https://www.halvorsen.blog

Raspberry Pi with MATLAB

Hans-Petter Halvorsen

Contents

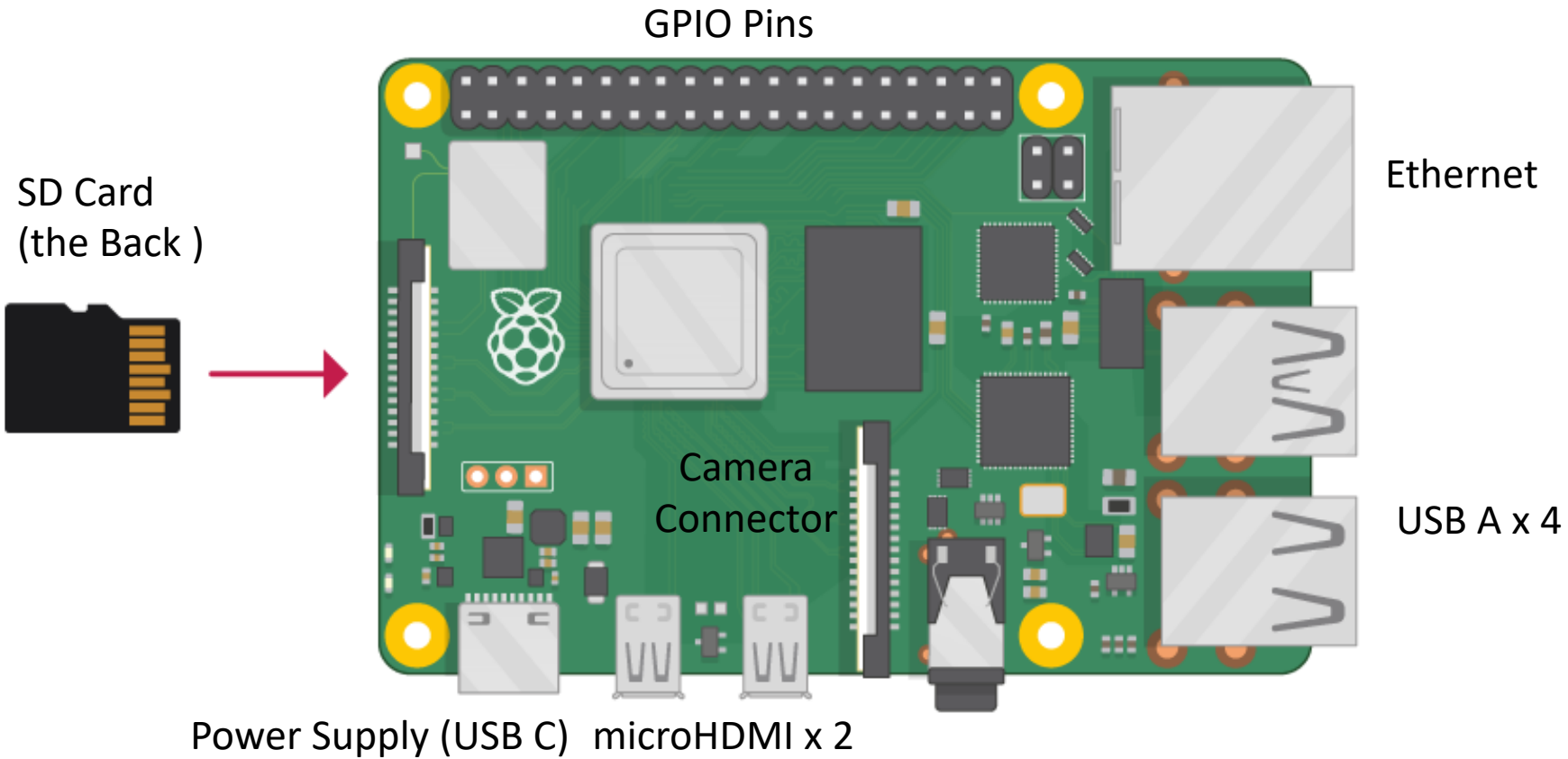
- Raspberry Pi
- MATLAB
- MATLAB Support Package for Raspberry Pi
- GPIO
- Examples
  - LED
  - PWM
  - Push Button
  - Camera

Raspberry Pi

Hans-Petter Halvorsen

Table of Contents

Raspberry Pi



Raspberry Pi

- The Raspberry Pi is a small computer that can do lots of things
- It has a small footprint (about 9x6cm) and it is cheap ($35+)
- You plug it into a monitor and attach a keyboard and mouse
- It has so-called GPIO pins (General Purpose Input/Output) for connection sensors and other electronic components like LEDs, etc.
- Raspberry Pi is as well suited for prototyping, datalogging and different electronics projects, a media center, etc.
- It can be used to learn programming, it and other technical skills, etc.
- RP has limited power (CPU, RAM, etc.) so it cannot normally replace a desktop computer or laptop for ordinary use

https://www.raspberrypi.org

MATLAB

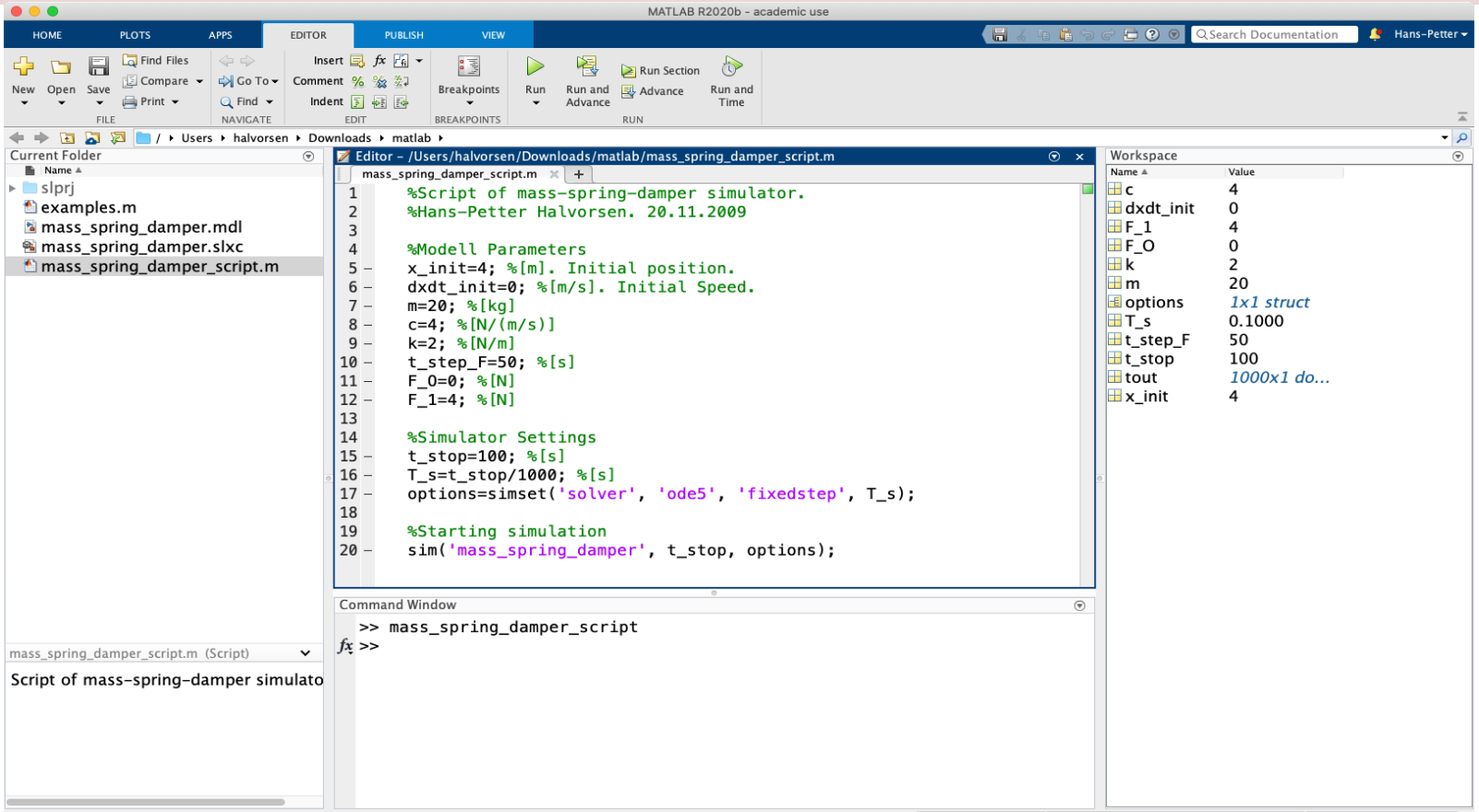
Hans-Petter Halvorsen

Table of Contents

MATLAB

- MATLAB is a tool for technical computing, computation and visualization in an integrated environment.
- MATLAB is an abbreviation for MATrix LABoratory
- It is well suited for matrix manipulation and problem solving related to Linear Algebra, Modelling, Simulation and Control applications, etc.
- MATLAB is popular in Universities, Teaching and Research and Development (R&D)
- www.mathworks.com

MATLAB


MATLAB R2020b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Hans-Petter
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
/ ▸ Users ▸ halvorsen ▸ Downloads ▸ matlab ▸
Current Folder
Name
slprj
examples.m
mass_spring_damper.mdl
mass_spring_damper.slxc
mass_spring_damper_script.m
mass_spring_damper_script.m (Script)
Script of mass-spring-damper simulato
Editor - /Users/halvorsen/Downloads/matlab/mass_spring_damper_script.m
mass_spring_damper_script.m
1 %Script of mass-spring-damper simulator.
2 %Hans-Petter Halvorsen. 20.11.2009
3
4 %Modell Parameters
5 x_init=4; %[m]. Initial position.
6 dxdt_init=0; %[m/s]. Initial Speed.
7 m=20; %[kg]
8 c=4; %[N/(m/s)]
9 k=2; %[N/m]
10 t_step_F=50; %[s]
11 F_O=0; %[N]
12 F_1=4; %[N]
13
14 %Simulator Settings
15 t_stop=100; %[s]
16 T_s=t_stop/1000; %[s]
17 options=simset('solver', 'ode5', 'fixedstep', T_s);
18
19 %Starting simulation
20 sim('mass_spring_damper', t_stop, options);
Command Window
>> mass_spring_damper_script
>>
Workspace
Name
Value
c 4
dxdt_init 0
F_1 4
F_O 0
k 2
m 20
options 1x1 struct
T_s 0.1000
t_step_F 50
t_stop 100
tout 1000x1 do...
x_init 4
UTF-8
Ln 1
Col 1

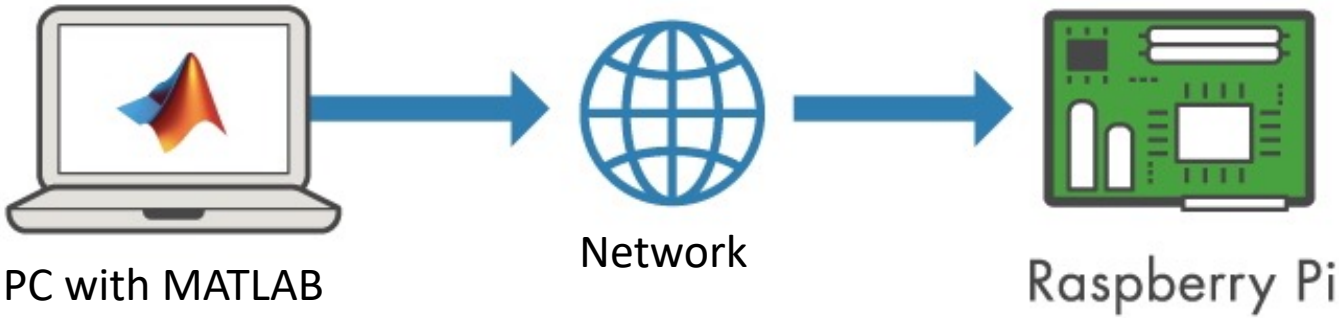
MATLAB Support Package for Raspberry Pi

Hans-Petter Halvorsen

Table of Contents

Raspberry Pi + MATLAB

MATLAB Support Package for Raspberry Pi

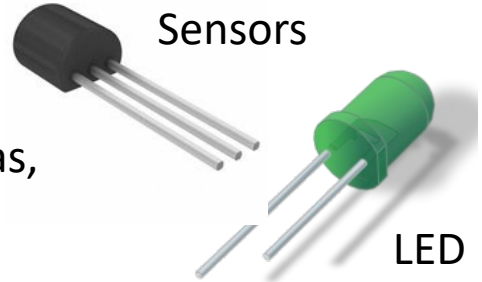


With **MATLAB Support Package for Raspberry Pi**, the Raspberry Pi is connected to a computer running MATLAB. Processing is done on the computer with MATLAB.

https://mathworks.com/hardware-support/raspberry-pi-matlab.html

Raspberry Pi + MATLAB

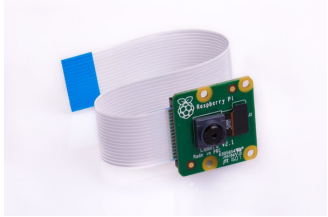
We can read data from Sensors, Cameras, control LEDs, etc. from our MATLAB environment on the Desktop Computer



Sensors

LED

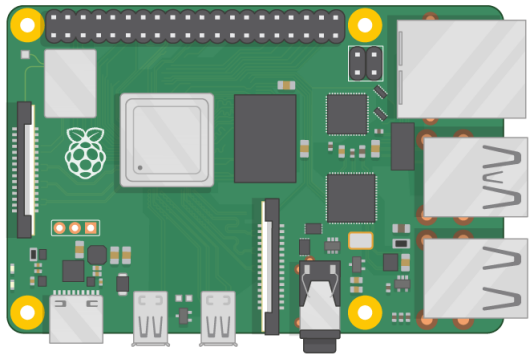
Push Button



Camera

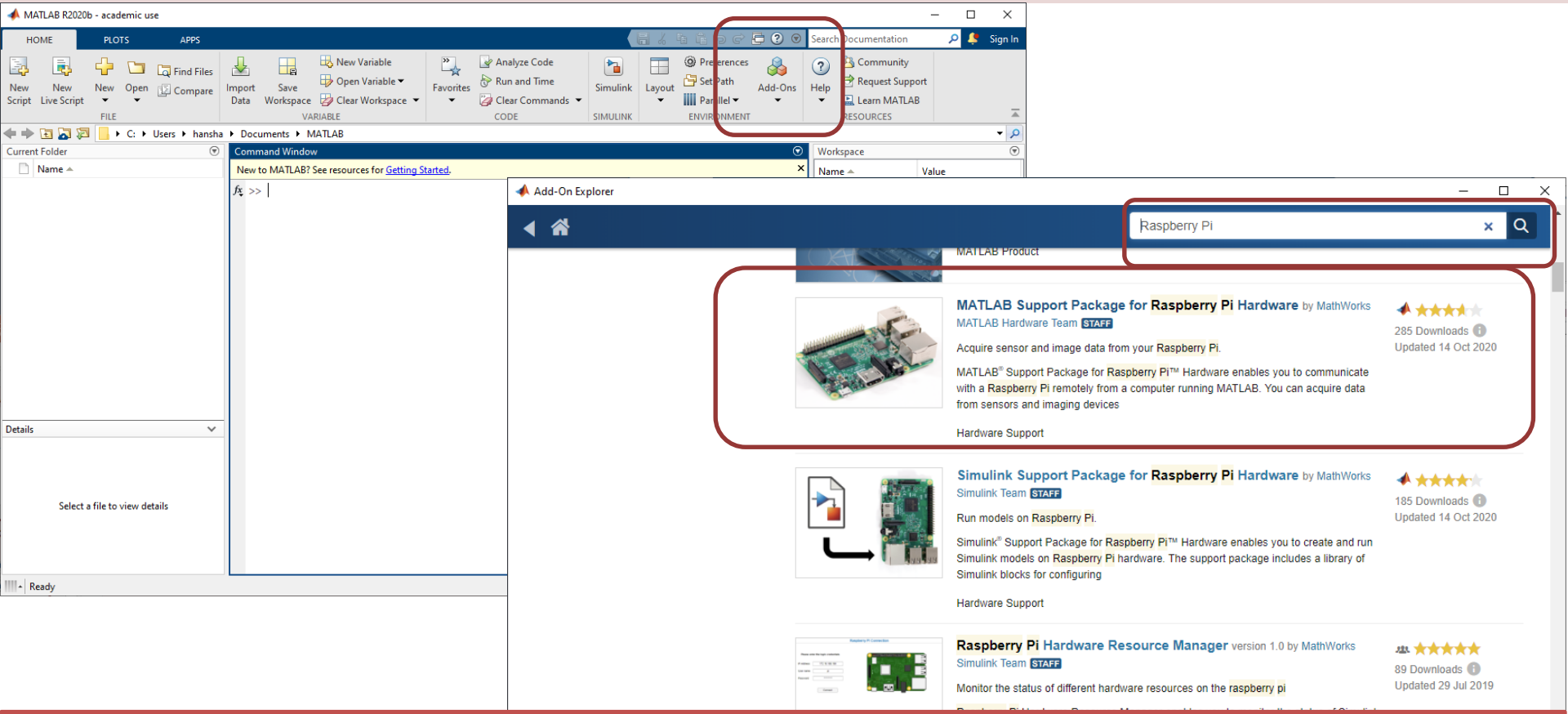
GPIO Pins

PC



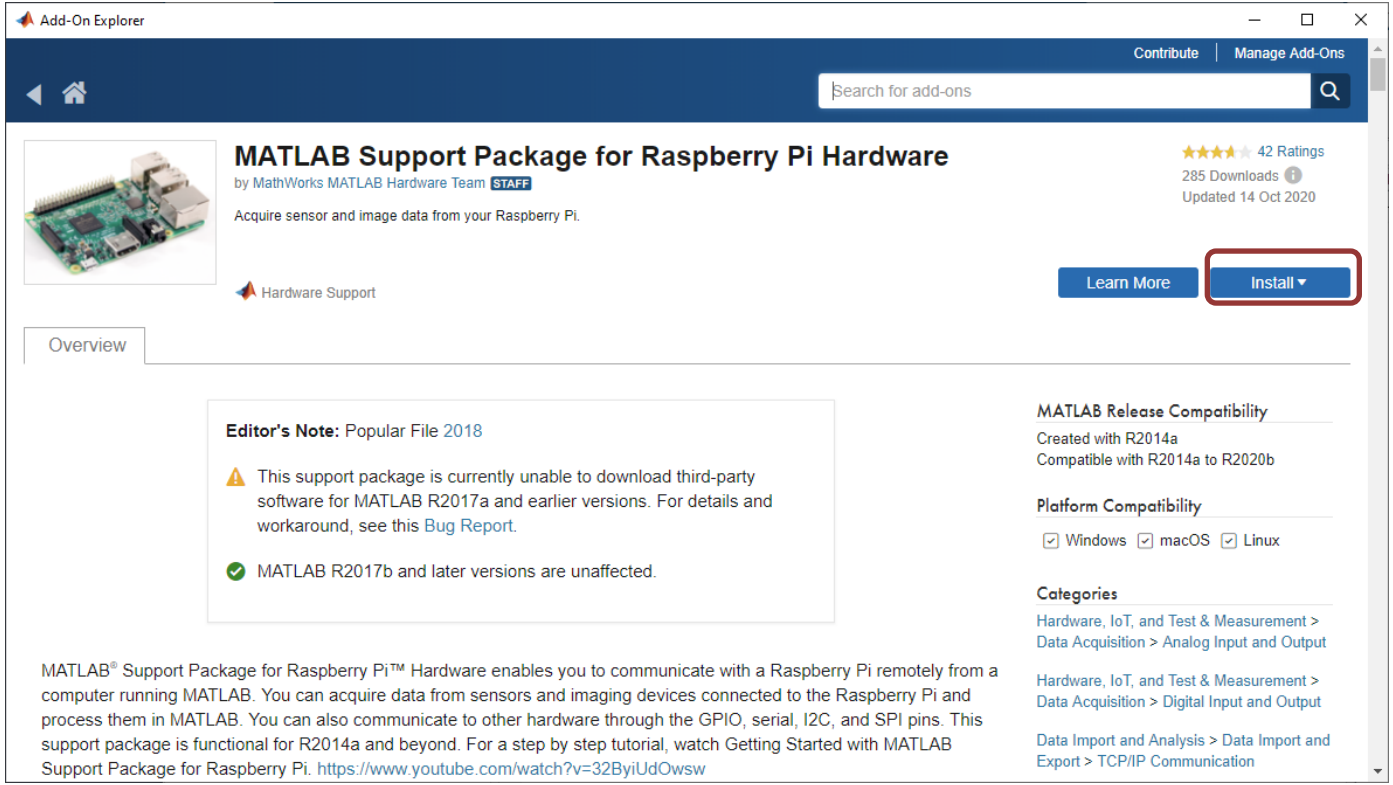
Raspberry Pi

MATLAB Support Package for Raspberry Pi



Getting Started with MATLAB Support Package for Raspberry Pi: https://youtu.be/32ByiUdOwsw

MATLAB Support Package for Raspberry Pi



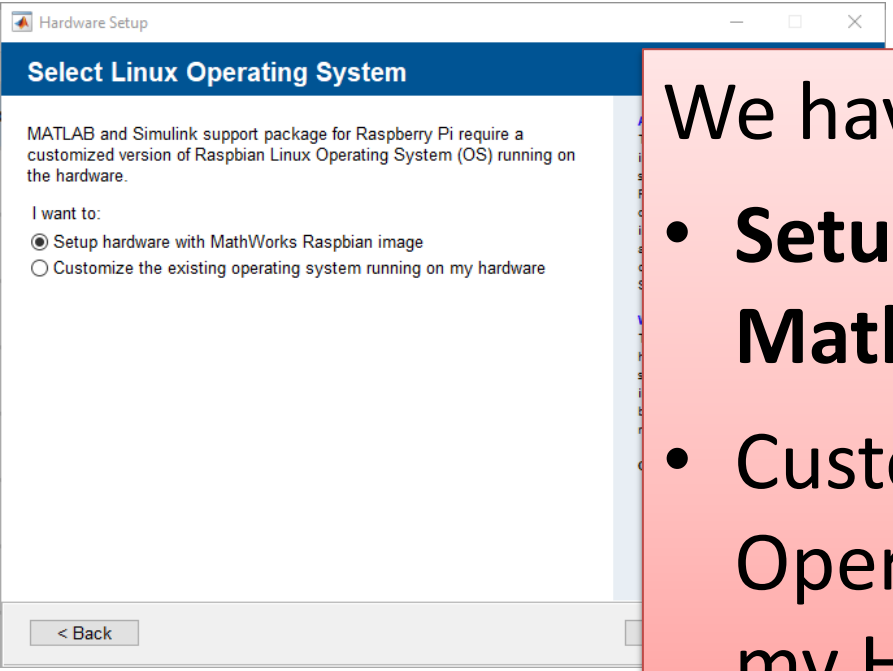
Getting Started with MATLAB Support Package for Raspberry Pi: https://youtu.be/32ByiUdOwsw

Hardware Setup

Hans-Petter Halvorsen

Table of Contents

Hardware Setup



We have 2 options/alternatives:

- **Setup Hardware with MathWorks Raspbian Image**
- Customize the existing Operating System running on my Hardware

Hardware Setup Alternatives

- Setup Hardware with MathWorks Raspbian Image
- Customize the existing Operating System running on my Hardware

Hardware Setup

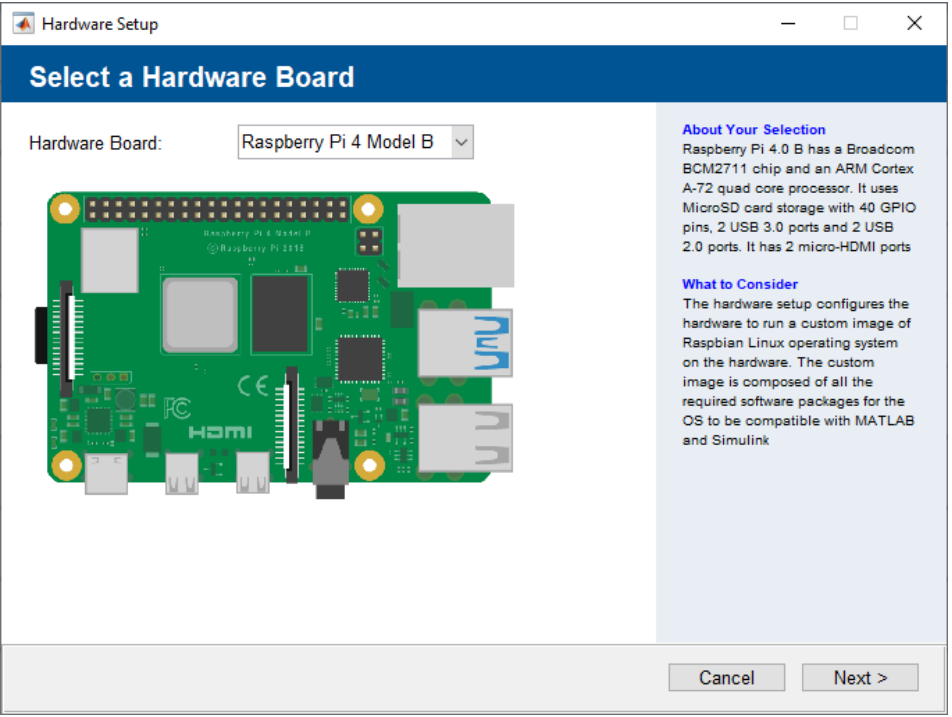
Setup Hardware with MathWorks Raspbian Image

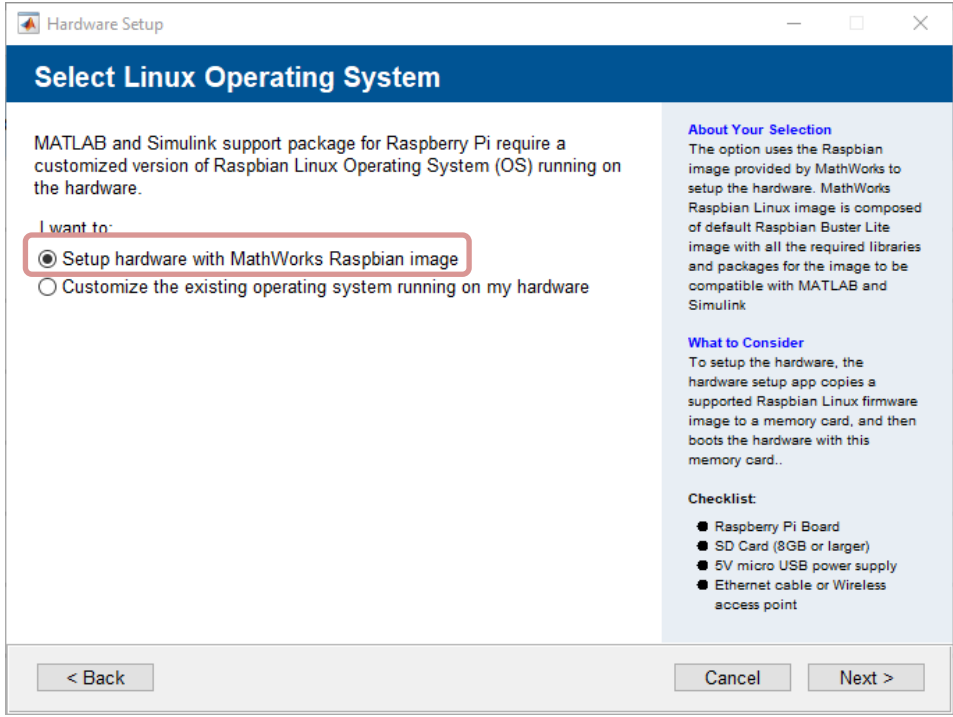
Hans-Petter Halvorsen

Table of Contents

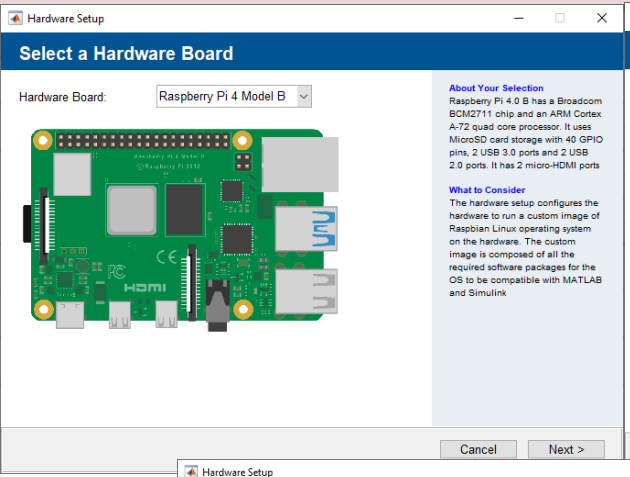
Hardware Setup

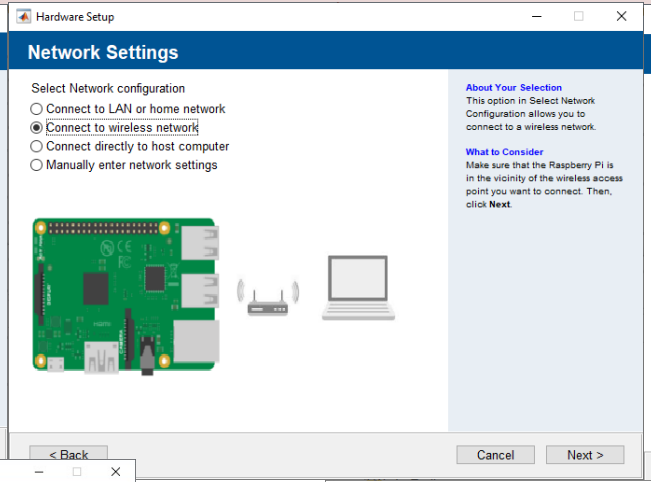
Setup Hardware with MathWorks Raspbian Image

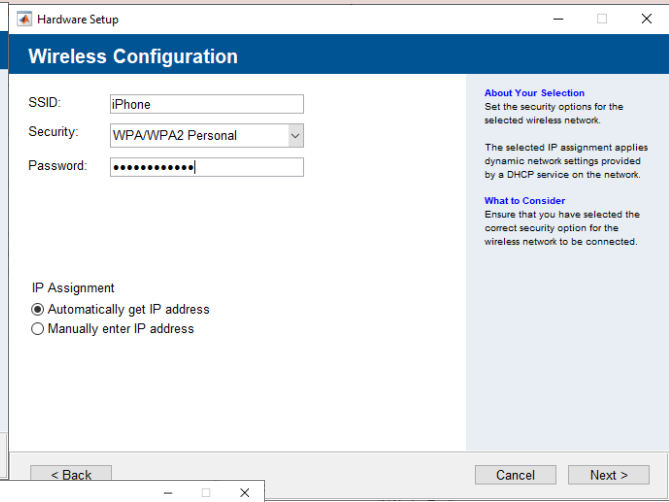


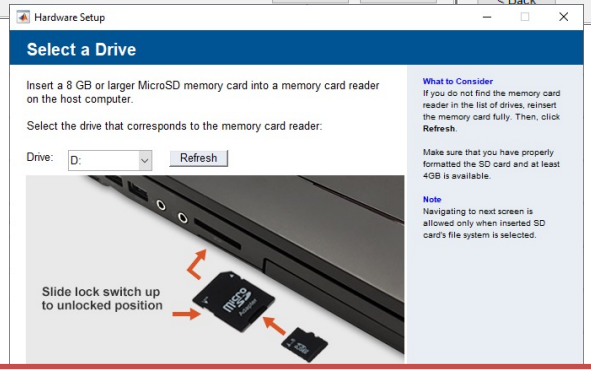


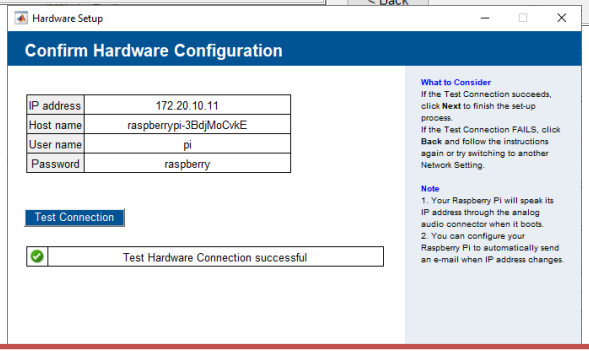
Hardware Setup





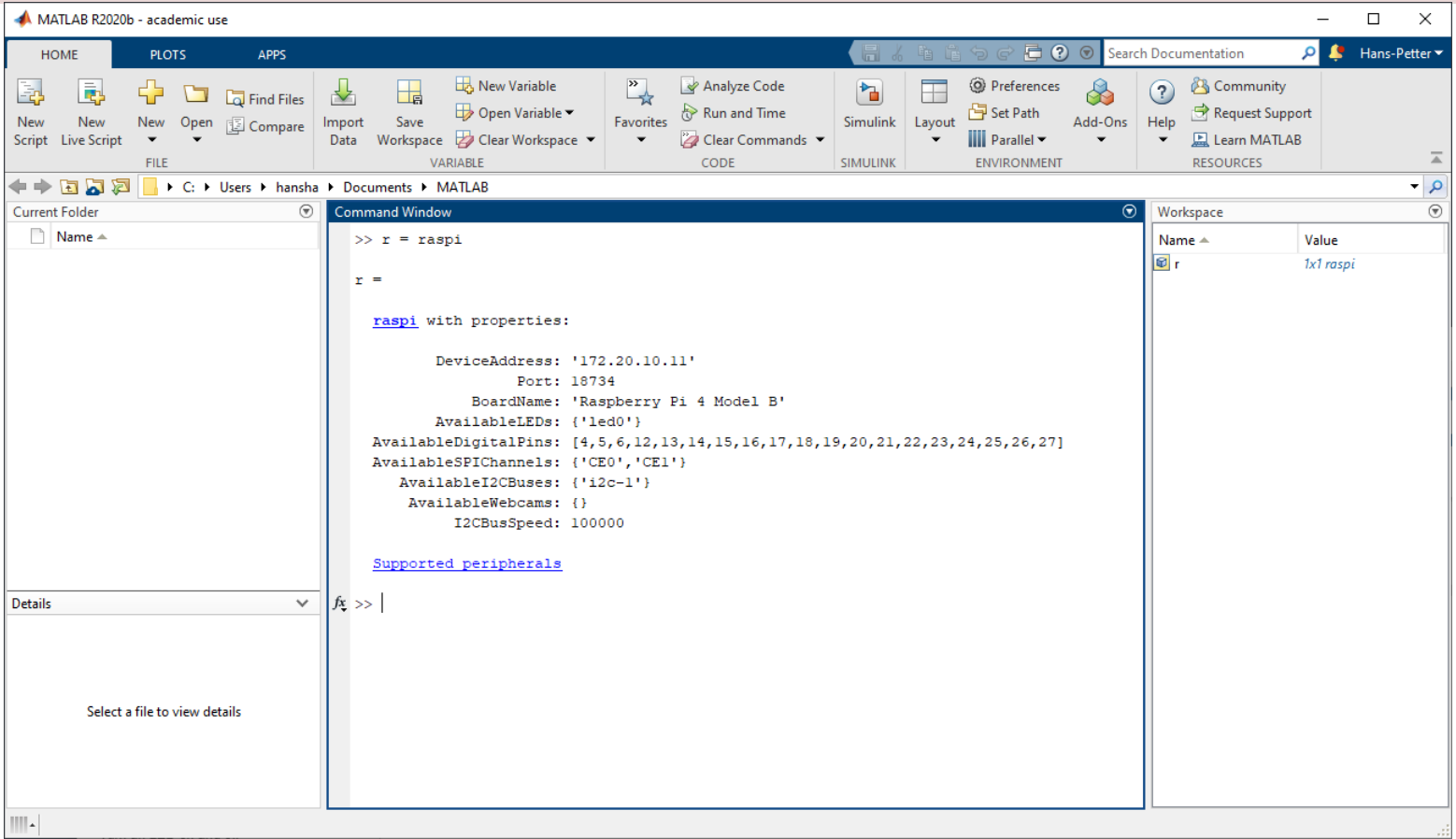




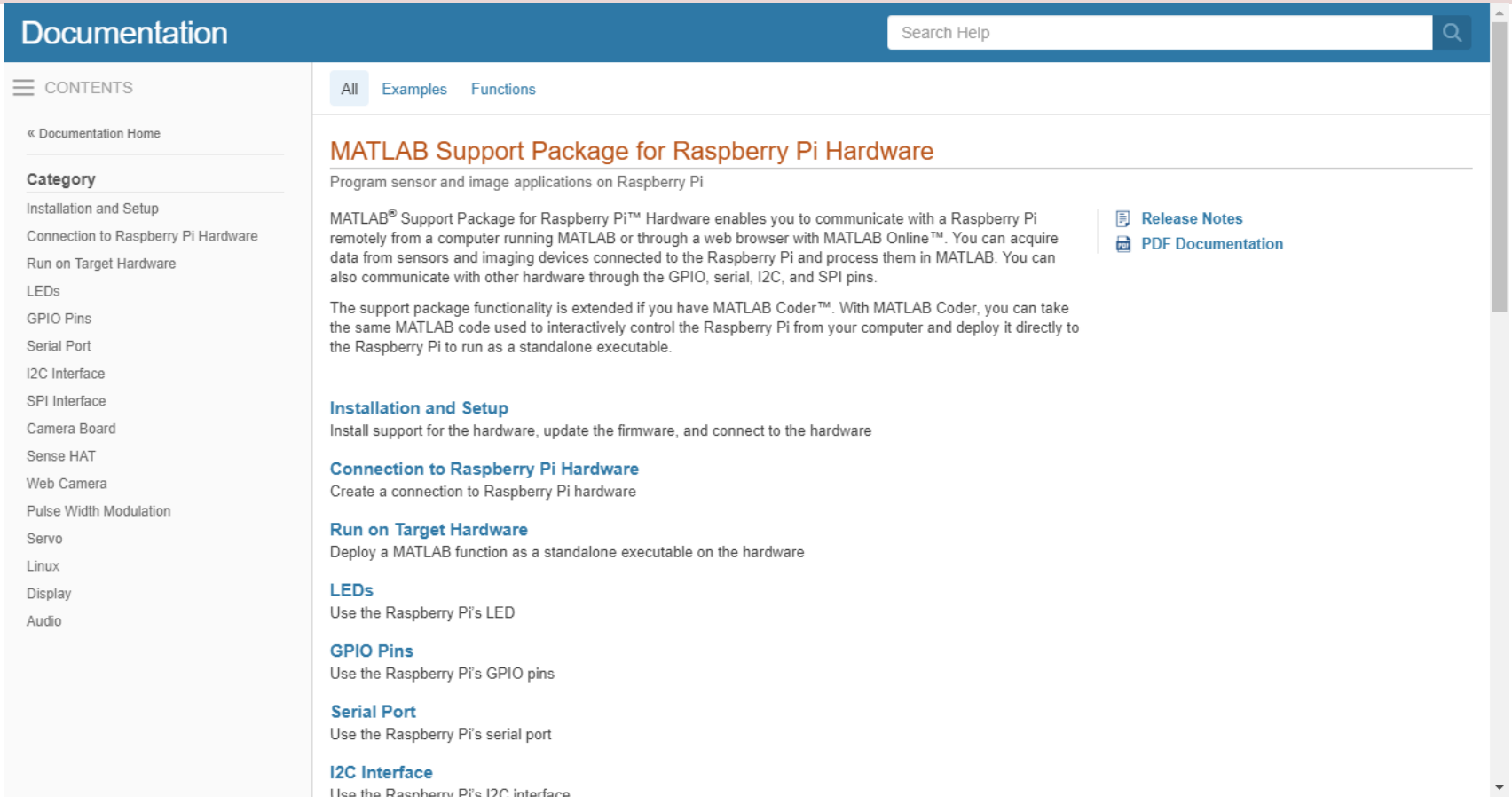


Getting Started with MATLAB Support Package for Raspberry Pi: https://youtu.be/32ByiUdOwsw

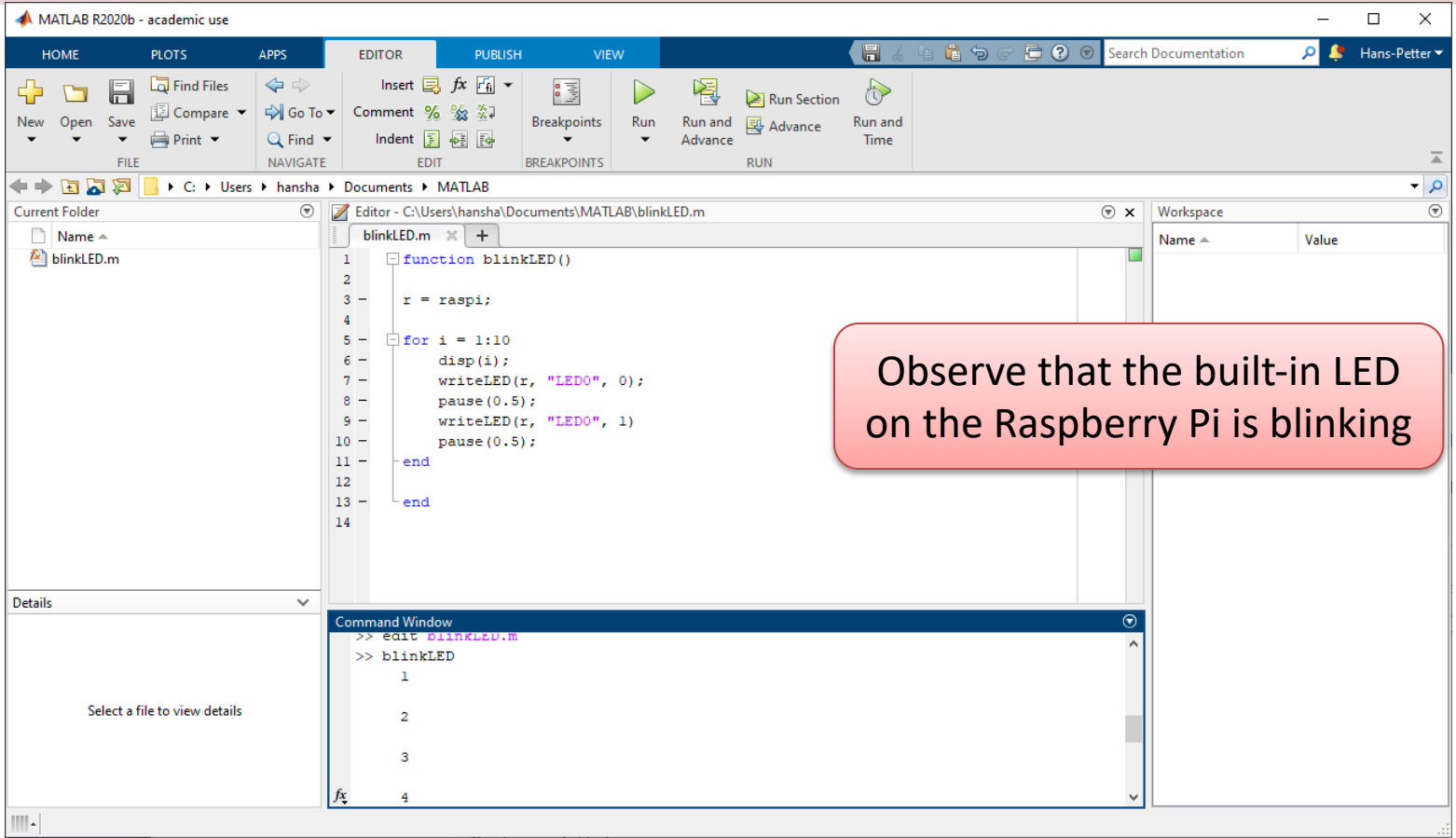
Test Hardware



Documentation and Examples



Blinking LED Example



Blinking LED Example

WE use the following Function:

```
writeLED(r, "LED0", 1);
```

```
clear
clc

r = raspi;

for i = 1:10
    disp(i);
    writeLED(r, "LED0", 0);
    pause(0.5);
    writeLED(r, "LED0", 1)
    pause(0.5);
end
```

Hardware Setup - Alternative 2

Hardware Setup

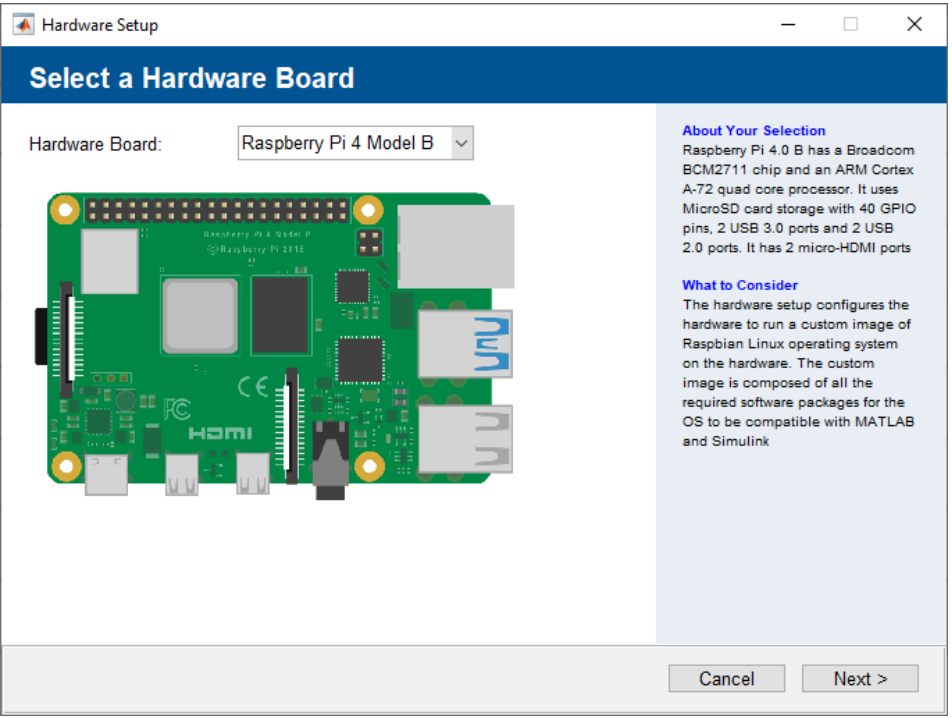
Customize the existing Operating System running on my Hardware

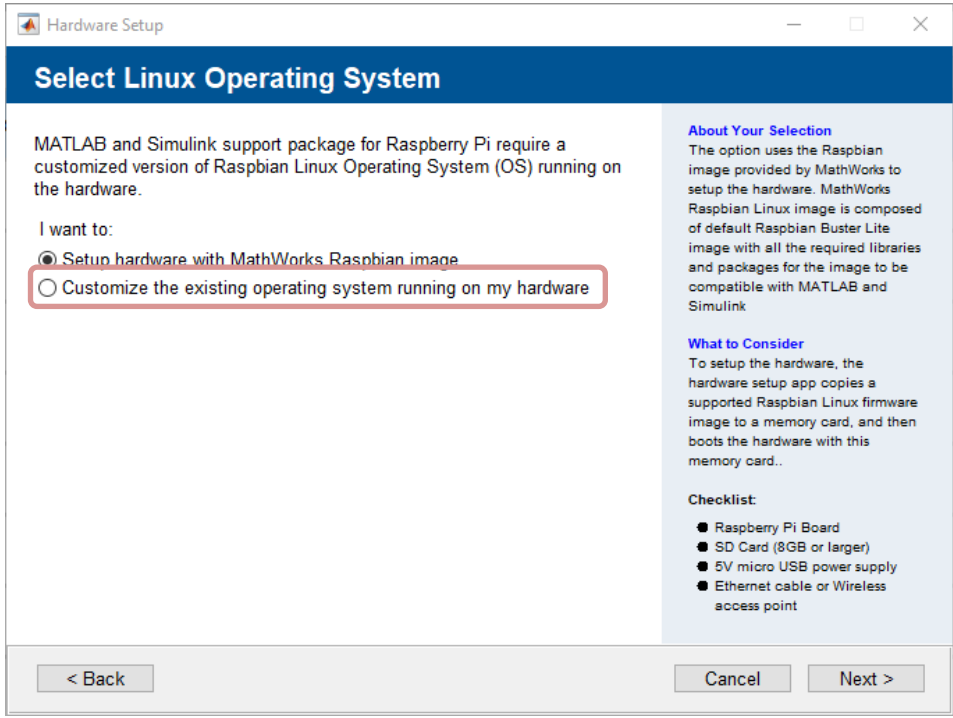
Hans-Petter Halvorsen

Table of Contents

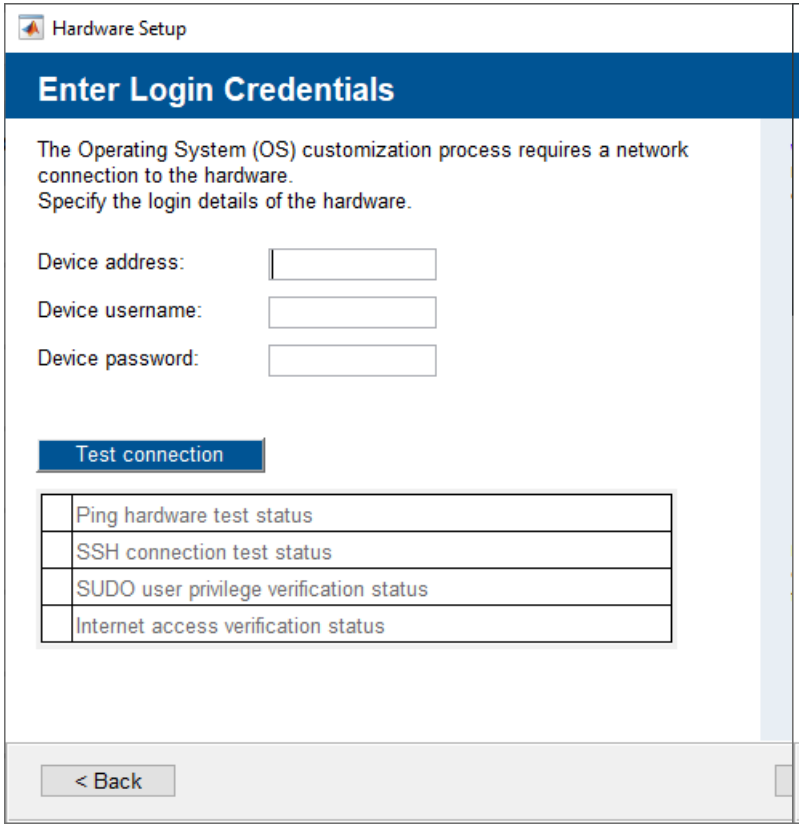
Hardware Setup

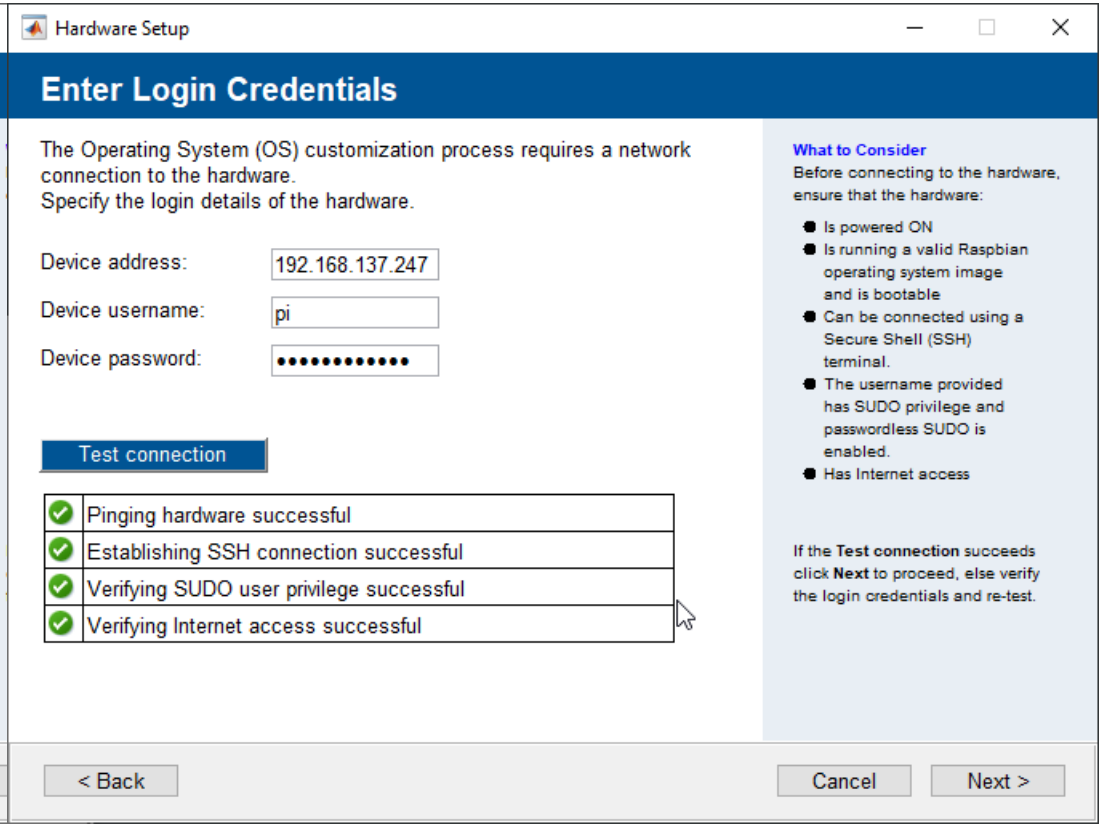
Customize the existing Operating System running on my Hardware



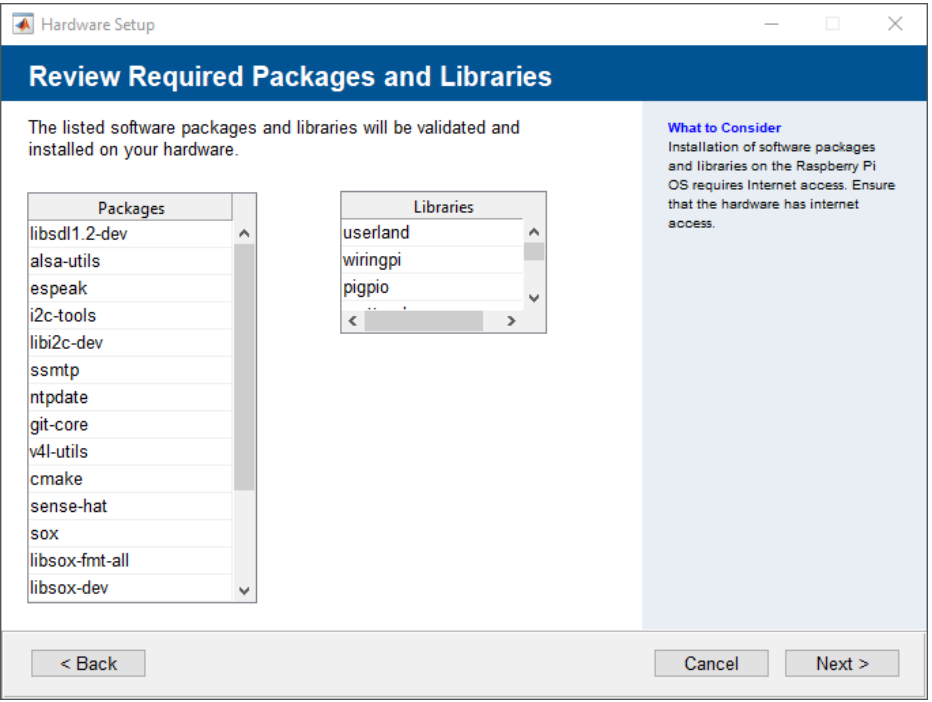


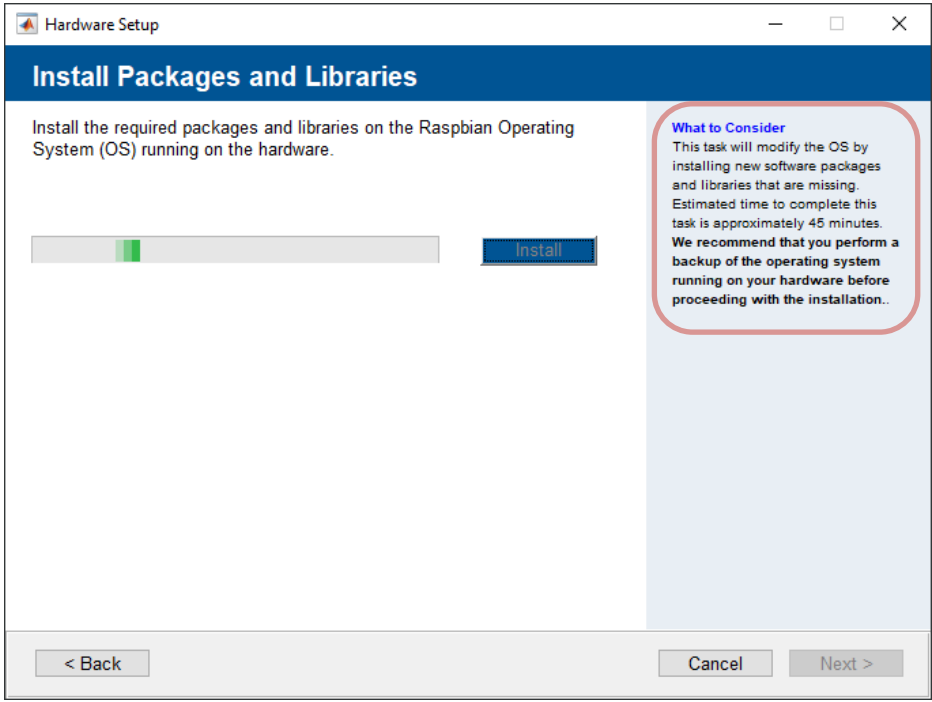
Hardware Setup





Hardware Setup



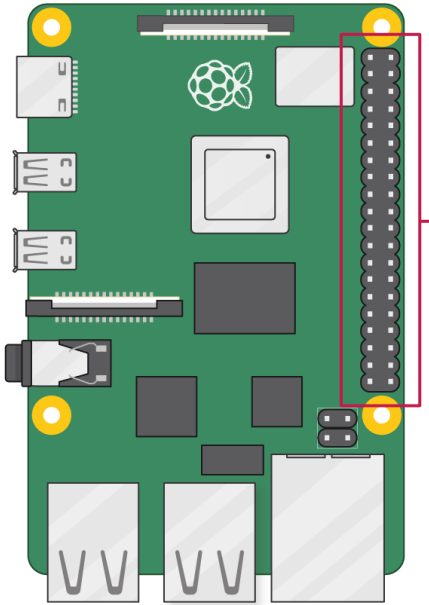


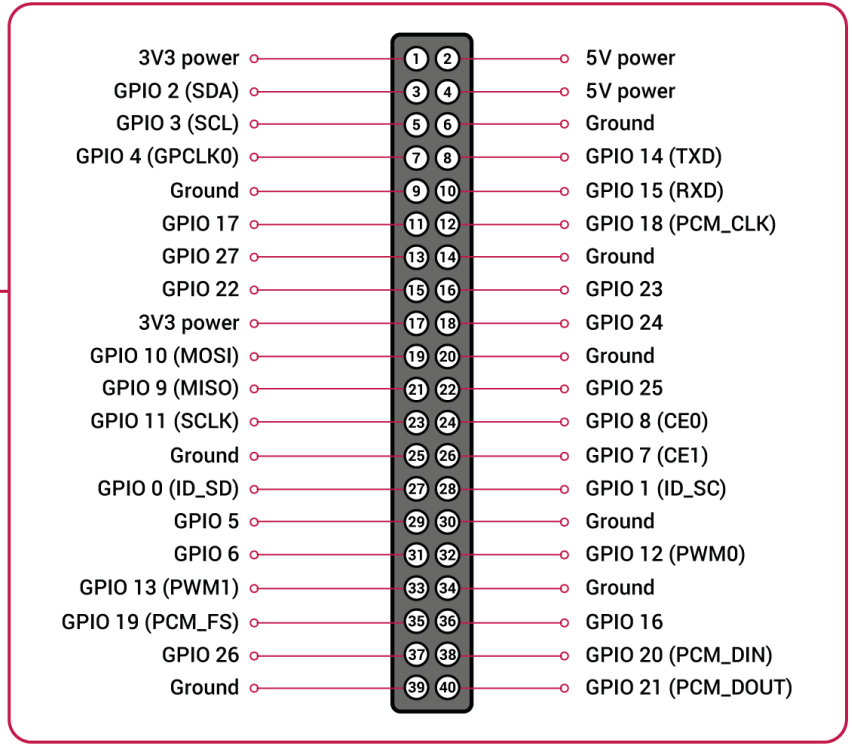
GPIO

Hans-Petter Halvorsen

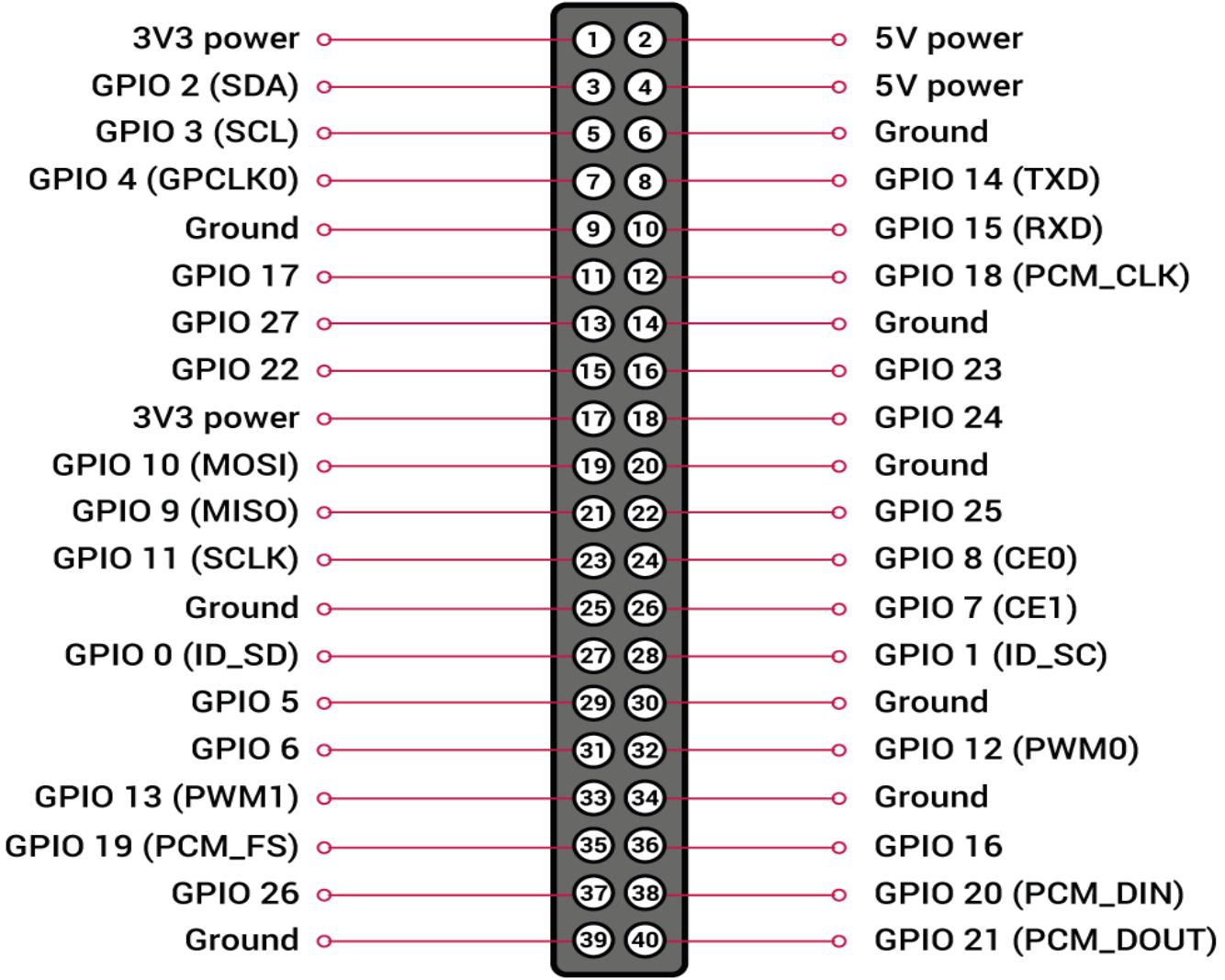
Table of Contents

GPIO




3V3 power
GPIO 2 (SDA)
GPIO 3 (SCL)
GPIO 4 (GPCLK0)
Ground
GPIO 17
GPIO 27
GPIO 22
3V3 power
GPIO 10 (MOSI)
GPIO 9 (MISO)
GPIO 11 (SCLK)
Ground
GPIO 0 (ID_SD)
GPIO 5
GPIO 6
GPIO 13 (PWM1)
GPIO 19 (PCM_FS)
GPIO 26
Ground
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30
31 32
33 34
35 36
37 38
39 40
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)

GPIO


3V3 power
GPIO 2 (SDA)
GPIO 3 (SCL)
GPIO 4 (GPCLK0)
Ground
GPIO 17
GPIO 27
GPIO 22
3V3 power
GPIO 10 (MOSI)
GPIO 9 (MISO)
GPIO 11 (SCLK)
Ground
GPIO 0 (ID_SD)
GPIO 5
GPIO 6
GPIO 13 (PWM1)
GPIO 19 (PCM_FS)
GPIO 26
Ground
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30
31 32
33 34
35 36
37 38
39 40
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)

GPIO Features

The GPIO pins are Digital Pins which are either True (+3.3V) or False (0V). These can be used to turn on/off LEDs, etc.

The Digital Pins can be either Output or Input.

In addition, some of the pins also offer some other Features:

- PWM (Pulse Width Modulation)

Digital Buses (for reading data from Sensors, etc.):

- SPI
- I2C

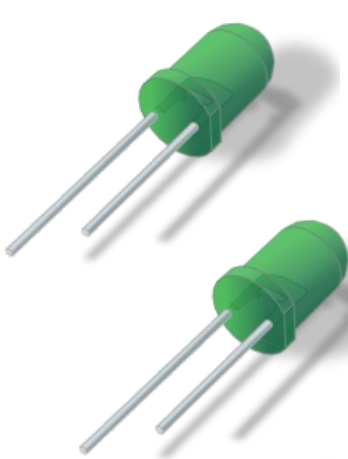
GPIO Code Examples

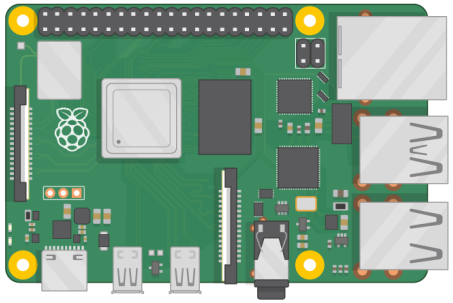
Hans-Petter Halvorsen

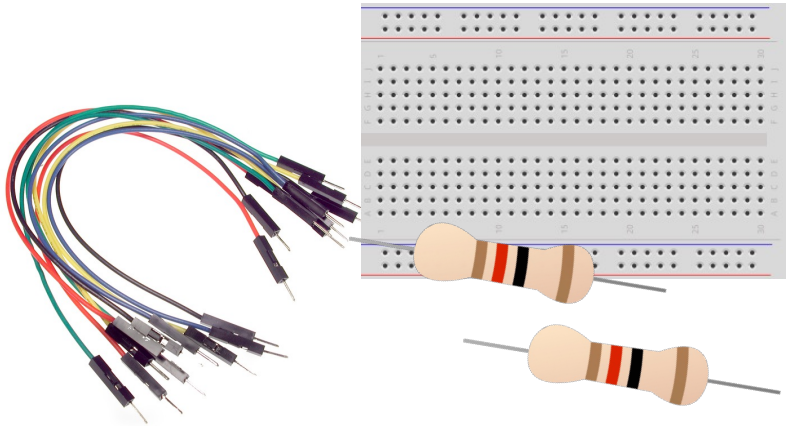
Table of Contents

Necessary Equipment

- Raspberry Pi
- Breadboard
- LEDs
- Push Buttons
- Resistors
- Wires (Jumper Wires)







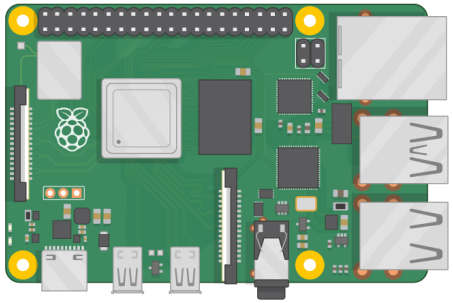
LED Example

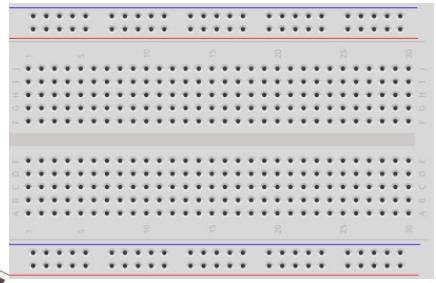
Hans-Petter Halvorsen

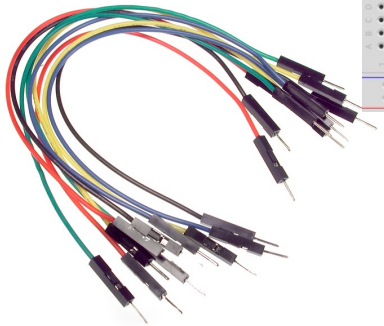
Table of Contents

Necessary Equipment

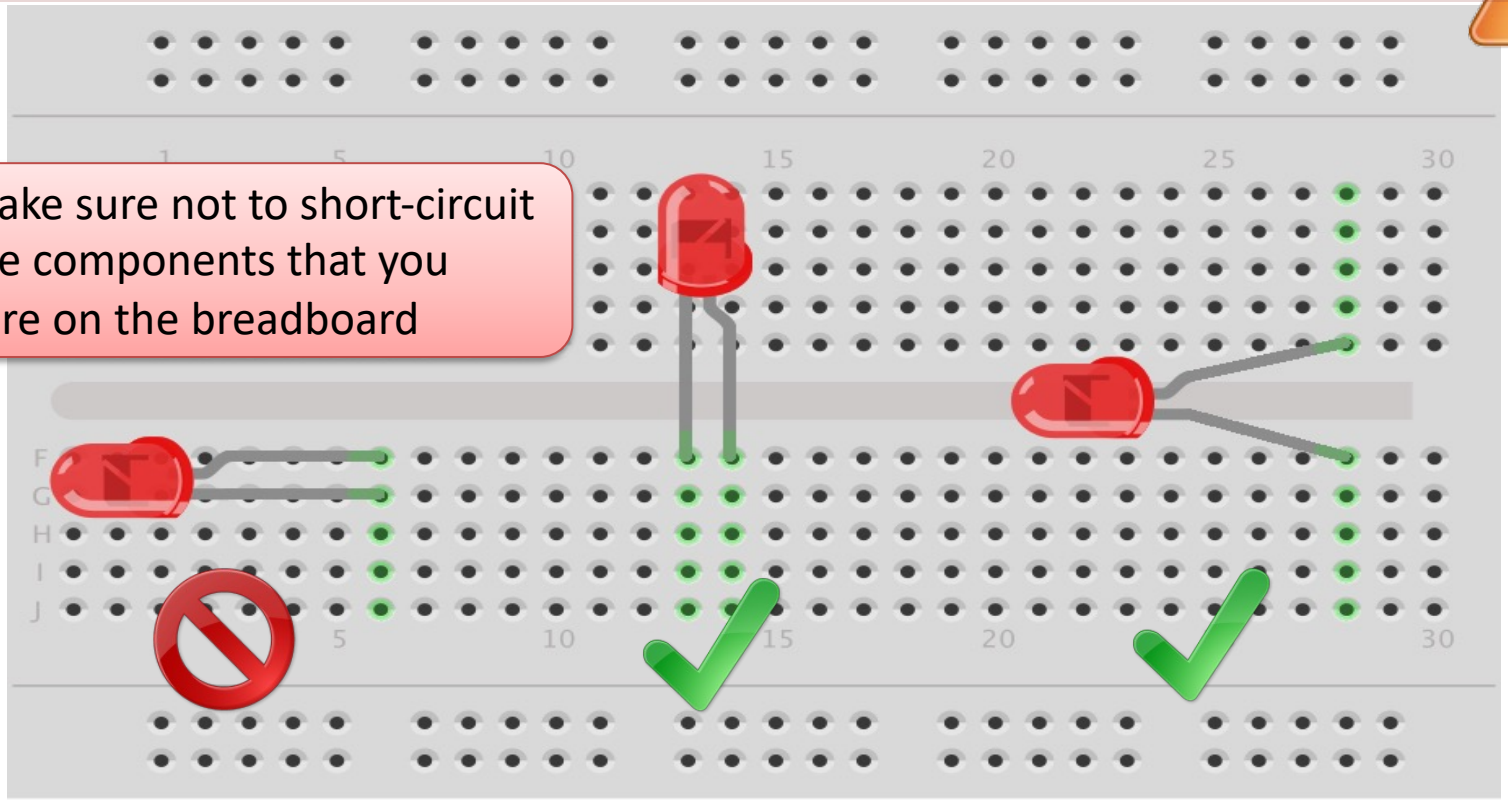
- Raspberry Pi
- Breadboard
- LED
- Resistor (270Ω)
- Wires (Jumper Wires)



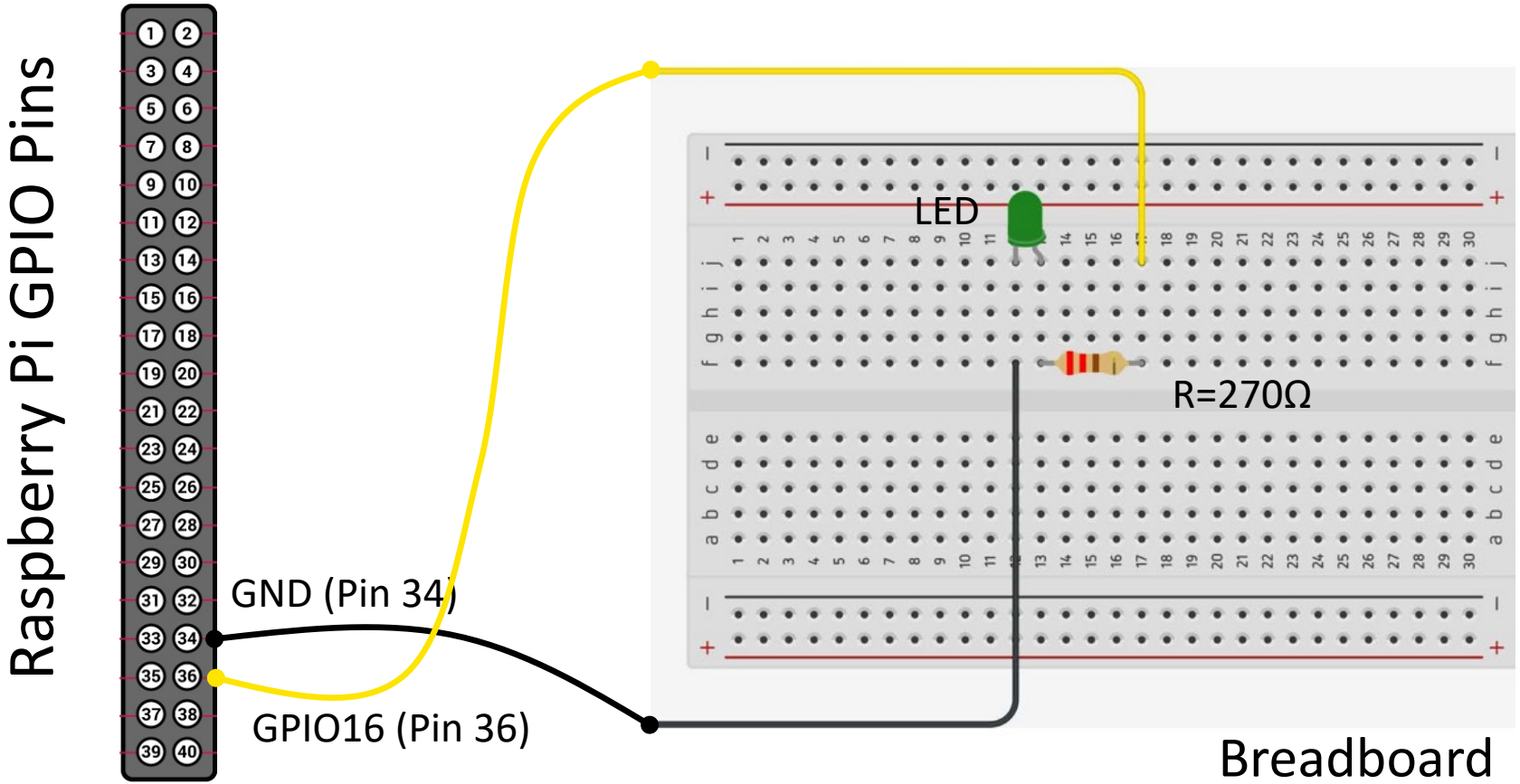




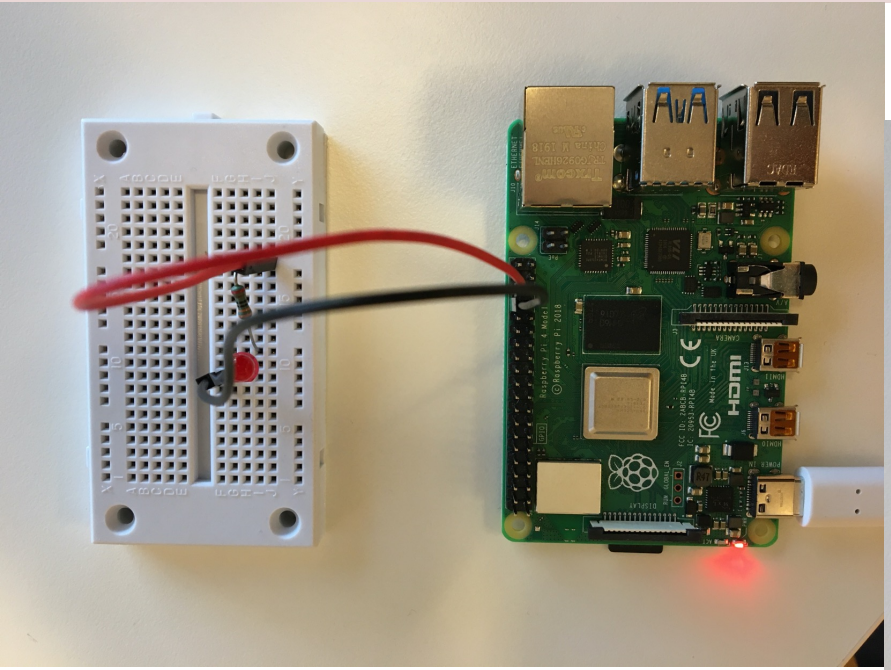
Breadboard Wiring

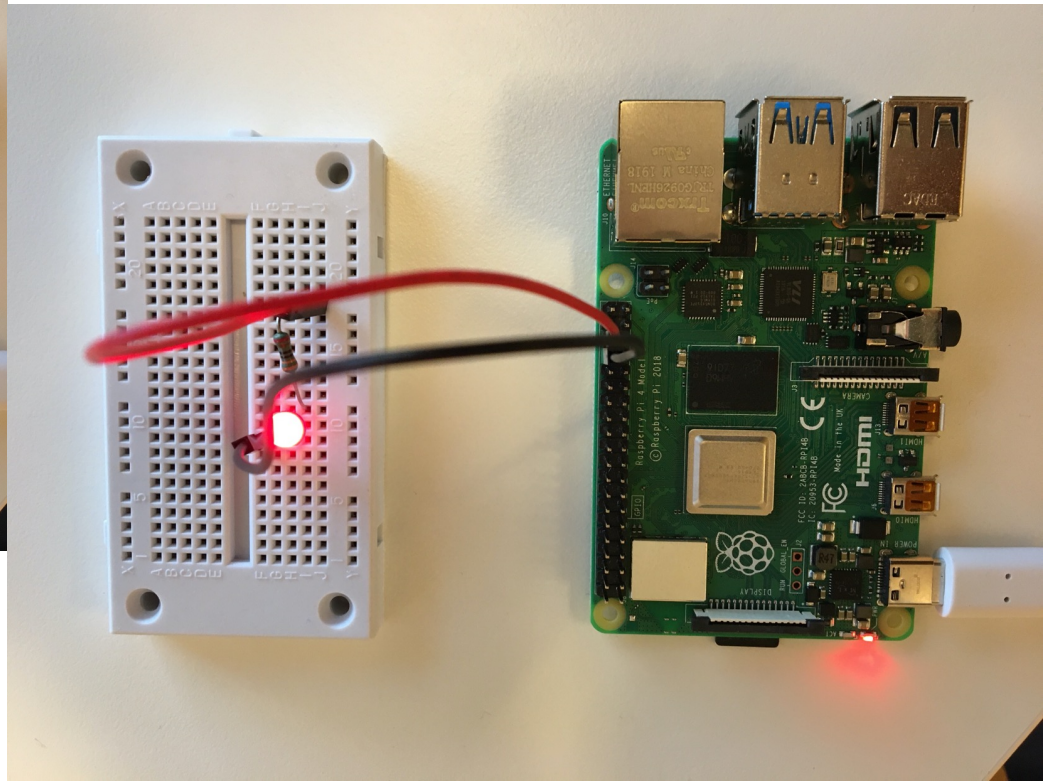


Wiring


Raspberry Pi GPIO Pins
GND (Pin 34)
GPIO16 (Pin 36)
LED
R=270Ω
Breadboard

Blinking LED GPIO Example





Turn LED ON Example


```
clear rpi
rpi = raspi();
gpiopin = 16;
ledvalue = 1;
writeDigitalPin(rpi, gpiopin, ledvalue);
```

Blinking LED GPIO Example

```
clear rpi
rpi = raspi();
gpiopin = 16;

for i = 1:10
    ledvalue = 1;
    writeDigitalPin(rpi, gpiopin, ledvalue);
    pause(0.5);
    ledvalue = 0;
    writeDigitalPin(rpi, gpiopin, ledvalue);
    pause(0.5);
end
```

Blinking LED GPIO Example

```
clear rpi
rpi = raspi();
gpiopin = 16;

for i = 1:10
    ledvalue = 1;
    writeDigitalPin(rpi, gpiopin, ledvalue);
    pause(0.5);
    ledvalue = 0;
    writeDigitalPin(rpi, gpiopin, ledvalue);
    pause(0.5);
end
```

PWM

Pulse Width Modulation

Hans-Petter Halvorsen

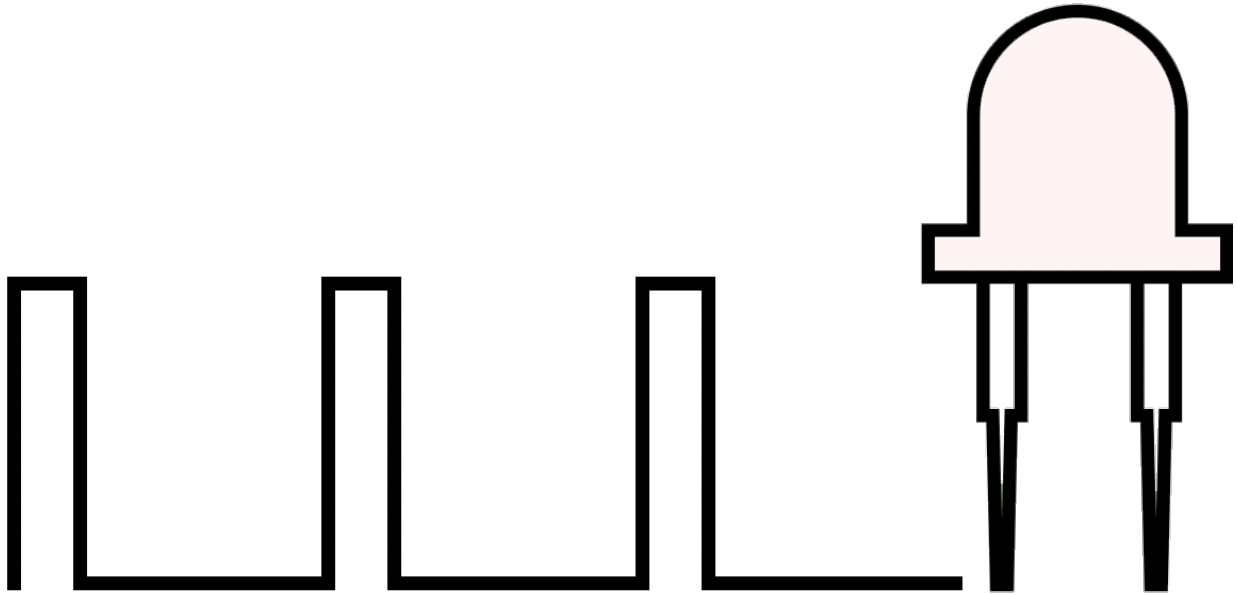
Table of Contents

Controlling LED Brightness using PWM

- 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).

Controlling LED Brightness using PWM

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

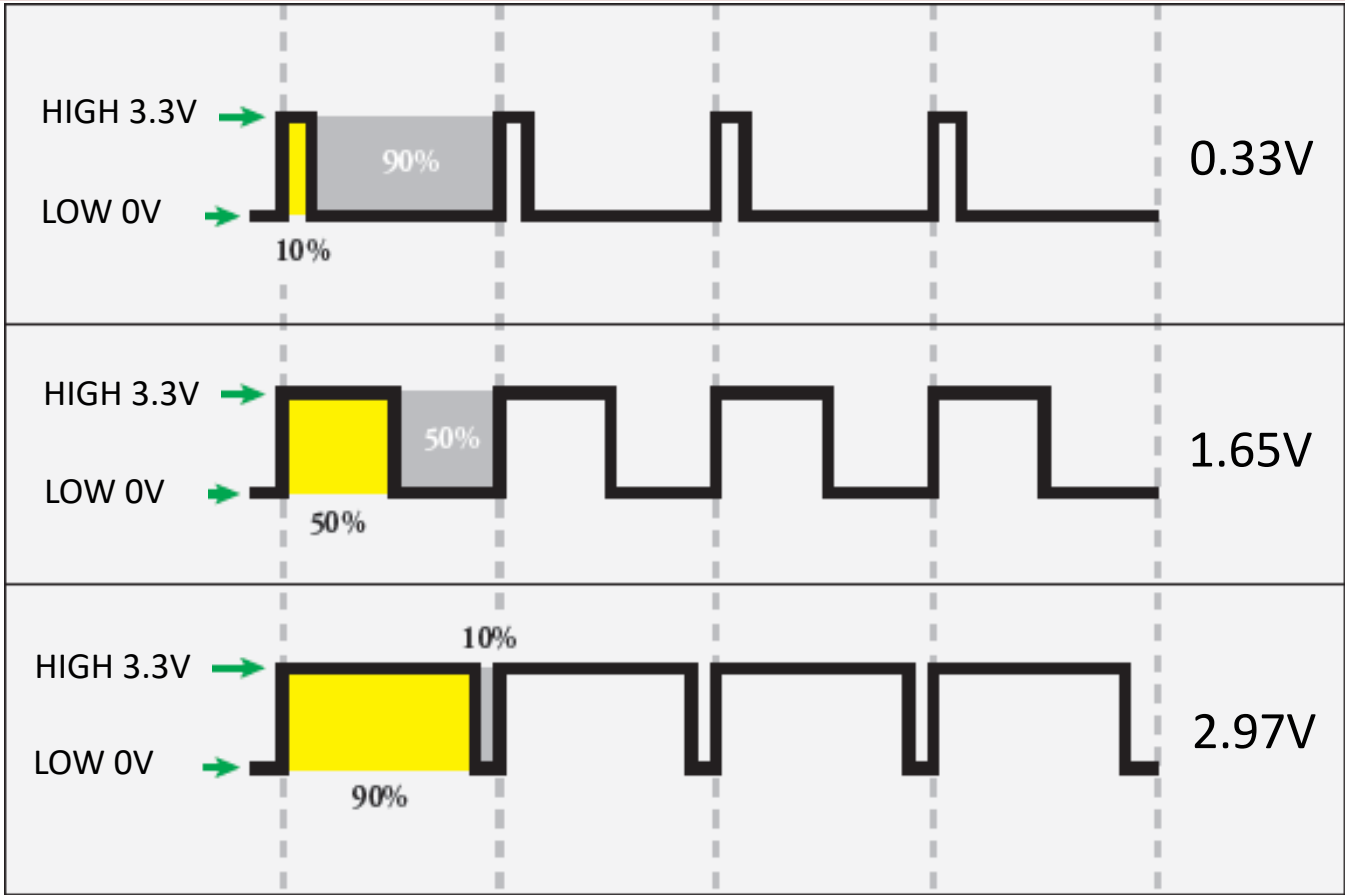


https://www.electronicwings.com/raspberry-pi/raspberry-pi-pwm-generation-using-python-and-c

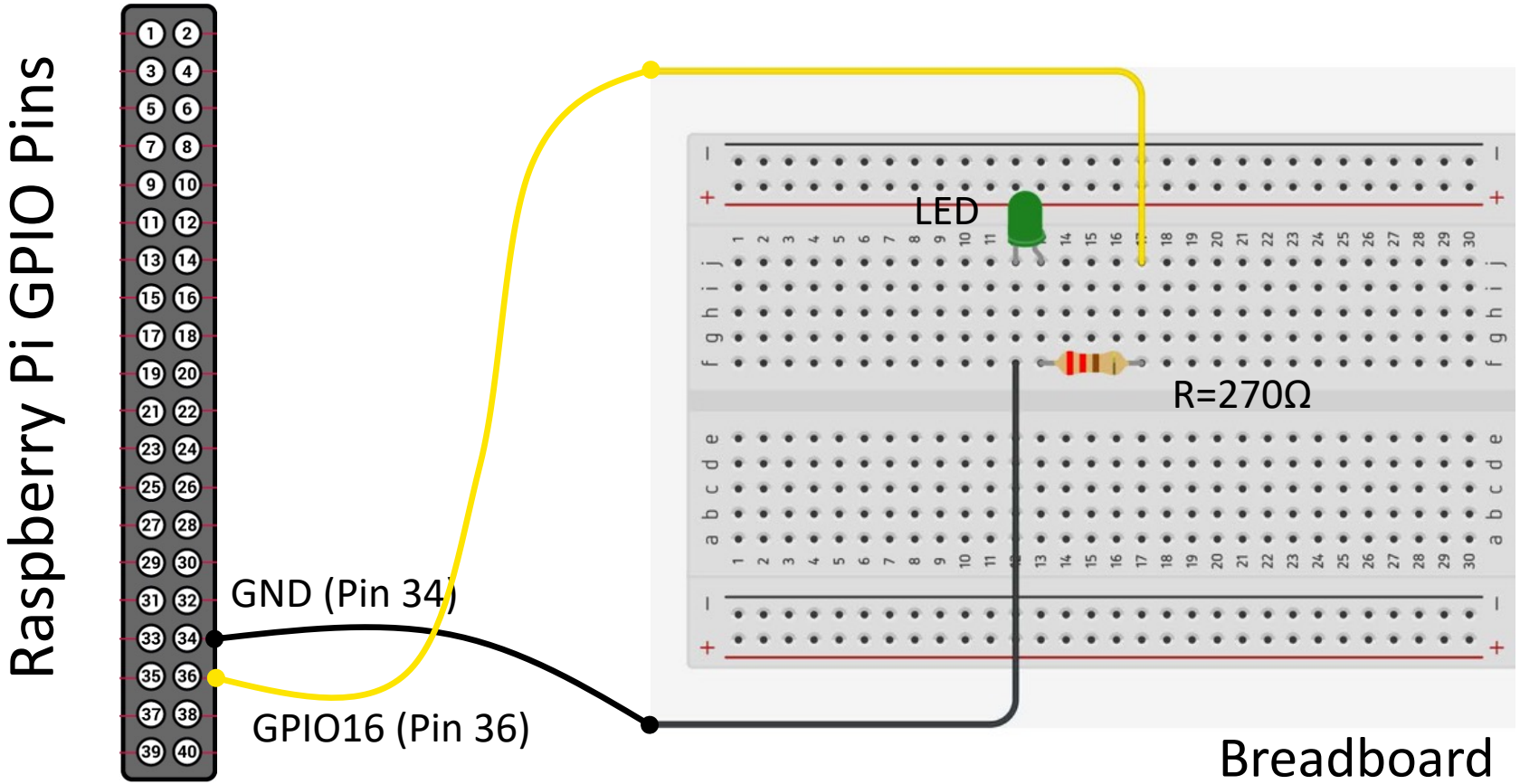
PWM as “Analog Out“

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.

0 – 3.3V



Wiring


Raspberry Pi GPIO Pins
GND (Pin 34)
GPIO16 (Pin 36)
LED
R=270Ω
Breadboard

writePWM**Voltage**

```
clear mypi
mypi = raspi()

gpiopin = 16;

configurePin(mypi, gpiopin, 'PWM');
writePWMFrequency(mypi, gpiopin, 2000);

volt = 1.65
writePWMVoltage(mypi, gpiopin, volt);
```

volt between 0V (0%) and 3.3V (100%)
This means 1.65V = 50% brightness of the LED

writePWM**DutyCycle**

```
clear mypi
mypi = raspi()

gpiopin = 16;

configurePin(mypi, gpiopin, 'PWM');
writePWMFrequency(mypi, gpiopin, 2000);

dutycycle = 0.5
writePWMDutyCycle(mypi, gpiopin, dutycycle);
```

dutycycle between 0 (0%) and 1 (100%)
This means 0.5 = 50% brightness of the LED

PWM in MATLAB - Summary

```
clear mypi
mypi = raspi()

gpiopin = 16;

configurePin(mypi, gpiopin, 'PWM');
writePWMFrequency(mypi, gpiopin, 2000);

volt = 1.65
writePWMVoltage(mypi, gpiopin, volt);
```

volt between 0V (0%) and 3.3V (100%)
This means 1.65V = 50% brightness of the LED

dutycycle between 0 (0%) and 1 (100%)
This means 0.5 = 50% brightness of the LED

```
clear mypi
mypi = raspi()

gpiopin = 16;

configurePin(mypi, gpiopin, 'PWM');
writePWMFrequency(mypi, gpiopin, 2000);

dutycycle = 0.5
writePWMDutyCycle(mypi, gpiopin, dutycycle);
```

PWM with MATLAB

```
clear rpi
mypi = raspi()

gpiopin = 16;

configurePin(mypi, gpiopin, 'PWM');
writePWMFrequency(mypi, gpiopin, 2000);


for dutycycle = 0:0.1:1
    writePWMDutyCycle(mypi, gpiopin, dutycycle);
    pause(0.5);
end

dutycycle = 0;
writePWMDutyCycle(mypi, gpiopin, dutycycle);
```

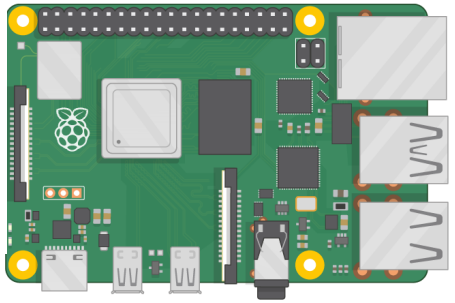
Push Button

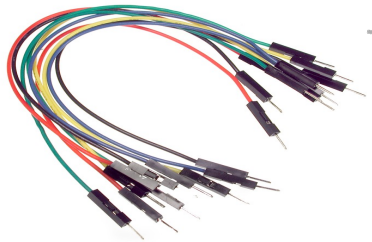
Hans-Petter Halvorsen

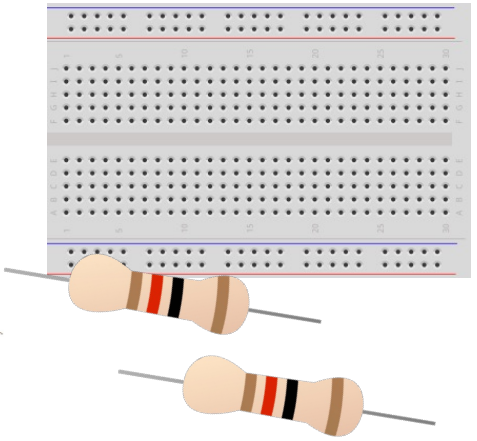
Table of Contents

Necessary Equipment

- Raspberry Pi
- Breadboard
- Push Button
- LED
- Resistors, $R = 270\Omega, R = 10k\Omega$
- Wires (Jumper Wires)

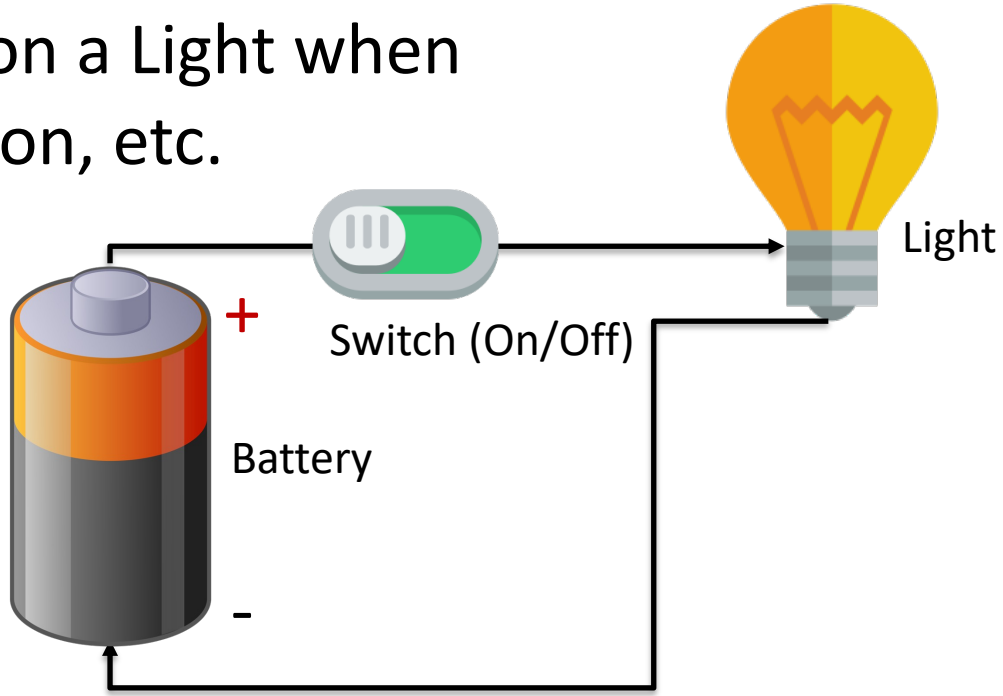






Push Button/Switch

- 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.

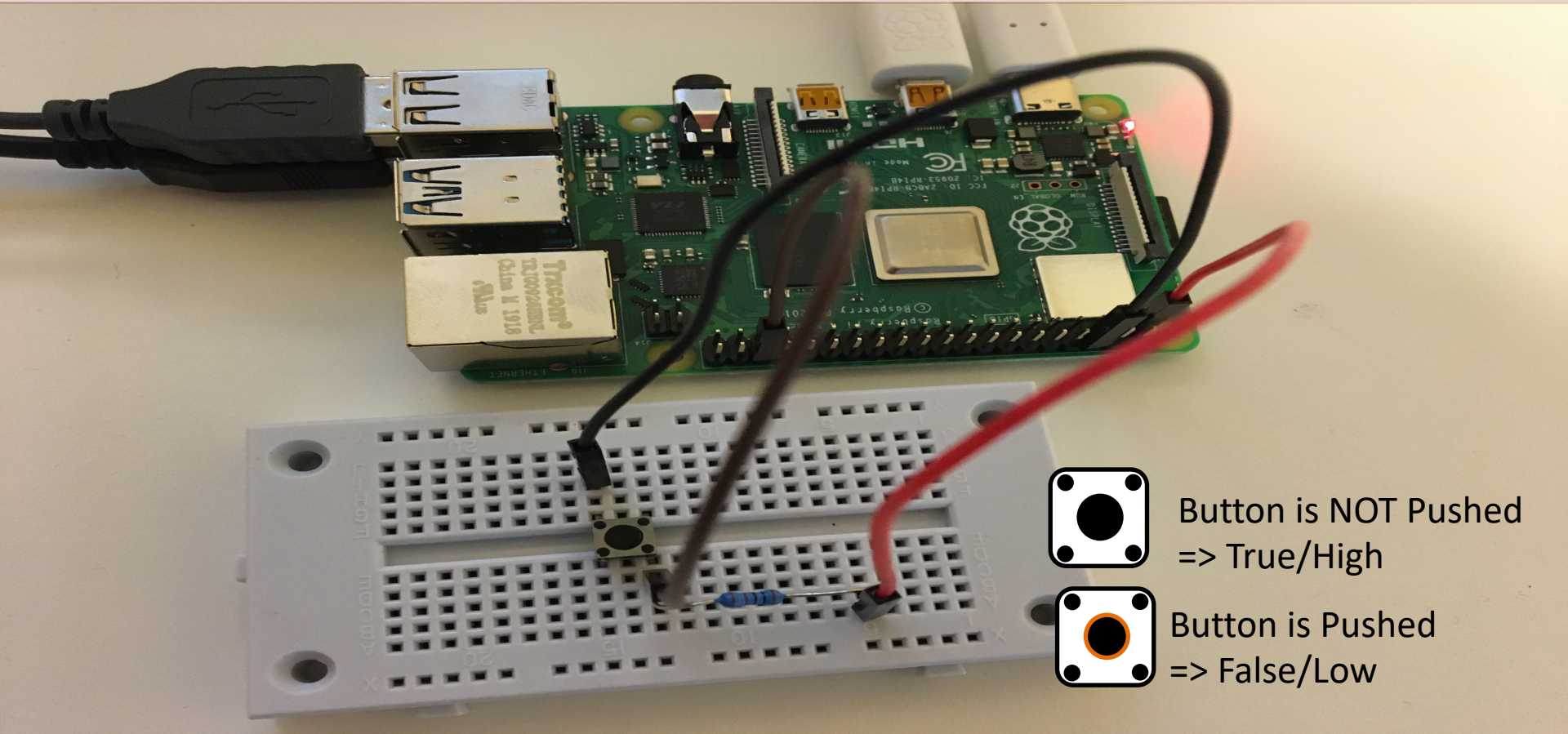


Push Button (Pull-up Resistor)

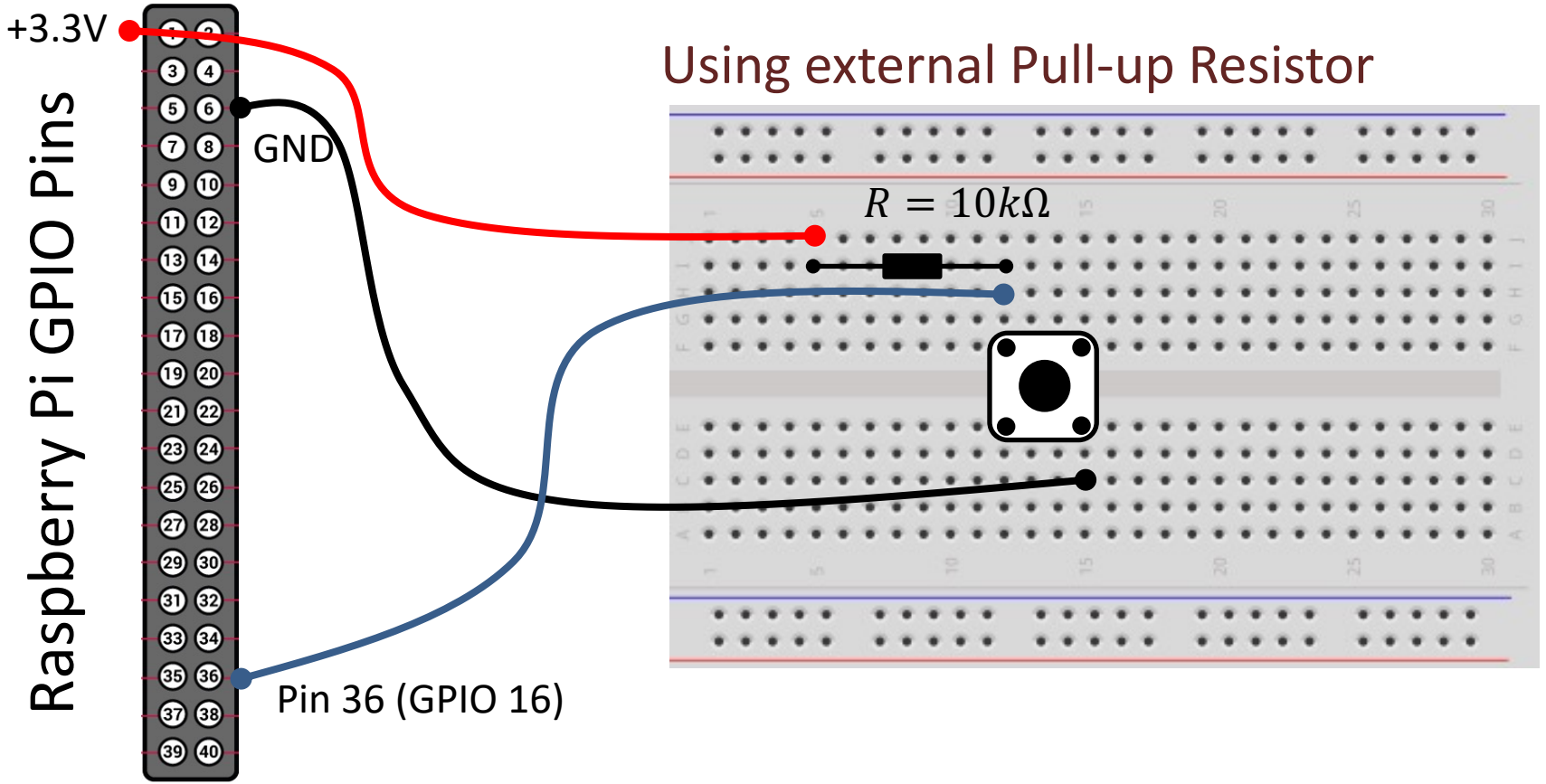
Hans-Petter Halvorsen

Table of Contents

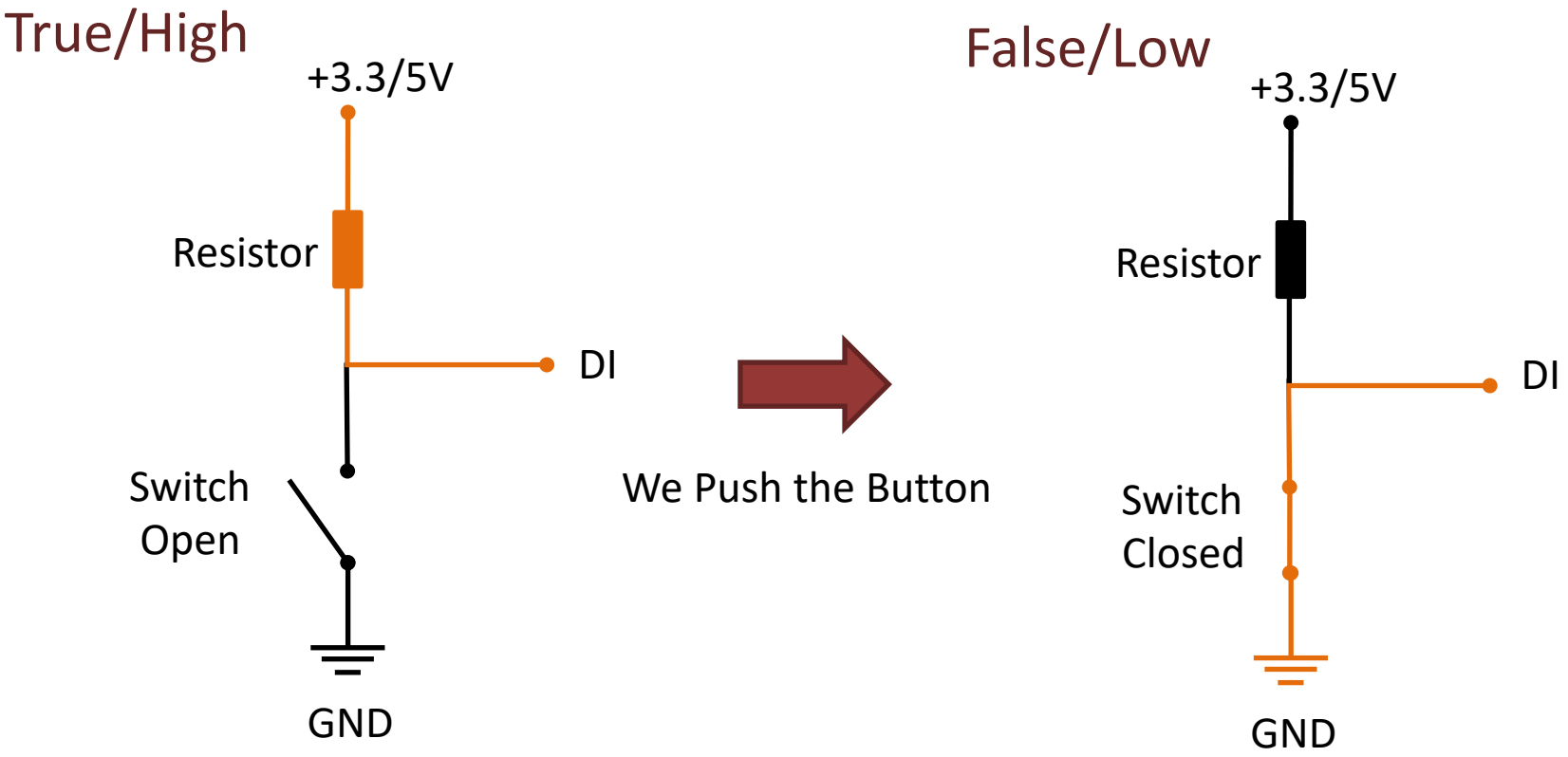
Push Button (Pull-up Resistor)



Button Setup (Pull-up Resistor)



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.

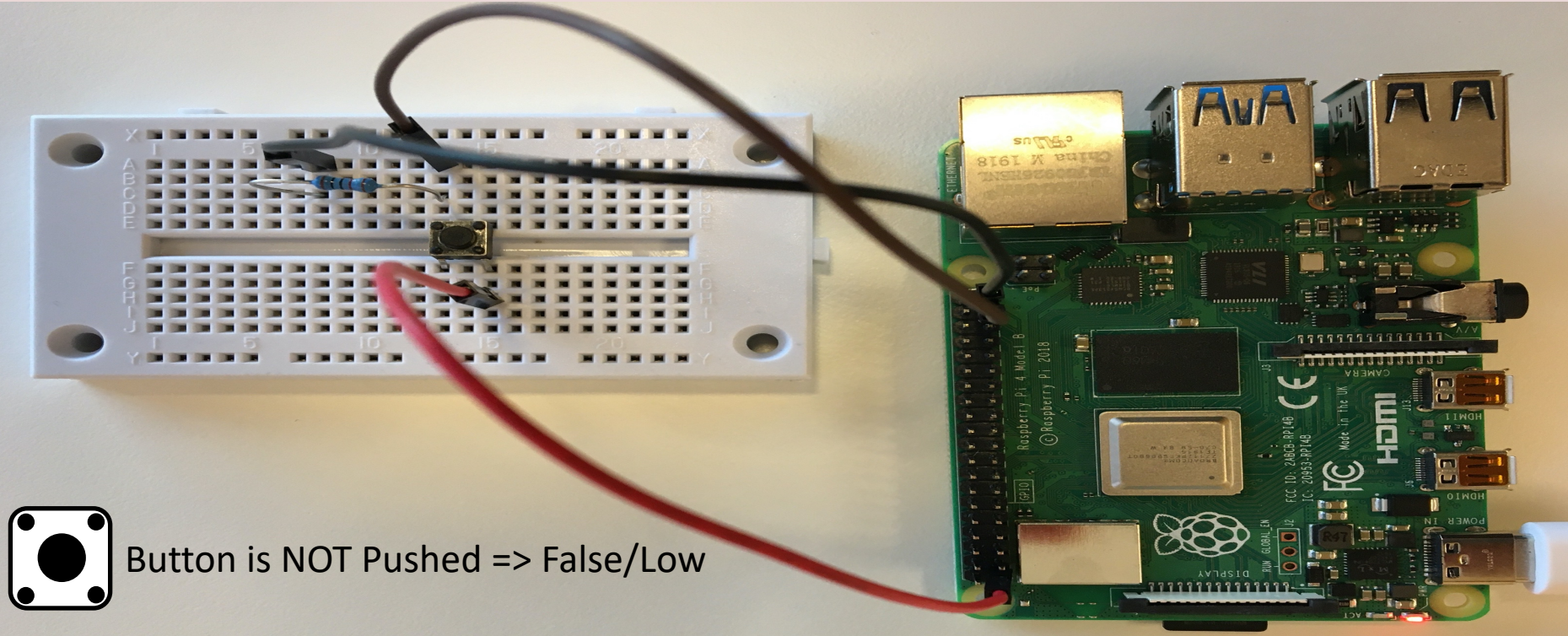
```
clear r
clc

r = raspi();
gpiopin_read = 16;

for i = 1:10
    status = readDigitalPin(r, gpiopin_read);

    if (status == 0)
        disp("Button Pushed")
    else
        disp("Please Push the Button")
    end

    pause(1);
end
```

Push Button (Pull-down Resistor)

Hans-Petter Halvorsen

Table of Contents

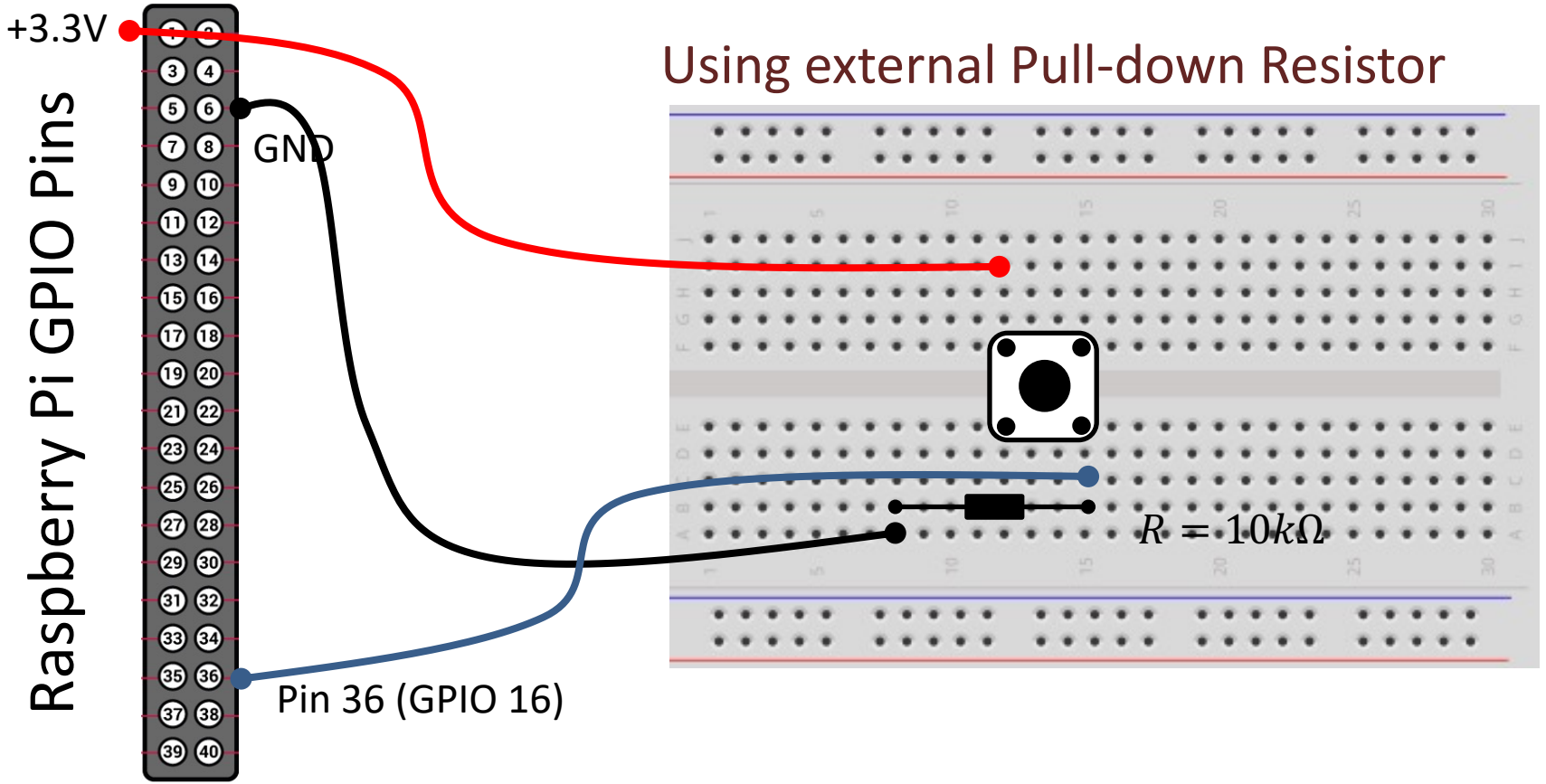
Push Button (Pull-down Resistor)



Button is NOT Pushed => False/Low

Button is Pushed => True/High

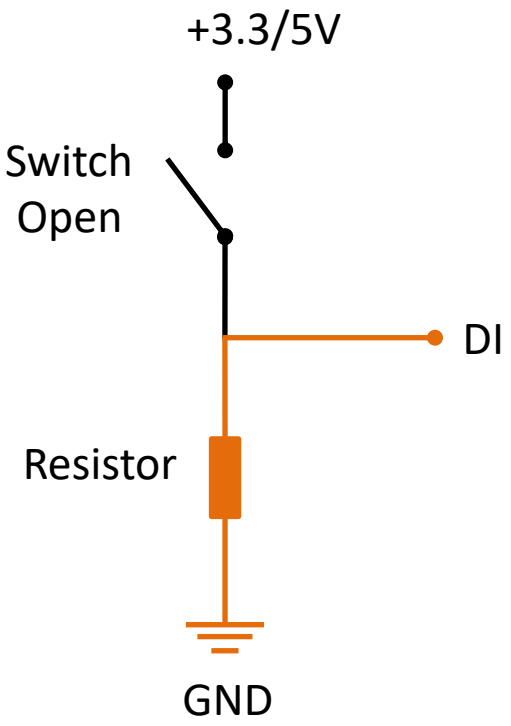
Button Setup (Pull-down Resistor)



Pull-down Resistor

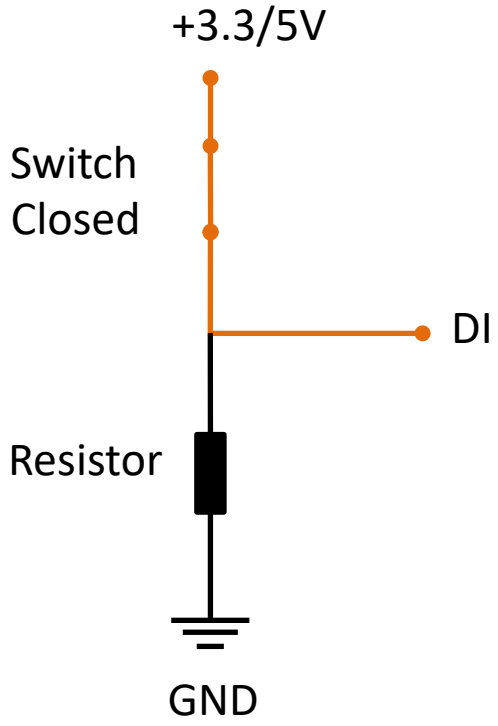
We could also have wired according to a “Pull-down” Resistor

False/Low



We Push the Button

True/High



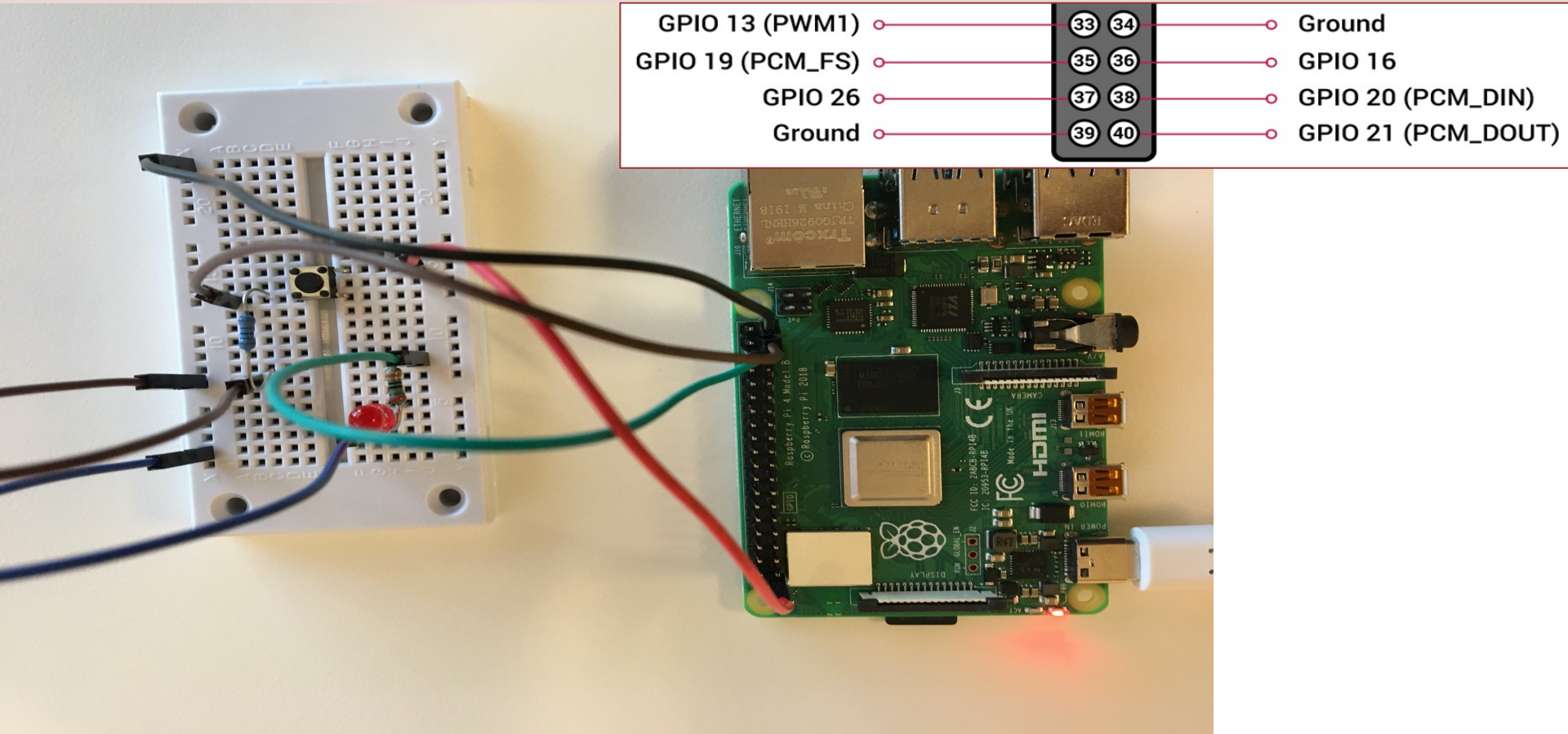
```
clear r
clc

r = raspi();
gpiopin_read = 16;

for i = 1:10
    status = readDigitalPin(r, gpiopin_read);

    if (status == 1)
        disp("Button Pushed")
    else
        disp("Please Push the Button")
    end

    pause(1);
end
```

Push Button + LED

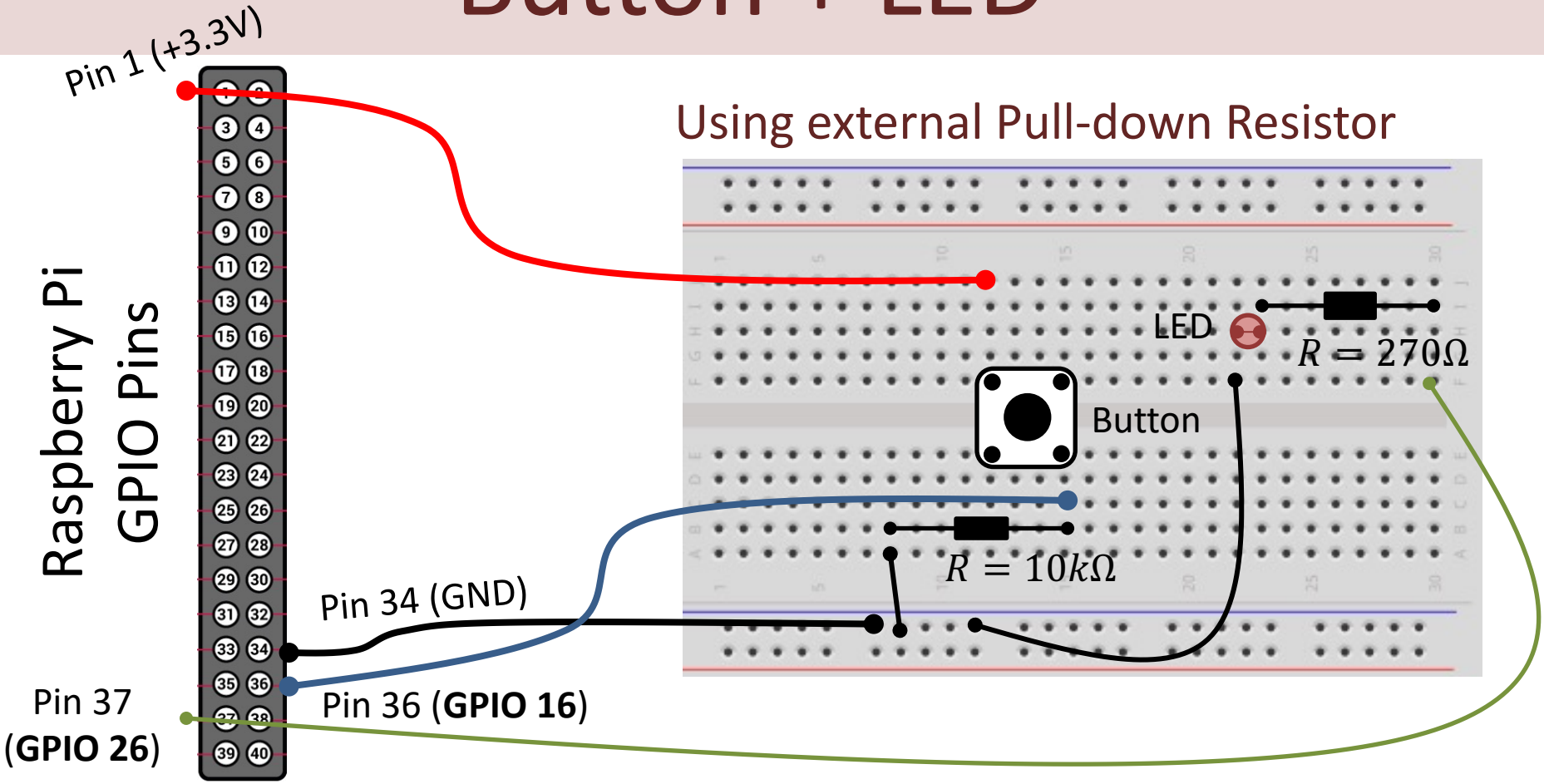
Hans-Petter Halvorsen

Table of Contents

Push Button + LED Example



Button + LED


Pin 1 (+3.3V)
Raspberry Pi
GPIO Pins
Using external Pull-down Resistor
LED
$R = 270\Omega$
Button
$R = 10k\Omega$
Pin 34 (GND)
Pin 36 (**GPIO 16**)
Pin 37
(**GPIO 26**)

```
clear r
clc
gpiopin_button = 16;
gpiopin_led = 26;

r = raspi();
disp("Raspberry Pi Ready")

for i = 1:10

    status = readDigitalPin(r, gpiopin_button);

    if (status == 1)
        ledvalue = 1;
        writeDigitalPin(r, gpiopin_led, ledvalue);
        disp("LED On")
    else
        ledvalue = 0;
        writeDigitalPin(r, gpiopin_led, ledvalue);
        disp("LED Off")
    end

    pause(1);

end
clear r
disp("Program is Finished")
```

Camera

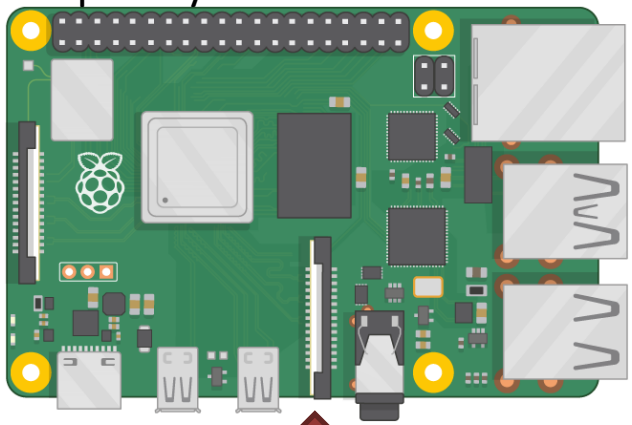
Hans-Petter Halvorsen

Table of Contents

Raspberry Pi + Camera

We can either use a specialized Raspberry Pi Camera or an ordinary/standard USB Camera

Raspberry Pi



USB Connector

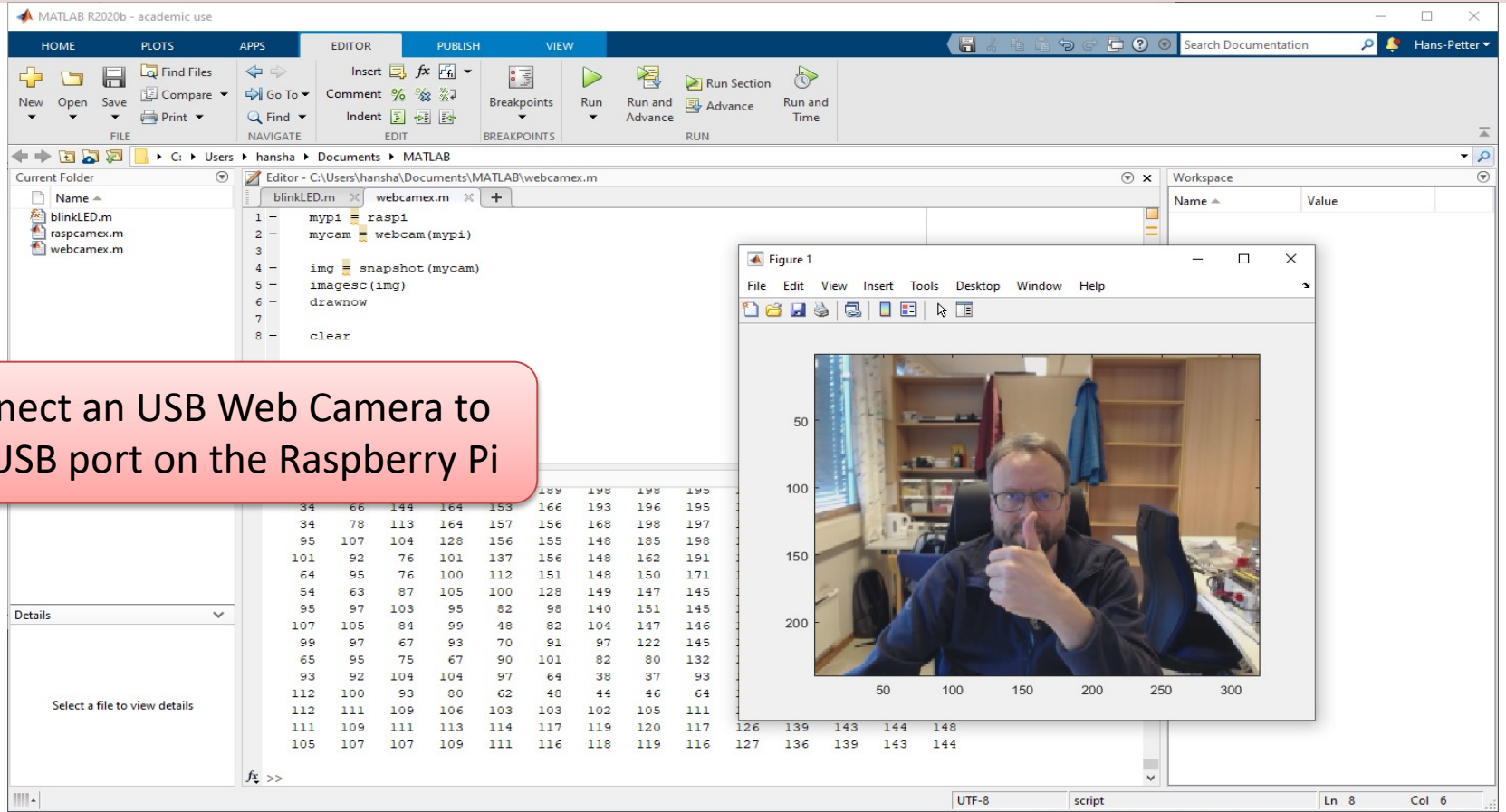
Raspberry Pi Camera Module v2

Camera Connector

CSI (Camera Serial Interface)

Standard USB Web Camera

Web Camera Example



Connect an USB Web Camera to the USB port on the Raspberry Pi

Web Camera Example

```
clear
clc

mypi = raspi
mycam = webcam(mypi)

img = snapshot(mycam)
imagesc(img)
drawnow
```

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog