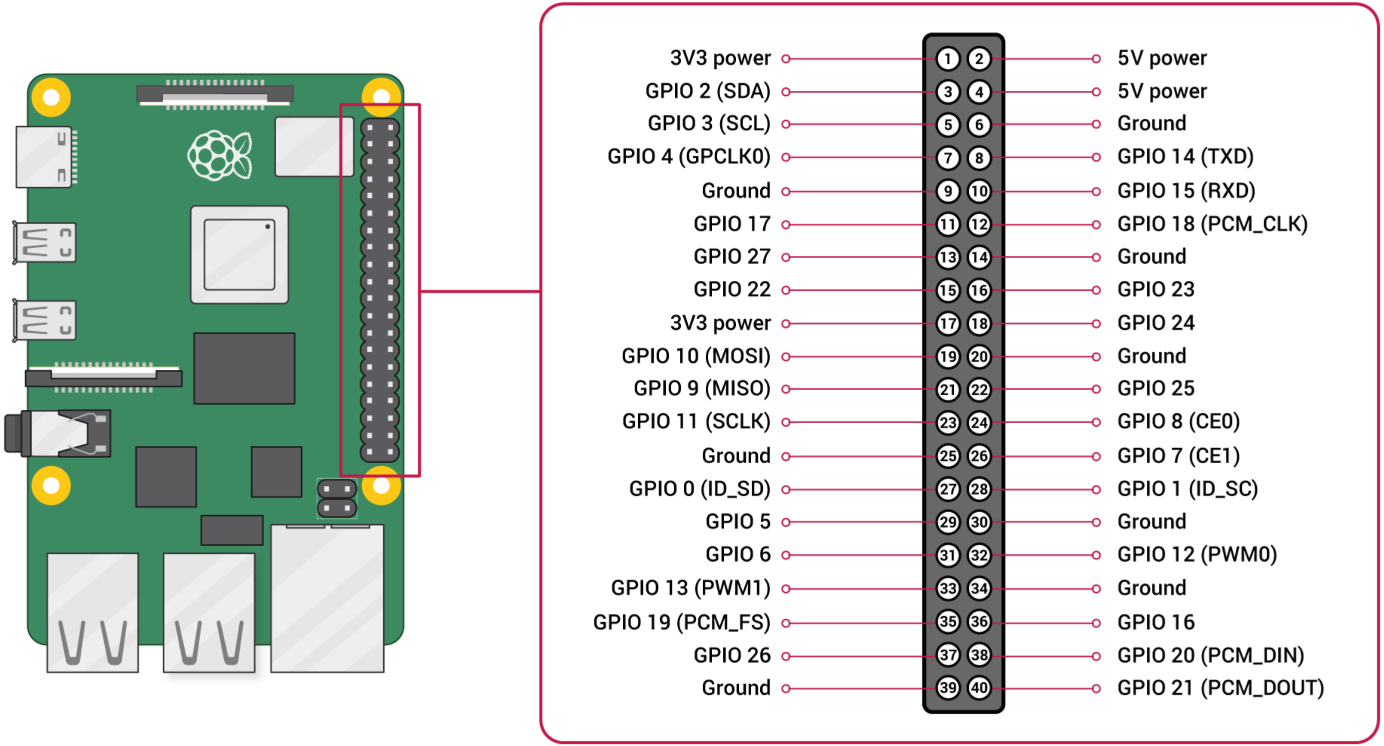
Hans-Petter Halvorsen

Table of Contents

# Raspberry Pi



# GPIO




A powerful feature of the Raspberry Pi is the GPIO (general-purpose input/output) pins. The Raspberry Pi has a 40-pin GPIO header as seen in the image

# Raspberry Pi OS

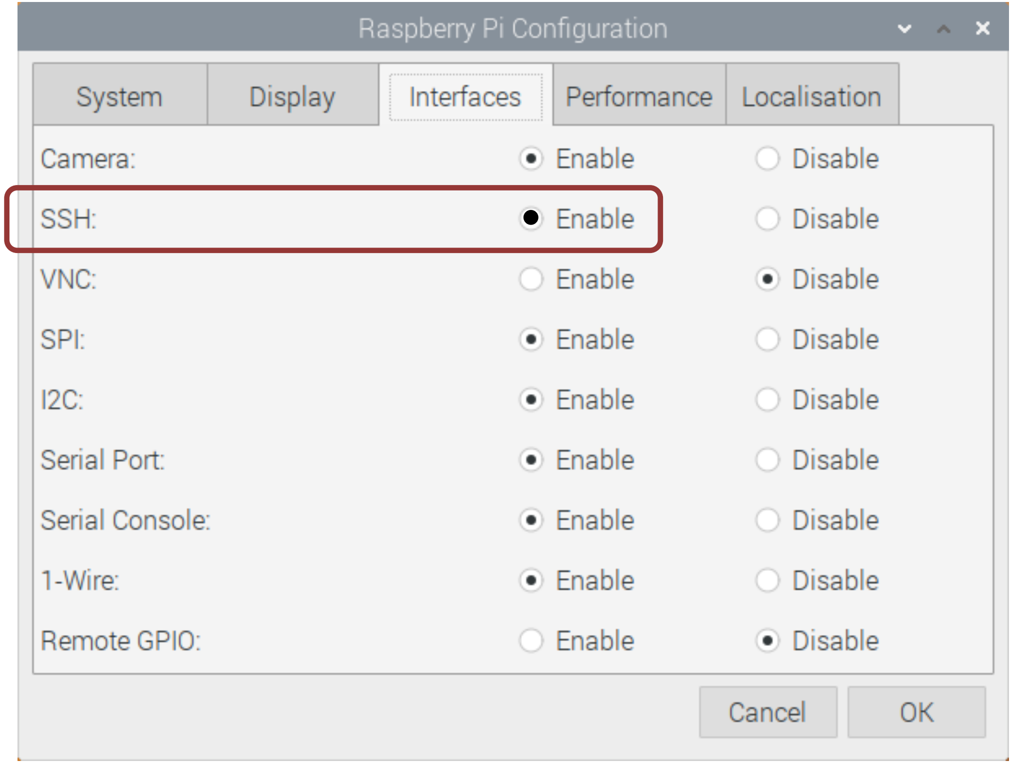
- In order make your Raspberry Pi up and running you need to install an Operating System (OS)
- The OS for Raspberry Pi is called “Raspberry Pi OS“ (previously known as Raspbian)
- Raspberry Pi runs a version of an operating system called Linux (Windows and macOS are other operating systems).
- To install the necessary OS, you need a microSD card
- Then you use the “Raspberry Pi Imager“ in order to download the OS to the microSD card.

https://www.raspberrypi.org/software/

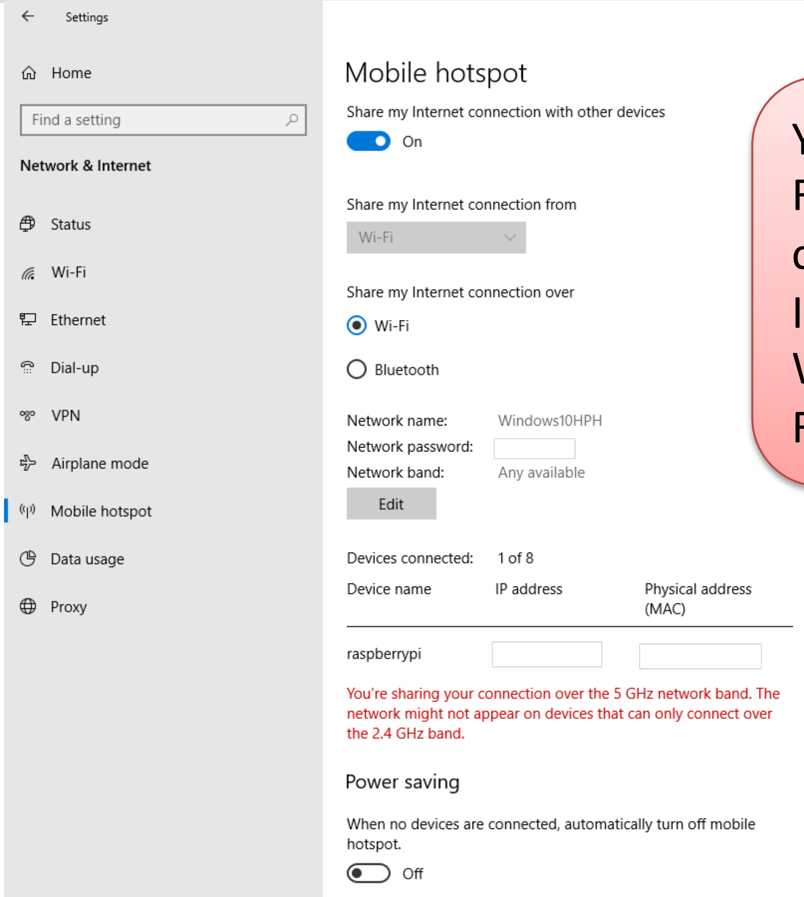
# Raspberry Pi Configuration

You need to Enable **SSH** so you can remotely get access to the Raspberry Pi from your Computer

SSH, also known as Secure Shell or Secure Socket Shell, is a Network Protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network.



# Mobile Wi-Fi hotspot on Windows10

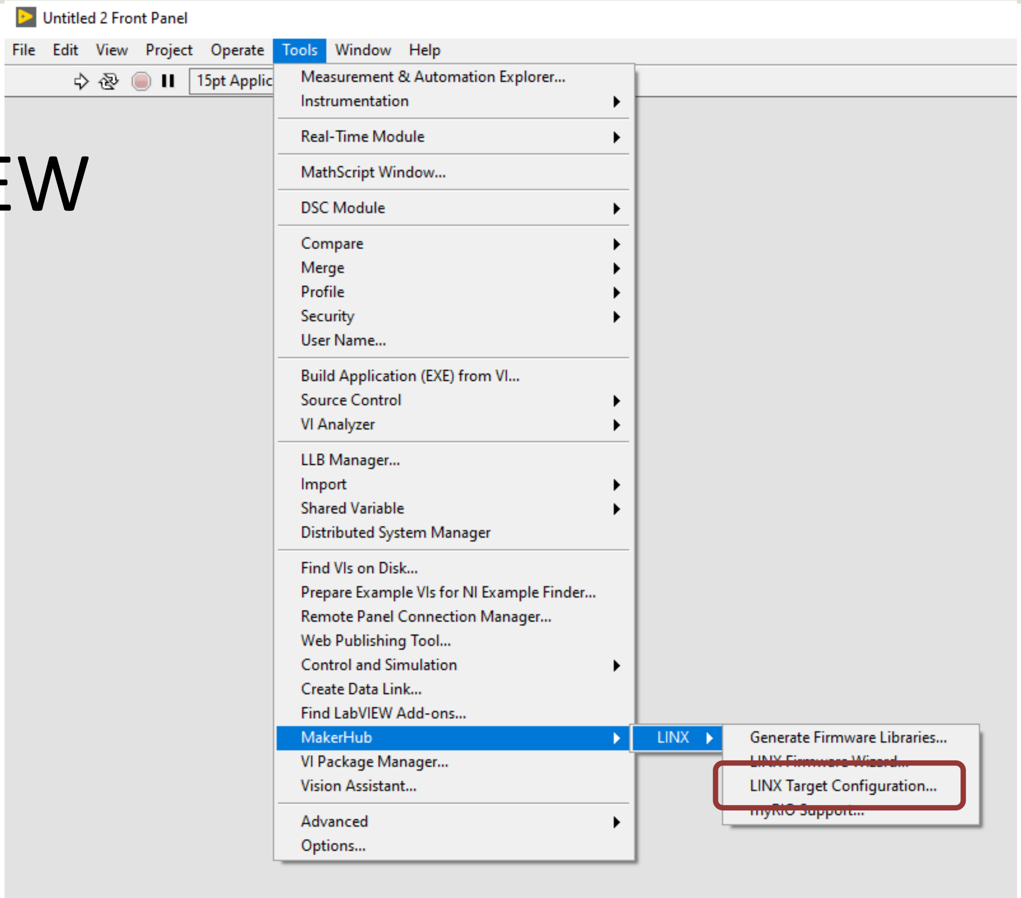



You can connect your PC and the Raspberry Pi together using an Ethernet cable or using Wi-Fi.

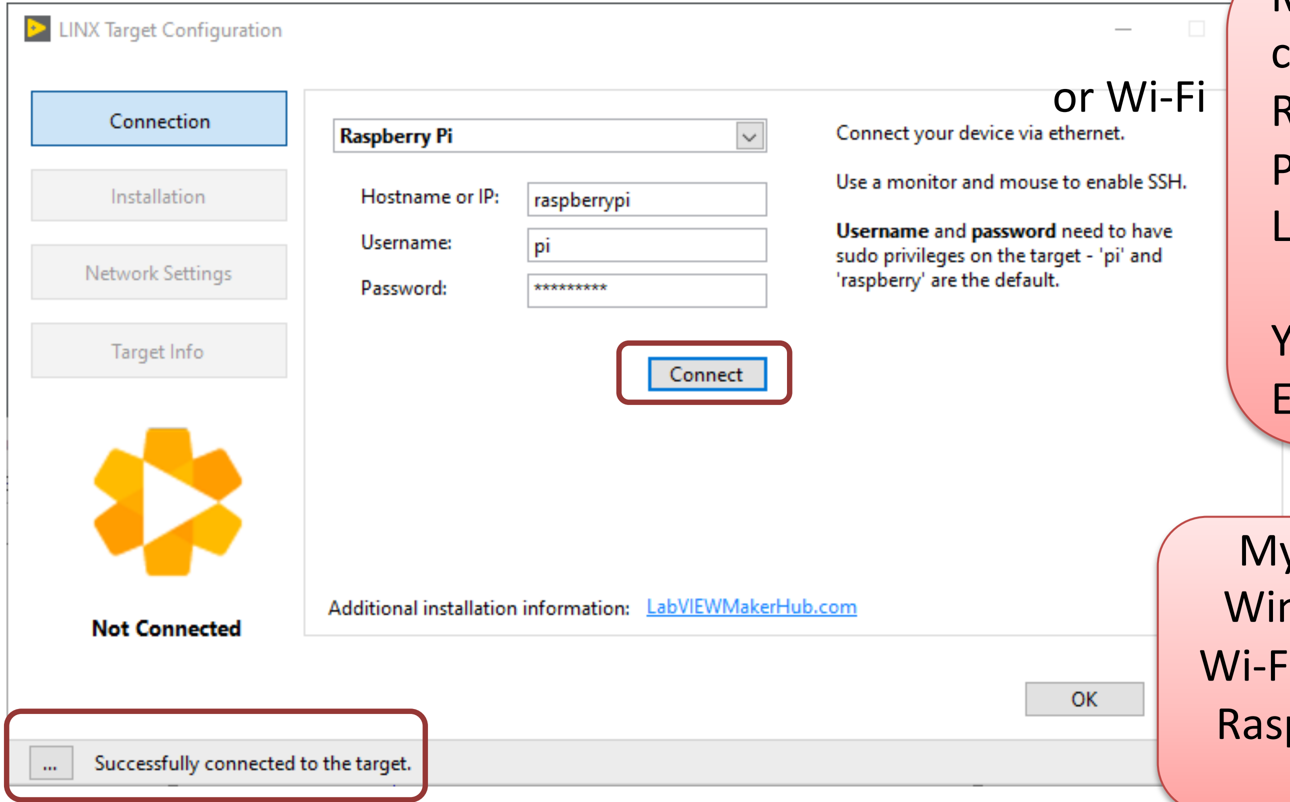
I configured Mobile Wi-Fi hotspot on my Windows10 PC. Then I connected my Raspberry Pi to this Wi-Fi network

# Raspberry Pi LINX Configuration

LabVIEW



# Raspberry Pi LINX Configuration



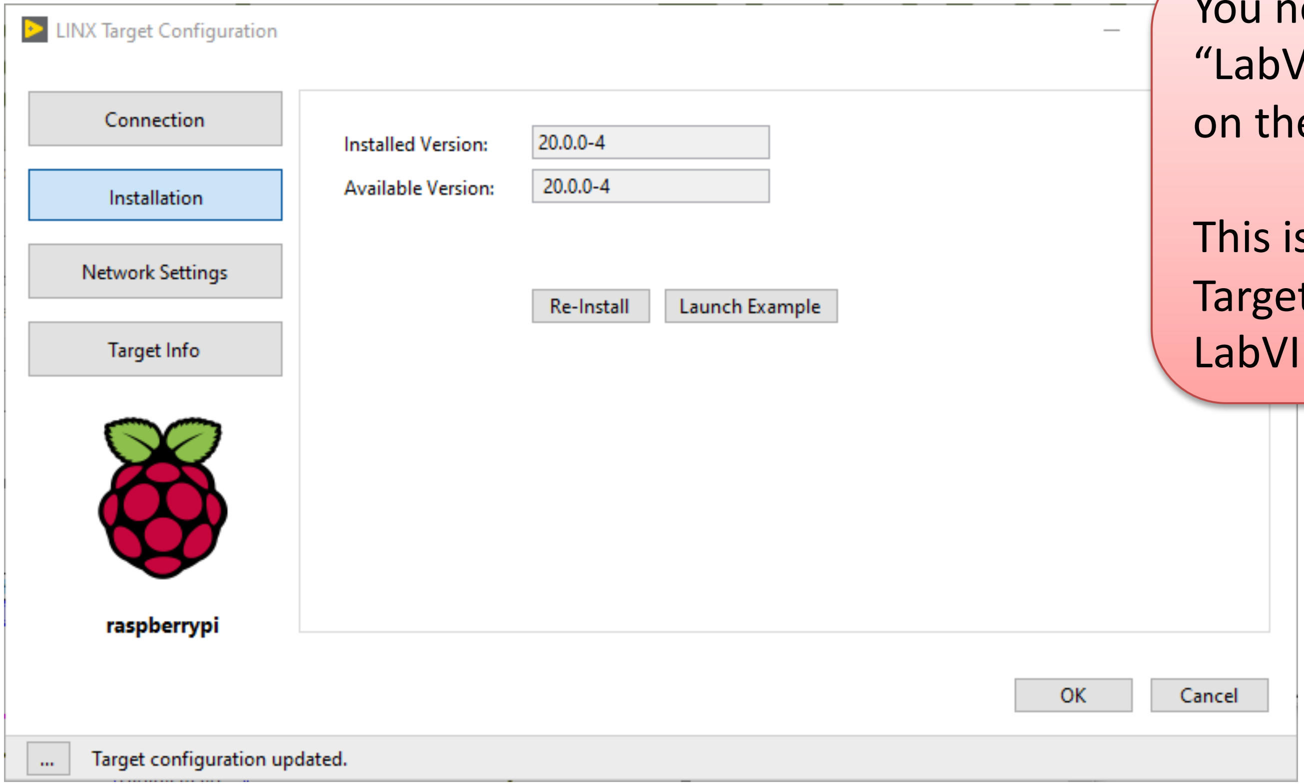

or Wi-Fi

Make sure you can connect to the Raspberry Pi from your PC where you have LabVIEW installed.

You can use Wi-Fi or an Ethernet cable

My Configuration: On my Windows PC I configured a Wi-Fi Mobile hotspot. On the Raspberry Pi I connected to this Wi-Fi hotspot

# Raspberry Pi LINX Configuration



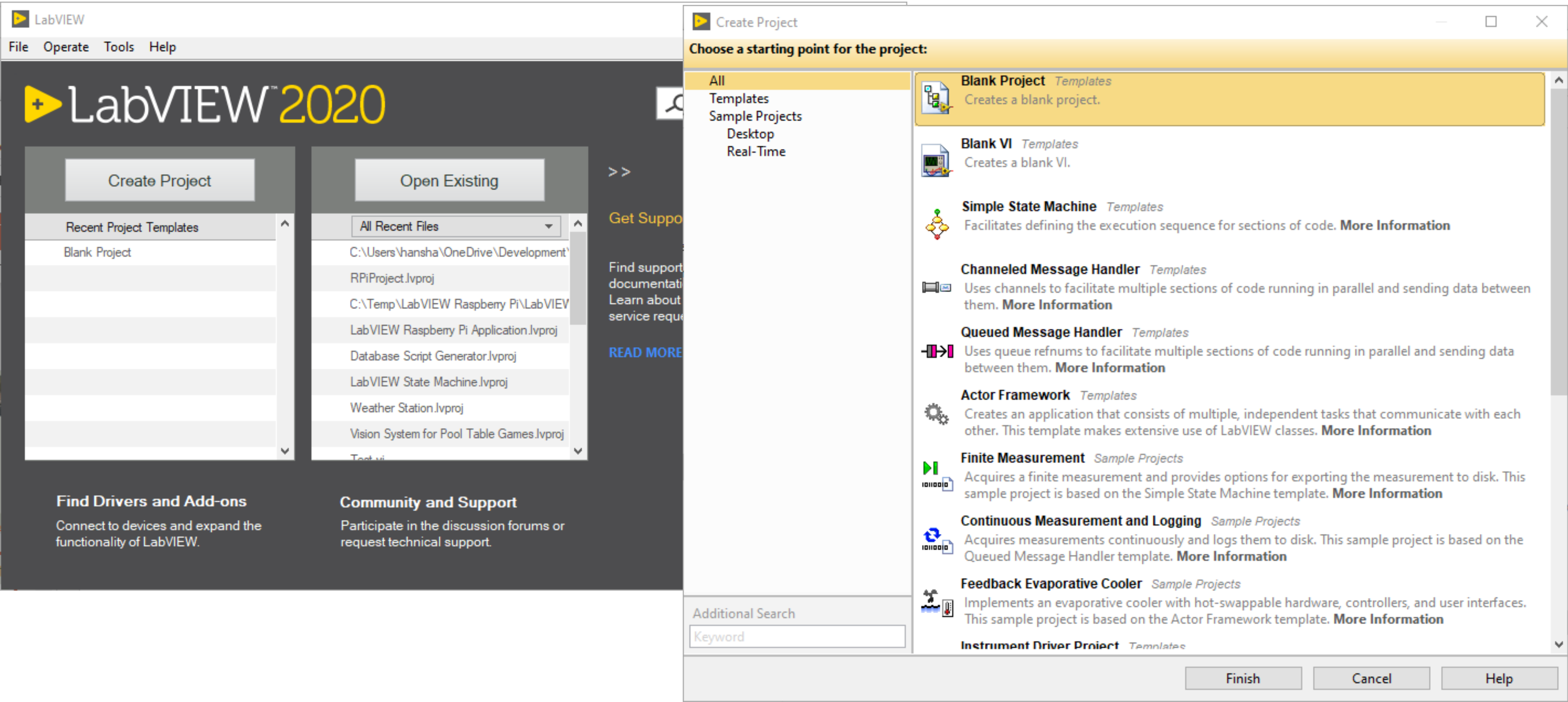

You need to install "LabVIEW Runtime Engine" on the Raspberry Pi device.

This is done from the LINX Target Configuration in LabVIEW on your PC

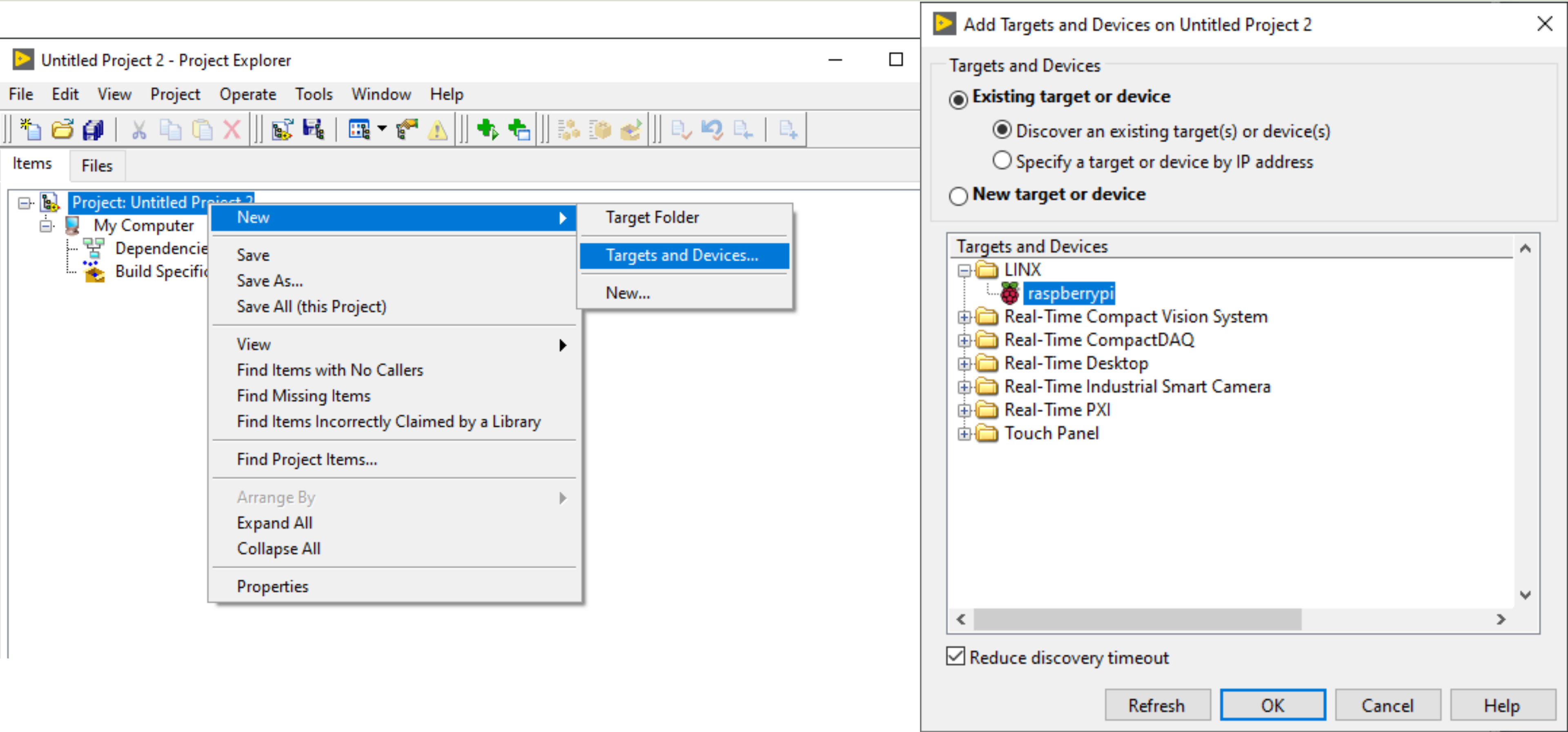
# LabVIEW Raspberry Pi Project

Hans-Petter Halvorsen

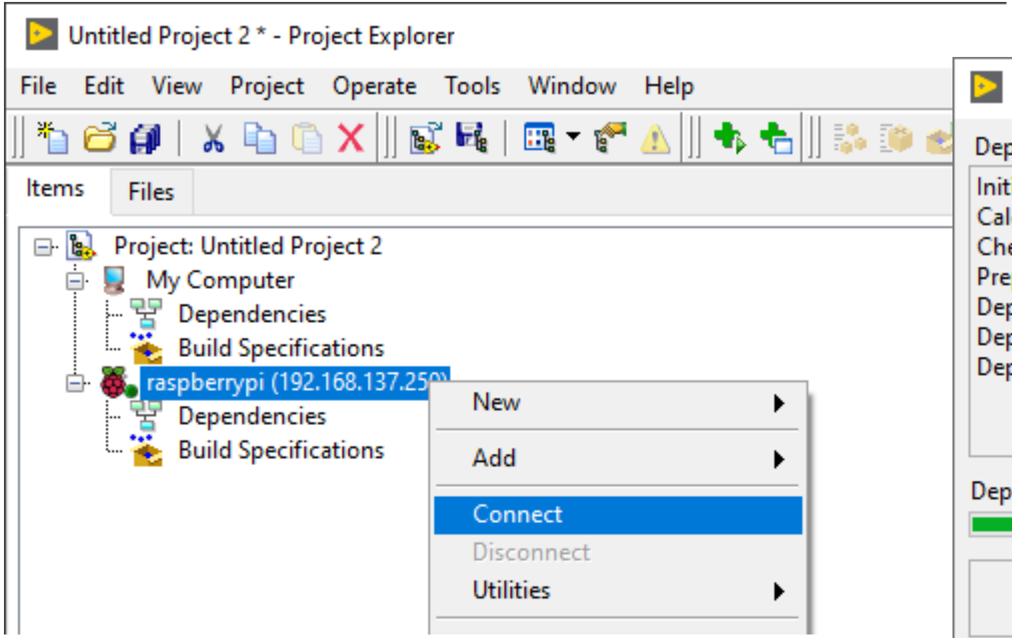
# Create your Raspberry Pi Project

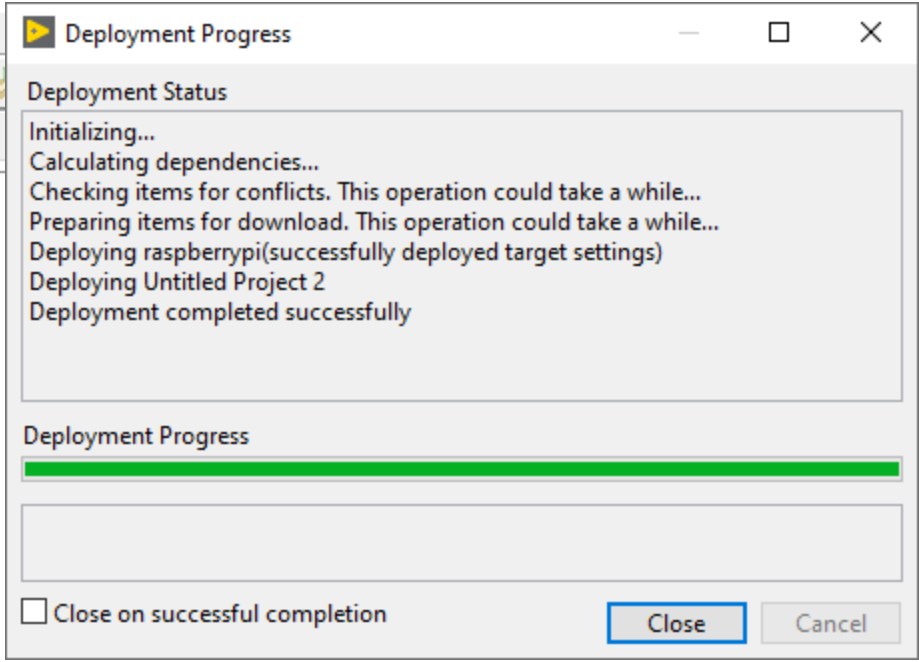


# Create your Raspberry Pi Project

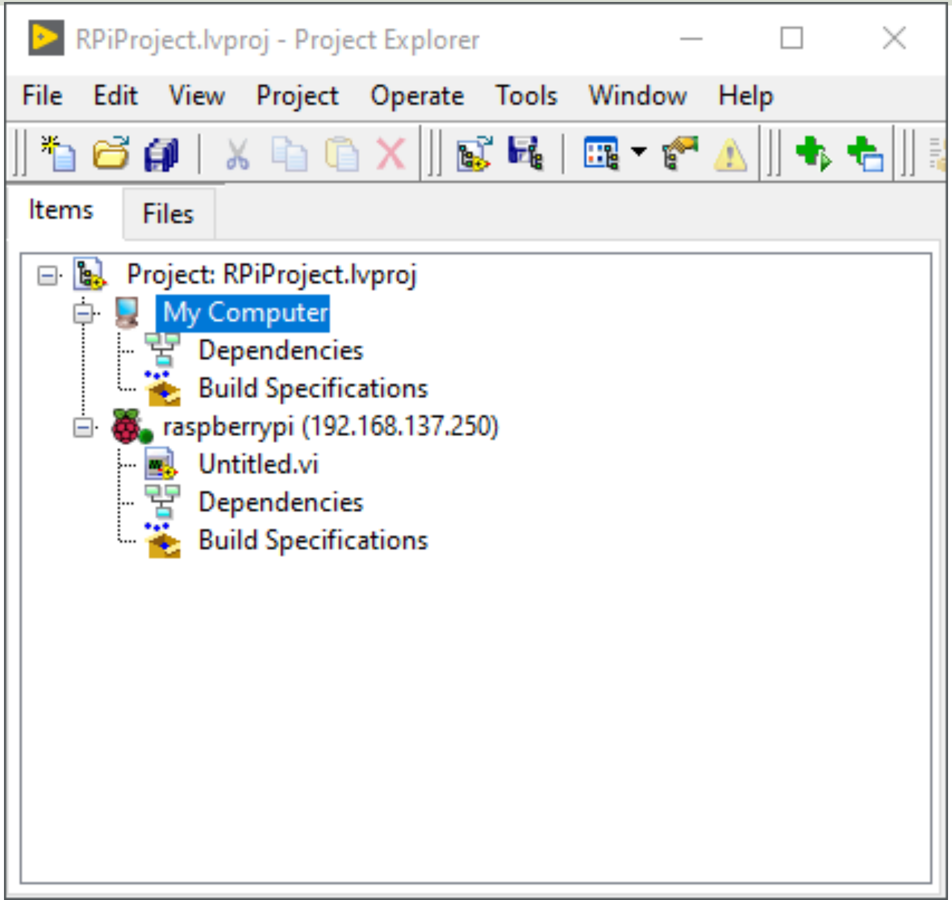


# Create your Raspberry Pi Project



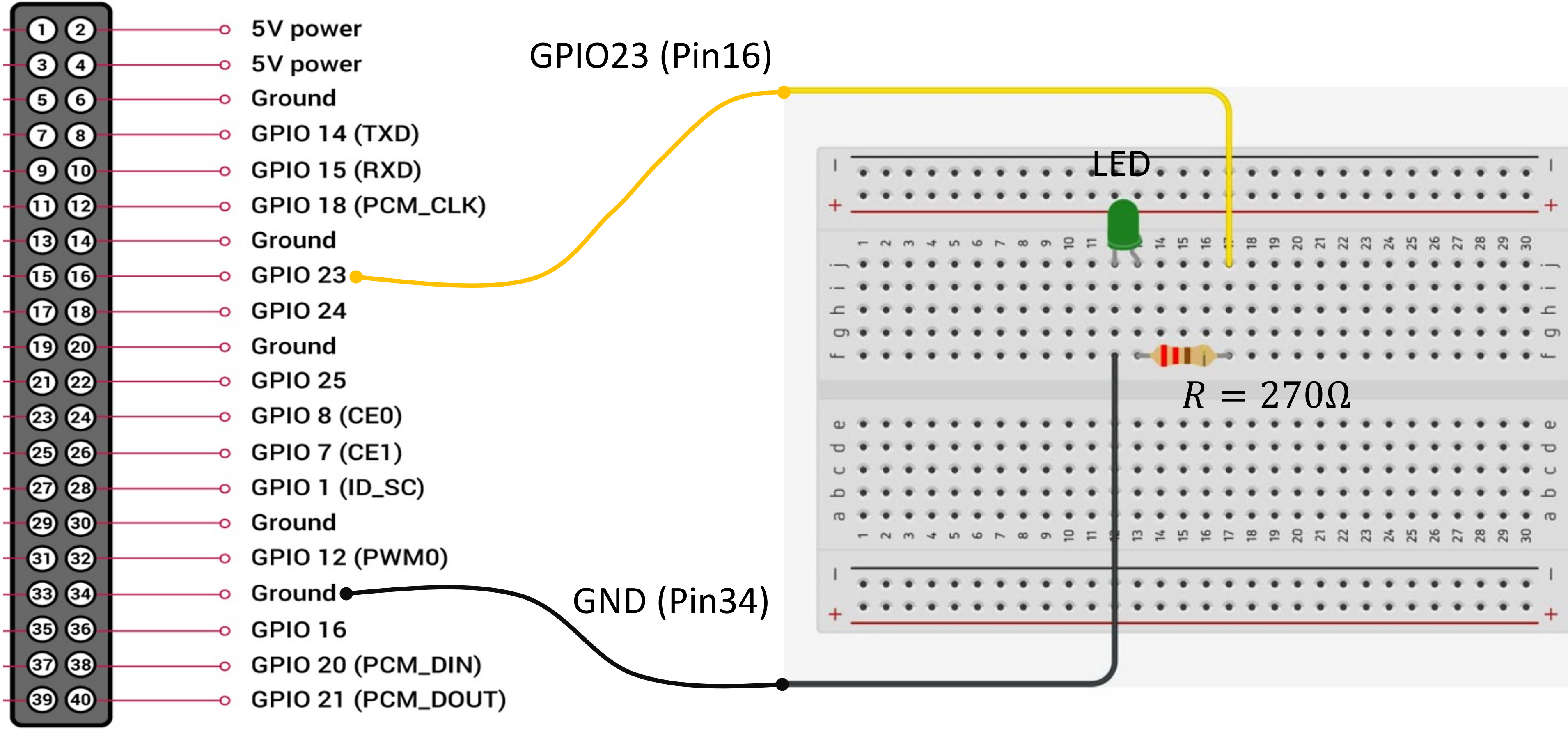
# LabVIEW Project Explorer




You are now ready to start creating LabVIEW Code that control the GPIO pins on the Raspberry Pi device

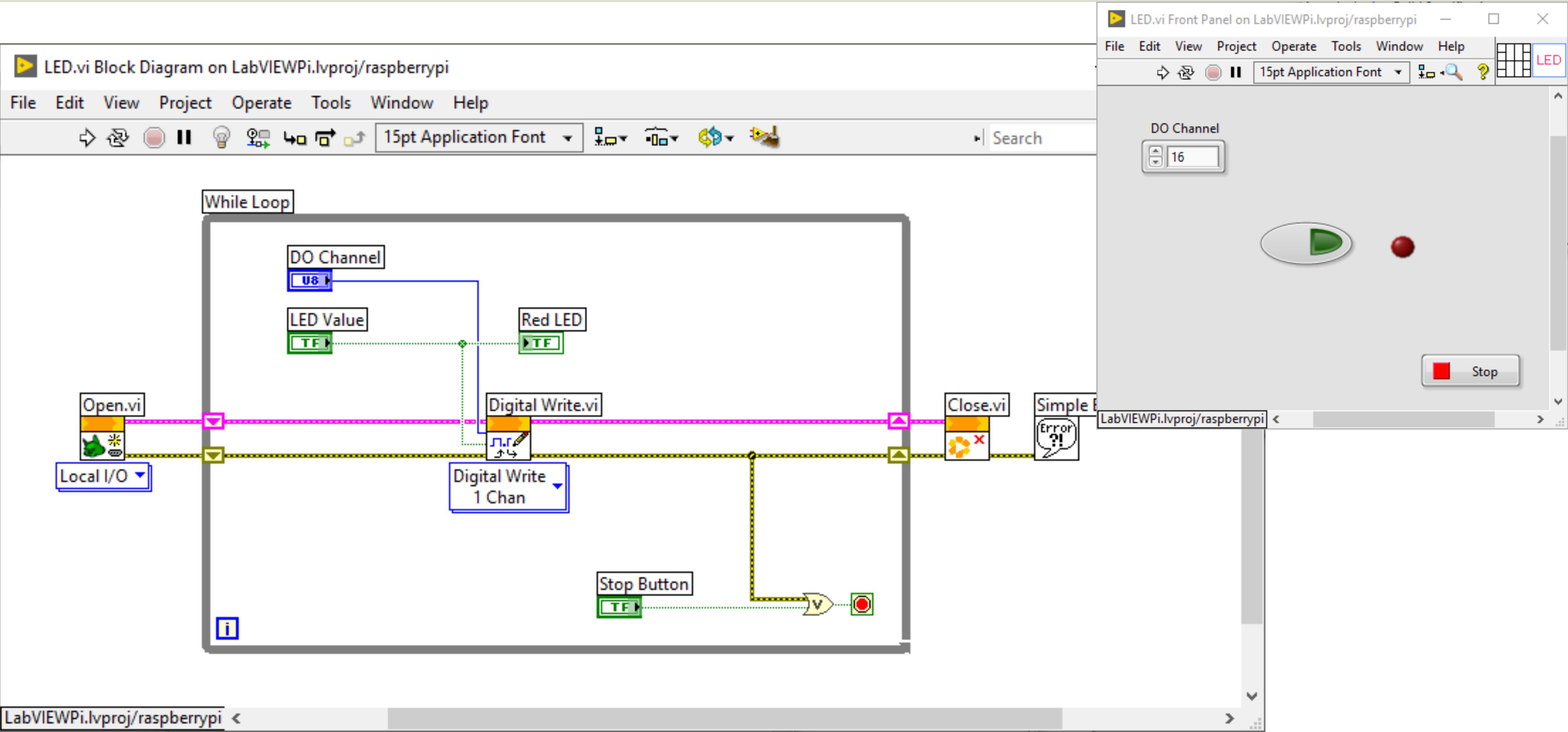
# Digital Out (DO)

# LED Wiring



5V power
5V power
Ground
GPIO 14 (TXD)
GPIO 15 (RXD)
GPIO 18 (PCM_CLK)
Ground
GPIO 23
GPIO 24
Ground
GPIO 25
GPIO 8 (CE0)
GPIO 7 (CE1)
GPIO 1 (ID_SC)
Ground
GPIO 12 (PWM0)
Ground
GPIO 16
GPIO 20 (PCM_DIN)
GPIO 21 (PCM_DOUT)
GPIO23 (Pin16)
LED
$R = 270\Omega$
GND (Pin34)

# LED ON/OFF - LabVIEW Example



# Digital In (DI)

Hans-Petter Halvorsen

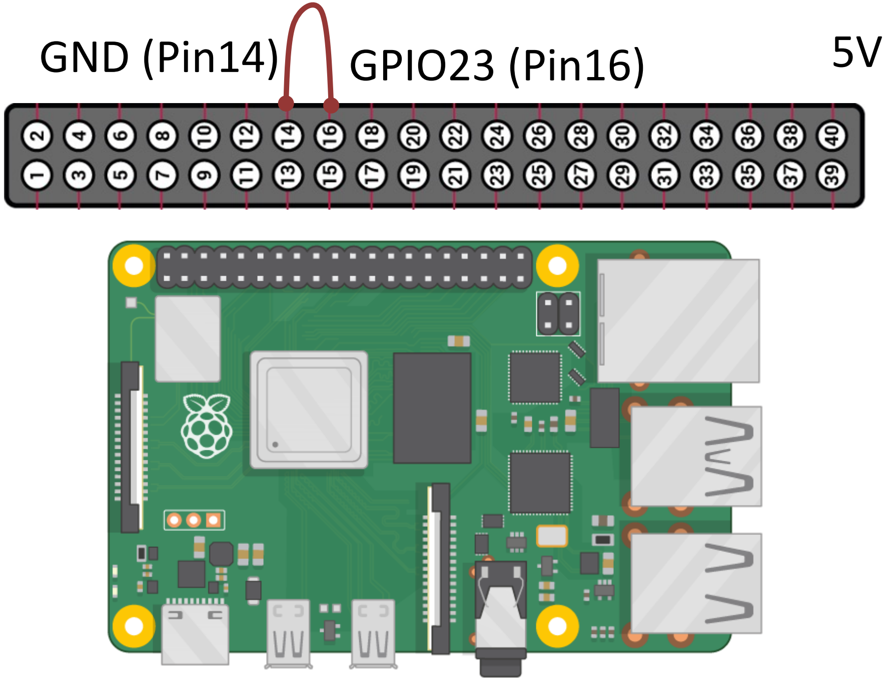
Table of Contents

# Test of Digital Read

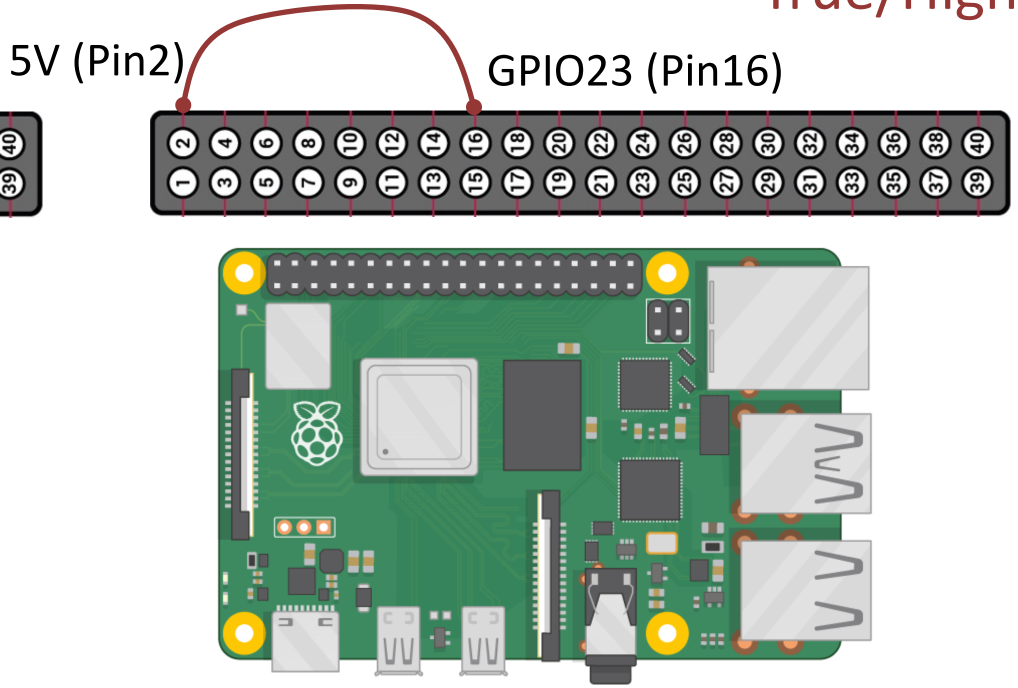
We can test the Digital In (Read) by wiring to GND (False/Low) or 5V (True/High)

GPIO23 (Pin16) is used in this example, but you can of course use another GPIO pin

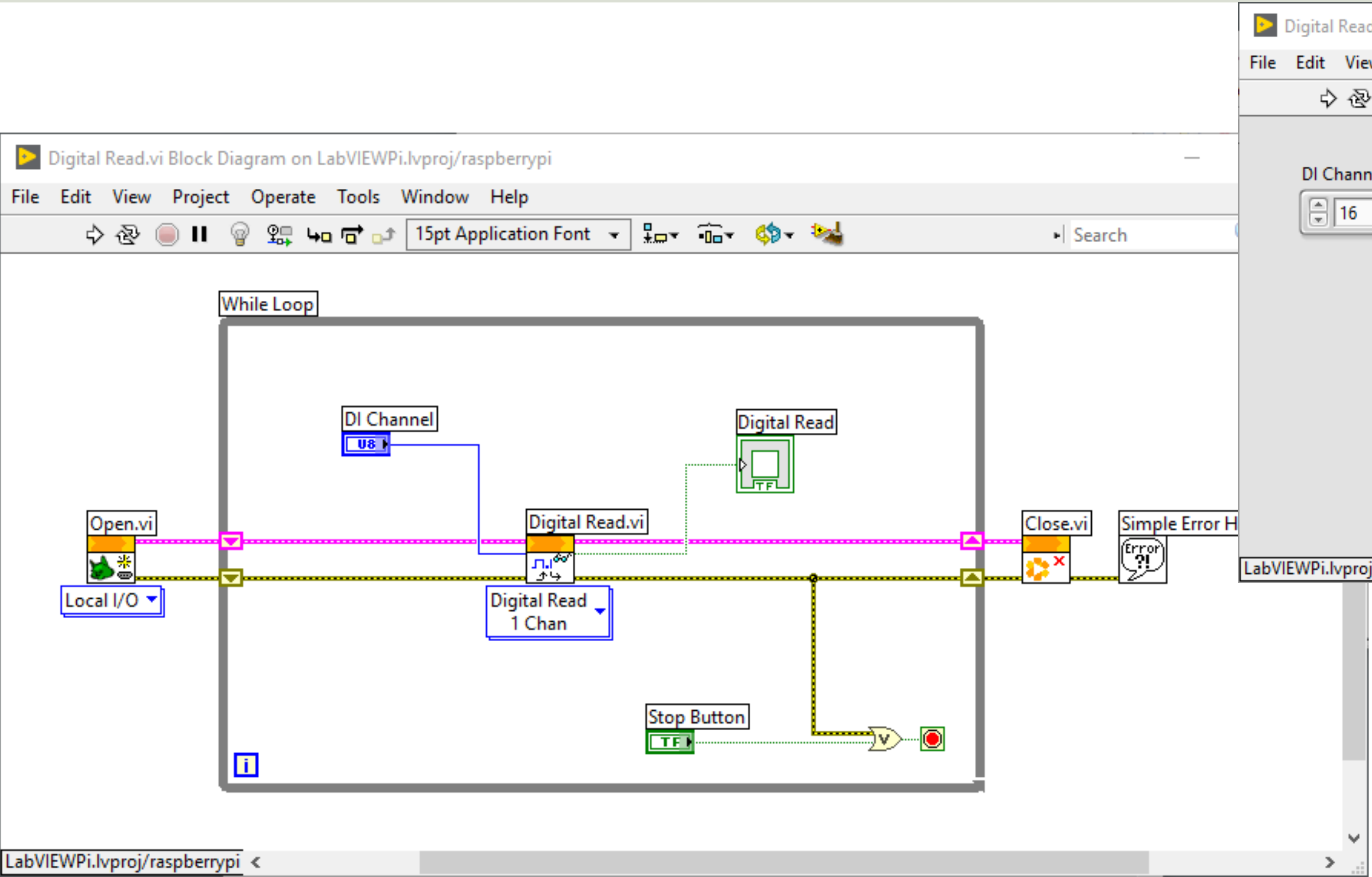
False/Low




True/High

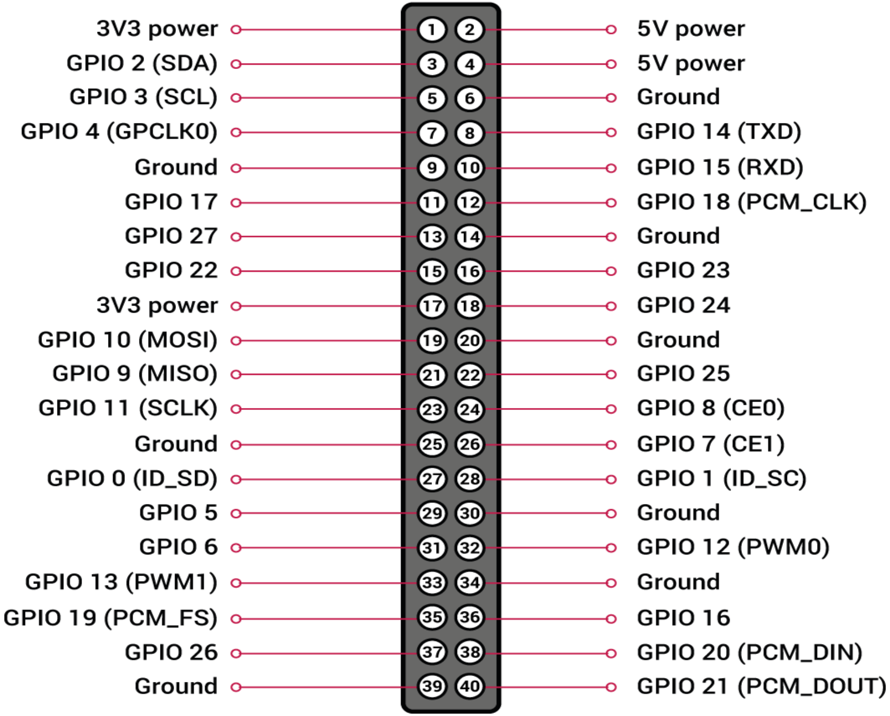


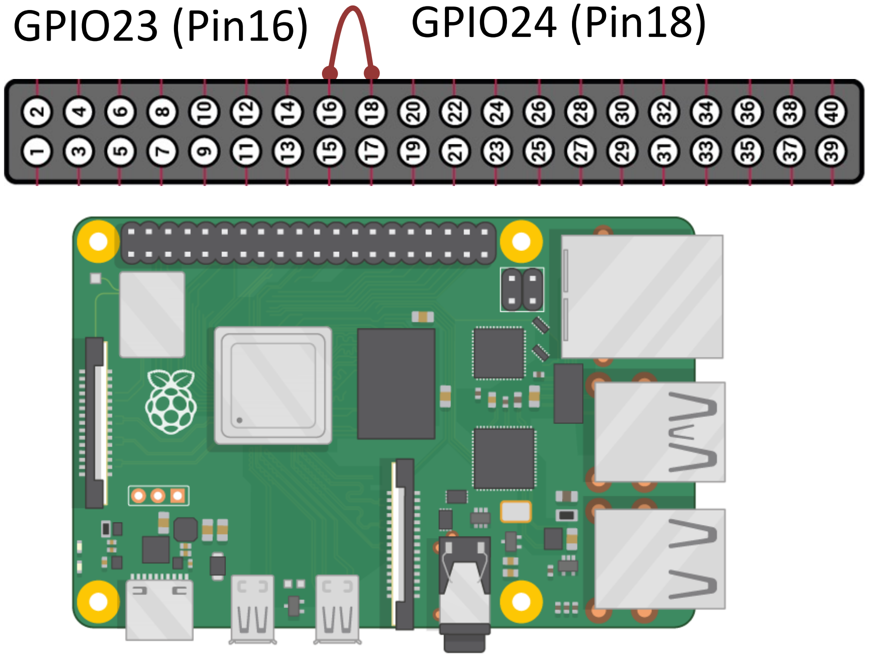
# LabVIEW - Digital Read



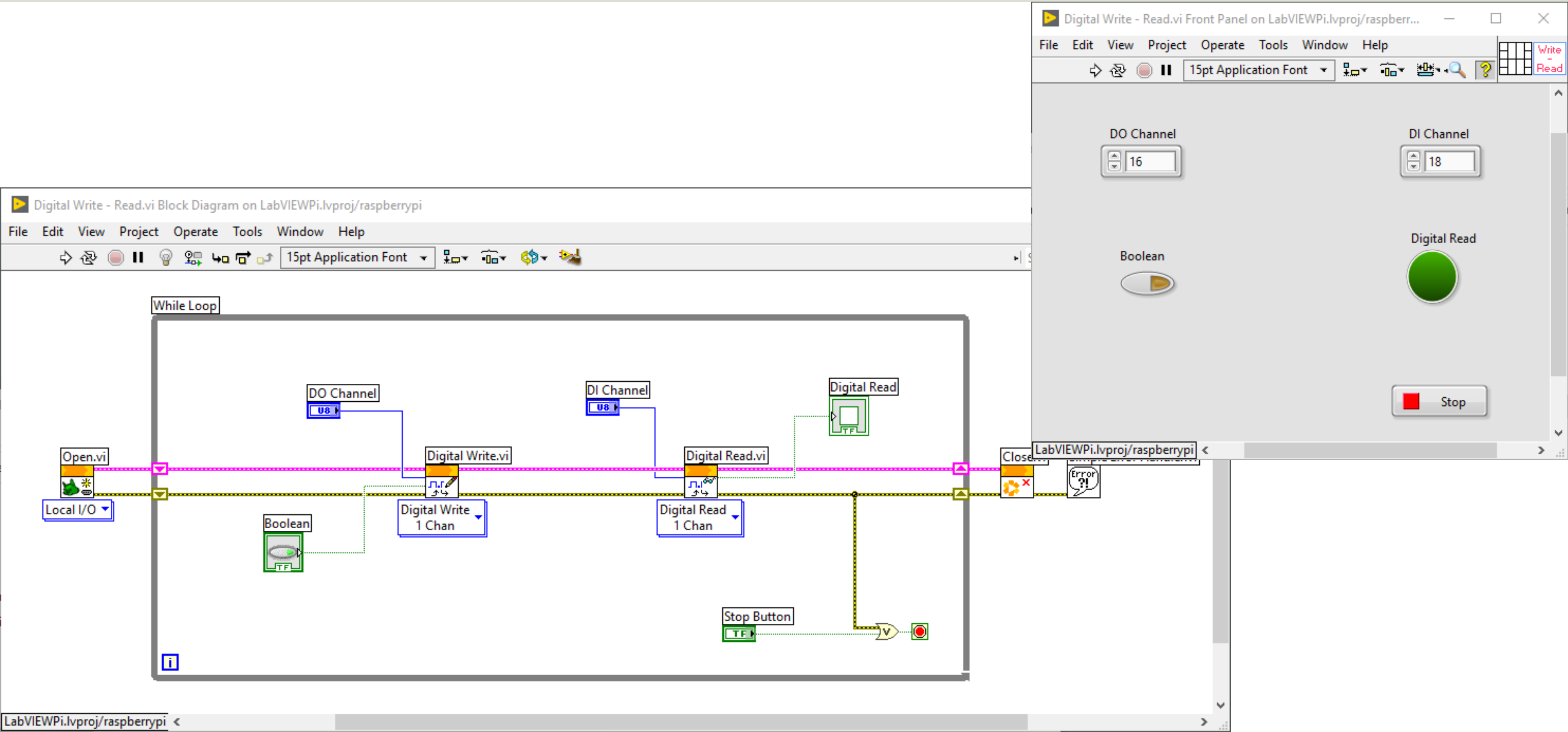
# LabVIEW Digital Write - Read

We can test the Digital Read by wiring a “Digital Out” (Write) Channel to the “Digital In” (Read) Channel



# LabVIEW Digital Write - Read



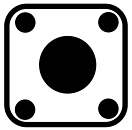
# Push Buttons

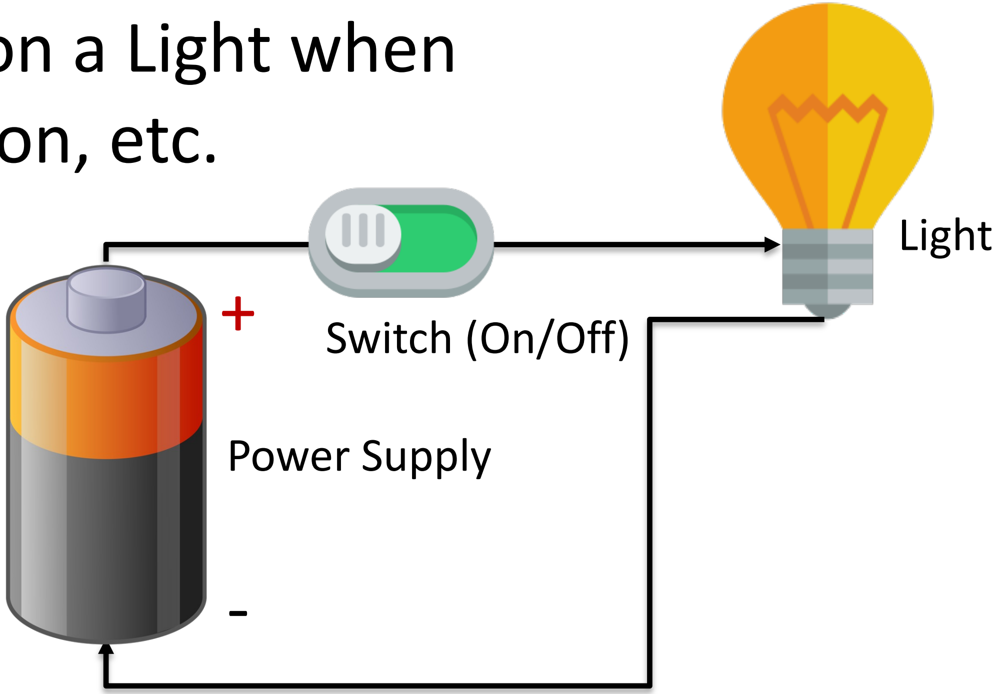
Hans-Petter Halvorsen

Table of Contents

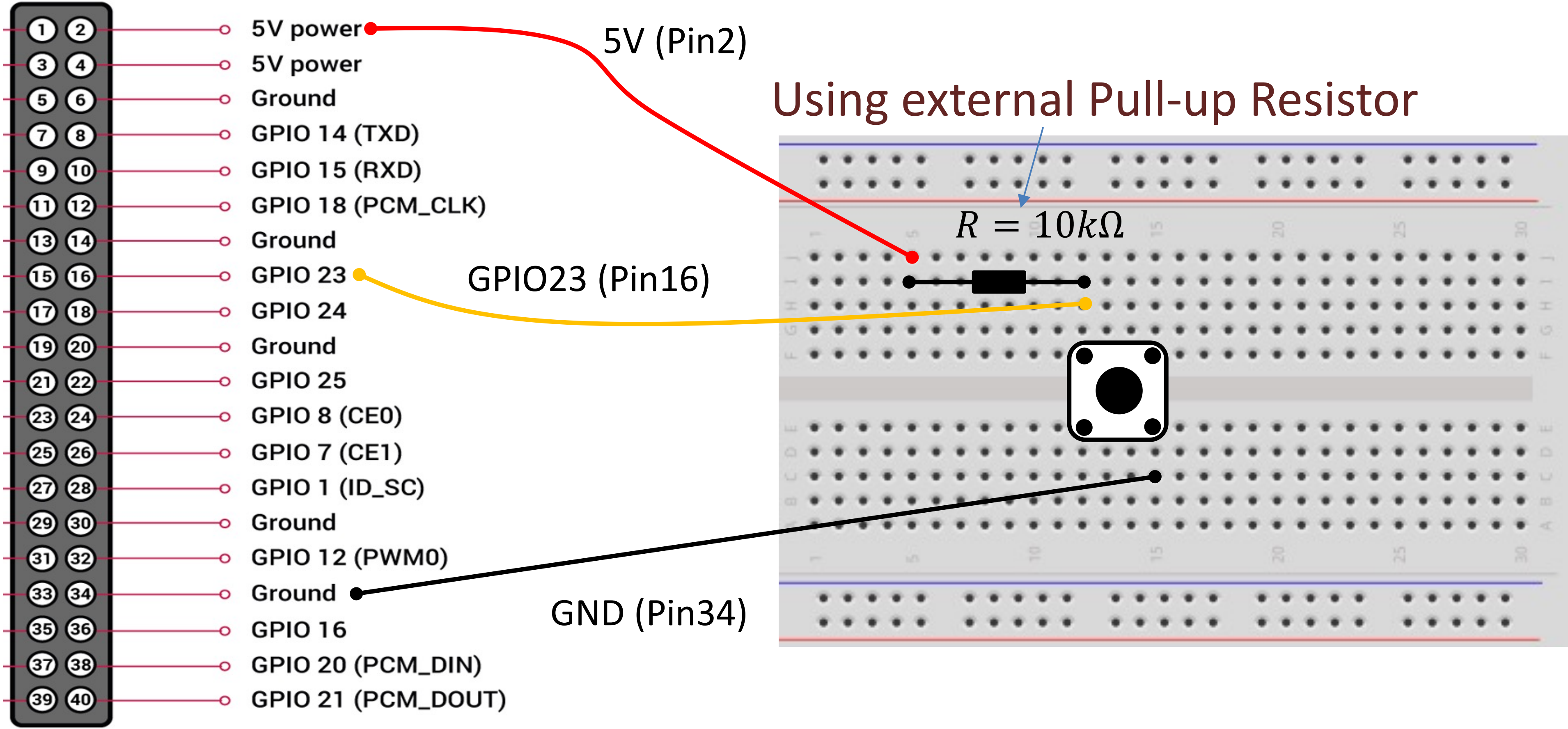
# Push Button/Switch

- Pushbuttons or switches connect two points in a circuit when you press them.
- You can use it to turn on a Light when holding down the button, etc.





# Wiring (Pull-up Resistor)

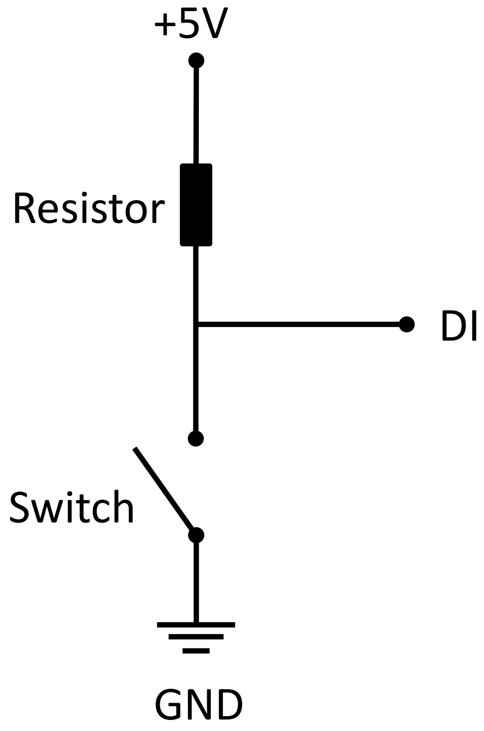


# Pull-down/Pull-up Resistor

Why do we need a pull-up or pull-down resistor in the circuit?

- If you disconnect the digital I/O pin from everything, it will behave in an irregular way.
- This is because the input is "floating" - that is, it will randomly return either HIGH or LOW.
- That's why you need a pull-up or pull-down resistor in the circuit.

# Pull-up Resistor

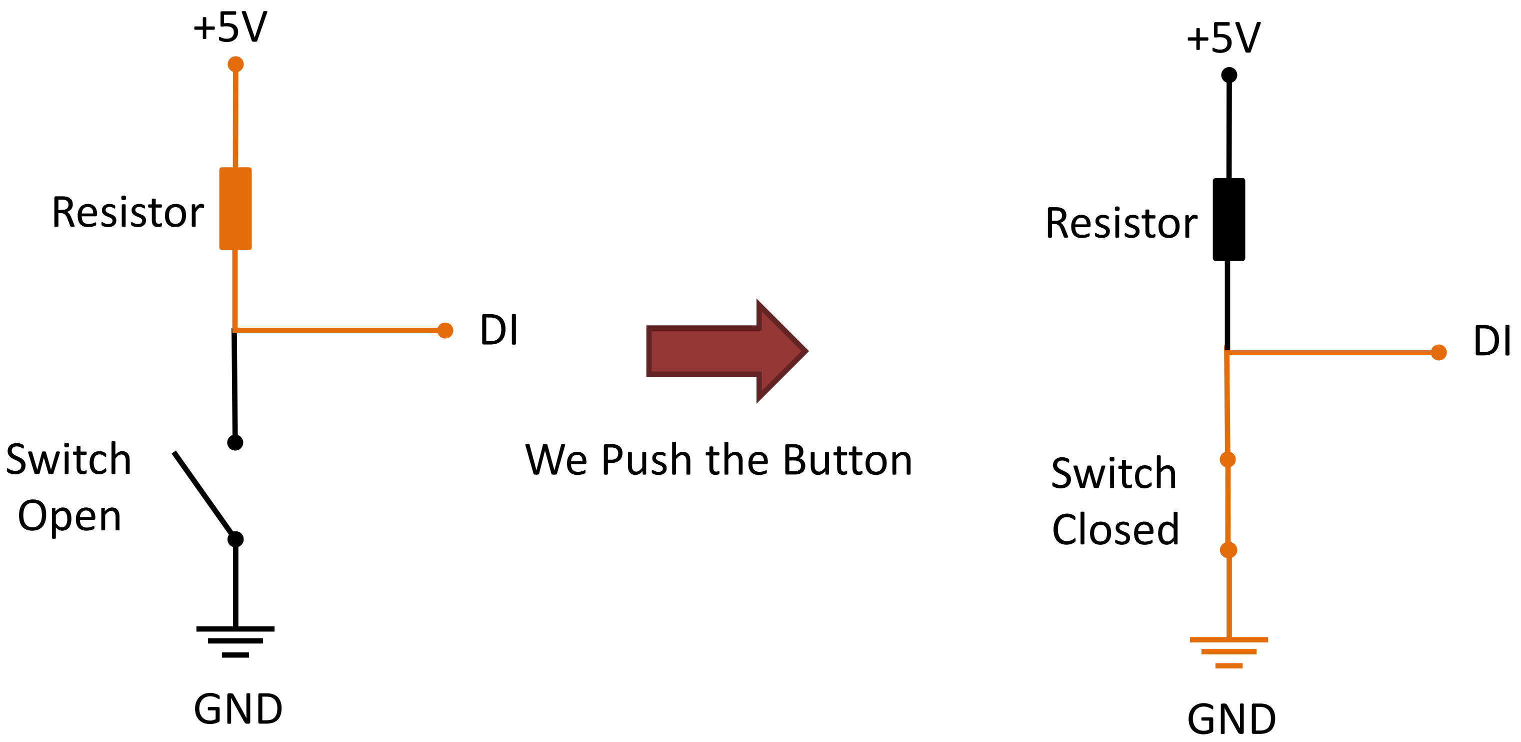



- When the pushbutton is open (unpressed) there is a connection between 5V and the DI pin.
- This means the default state is **True** (High).
- When the button is closed (pressed), the state goes to **False** (Low).

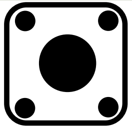
# Pull-up Resistor

## True/High

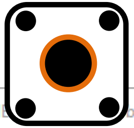
## False/Low



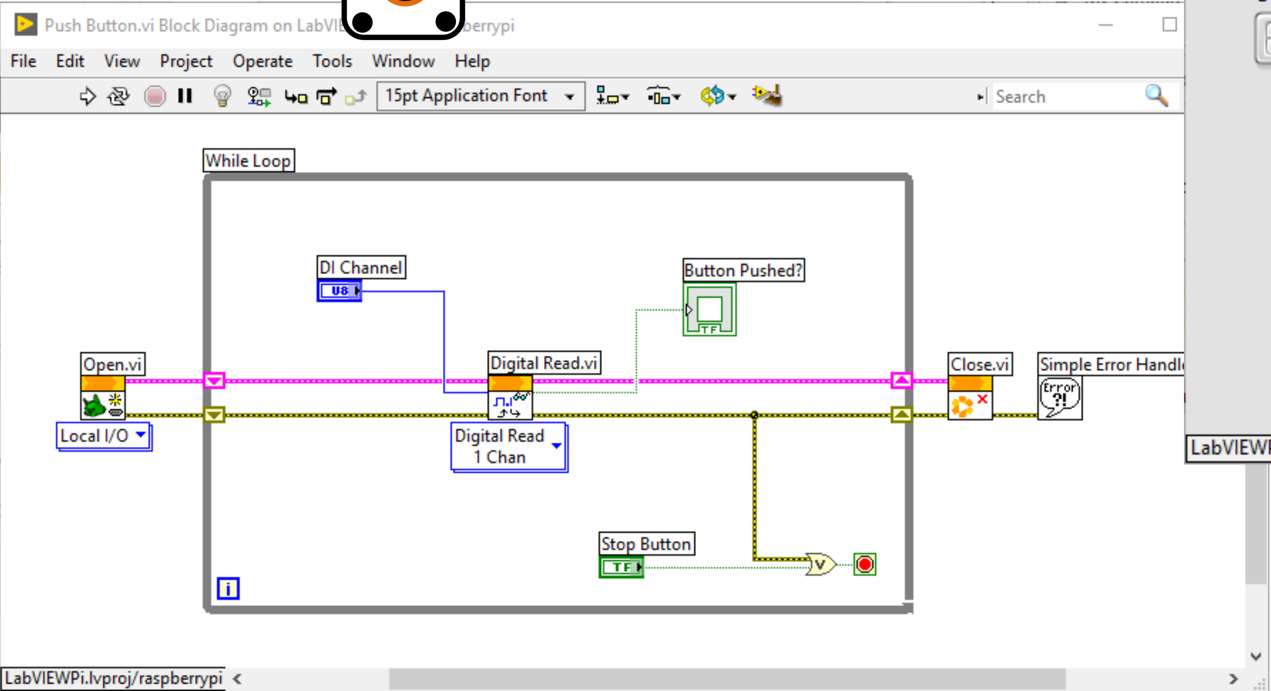
# Push Button (Pull-up Resistor)

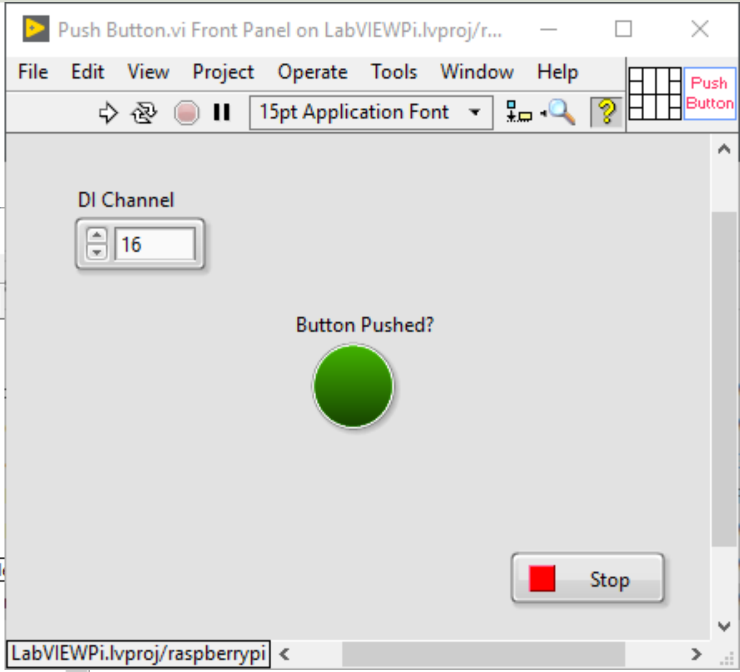


Button is NOT Pushed => True/High

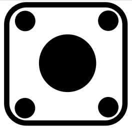


Button is Pushed => False/Low

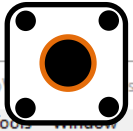


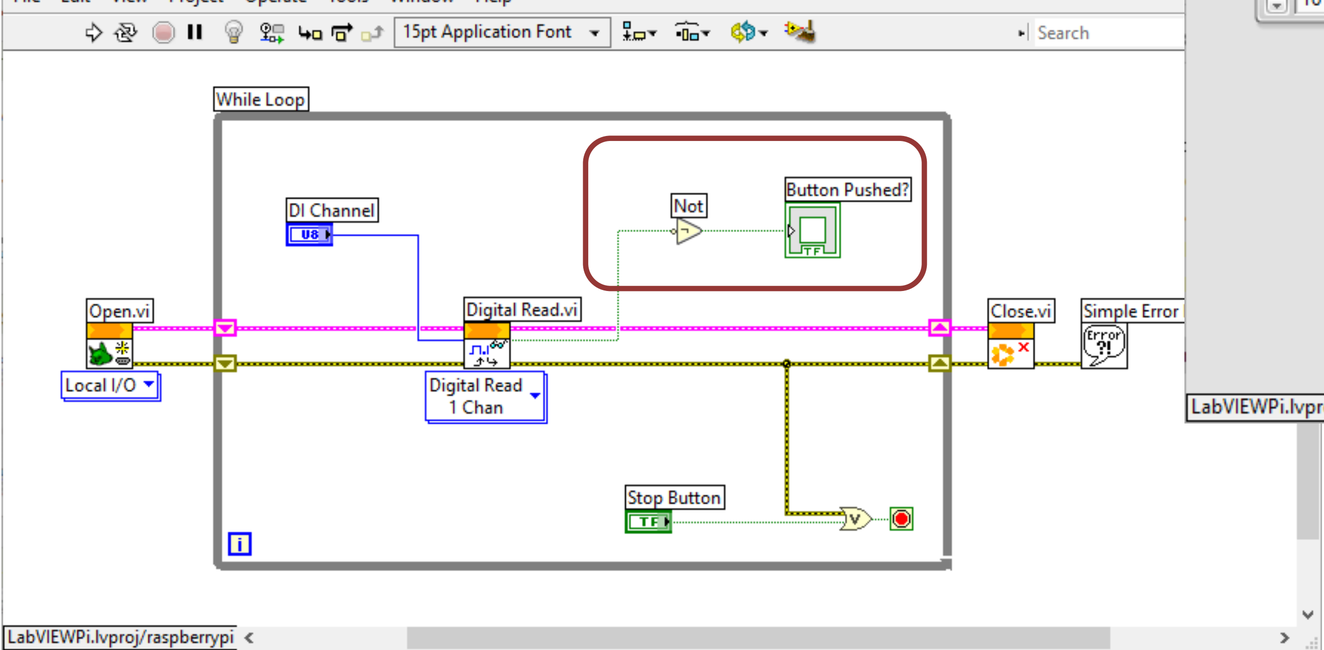
# Adding a “NOT” block

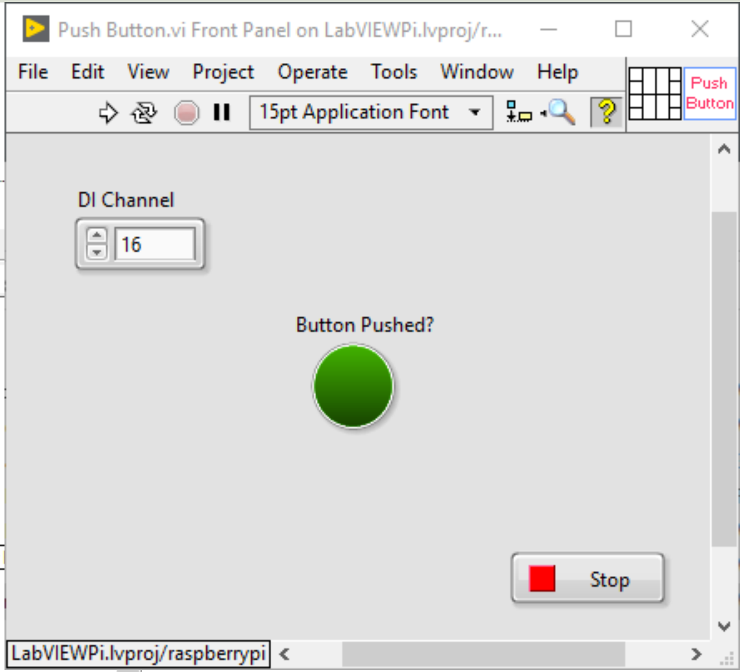


Button is NOT Pushed => False/Low



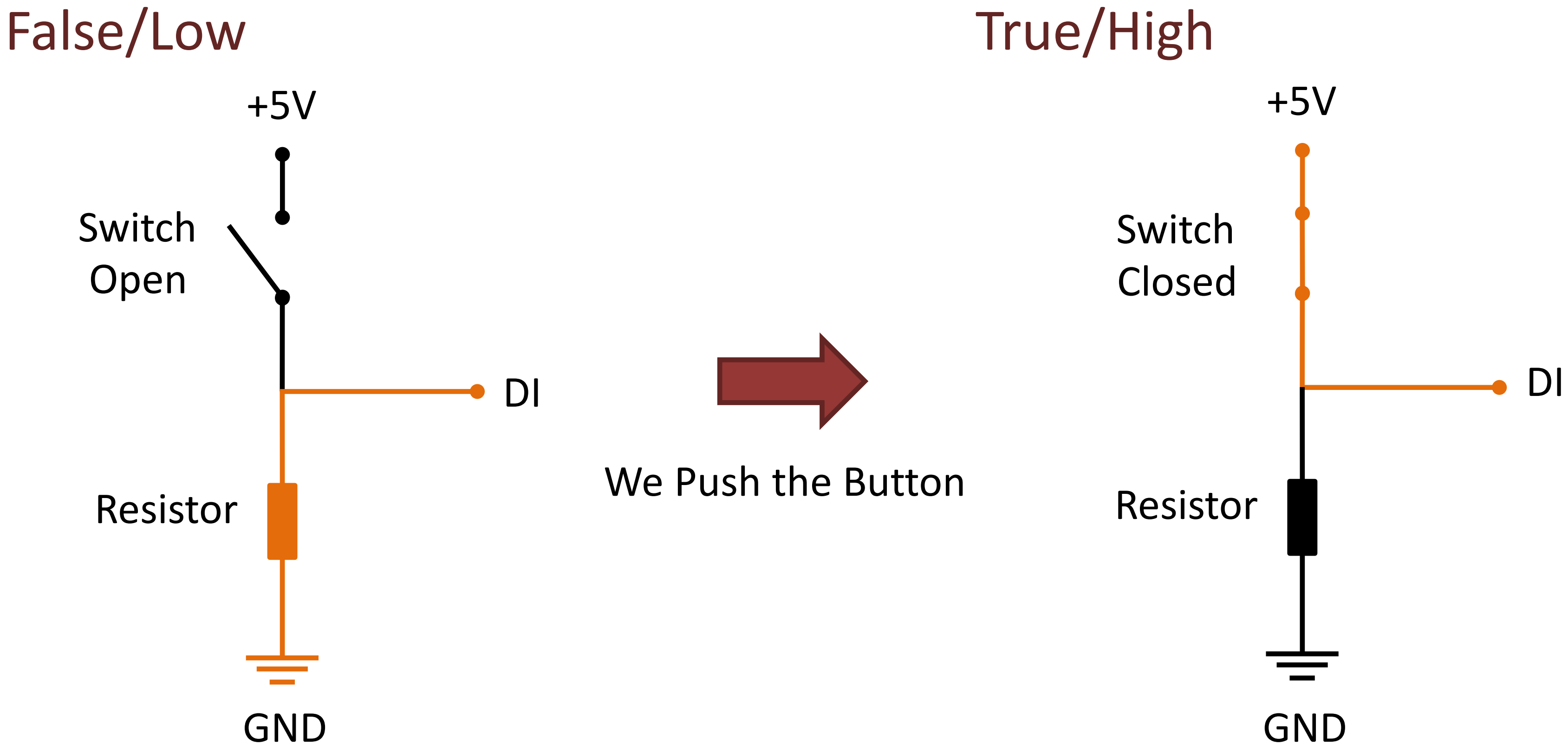
Button is Pushed => True/High



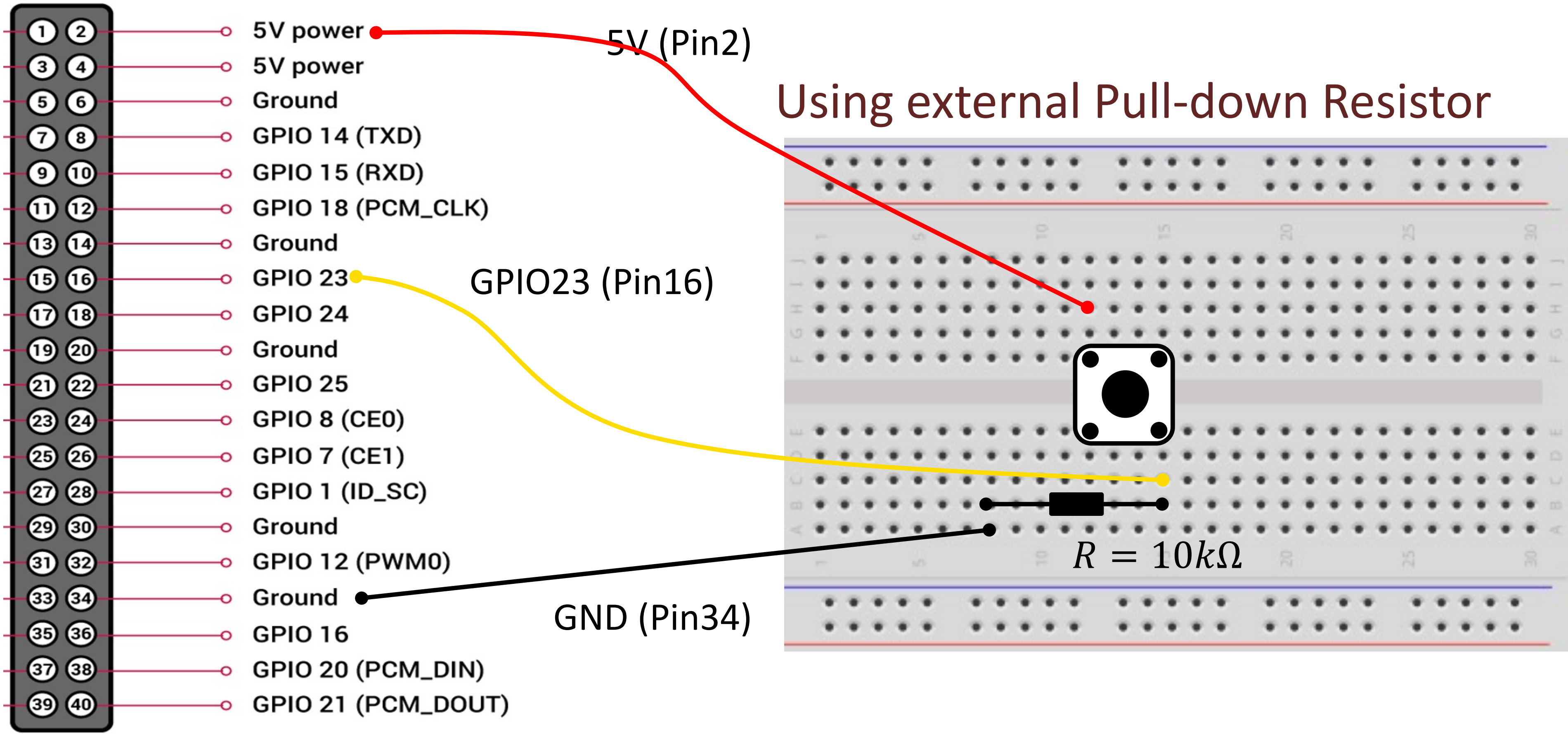



# Pull-down Resistor

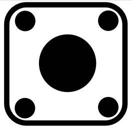
We could also have wired according to a “Pull-down“ Resistor



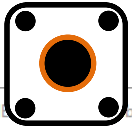
# Wiring (Pull-down Resistor)



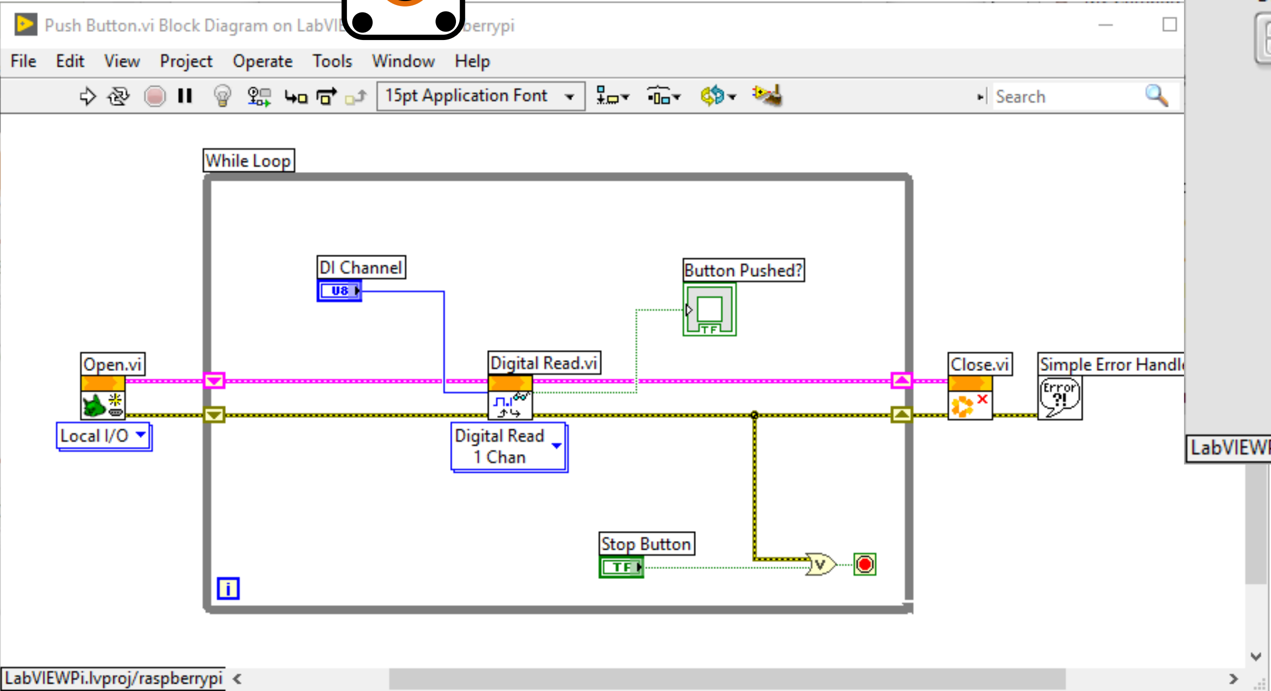
# Push Button (Pull-down Resistor)

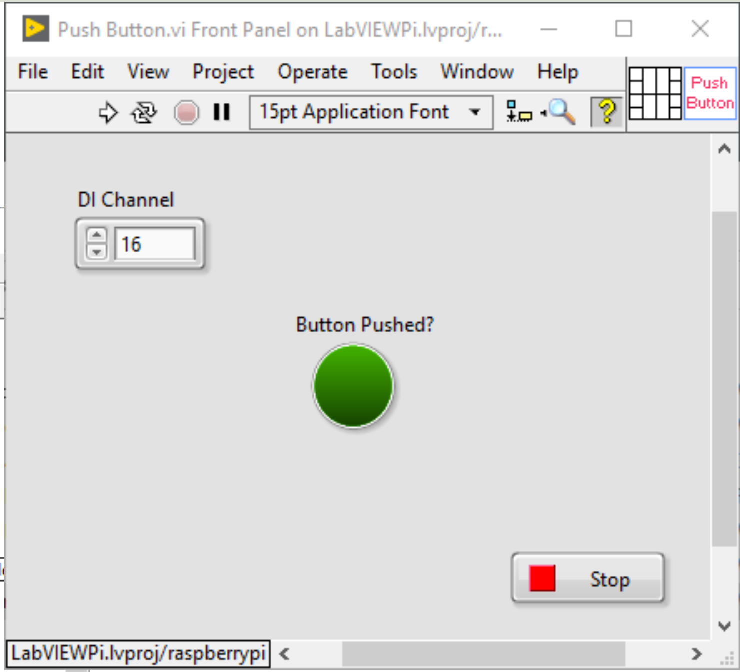


Button is NOT Pushed => False/Low



Button is Pushed => True/High

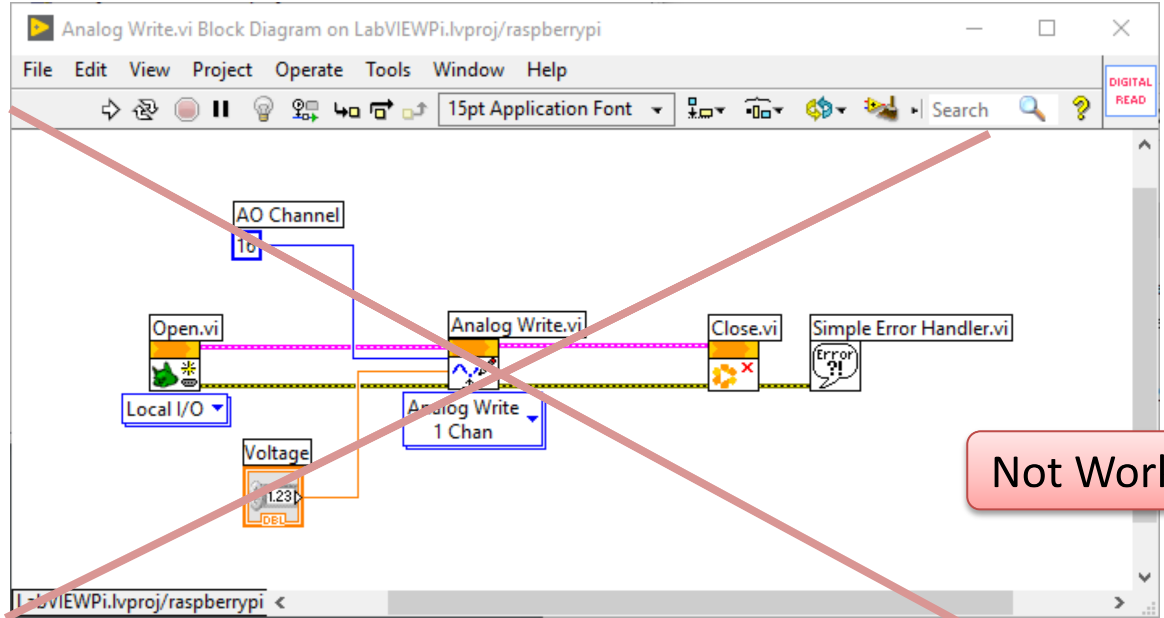


# PWM

# Analog Out

- Raspberry Pi has no Analog Out
- We need to use Pulse Width Modulation (PWM)

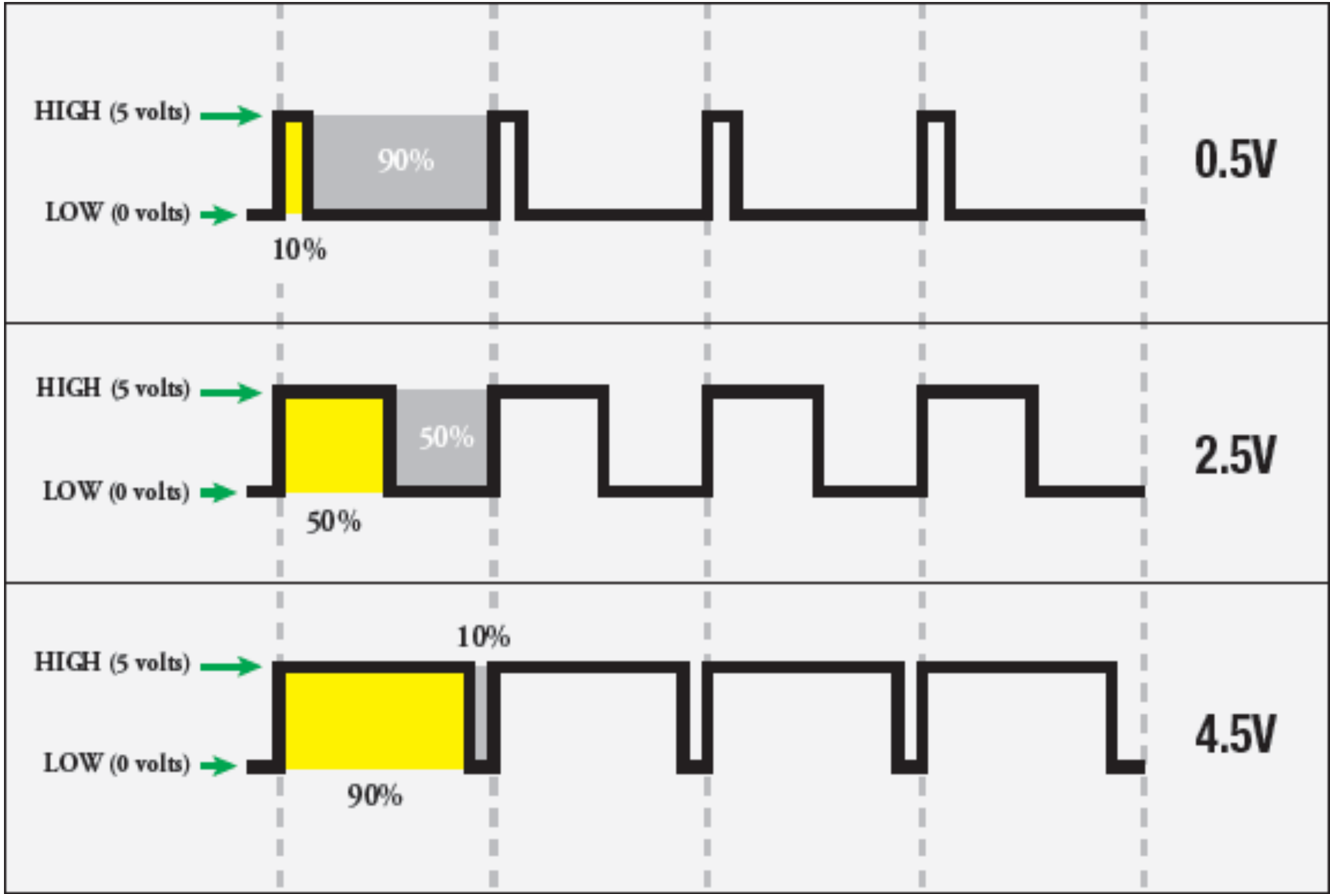



Not Working!

# PWM as "Analog Out"

The Raspberry Pi has no real Analog Out pins, but we can use a PWM pin.

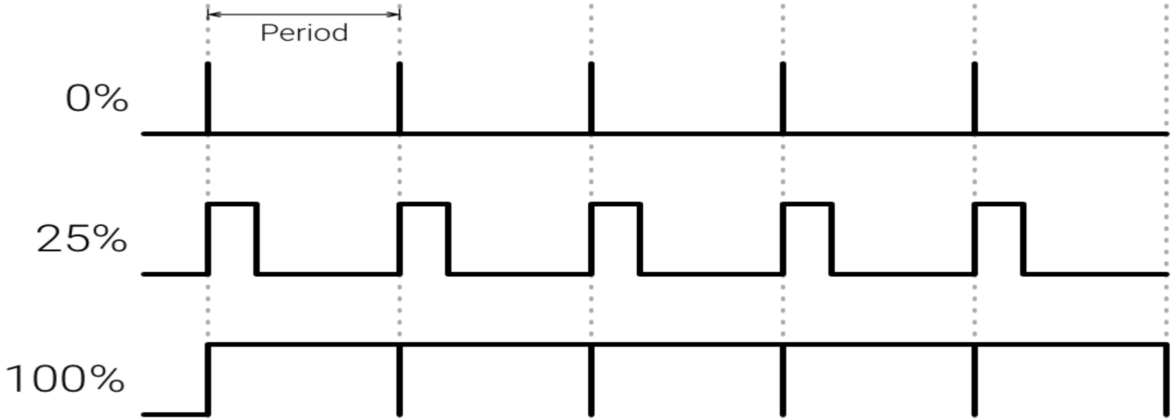
PWM can be used to control brightness of a LED, control the speed of a Fan, control a DC Motor, etc.



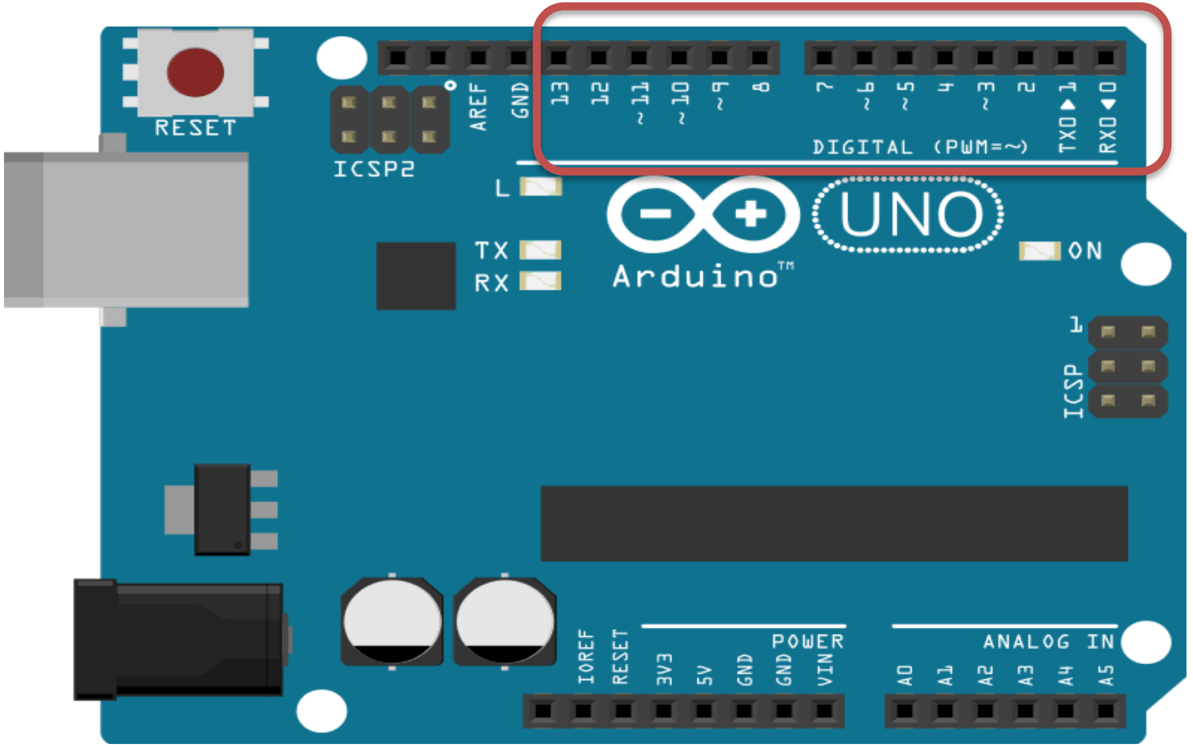
# PWM

PWM is a digital (i.e., square wave) signal that oscillates according to a given *frequency* and *duty cycle*.
The frequency (expressed in Hz) describes how often the output pulse repeats.
The period is the time each cycle takes and is the inverse of frequency.
The duty cycle (expressed as a percentage) describes the width of the pulse within that frequency window.

You can adjust the duty cycle to increase or decrease the average "on" time of the signal. The following diagram shows pulse trains at 0%, 25%, and 100% duty:



# PWM



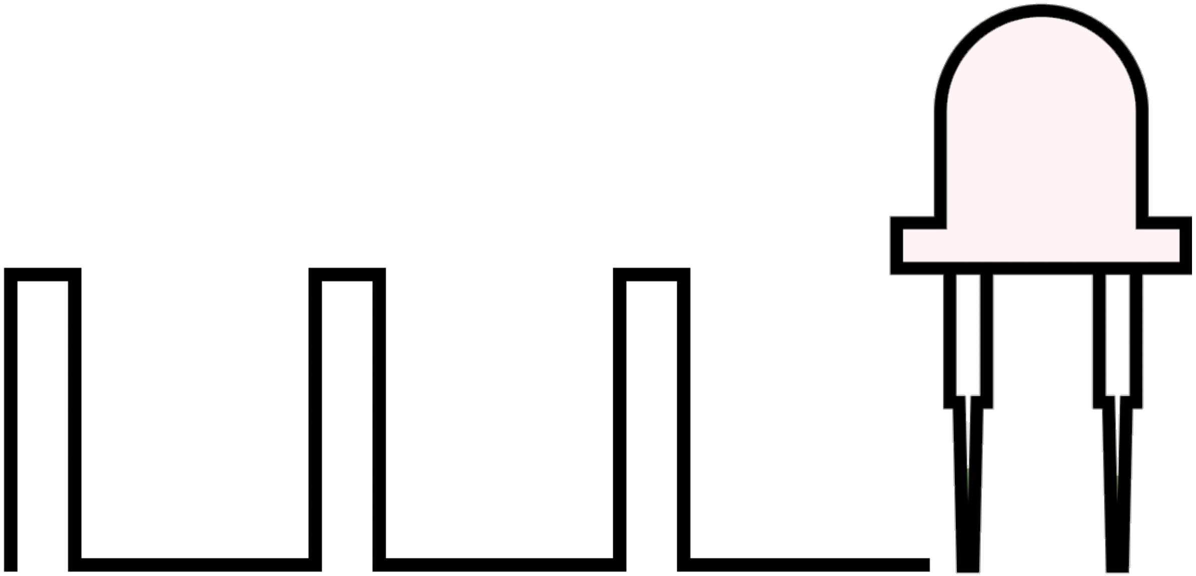

The Digital Pins marked with ~ can be used as "Analog Outputs", so-called PWM outputs

# Control Brightness of a LED

- We've seen how to turn an LED on and off, but how do we control its brightness levels?
- An LED's brightness is determined by controlling the amount of current flowing through it, but that requires a lot more hardware components.
- A simple trick we can do is to flash the LED faster than the eye can see!
- By controlling the amount of time the LED is on versus off, we can change its perceived brightness.
- This is known as *Pulse Width Modulation* (PWM).

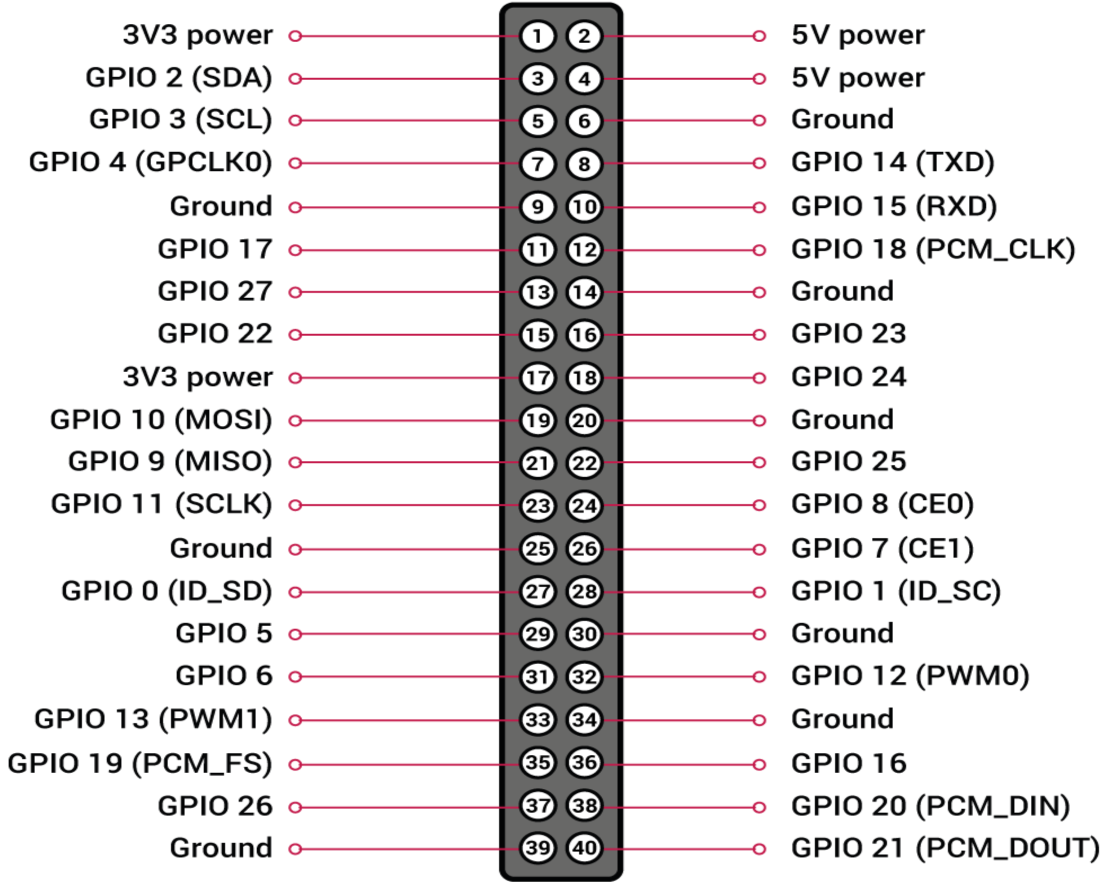
# Control Brightness of a LED

Below we see how we can use PWM to control the brightness of a LED

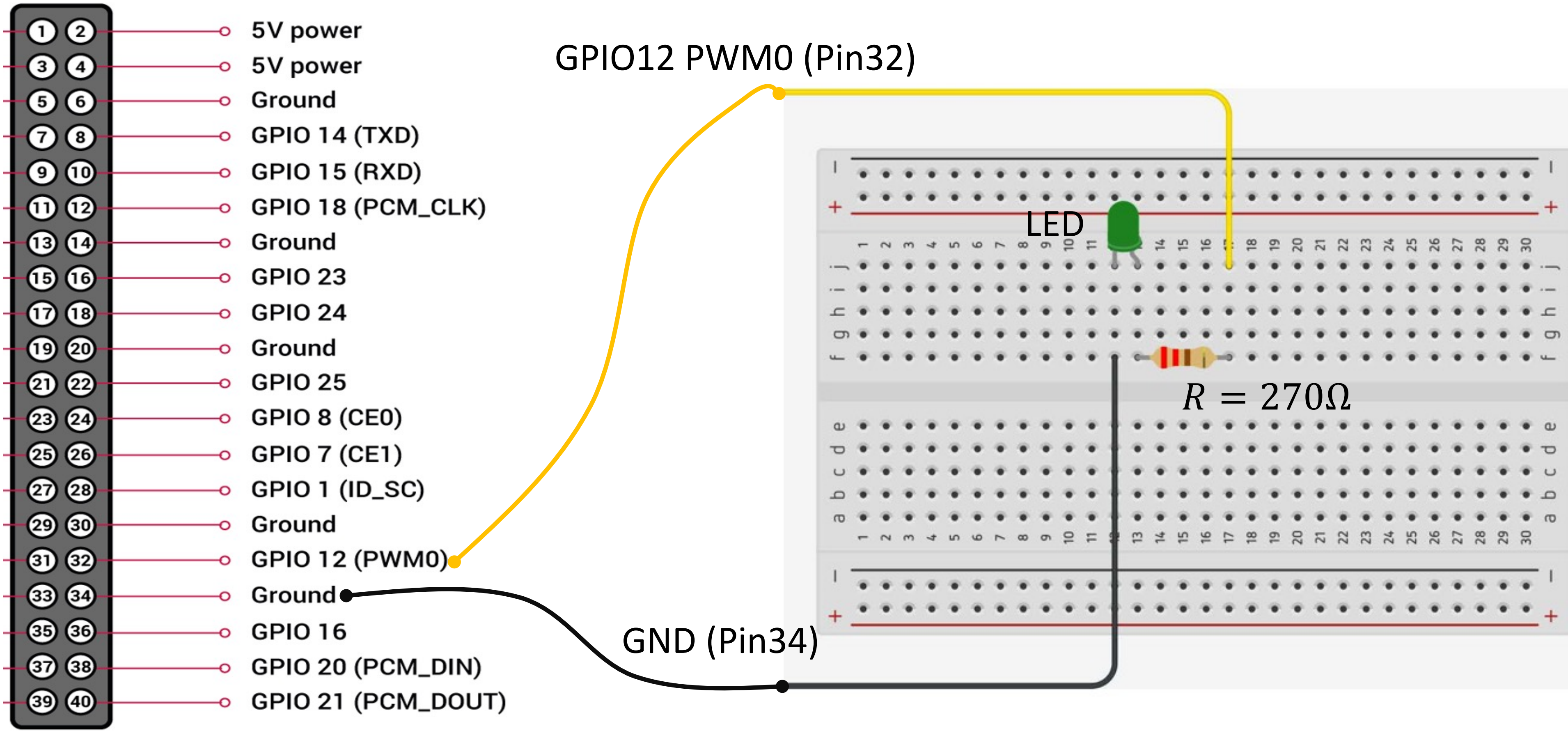


https://www.electronicwings.com/raspberry-pi/raspberry-pi-pwm-generation-using-python-and-c

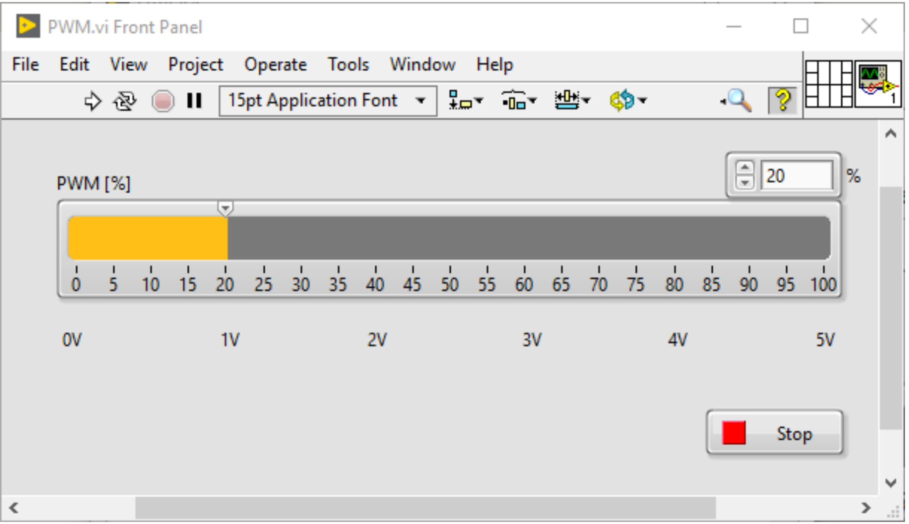
# PWM pins on Raspberry Pi



# PWM LED Wiring

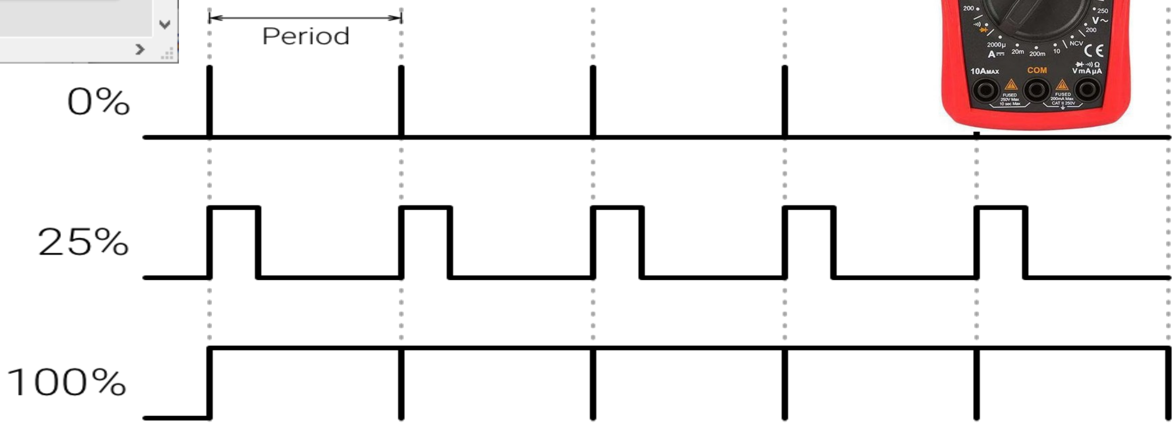


# PWM Example

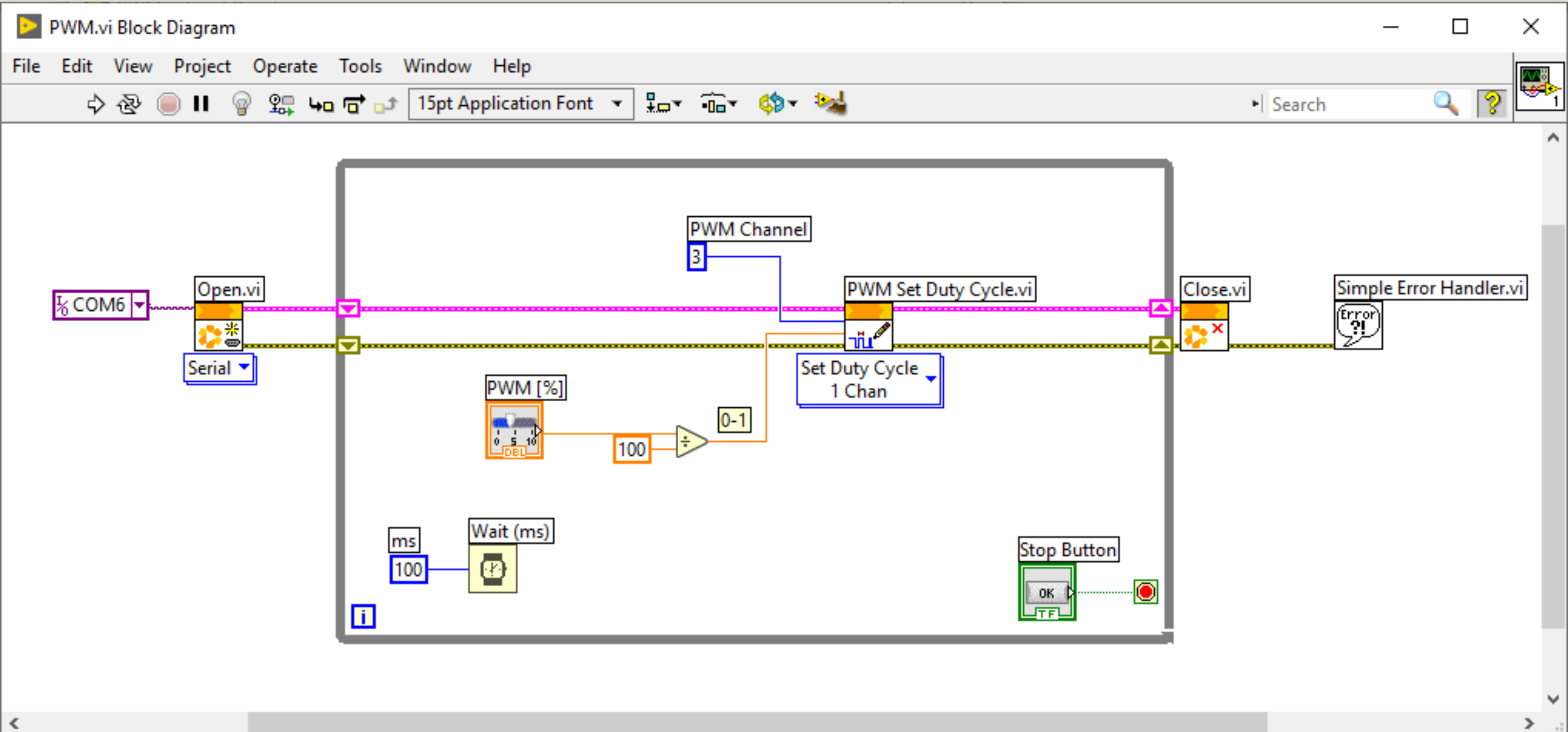



We will see the brightness of the LED will increase.

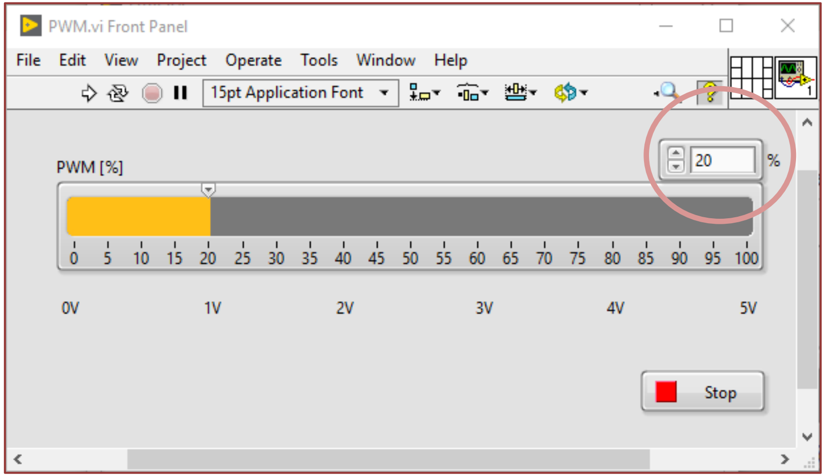
Or you can use a Multimeter and see the (average) voltage will increase

# PWM Example


PWM.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
COM6
Open.vi
Serial
PWM Channel
3
PWM Set Duty Cycle.vi
Set Duty Cycle
1 Chan
Close.vi
Simple Error Handler.vi
Error
PWM [%]
DBL
100
0-1
ms
100
Wait (ms)
Stop Button
OK
TF

# PWM Example



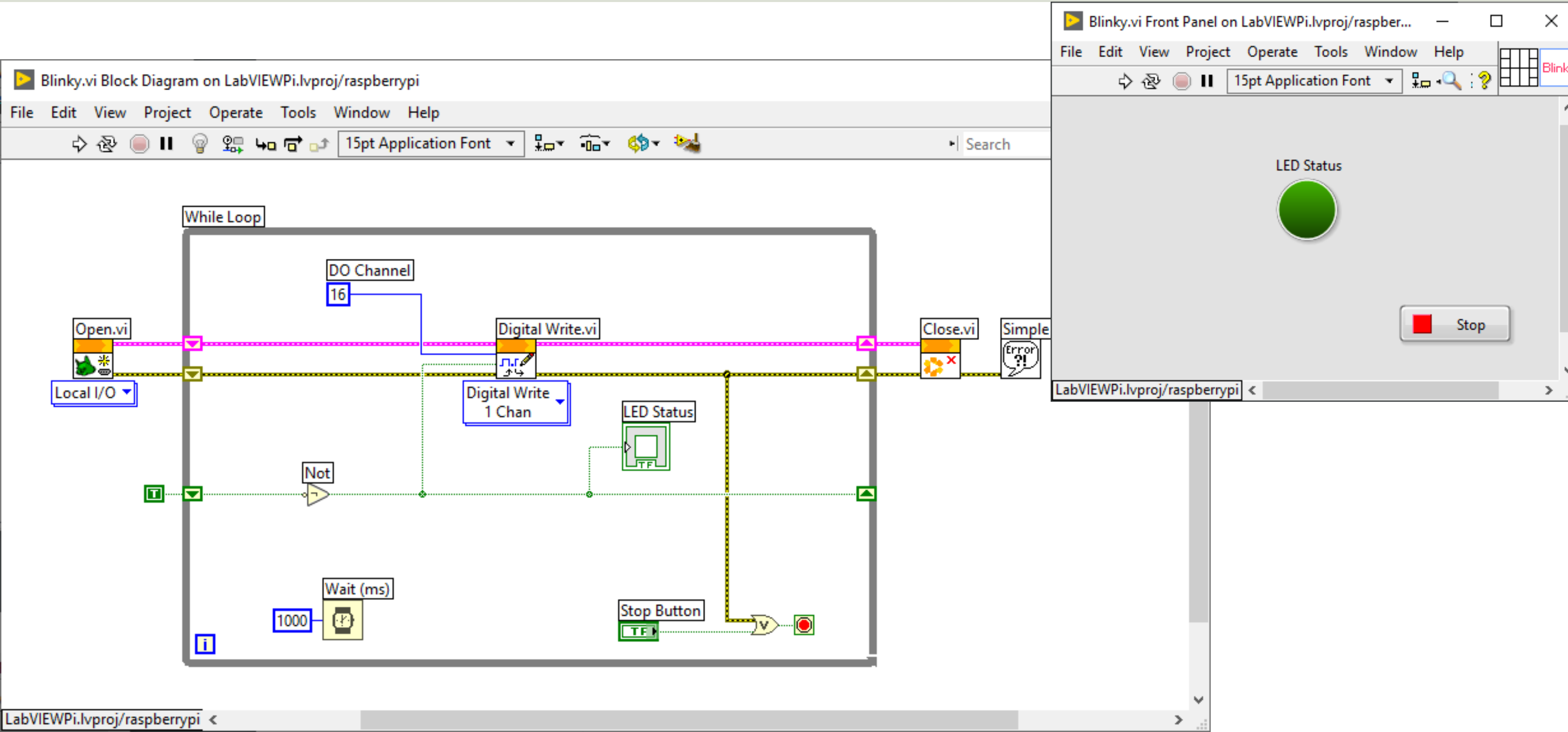

20% → 1V


0-100% → 0-5V

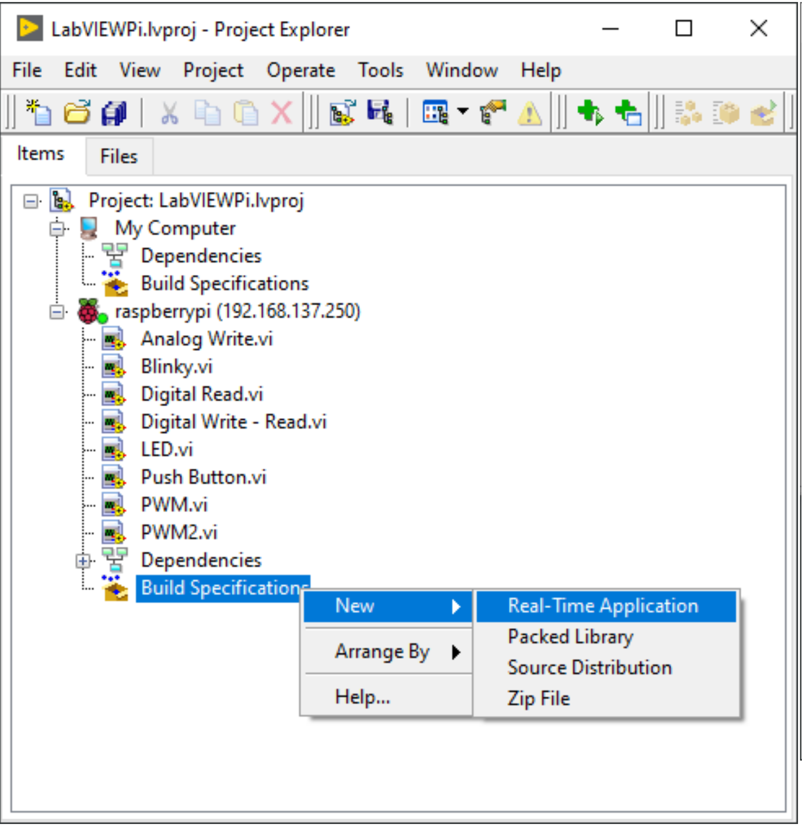
# Build and Deploy Executable LabVIEW Application

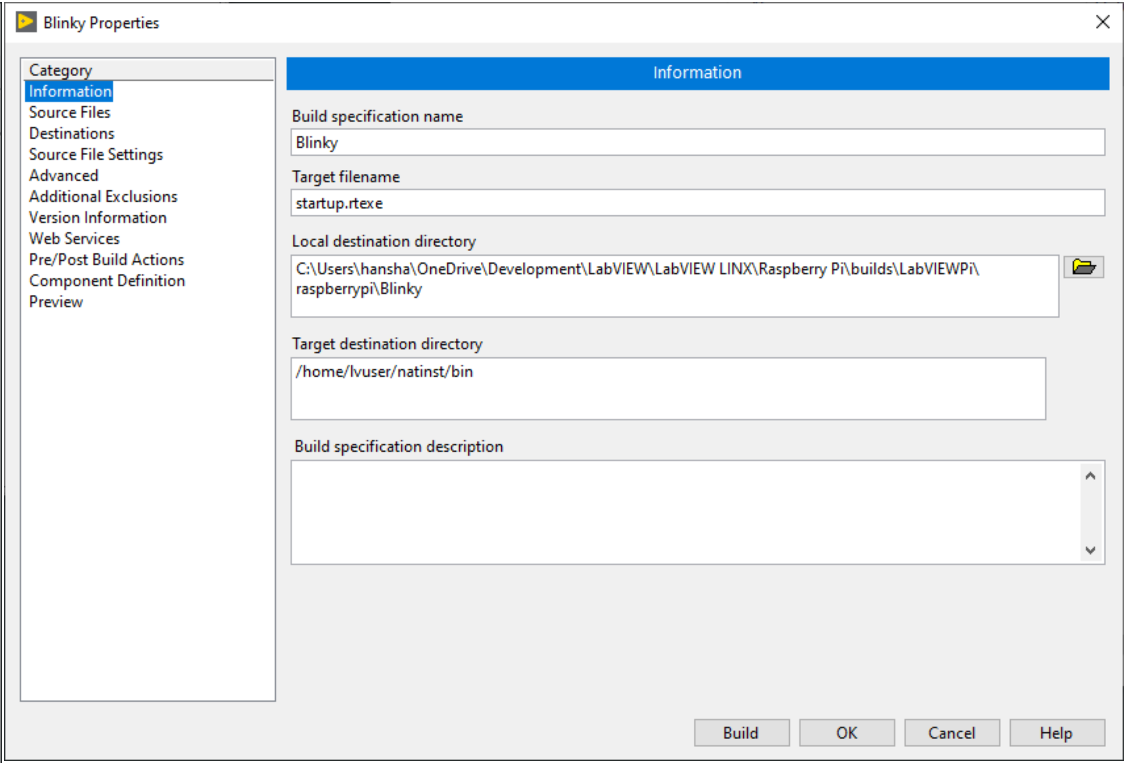
Hans-Petter Halvorsen

# Blinky Application

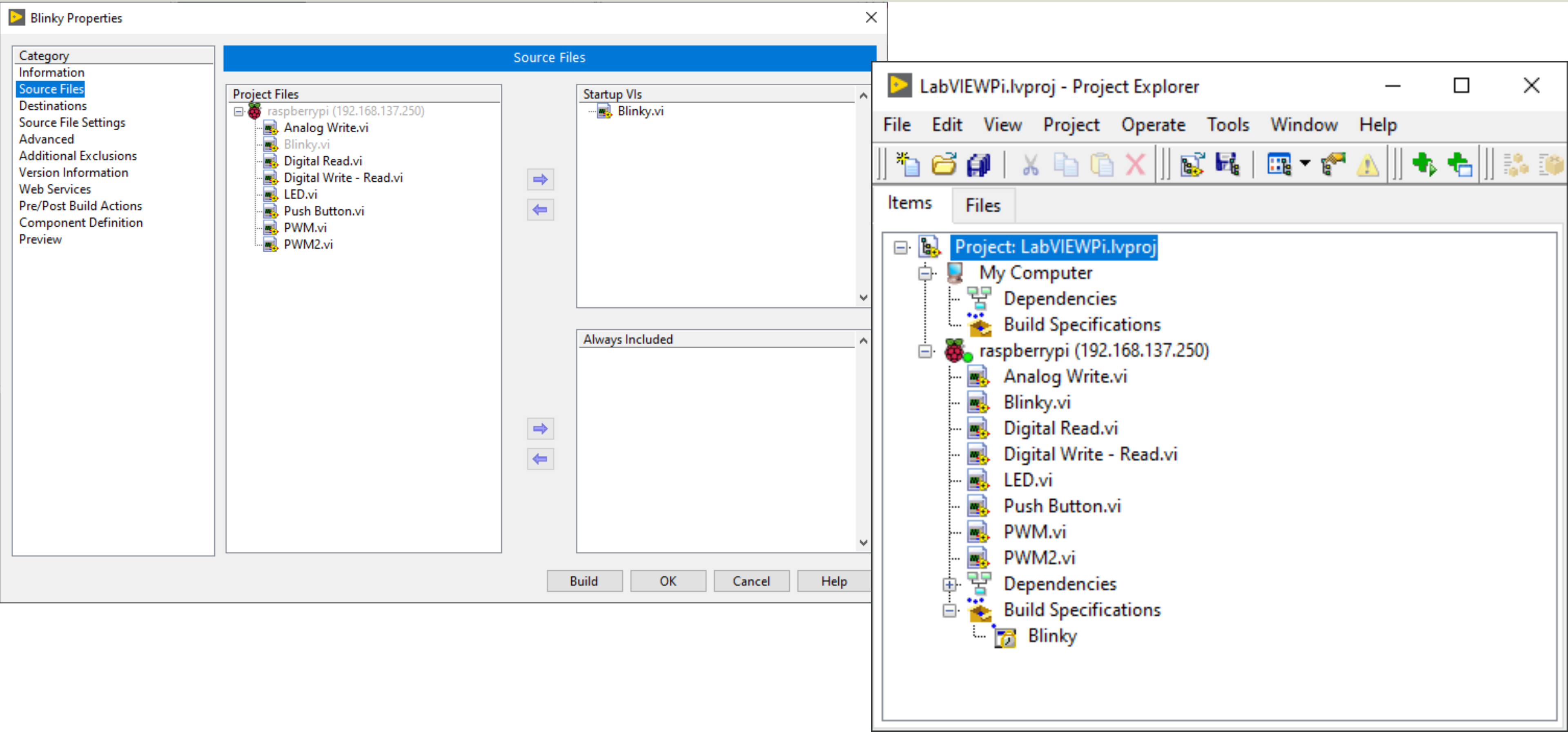


# Build Application

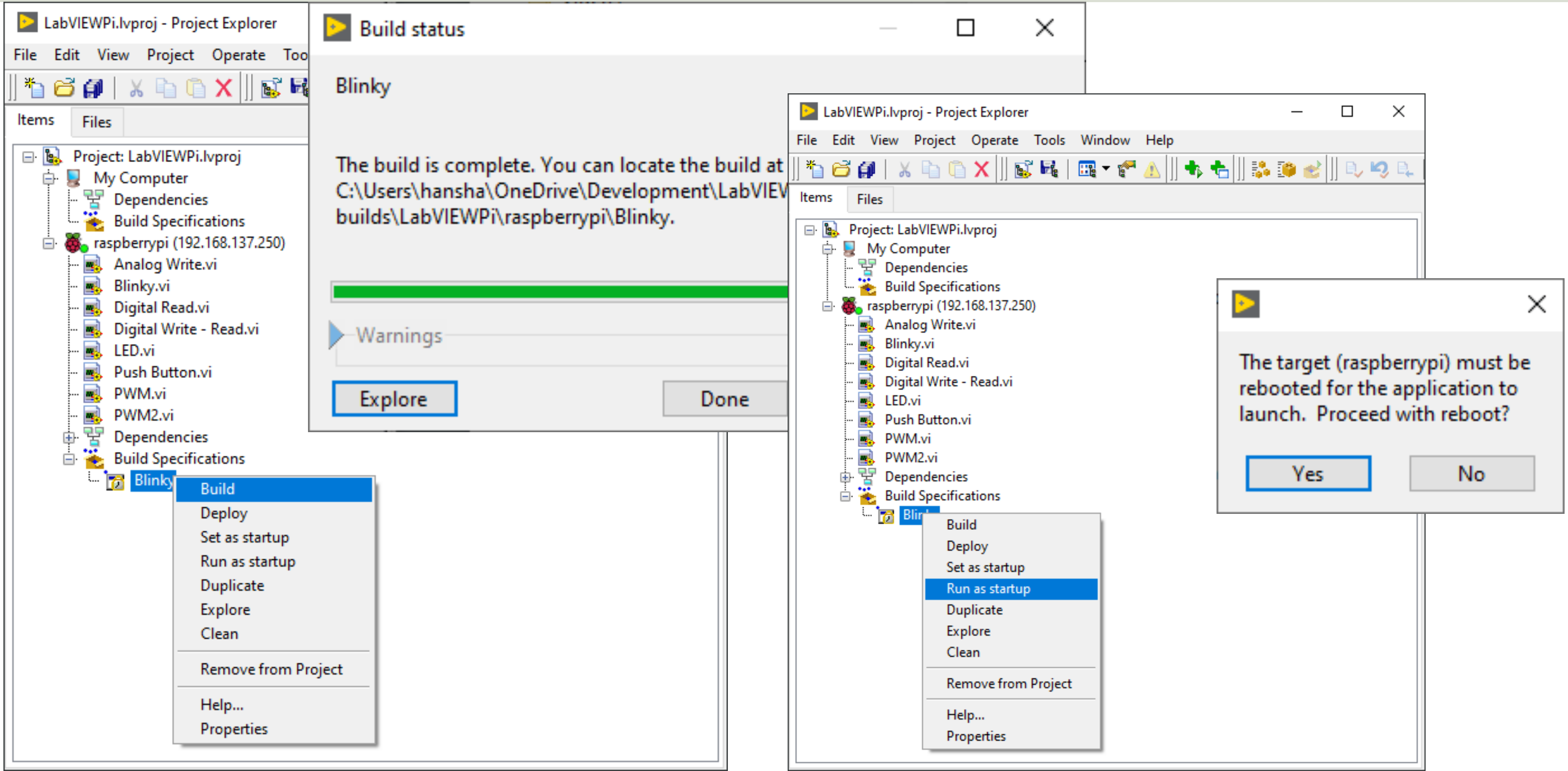


# Build Application



# Build Application



# Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog