https://www.halvorsen.blog

Raspberry Pi with MATLAB

Using SPI and I2C

Hans-Petter Halvorsen

Contents

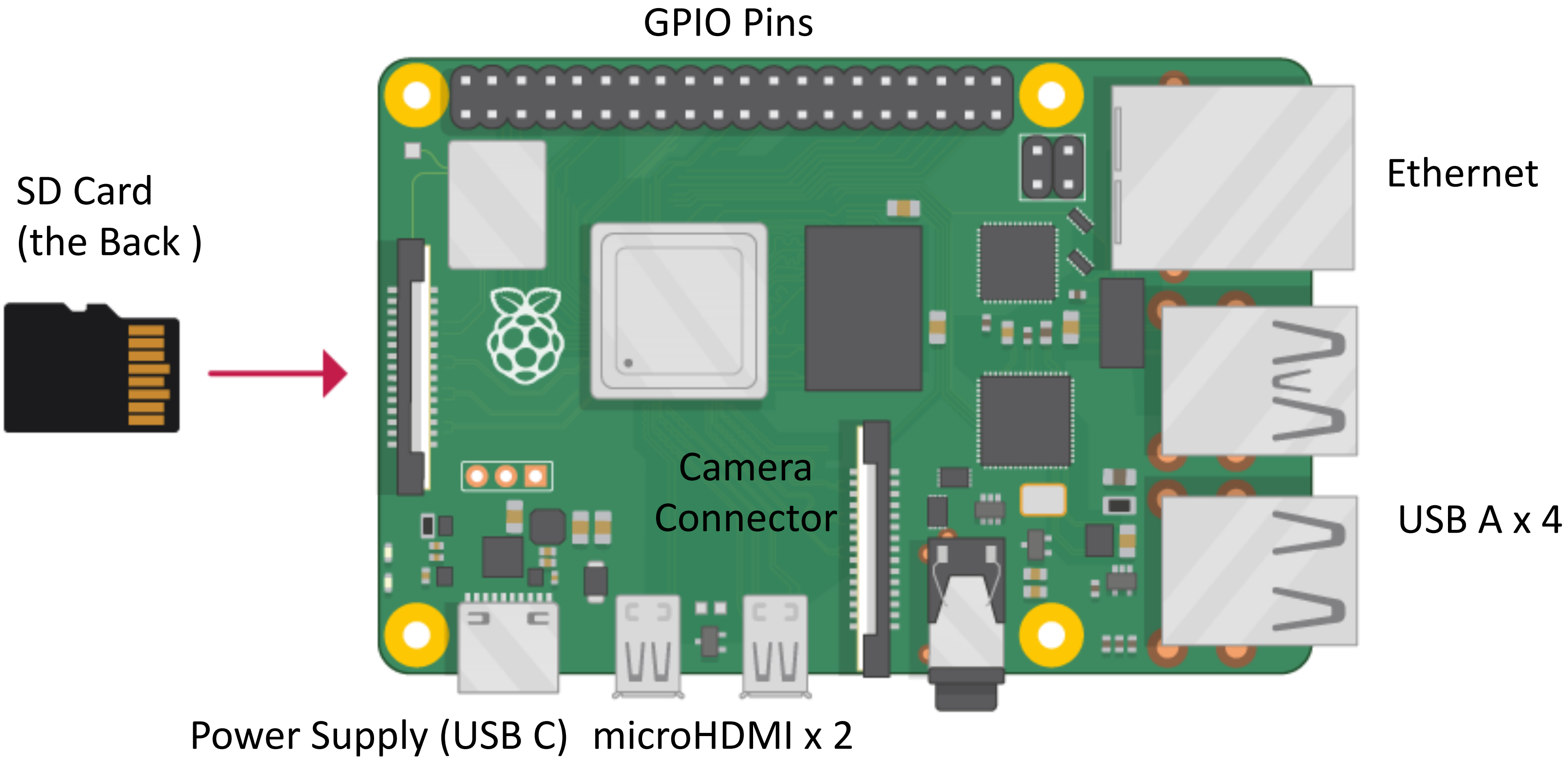
These parts are covered in more detail in a previous Tutorial

Raspberry Pi

Hans-Petter Halvorsen

Table of Contents

Raspberry Pi

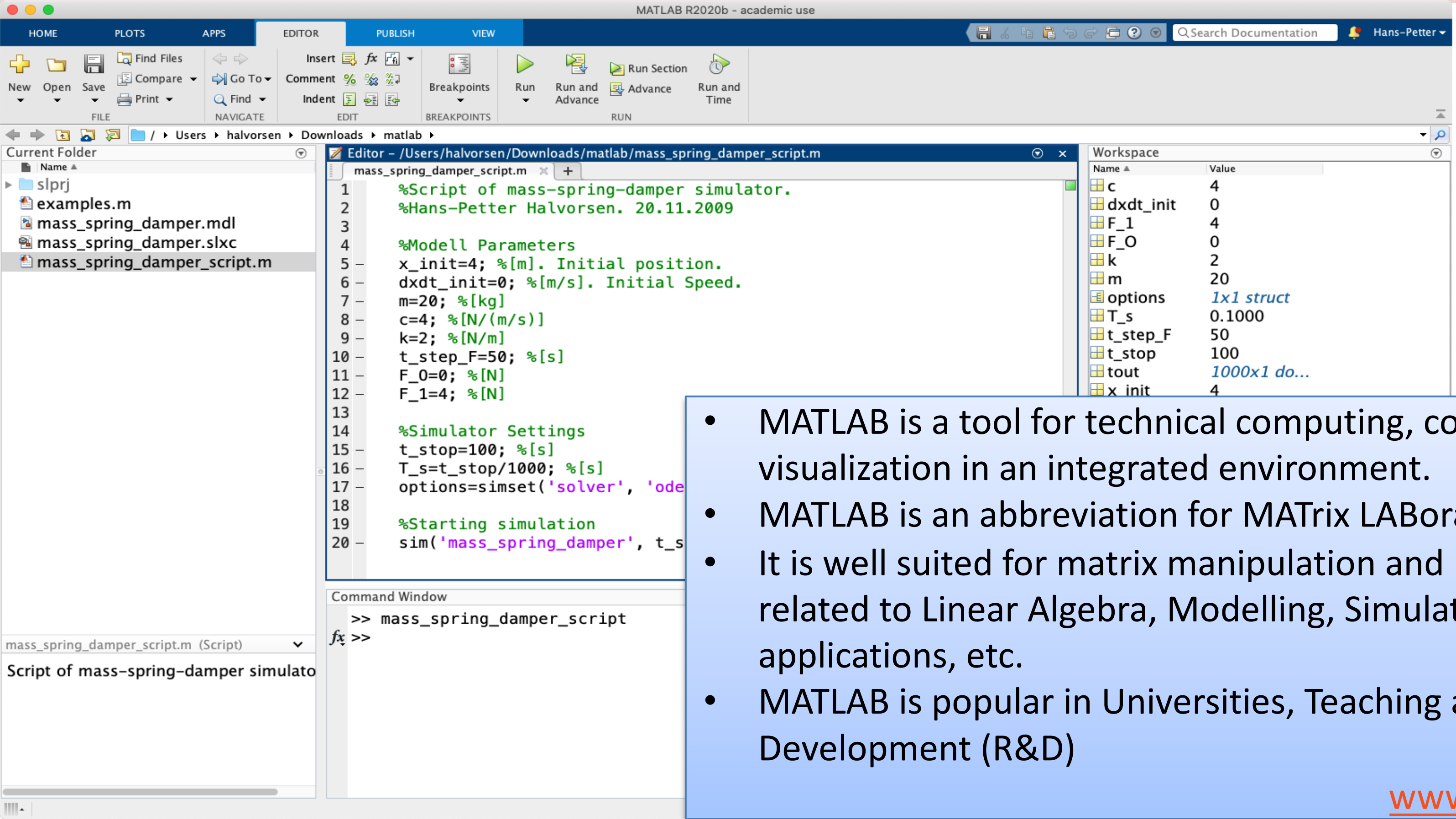


MATLAB

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

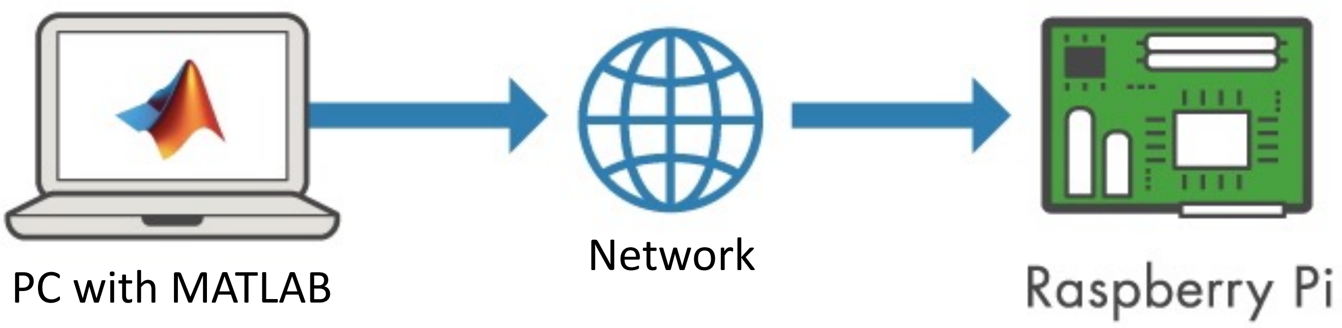
MATLAB Support Package for Raspberry Pi

Hans-Petter Halvorsen

Table of Contents

Raspberry Pi + MATLAB

MATLAB Support Package for Raspberry Pi

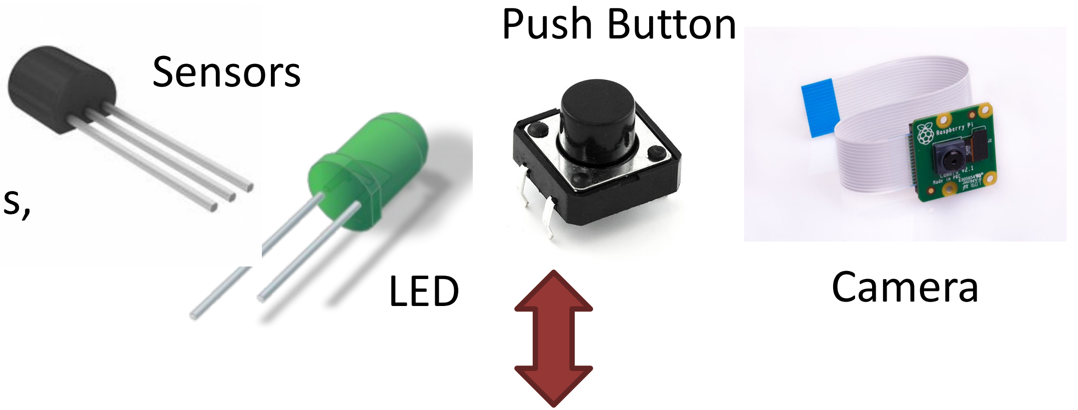


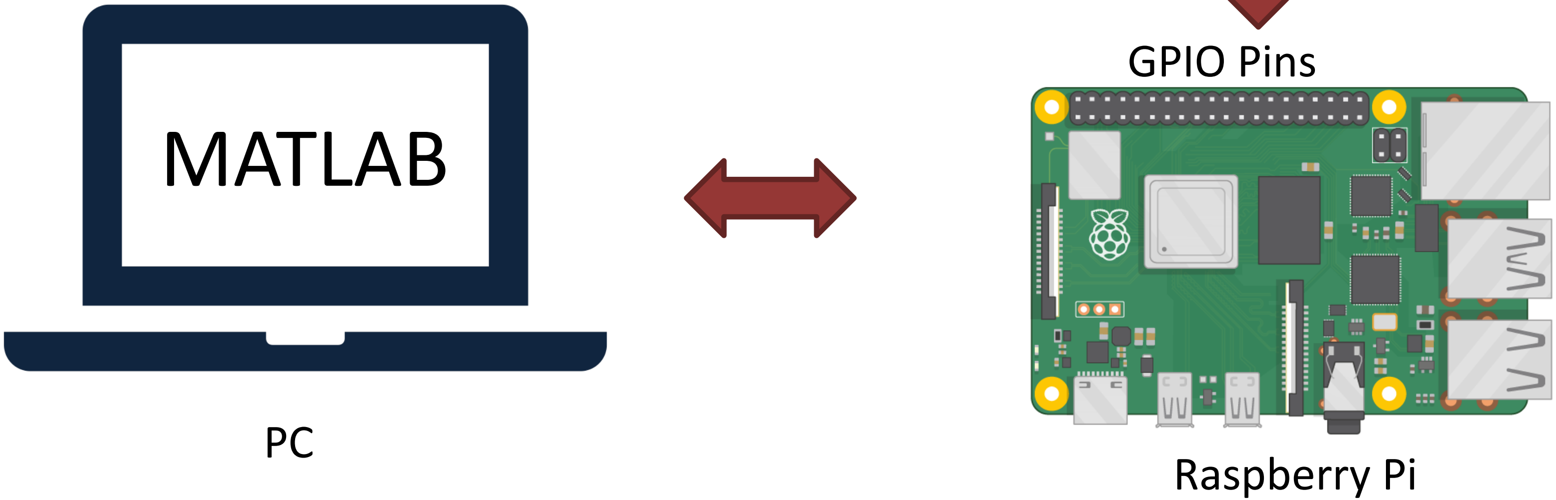
With MATLAB Support Package for Raspberry Pi, the Raspberry Pi is connected to a computer running MATLAB. Processing is done on the computer with MATLAB.

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

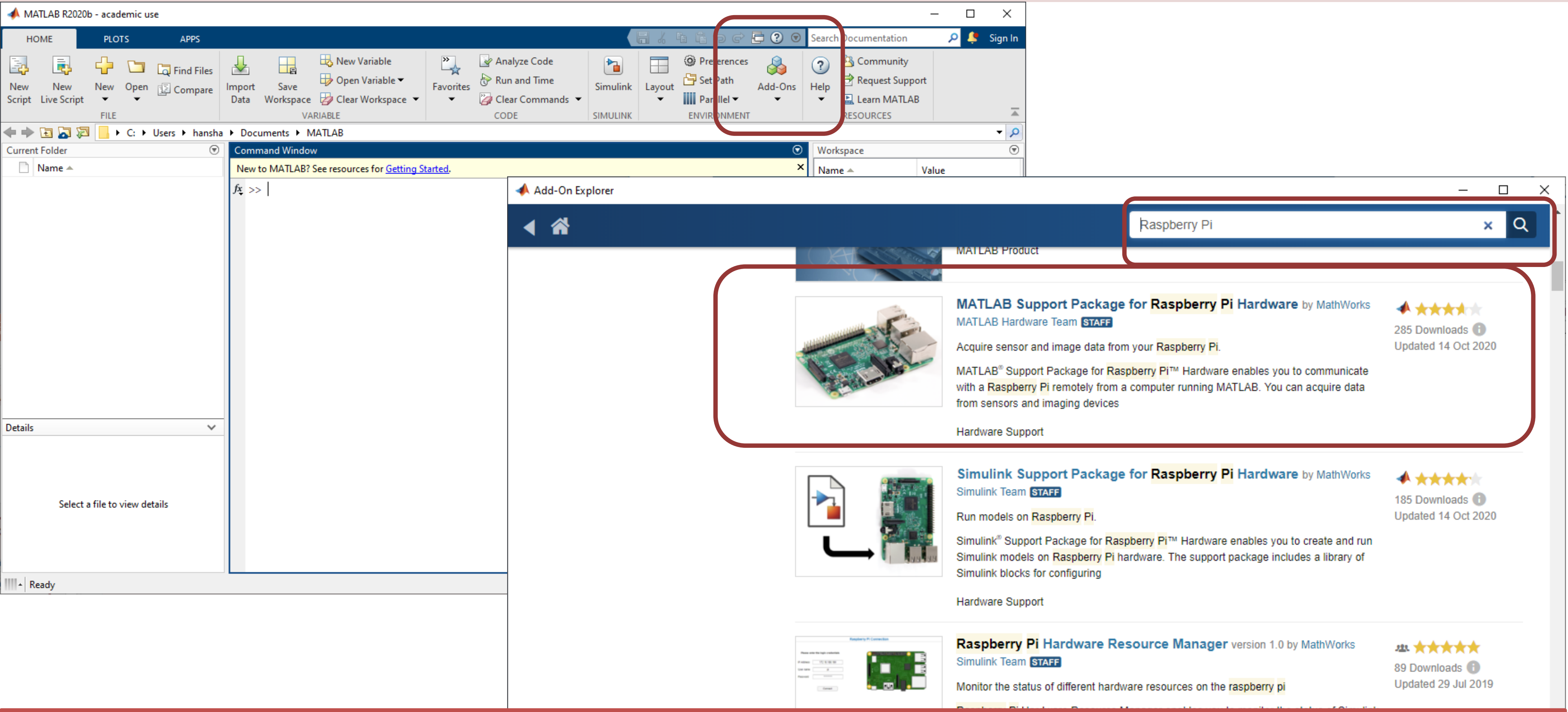
Raspberry Pi + MATLAB

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



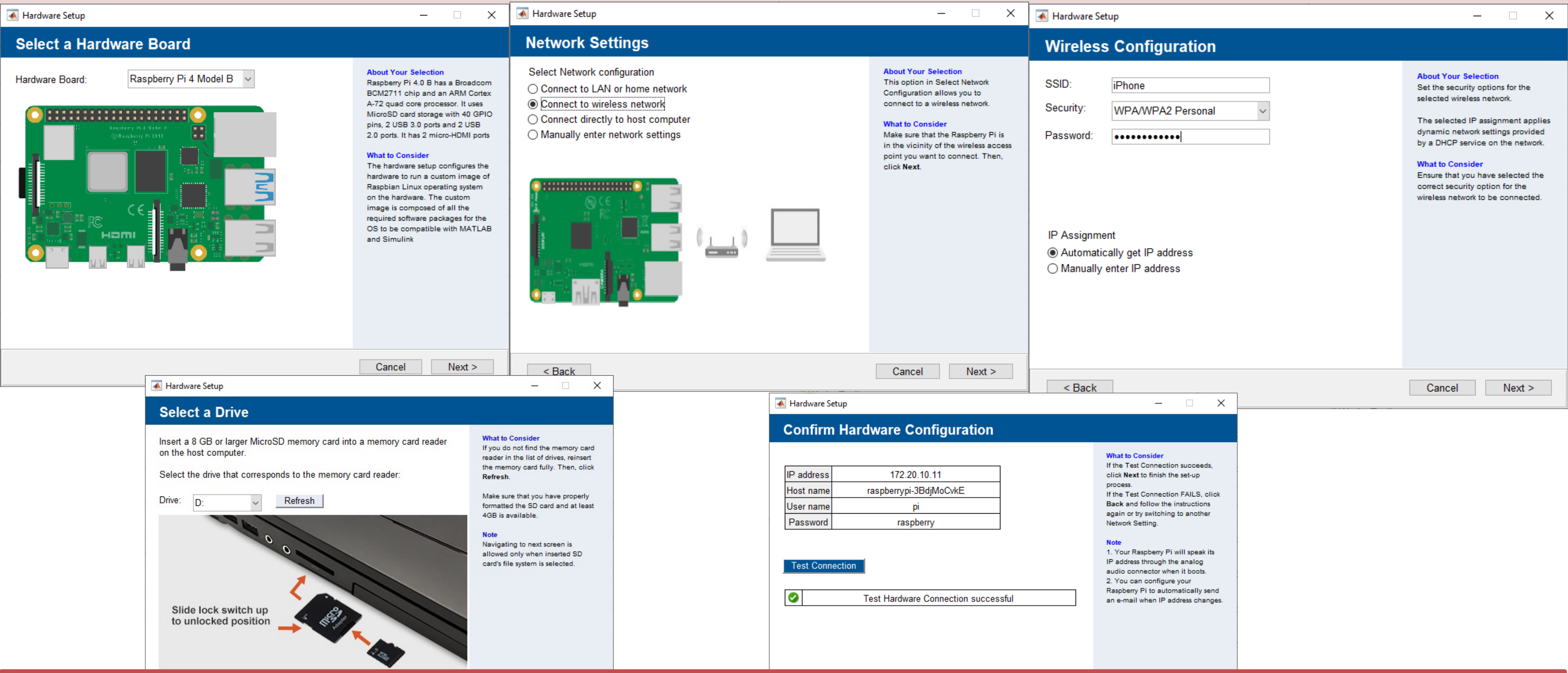


MATLAB Support Package for Raspberry Pi



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

Hardware Setup



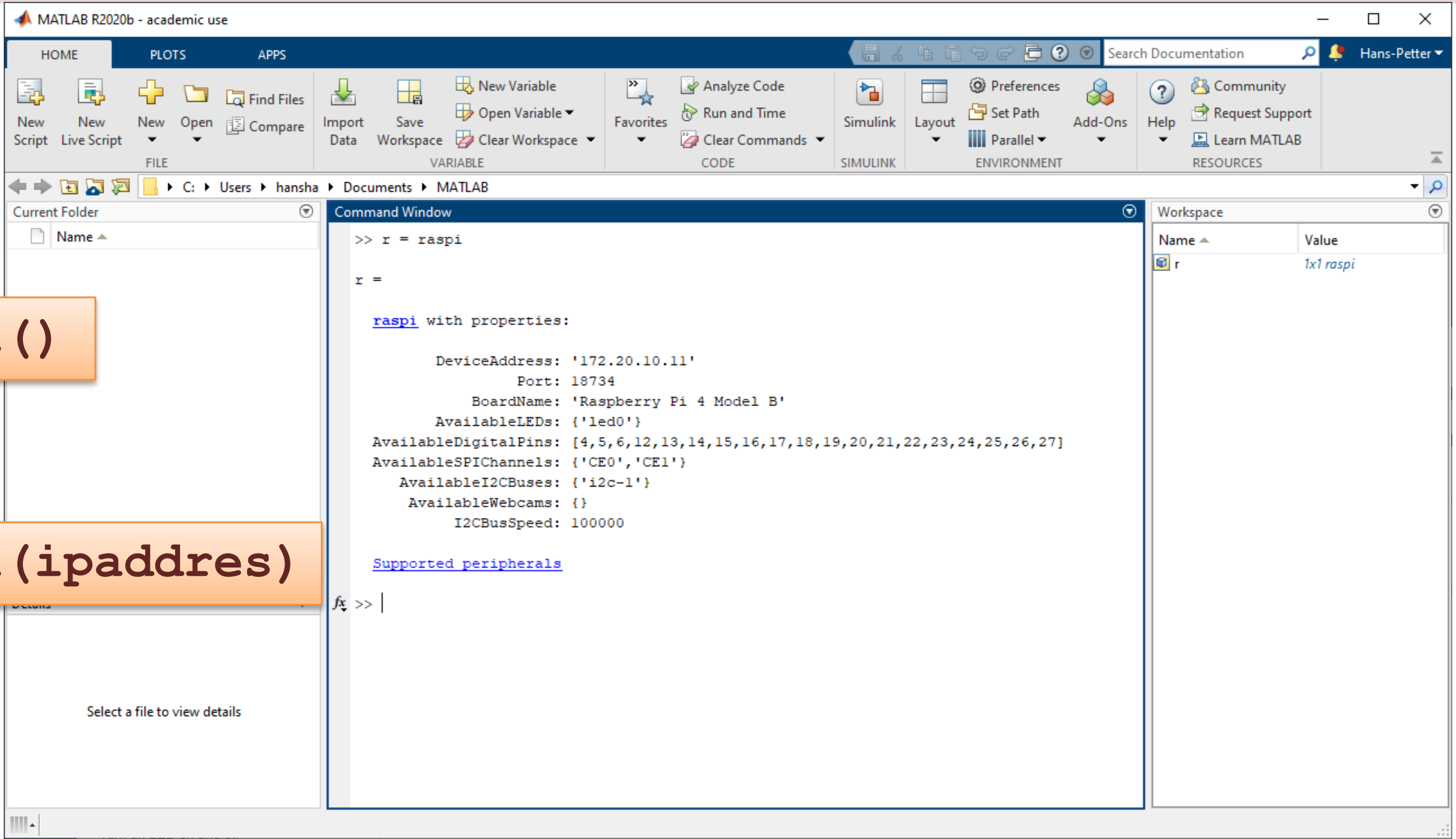
Getting Started with MATLAB Support Package for Raspberry Pi: https://youtu.be/32ByiUdOwsw

Test Hardware

```
r = raspi()
```

or

```
r = raspi(ipaddres)
```

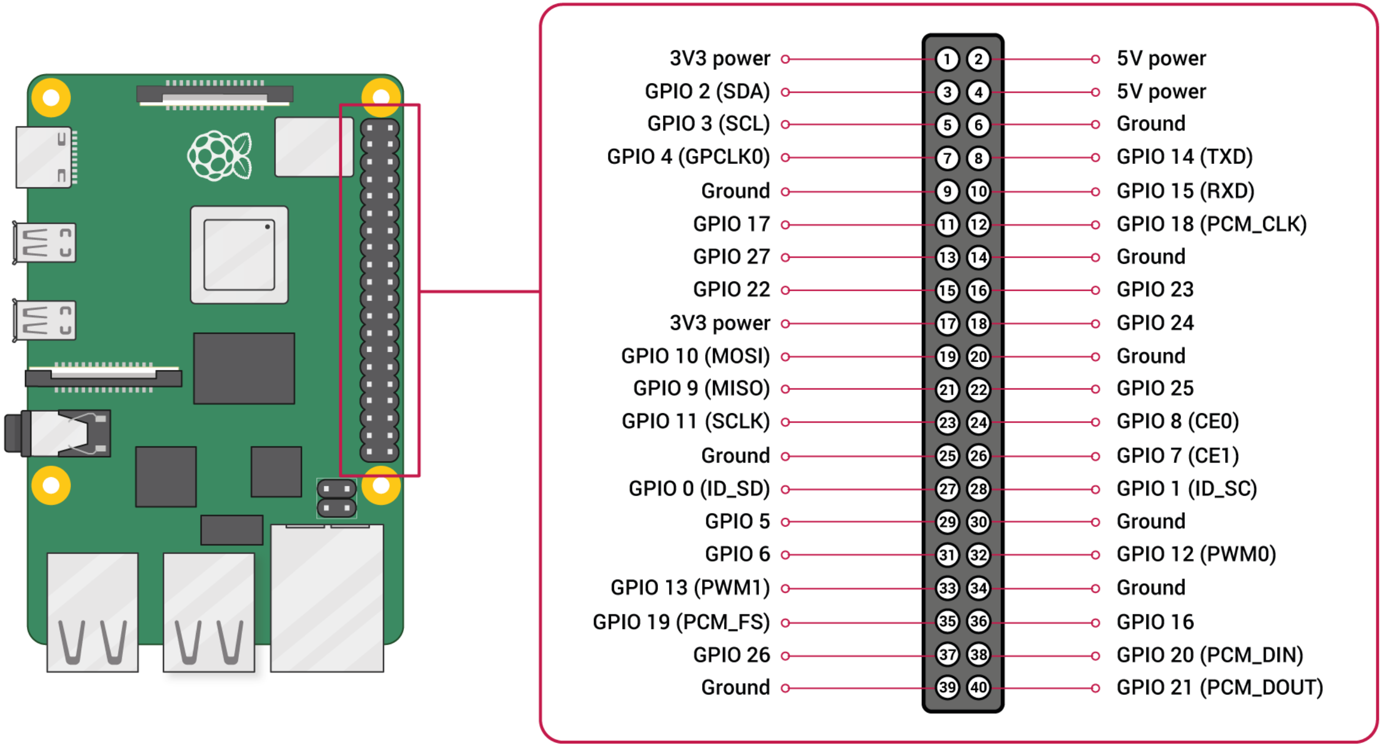


Raspberry Pi GPIO

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 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)

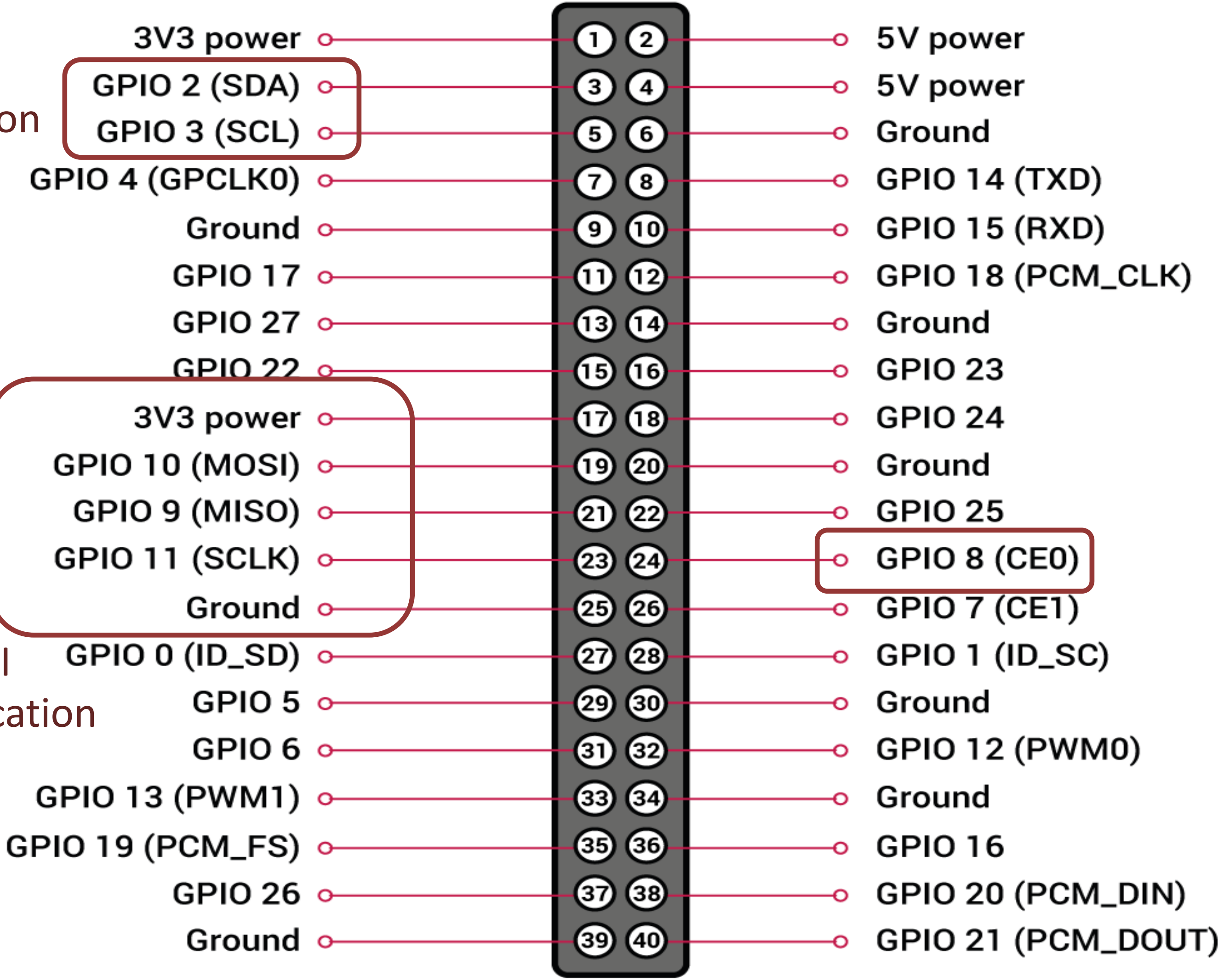
Digital Buses (for reading data from Sensors, etc.):

- **I2C**
- **SPI**

GPIO

Pins for I2C Communication

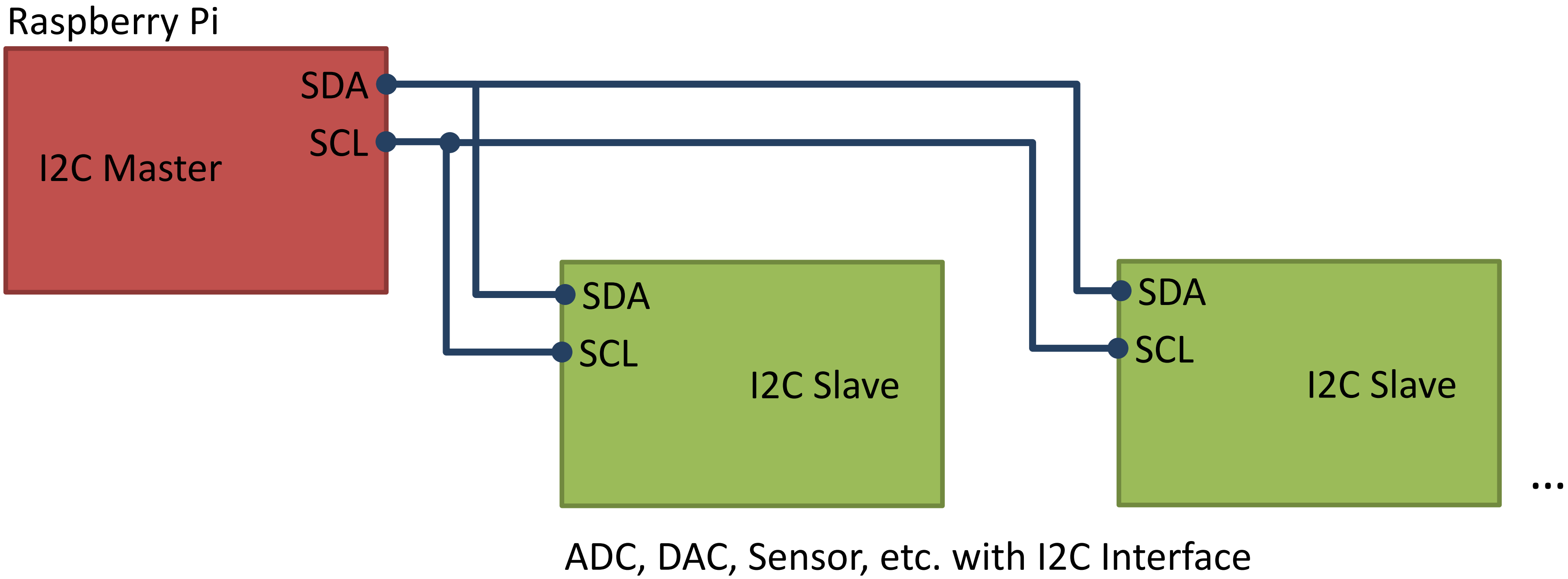
Pins for SPI Communication



I2C

I2C

Multiple devices can be connected to the I2C pins on the Raspberry Pi
Master – Device that generates the clock and initiates communication with slaves
Slave – Device that receives the clock and responds when addressed by the master.



Access I2C on Raspberry Pi

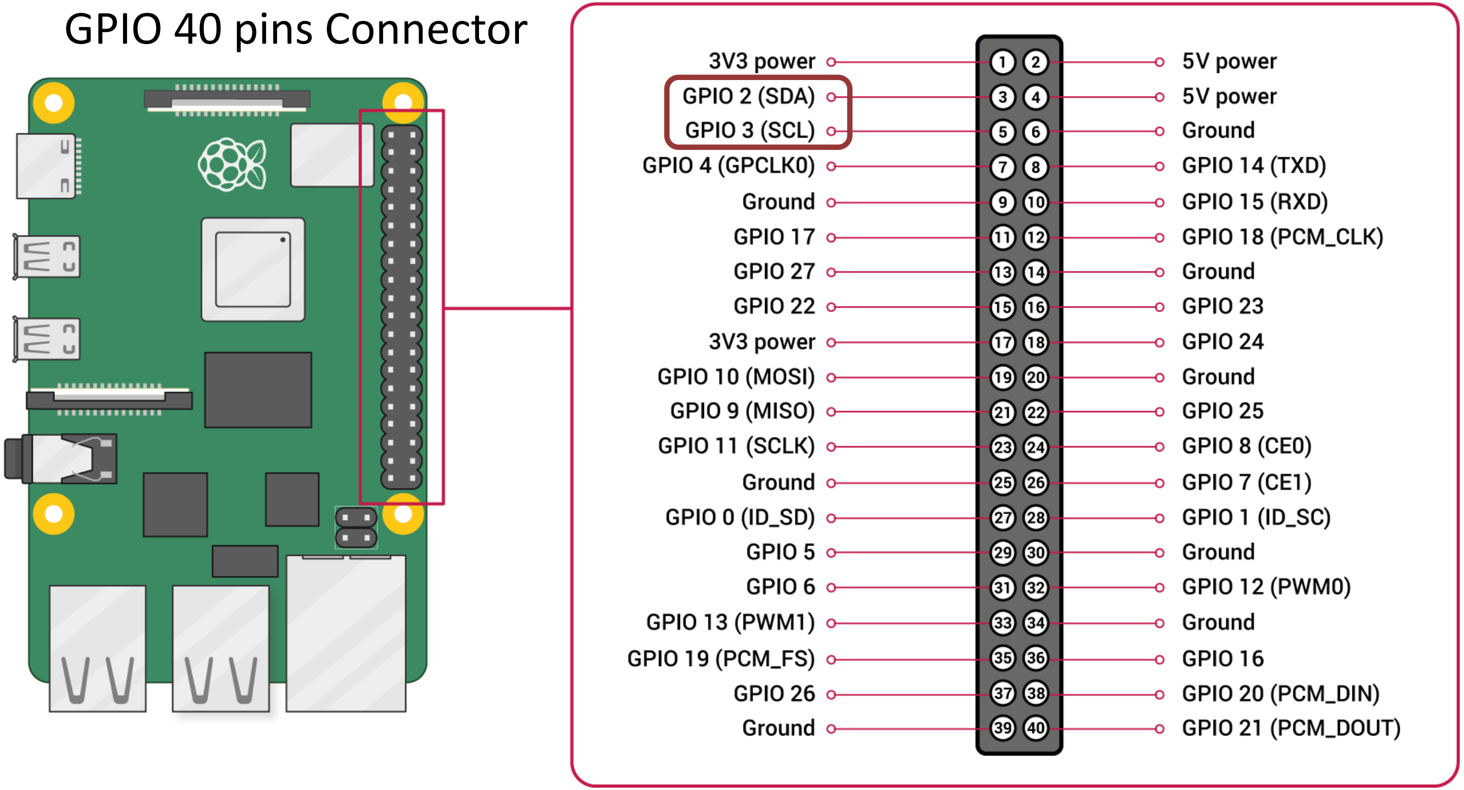
enableI2C(mypi) enables the I2C bus at its default bus speed of 100000 bps.

You can then use the Raspberry Pi hardware board pins I2C1_SDA (GPIO 2) and I2C1_SCL (GPIO 3) as I2C pins and interface the Raspberry Pi board with any I2C device to exchange data.

The I2C bus is enabled by default. To disable I2C, use disableI2C.

I2C Wiring on Raspberry Pi

GPIO 40 pins Connector



Note! The I2C pins include a fixed 1.8 kΩ pull-up resistor to 3.3v.

MATLAB - I2C Interface

I2C Interface

Use the Raspberry Pi™'s I2C interface

Objects

raspi	Connection to Raspberry Pi board
i2cdev	Connection to device on Raspberry Pi hardware

Functions

scanI2CBus	Scan I2C bus device addresses
read	Read data from I2C device
write	Write data to I2C device
readRegister	Read from register on I2C device
writeRegister	Write to register on I2C device
enableI2C	Enable I2C interface
disableI2C	Disable I2C interface

https://se.mathworks.com/help/supportpkg/raspberrypiio/i2c-interface.html?s_tid=CRUX_lftnav

Get Raspberry Pi Information

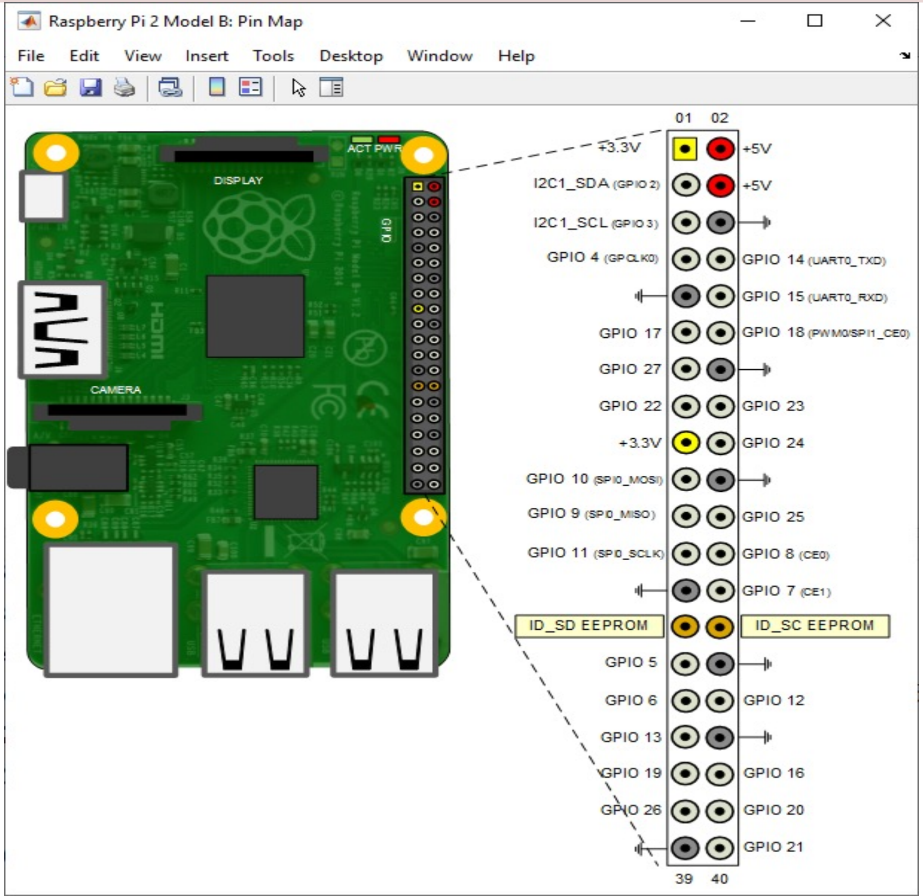
```
clear rpi;
rpi = raspi();
```

Command Window

```
raspi with properties:

         DeviceAddress: '192.168.137.125'
                  Port: 18734
             BoardName: 'Raspberry Pi 2 Model B'
         AvailableLEDs: {'led0'}
  AvailableDigitalPins: [4,5,6,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]
  AvailableSPIChannels: {'CE0','CE1'}
     AvailableI2CBuses: {'i2c-1'}
      AvailableWebcams: {}
           I2CBusSpeed: 100000

Supported peripherals
```

Get Raspberry Pi Information

```
clear rpi;
rpi = raspi()

showPins(rpi);
```



Get I2C Address

Get the addresses of I2C devices that are attached to the I2C bus

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

scanI2CBus(rpi,'i2c-1')
```

Command Window

```
>> i2c_get_devices

ans =

  1×1 cell array

    {'0x48'}
```

Basic Read Example

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

i2caddress = '0x48';
device = i2cdev(rpi,'i2c-1', i2caddress);

value = read(device,1);
disp(value);
```

Basic Write Example

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

i2caddress = '0x62';
device = i2cdev(rpi,'i2c-1', i2caddress);

value = 4092;
write(device, value);
```

Read Register

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

i2caddress = '0x48';
device = i2cdev(rpi,'i2c-1', i2caddress);

data = readRegister(device,14);
disp(data);
```

Read the value of register 14 from the I2C device

Write Register

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

i2caddress = '0x62';
device = i2cdev(rpi,'i2c-1', i2caddress);

data = hex2dec('08');
type = 'uint8'
writeRegister(device,3,data,type)
```

Write a scalar hexadecimal value, hex2dec('08'), to register 3 on the I2C device

TC74

Temperature Sensor with I2C Interface

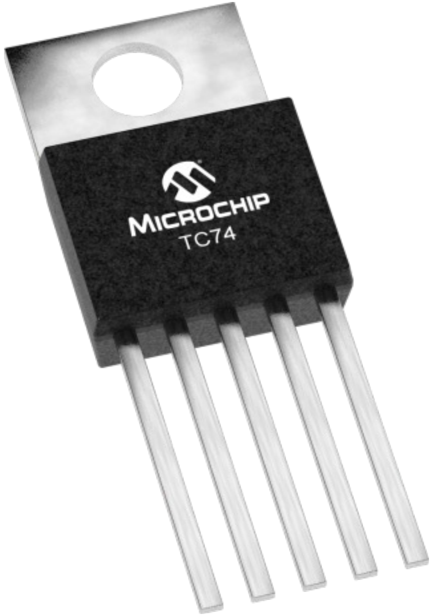
Hans-Petter Halvorsen

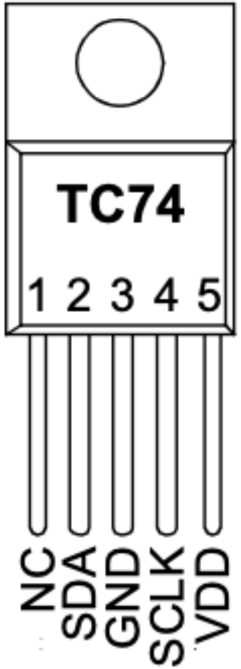
Table of Contents

TC74 Temperature Sensor

I2C Interface

TC74A0-5.0VAT

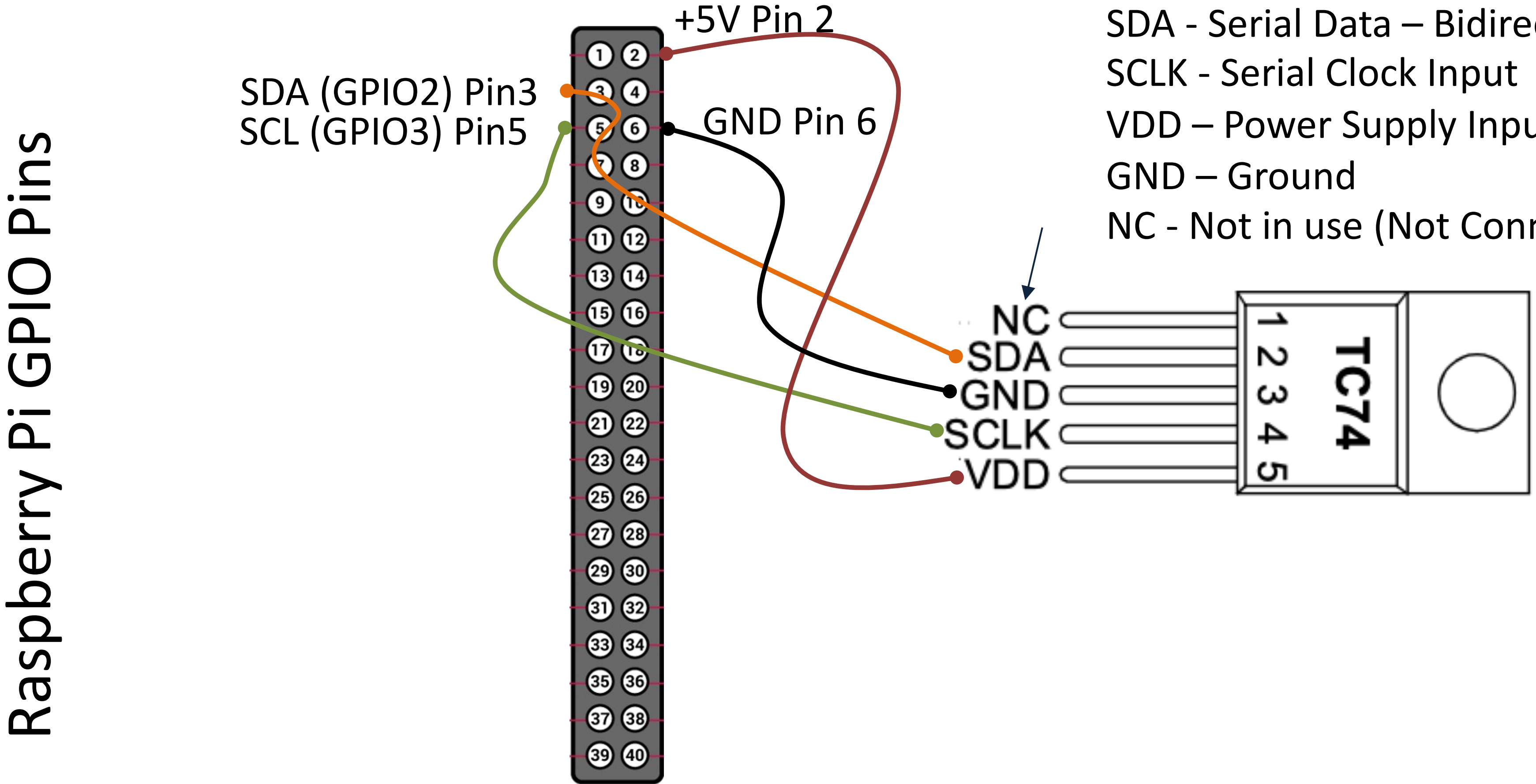




- The TC74 acquires and converts temperature information from its onboard solid-state sensor with a resolution of ±1°C.
- It stores the data in an internal register which is then read through the serial port.
- The system interface is a slave SMBus/I2C port, through which temperature data can be read at any time.

Datasheet: https://ww1.microchip.com/downloads/en/DeviceDoc/21462D.pdf

TC74 Wiring



SDA - Serial Data – Bidirectional

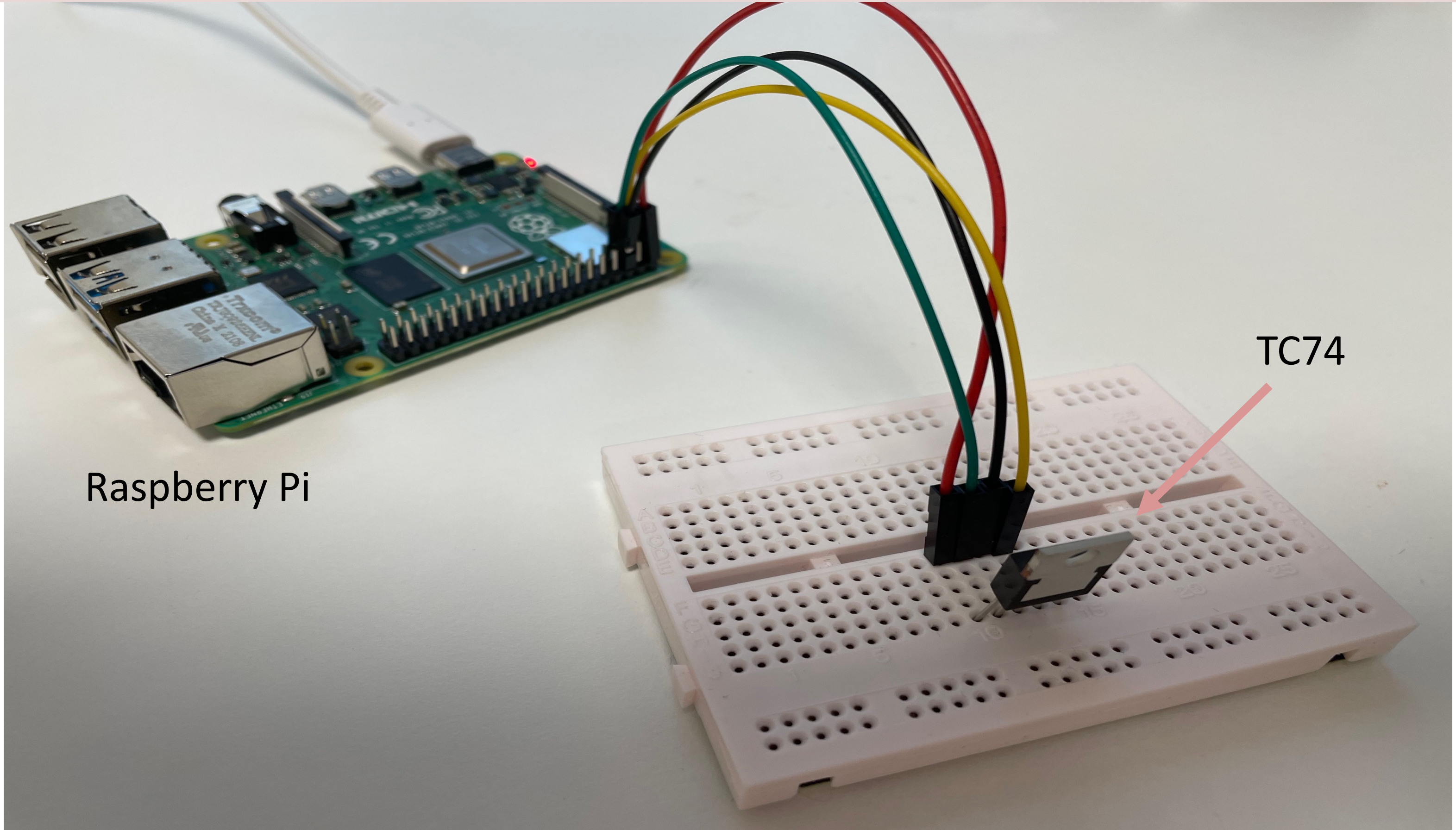
SCLK - Serial Clock Input

VDD – Power Supply Input

GND – Ground

NC - Not in use (Not Connected)

Wiring

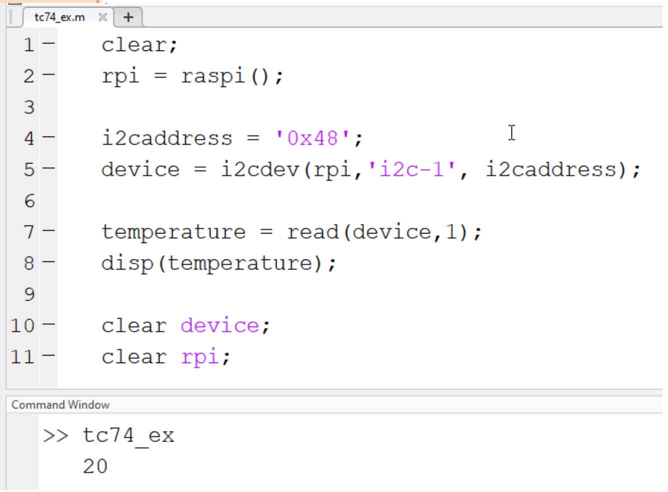


Read Temperature Example

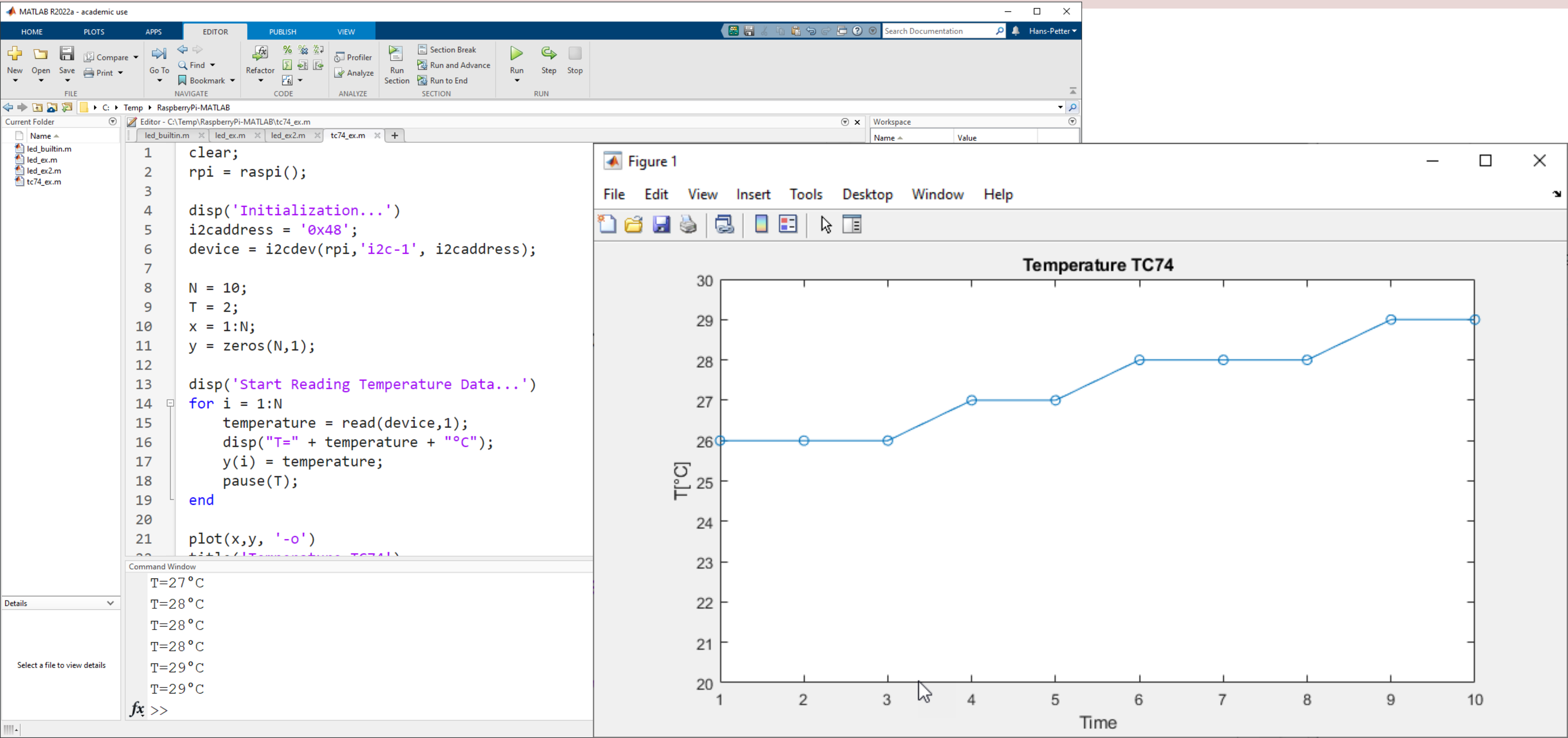
You read the temperature value directly as a number in degrees Celsius. No conversion or anything is necessary

```
clear;
rpi = raspi();

i2caddress = '0x48';
device = i2cdev(rpi,'i2c-1', i2caddress);

temperature = read(device,1);
disp(temperature);

clear device;
clear rpi;
```

Read 1 Byte



Reading Temperature and Plotting



```
clear;
rpi = raspi();

disp('Initialization...')
i2caddress = '0x48';
device = i2cdev(rpi,'i2c-1', i2caddress);

N = 10;
T = 2;
x = 1:N;
y = zeros(N,1);

disp('Start Reading Temperature Data...')
for i = 1:N
    temperature = read(device,1);
    disp("T=" + temperature + "°C");
    y(i) = temperature;
    pause(T);
end

plot(x,y, '-o')
title('Temperature TC74')
xlabel('Time')
ylabel('T[°C]')
xmin=1;xmax=N;ymin=20;ymax=30;
axis([xmin xmax ymin ymax])

clear device;
clear rpi;
```

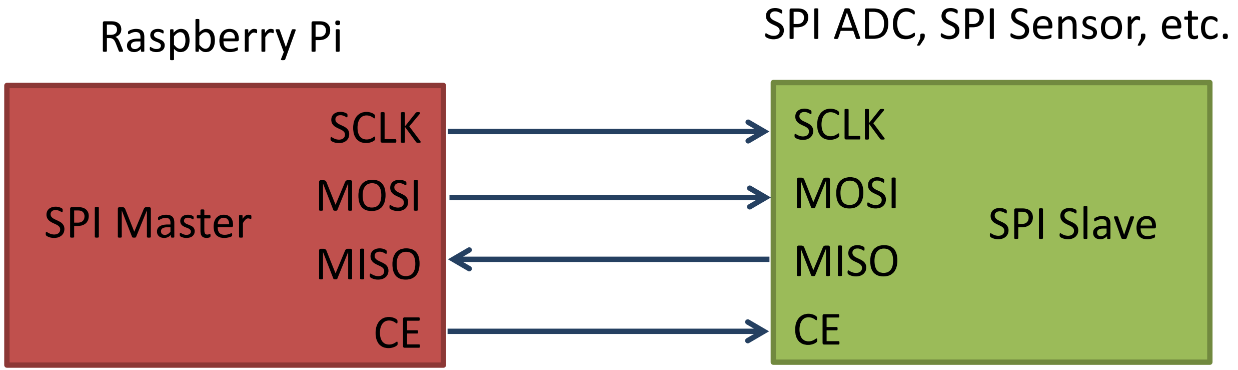
SPI

SPI

- Serial Peripheral Interface (SPI)
- SPI is an interface to communicate with different types of electronic components like Sensors, Analog to Digital Converts (ADC), etc. that supports the SPI interface
- Thousands of different Components and Sensors supports the SPI interface

SPI Interface

SPI devices communicate in full duplex mode using a master-slave architecture with a single master

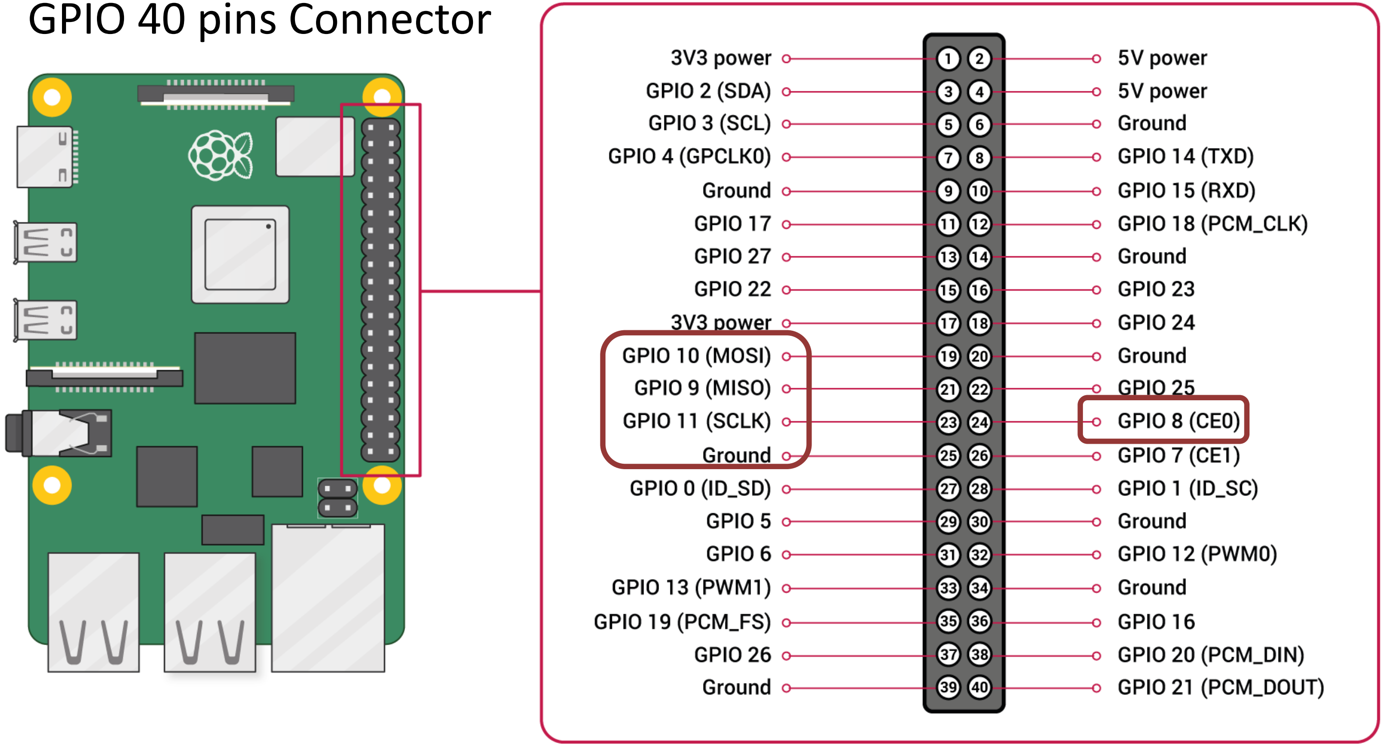


The SPI bus specifies four logic signals:

- **SCLK**: Serial Clock (output from master)
- **MOSI**: Master Out Slave In (data output from master)
- **MISO**: Master In Slave Out (data output from slave)
- **CE** (often also called SS - Slave Select): Chip Select (often active low, output from master)

SPI Wiring on Raspberry Pi

GPIO 40 pins Connector



MATLAB - SPI Interface

https://se.mathworks.com/help/supportpkg/raspberrypiio/spi-interface.html?s_tid=CRUX_lftnav

SPI Interface

Use the Raspberry Pi™'s SPI interface

Objects

raspi	Connection to Raspberry Pi board
spidev	Connection to SPI device on Raspberry Pi hardware

Functions

writeRead	Write data to and read data from SPI device
enableSPI	Enable SPI interface
disableSPI	Disable SPI interface

Find SPI Channels

```
clear rpi
rpi = raspi();
rpi.AvailableSPIChannels
```

Command Window

```
>> rp_ex

ans =

  1×2 cell array

    {'CE0'}    {'CE1'}
```

SPI Example

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

spidevice = spidev(mypi,'CE1',0)

wrdata = [hex2dec('08') hex2dec('D4')]; %Just an example

dataout = writeRead(spidevice, wrdata)
```

Write Data

Read Data

ADC with SPI interface

Analog to Digital Converter

Hans-Petter Halvorsen

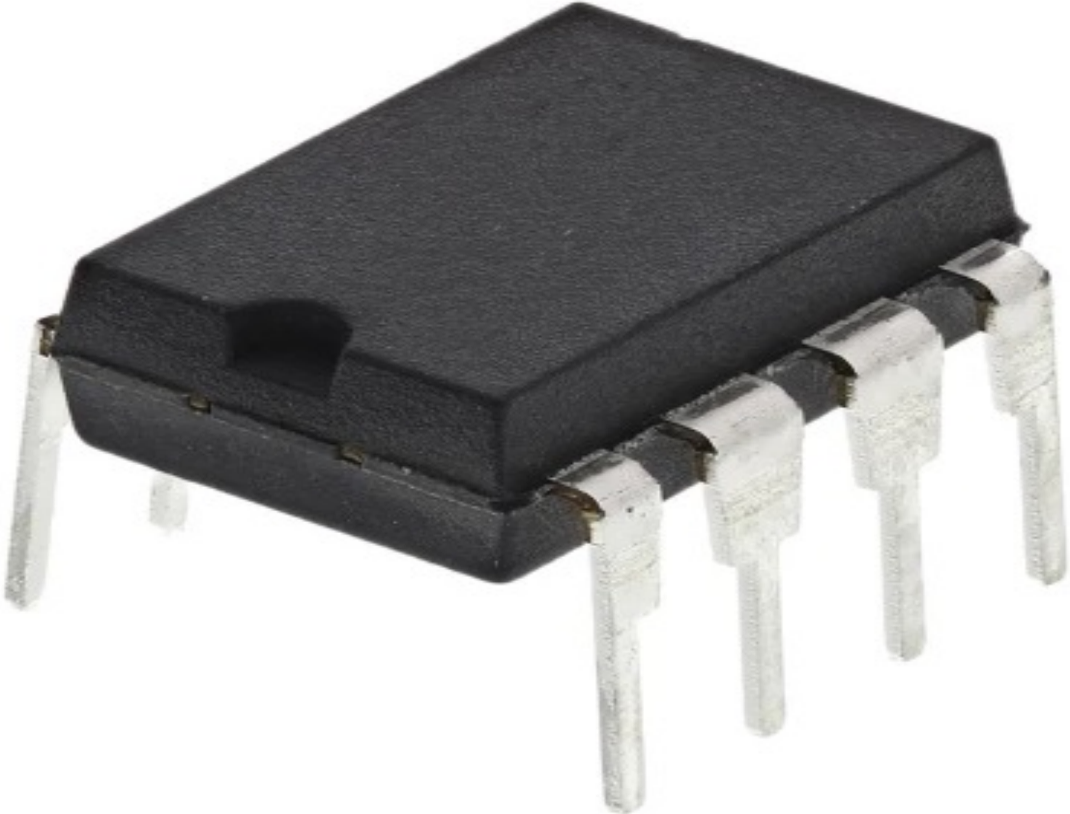
Table of Contents

ADC

- The Raspberry Pi has only Digital pins on the GPIO connector
- If you want to use an Analog electric component or an Analog Sensor together with Raspberry Pi, you need to connect it through an external ADC chip
- ADC – Analog to Digital Converter

https://en.wikipedia.org/wiki/Analog-to-digital_converter

ADC



MCPxxx ADC chip

The MCPxxx family of ADC chips uses a SPI Interface

10-bit analog to digital converters (ADC):

- MCP3008/3208/3304 have 8 channels (0-7)
- MCP3004/3204/3302 have 4 channels (0-3)
- **MCP3002**/3202 have 2 channels (0-1)
- MCP3001/3201/3301 only have 1 channel.

MCP3002 will be used in this Tutorial, but all should work in the same manner

Analog Input Using SPI (MCP3008):

https://se.mathworks.com/help/supportpkg/raspberrypiio/ref/analog-input-using-spi.html

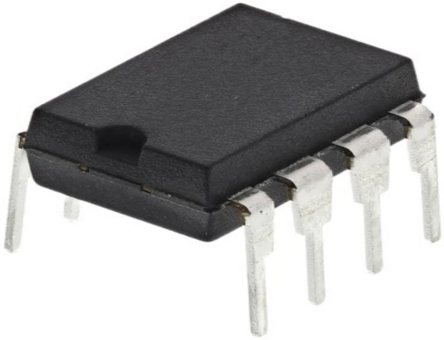
Sensors:

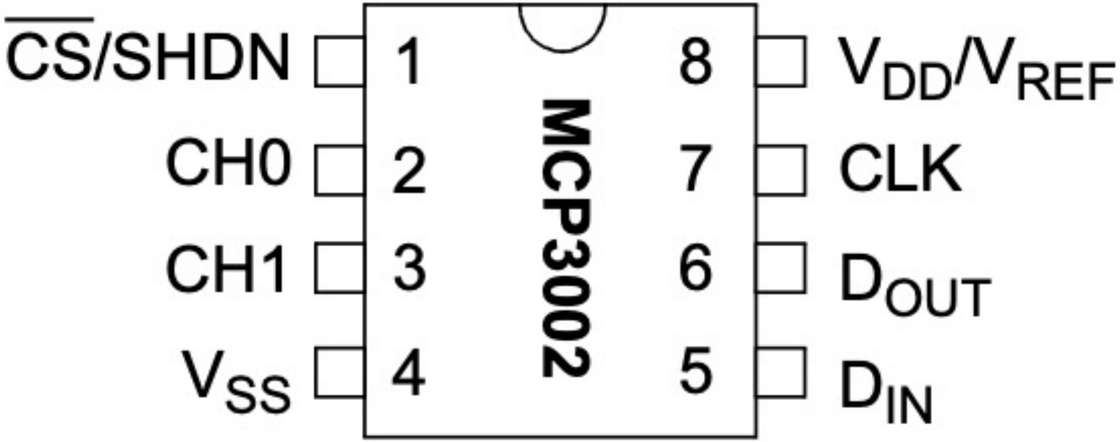
https://se.mathworks.com/help/supportpkg/raspberrypiio/sensors.html?s_tid=CRUX_lftnav

MCP3002 ADC chip

The MCP3002 is a 10-bit analog to digital converter with 2 channels (0-1).

The MCP3002 uses a SPI Interface

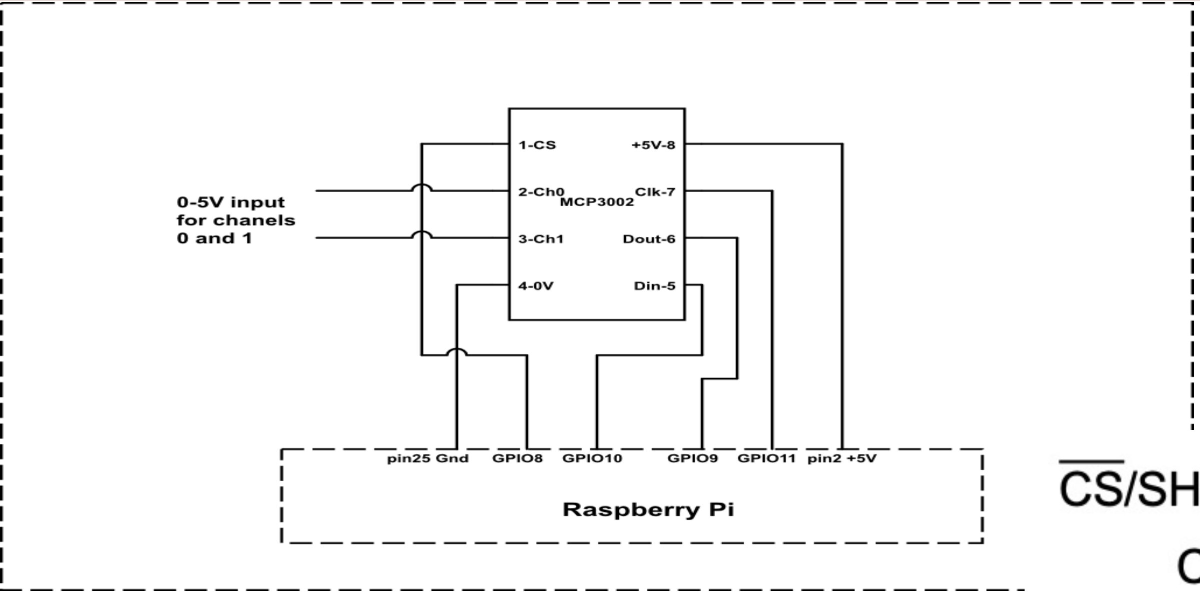


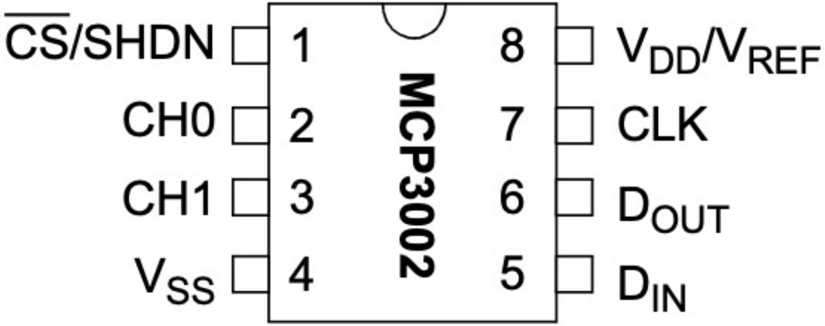


http://ww1.microchip.com/downloads/en/DeviceDoc/21294E.pdf

https://learn.sparkfun.com/tutorials/python-programming-tutorial-getting-started-with-the-raspberry-pi/experiment-3-spi-and-analog-input

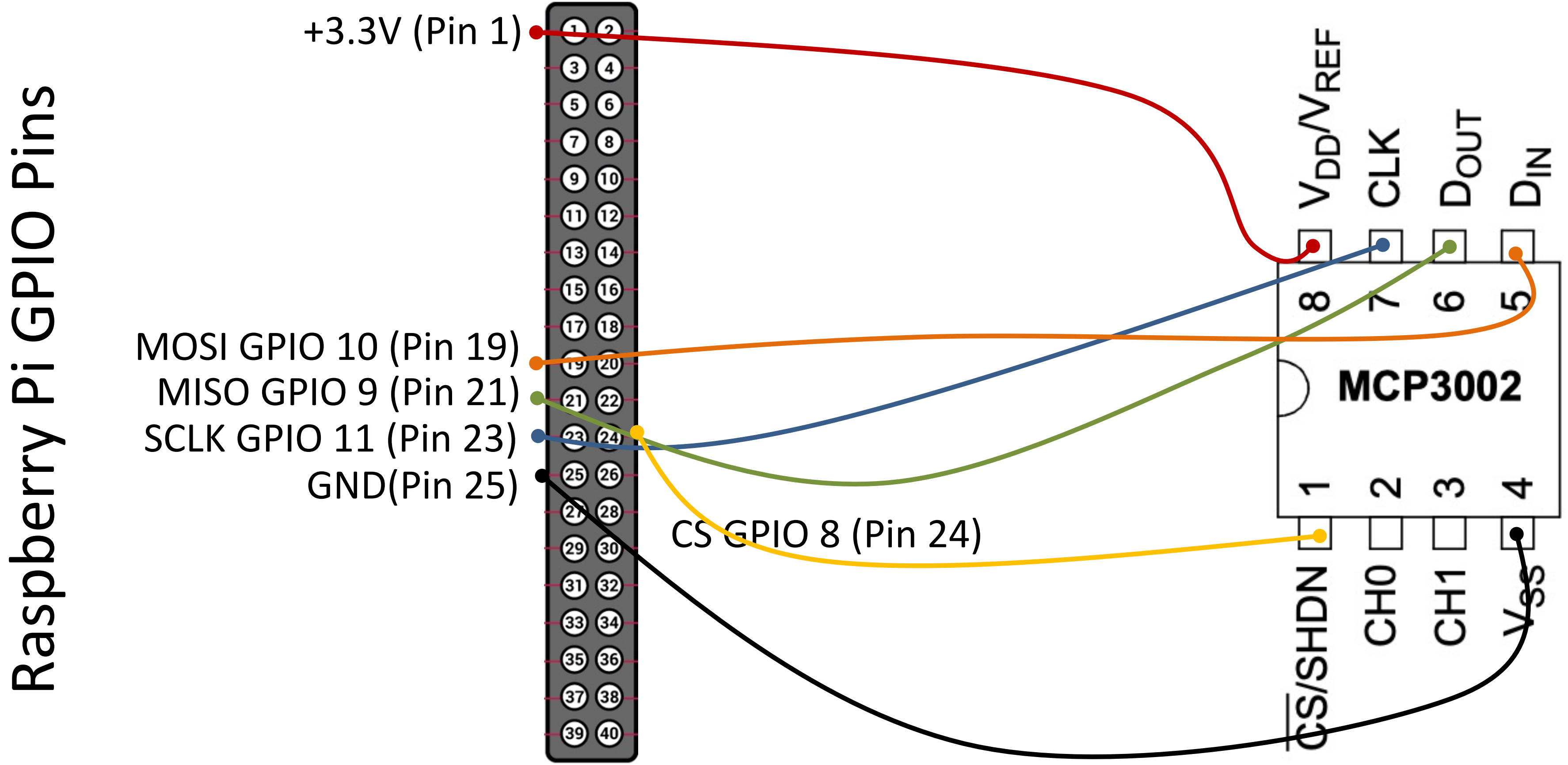
Wiring





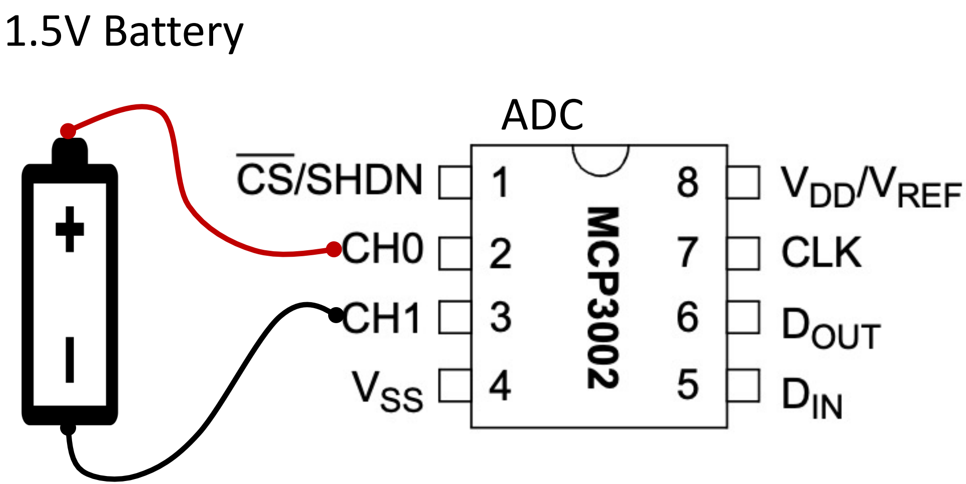
https://sites.google.com/a/joekamphaus.net/raspberry-pi-spi-interface-to-mcp3002/

Wiring


Raspberry Pi GPIO Pins
+3.3V (Pin 1)
MOSI GPIO 10 (Pin 19)
MISO GPIO 9 (Pin 21)
SCLK GPIO 11 (Pin 23)
GND(Pin 25)
CS GPIO 8 (Pin 24)
VDD/VREF
CLK
DOUT
DIN
MCP3002
CS/SHDN
CH0
CH1
VSS

Read Data from ADC

For test purpose we start by wiring a 1.5V Battery to the CH0 (+) and CH1(-) pins on the ADC



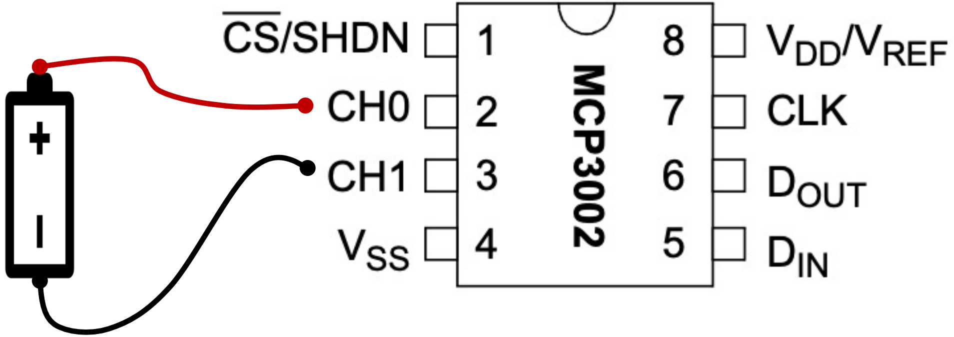
Wire the other pins to the Raspberry Pi according to previous wiring schema

Channel Configuration

From MCP3002 Datasheet

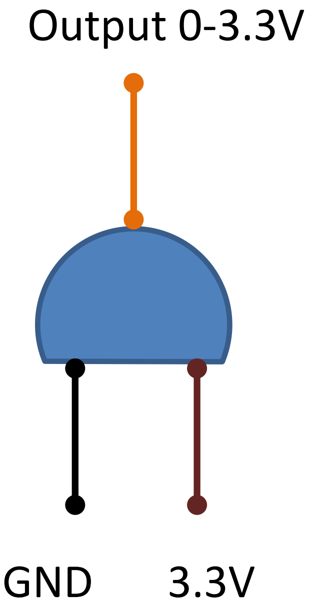
	CONFIG BITS		CHANNEL SELECTION		GND	
	SGL/$\overline{\text{DIFF}}$	ODD/$\overline{\text{SIGN}}$	0	1		
Single-Ended Mode	1	0	+		—	**1000**
	1	1		+	—	**1100**
Pseudo-Differential Mode	0	0	IN+	IN-	—	**0000**
	0	1	IN-	IN+	—	**0100**

For test purpose we can connect a battery, or even better using a **Potentiometer**

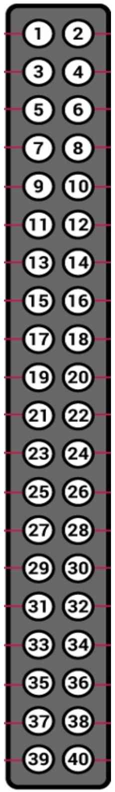


Potentiometer

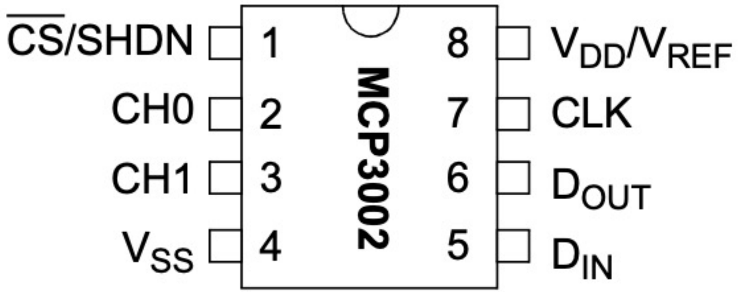
Potentiometer



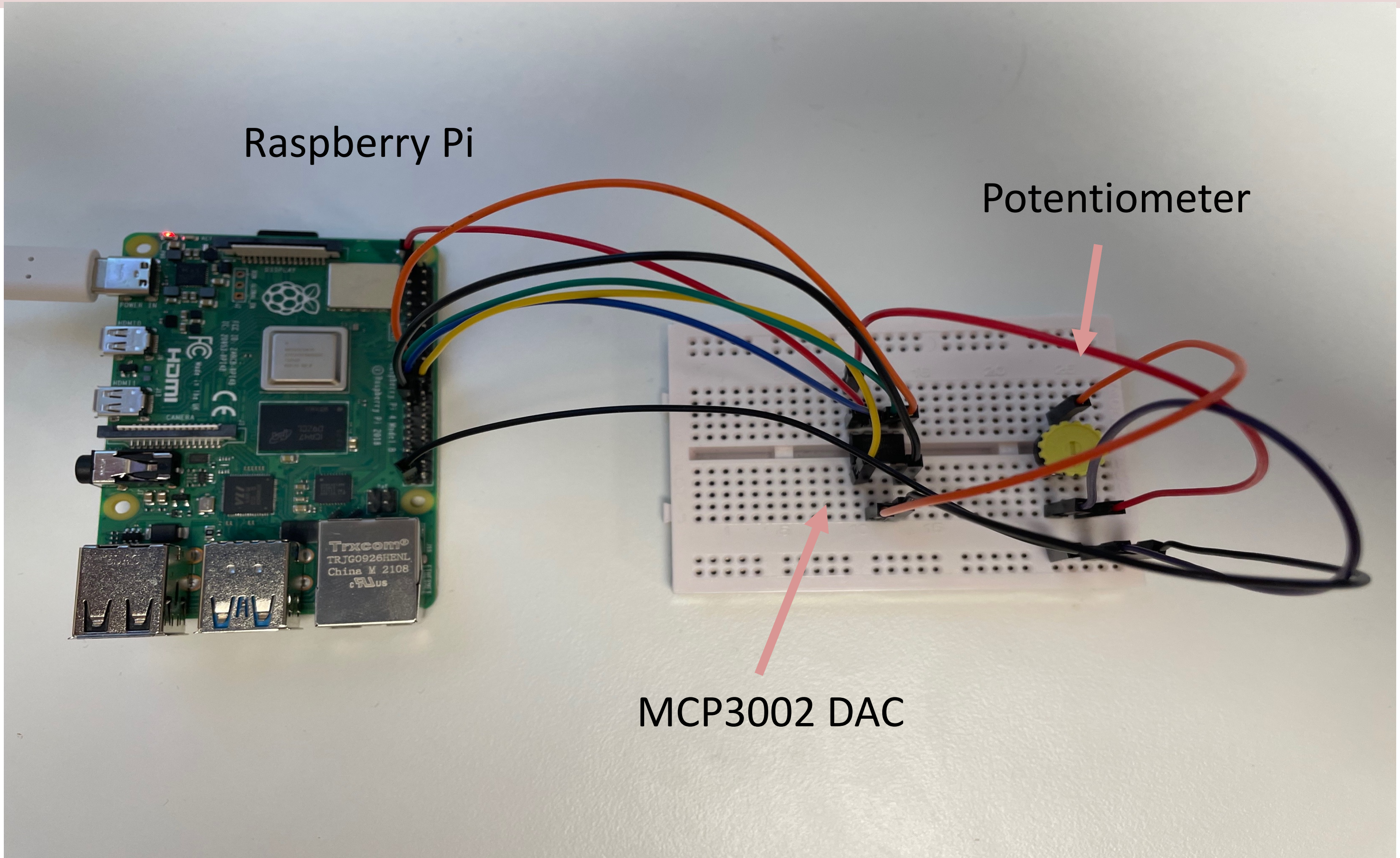
Raspberry Pi



MCP3002 DAC



Wiring



Read Data from ADC

```
clear;
rpi = raspi();
spidevice = spidev(rpi, 'CE0', 0);

data = uint16(writeRead(spidevice,[1, bin2dec('00000000'), 0]));
highbits = bitand(data(2), bin2dec('11'));
adcvalue = double(bitor(bitshift(highbits, 8), data(3)));
voltage = (3.3/1024) * adcvalue;

disp(voltage);

clear spidevice;
Clear rpi;
```

You should improve the Code by making a MATLAB Function, then use the function inside a Loop for continuous readings

https://se.mathworks.com/help/supportpkg/raspberrypiio/ref/analog-input-using-spi.html

Measure Voltage at CH0

- MCP300x uses the SPI interface to communicate with the SPI controller which in this case is Raspberry Pi hardware.
- An SPI transaction between MCP300x and Raspberry Pi consist of 3 bytes.
- Raspberry Pi hardware sends a byte containing a value of '**1**' to MCP300x.
- At the same time, MCP300x sends a “do not care” byte to Raspberry Pi hardware.
- Raspberry Pi hardware sends another byte to the MCP300x with the most significant 4 bits containing a value of '**1000**'. This byte indicates to the MCP300x that a single-ended voltage measurement at CH0 is requested.
- At the same time, MCP300x sends the **bits 9 and 10** of the ADC measurement.
- Finally, Raspberry Pi hardware sends a “do not care” byte and at the same time reads the **least significant 8 bits** of the voltage measurement.
- The 10-bit value read from MCP300x is then converted to a voltage value.

```
clear;
rpi = raspi();
spidevice = spidev(rpi, 'CE0', 0);

N = 20;

for i = 1:N
    data = uint16(writeRead(spidevice,[1, bin2dec('00000000'), 0]));
    highbits = bitand(data(2), bin2dec('11'));
    adcvalue = double(bitor(bitshift(highbits, 8), data(3)));
    %disp(adcvalue);
    voltage = (3.3/1024) * adcvalue;
    value = sprintf('%.2f', voltage);
    disp(value);
    pause(1);
end

clear mcp3002;
clear rpi;
```

But turning on the Potentiometer, you should see the voltage goes from 0V to 3.3V (max)

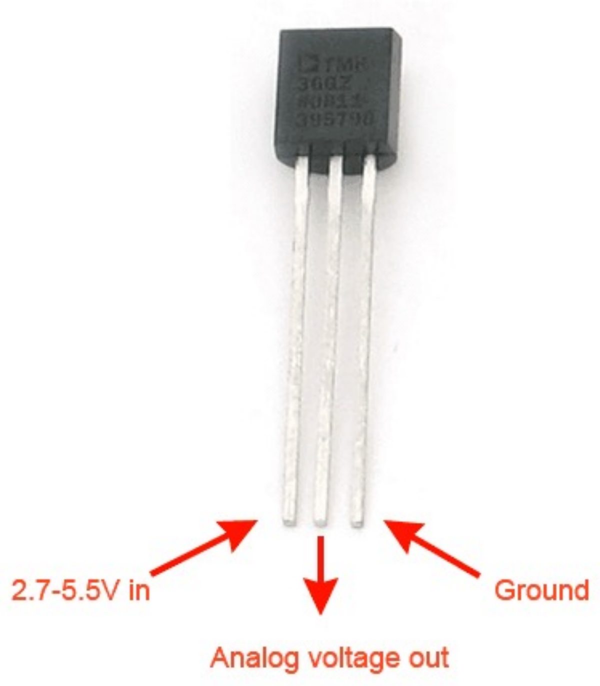
TMP36

Analog Temperature Sensor

Hans-Petter Halvorsen

Table of Contents

TMP36 Temperature Sensor

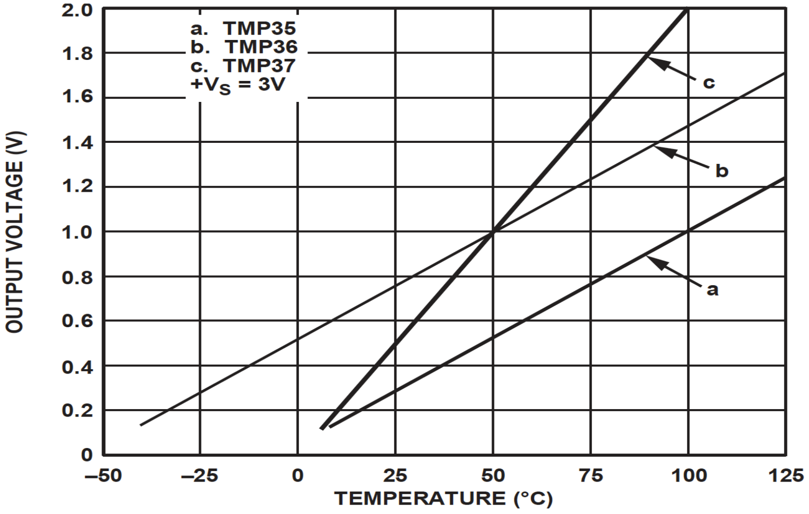


A Temperature sensor like TM36 use a solid-state technique to determine the temperature.

They use the fact as temperature increases, the voltage across a diode increases at a known rate.

https://learn.adafruit.com/tmp36-temperature-sensor

TMP36 Temperature Sensor



Convert form Voltage (V) to degrees Celsius

From the Datasheet we have:

$$(x_1, y_1) = (0.75V, 25°C)$$
$$(x_2, y_2) = (1V, 50°C)$$

There is a linear relationship between Voltage and degrees Celsius:

$$y = ax + b$$

We can find a and b using the following known formula:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

This gives:

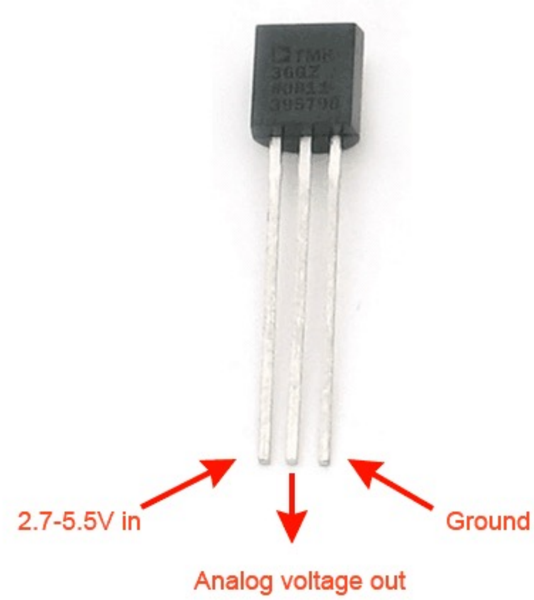
$$y - 25 = \frac{50 - 25}{1 - 0.75}(x - 0.75)$$

Then we get the following formula:

$$y = 100x - 50$$

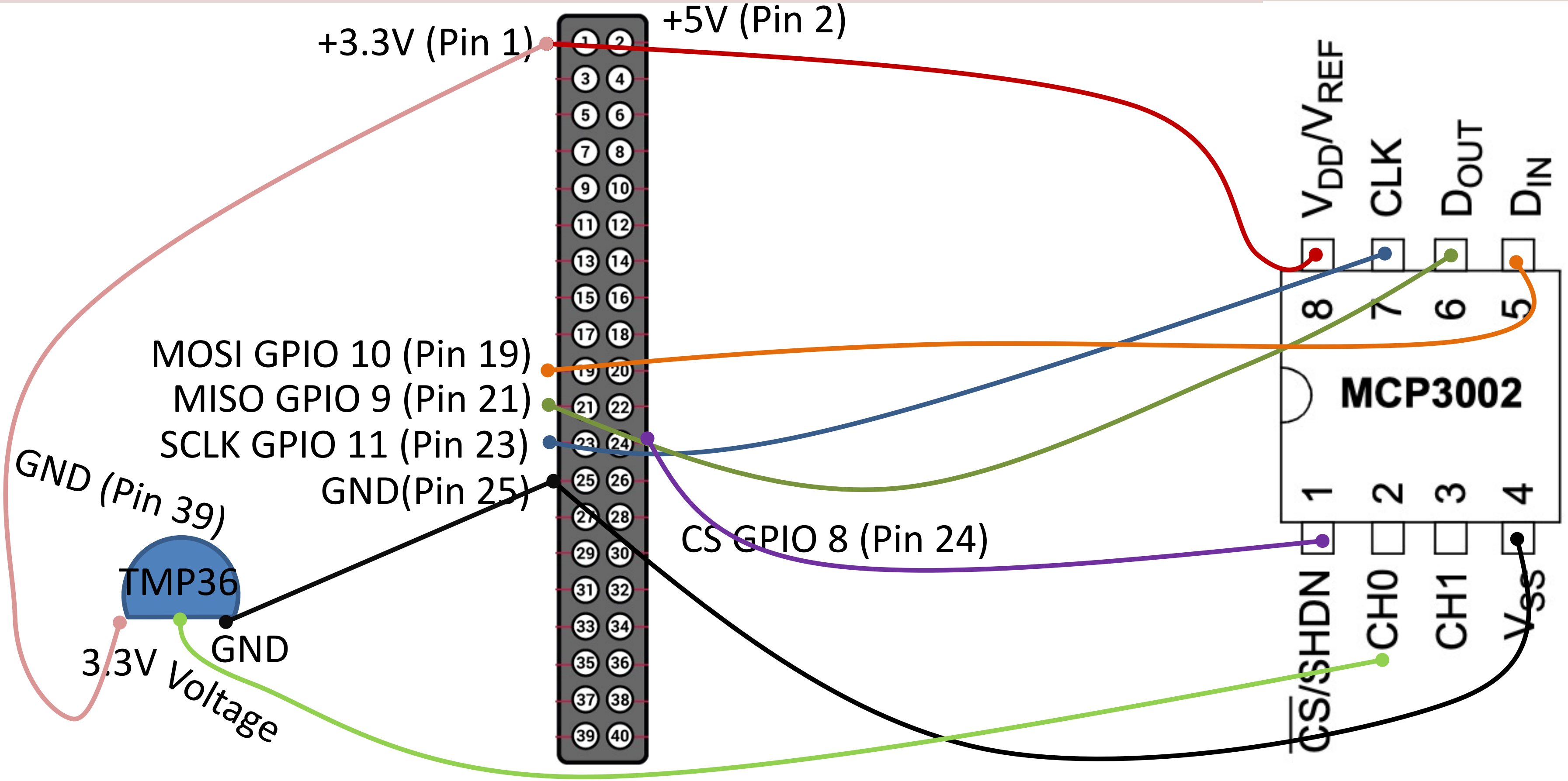
Measure Temperature with an ADC

TMP36 Temperature Sensor

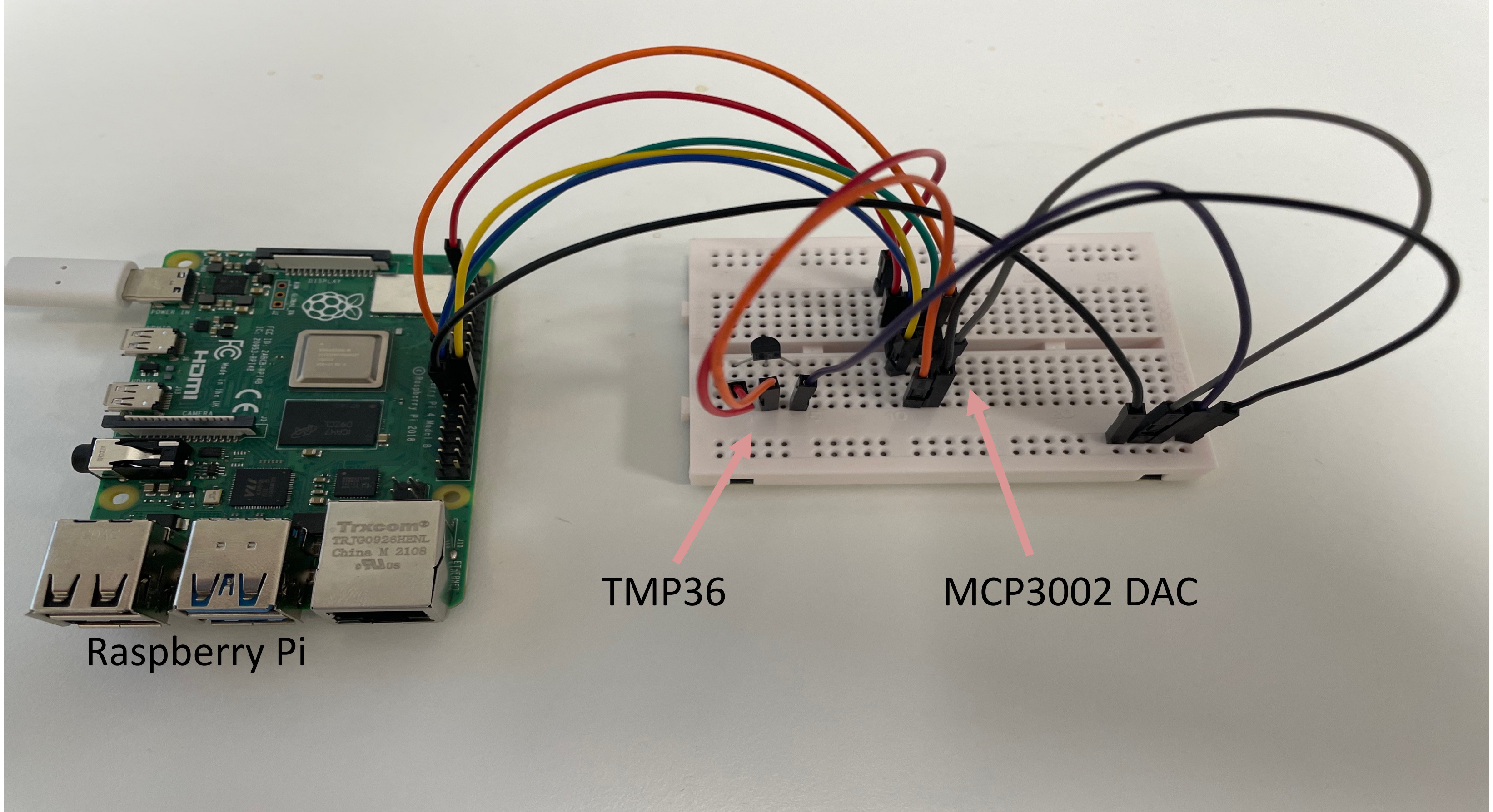


Wire a TMP36 temperature sensor to the first channel of an MCP3002 analog to digital converter and the other pins to +3.3V and GND

Wiring


+3.3V (Pin 1)
+5V (Pin 2)
MOSI GPIO 10 (Pin 19)
MISO GPIO 9 (Pin 21)
SCLK GPIO 11 (Pin 23)
GND(Pin 25)
GND (Pin 39)
CS GPIO 8 (Pin 24)
TMP36
3.3V Voltage
GND
MCP3002
VDD/VREF
CLK
DOUT
DIN
CS/SHDN
CH0
CH1
VSS

Wiring



```
clear;
rpi = raspi();
spidevice = spidev(rpi, 'CE0', 0);

N = 20;
for i = 1:N
    data = uint16(writeRead(spidevice,[1, bin2dec('10000000'), 0]));
    highbits = bitand(data(2), bin2dec('11'));
    adcvalue = double(bitor(bitshift(highbits, 8), data(3)));
    %disp(adcvalue);
    voltage = (3.3/1024) * adcvalue;
    tempC = 100*voltage-50;
    value = sprintf('%.2f', tempC);
    disp(value);
    pause(5);
end

clear spidevice;
clear rpi;
```

Summary

- Raspberry Pi + MATLAB
- MATLAB Support Package for Raspberry Pi
- I2C Overview and Examples
  - **TC74 I2C Temperature Sensor**
- SPI Overview and Examples
  - **ADC MCP3002** + **TMP36** Analog Temperature Sensor

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog