

<https://www.halvorsen.blog>



Raspberry Pi with MATLAB

Using SPI and I2C

Hans-Petter Halvorsen

Contents

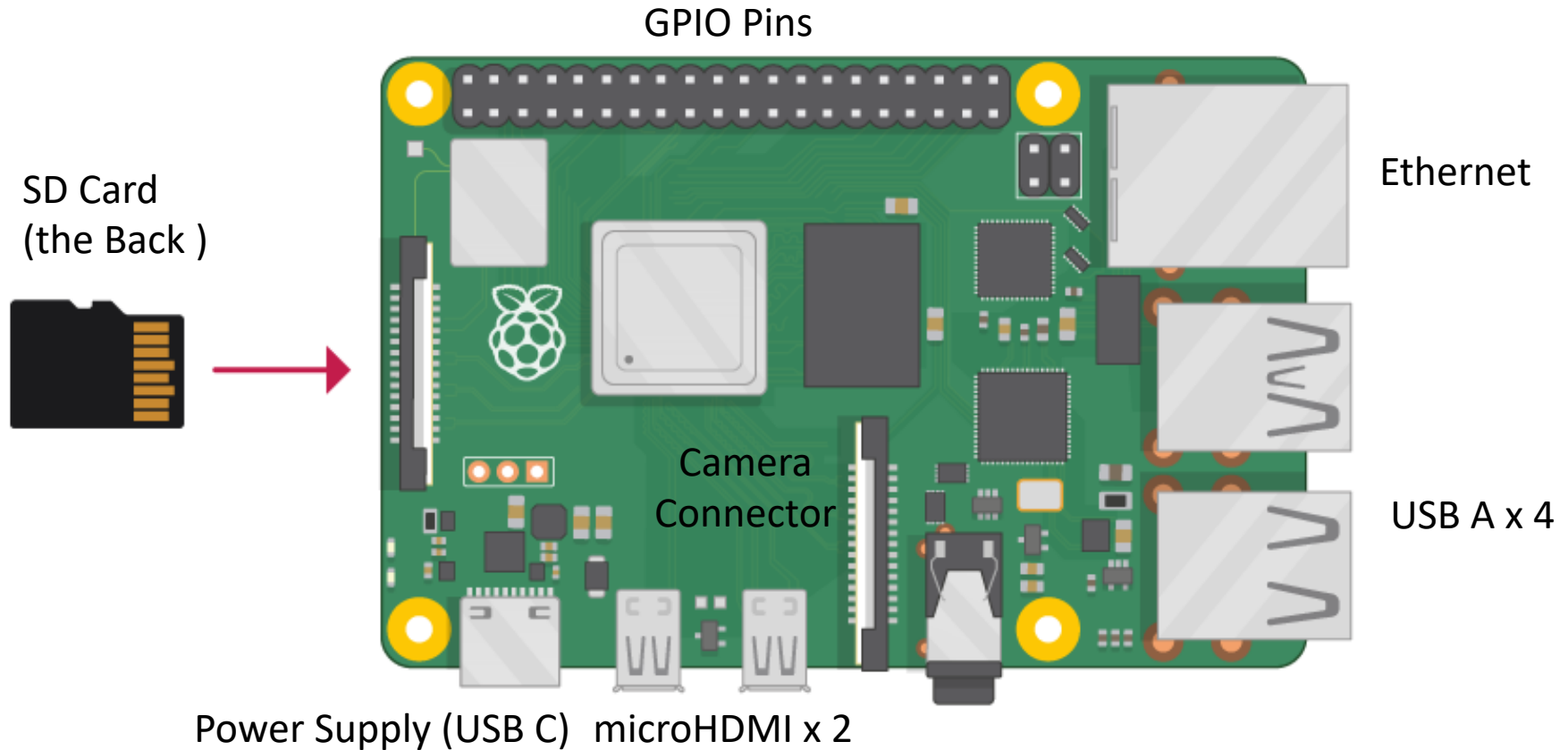
- Raspberry Pi
- MATLAB
- MATLAB Support Package for Raspberry Pi
- Raspberry PI GPIO
- I2C Overview and Examples
 - TC74 I2C Temperature Sensor
- SPI Overview and Examples
 - ADC MCP3002
 - TMP36 Analog Temperature Sensor + ADC MCP3002

These parts are
covered in more detail
in a previous Tutorial



Raspberry Pi

Raspberry Pi





MATLAB

MATLAB

The screenshot displays the MATLAB R2020b environment. The main window is the Editor, showing a script named 'mass_spring_damper_script.m'. The script contains comments and code for a mass-spring-damper simulator. The Workspace window on the right shows the current state of variables, including a 1x1 struct named 'options'. The Command Window at the bottom shows the execution of the script.

```
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_0=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','ode
18
19 %Starting simulation
20 sim('mass_spring_damper', t_s
```

Name	Value
c	4
dxdt_init	0
F_1	4
F_0	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

```
>> mass_spring_damper_script
fx >>
```

- 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)



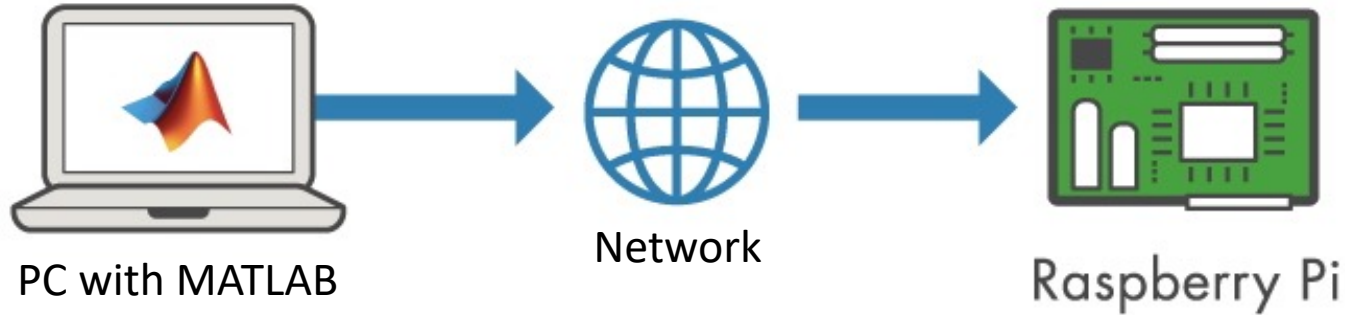
MATLAB Support Package for Raspberry Pi

Hans-Petter Halvorsen

[Table of Contents](#)

Raspberry Pi + MATLAB

MATLAB Support Package for Raspberry Pi

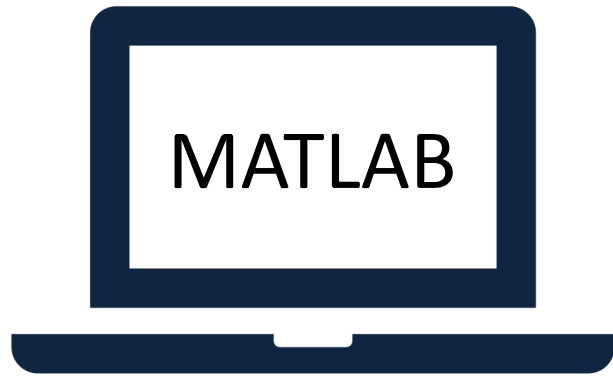


With MATLAB Support Package for Raspberry Pi, the Raspberry Pi is connected to a computer running MATLAB. Processing is done on the computer with MATLAB.

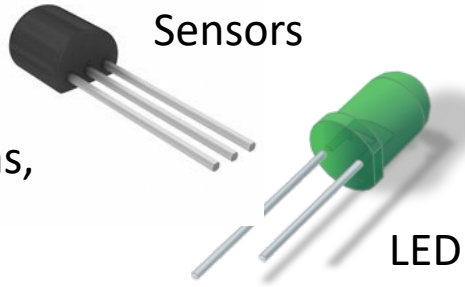
<https://mathworks.com/hardware-support/raspberry-pi-matlab.html>

Raspberry Pi + MATLAB

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



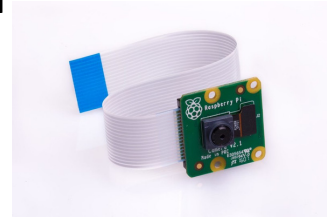
PC



Sensors

LED

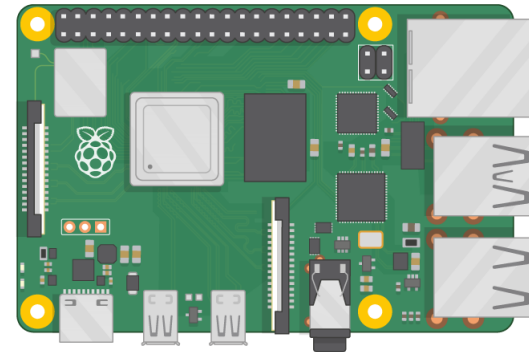
Push Button



Camera



GPIO Pins



Raspberry Pi



MATLAB Support Package for Raspberry Pi

The image shows the MATLAB R2020b interface. The top toolbar includes buttons for 'Add-Ons' and 'Help', which are circled in red. The 'Add-On Explorer' window is open, displaying search results for 'Raspberry Pi'. The search bar at the top of the Add-On Explorer contains the text 'Raspberry Pi'. Three results are listed:

- MATLAB Support Package for Raspberry Pi Hardware** by MathWorks
MATLAB Hardware Team **STAFF**
285 Downloads
Updated 14 Oct 2020
Acquire sensor and image data from your Raspberry Pi.
MATLAB® Support Package for Raspberry Pi™ Hardware enables you to communicate with a Raspberry Pi remotely from a computer running MATLAB. You can acquire data from sensors and imaging devices.
Hardware Support
- Simulink Support Package for Raspberry Pi Hardware** by MathWorks
Simulink Team **STAFF**
185 Downloads
Updated 14 Oct 2020
Run models on Raspberry Pi.
Simulink® Support Package for Raspberry Pi™ Hardware enables you to create and run Simulink models on Raspberry Pi hardware. The support package includes a library of Simulink blocks for configuring
Hardware Support
- Raspberry Pi Hardware Resource Manager** version 1.0 by MathWorks
Simulink Team **STAFF**
89 Downloads
Updated 29 Jul 2019
Monitor the status of different hardware resources on the raspberry pi

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

Hardware Setup

The image displays five sequential screenshots of the Raspberry Pi Hardware Setup application:

- Select a Hardware Board:** Shows the selection of a Raspberry Pi 4 Model B. It includes an image of the board and descriptive text about its specifications (Broadcom BCM2711 chip, ARM Cortex A-72 processor, MicroSD storage, GPIO pins, USB ports, and micro-HDMI ports).
- Network Settings:** Offers three options: Connect to LAN or home network, **Connect to wireless network** (selected), or Connect directly to host computer. It also includes a 'Manually enter network settings' option.
- Wireless Configuration:** Configures the wireless network. Fields include SSID (iPhone), Security (WPA/WPA2 Personal), and Password (masked). It also allows for IP Assignment, with 'Automatically get IP address' selected.
- Select a Drive:** Prompts the user to insert a MicroSD card into a reader. It shows a drive 'D:' selected and a 'Refresh' button. A note indicates that navigating to the next screen is only allowed when the SD card is selected.
- Confirm Hardware Configuration:** Displays the final configuration details: IP address (172.20.10.11), Host name (raspberrypi-3BdMoCwKE), User name (pi), and Password (raspberrypi). A 'Test Connection' button is present, and a status bar shows 'Test Hardware Connection successful'.

Test Hardware

The image shows the MATLAB R2020b interface with the Command Window and Workspace panes. The Command Window displays the output of the command `r = raspi`, which returns a structure `r` containing properties for a Raspberry Pi 4 Model B. The Workspace pane shows the variable `r` as a `1x1 raspi` structure.

```
>> r = raspi

r =

  raspi with properties:

    DeviceAddress: '172.20.10.11'
      Port: 18734
  BoardName: 'Raspberry Pi 4 Model B'
 AvailableLEDs: {'led0'}
 AvailableDigitalPins: [4,5,6,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]
 AvailableSPICannels: {'CE0','CE1'}
 AvailableI2CBuses: {'i2c-1'}
 AvailableWebcams: {}
    I2CBusSpeed: 100000

Supported peripherals

fu >> |
```

Name	Value
r	1x1 raspi

`r = raspi()`

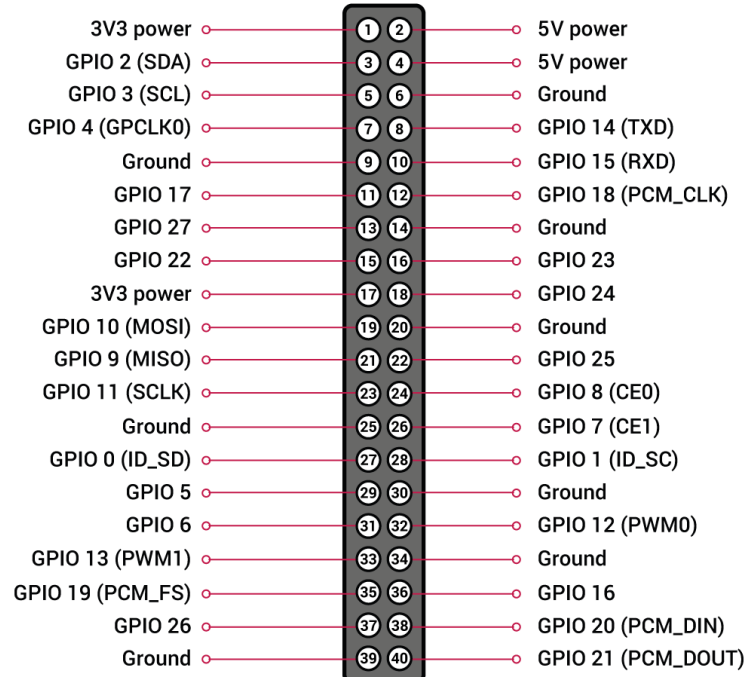
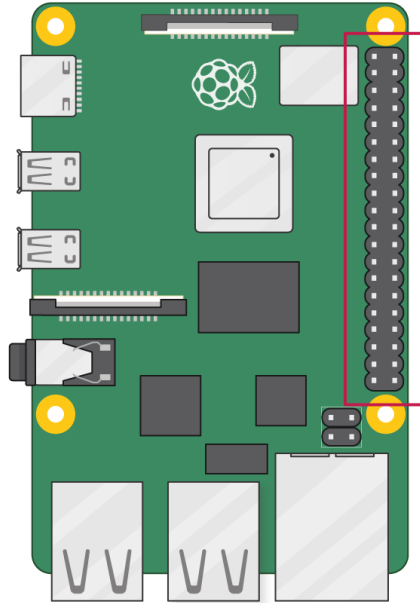
or

`r = raspi(ipaddress)`



Raspberry Pi GPIO

GPIO



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:

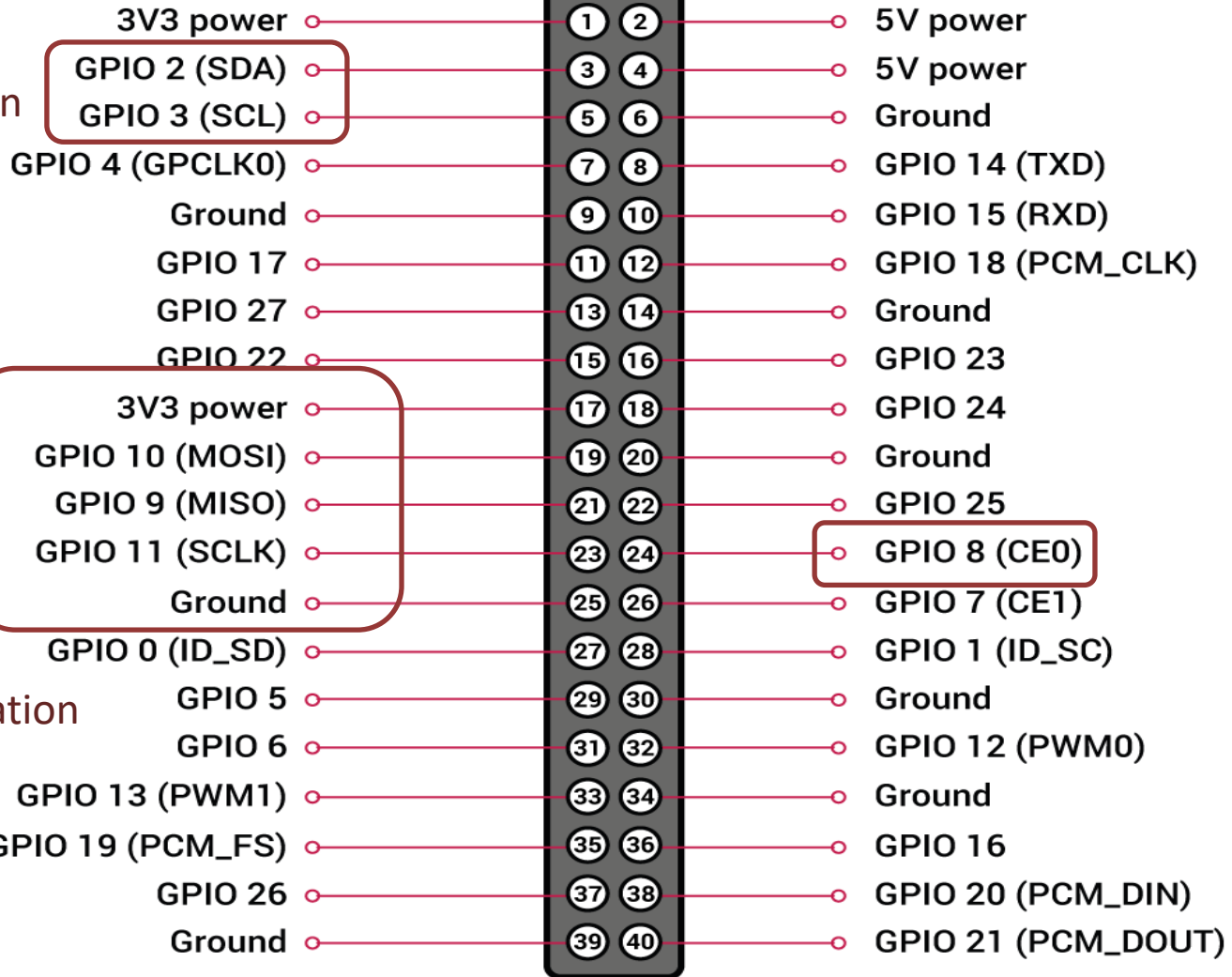
- PWM (Pulse Width Modulation)

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

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

GPIO

Pins for I2C
Communication



Pins for SPI
Communication



I2C

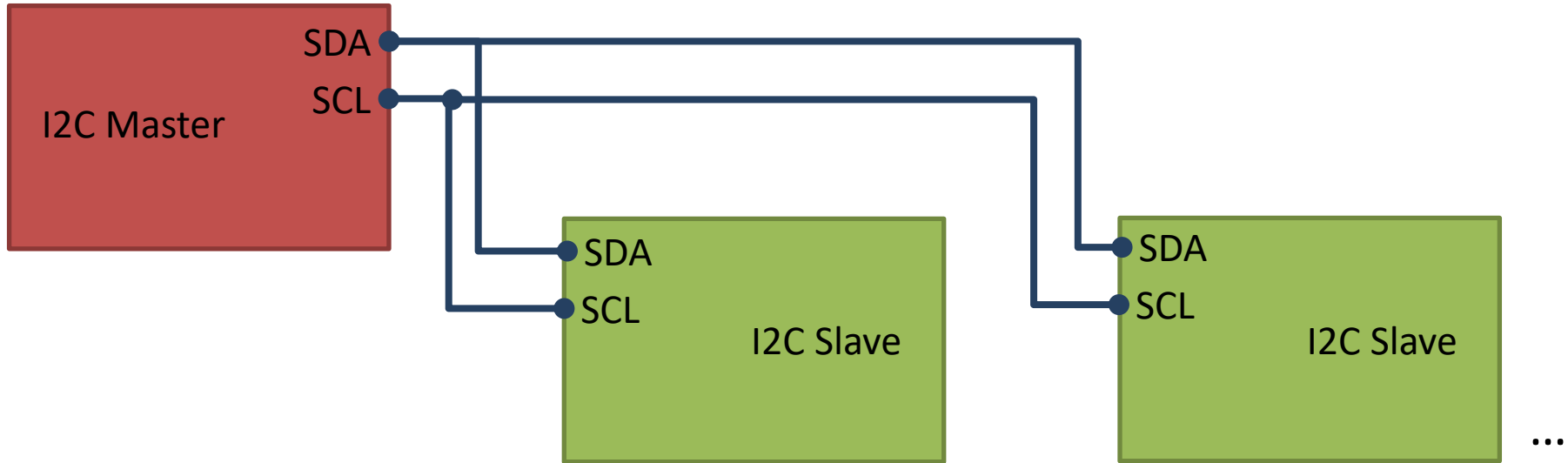
I2C

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.

Raspberry Pi



ADC, DAC, Sensor, etc. with I2C Interface

Access I2C on Raspberry Pi

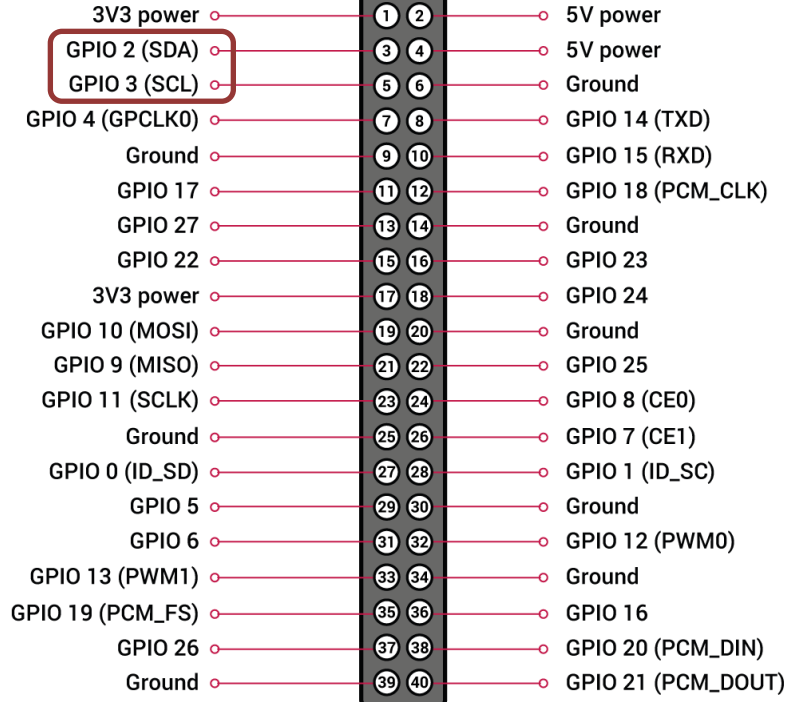
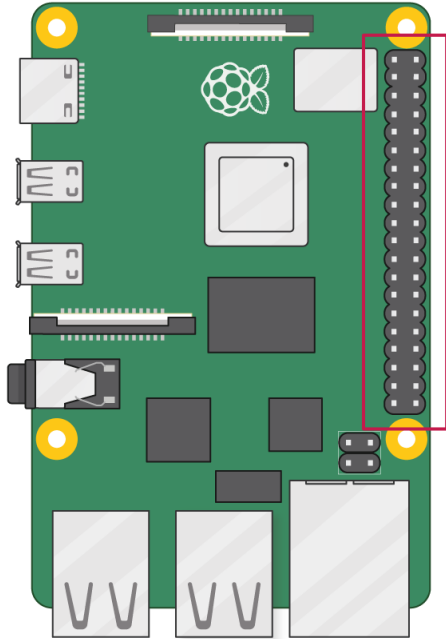
`enableI2C(mypi)` enables the I2C bus at its default bus speed of 100000 bps.

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();
```

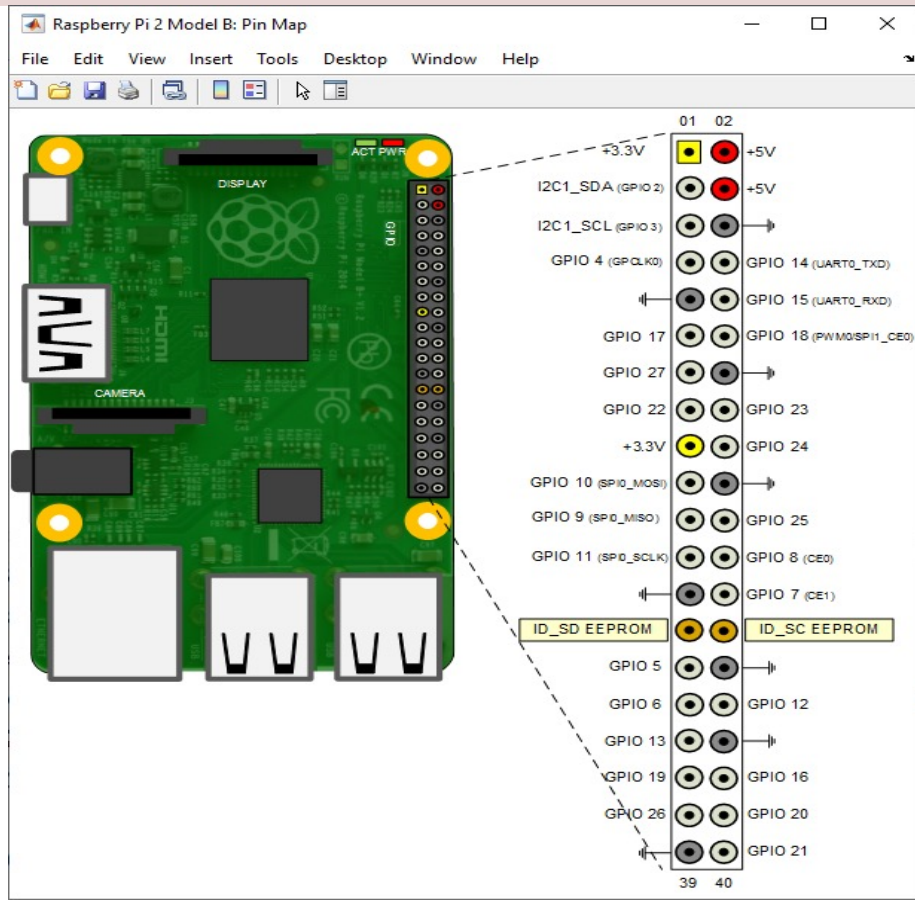
[raspi](#) with properties:

```
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]  
AvailableSPICannels: {'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

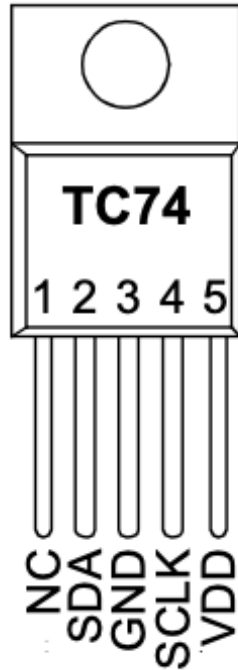
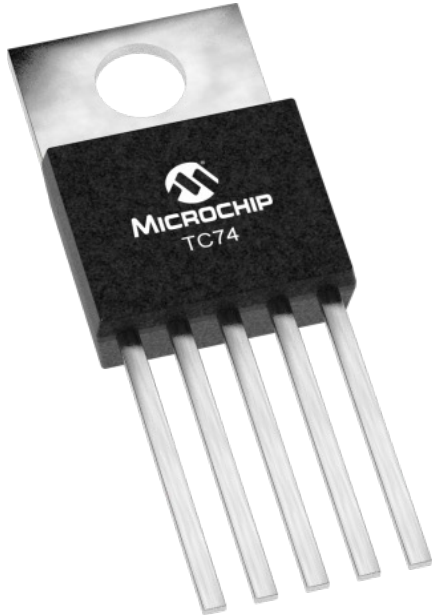
Hans-Petter Halvorsen

[Table of Contents](#)

TC74 Temperature Sensor

I2C Interface

TC74A0-5.0VAT

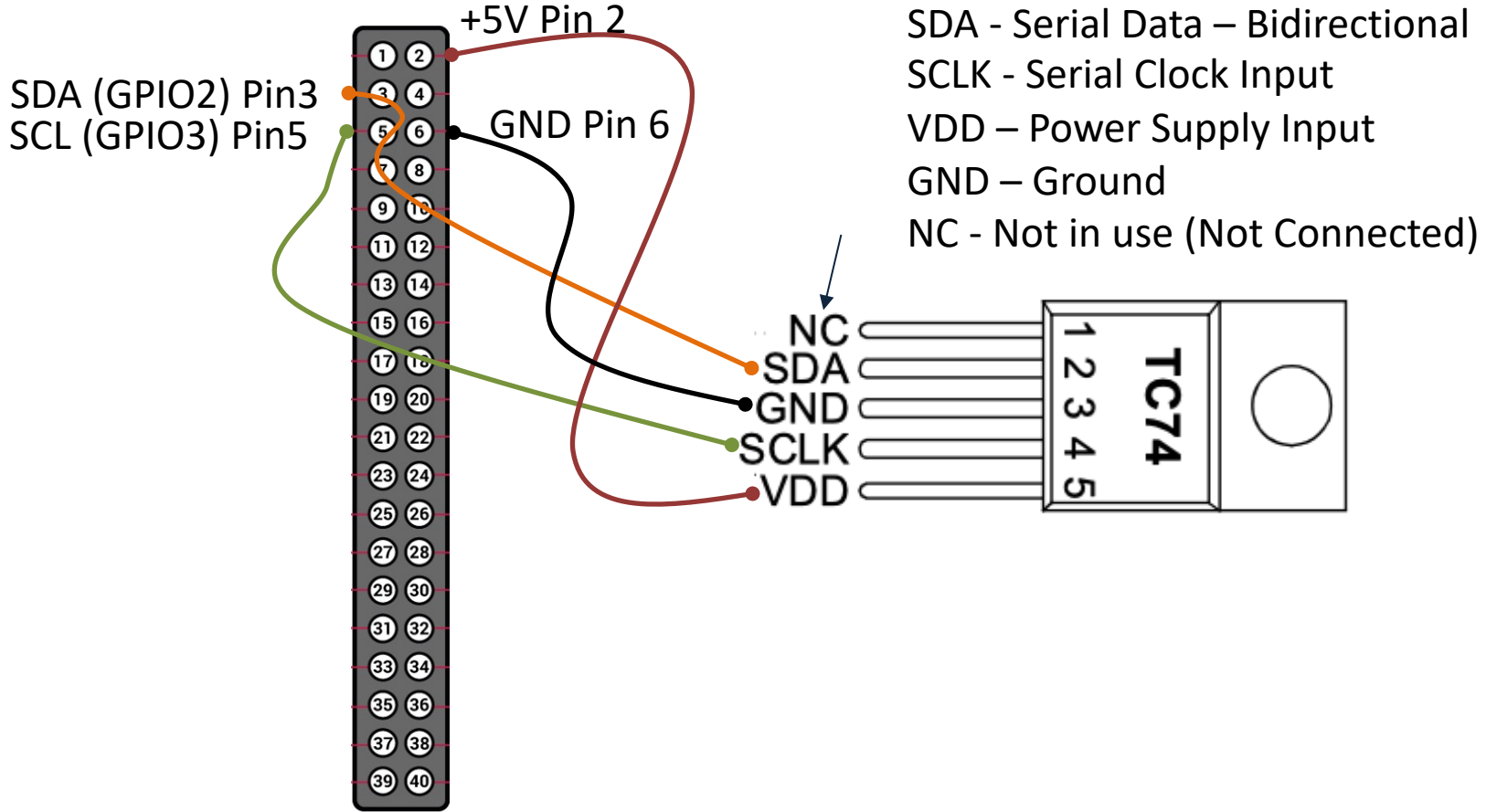


- The TC74 acquires and converts temperature information from its onboard solid-state sensor with a resolution of $\pm 1^{\circ}\text{C}$.
- It stores the data in an internal register which is then read through the serial port.
- The system interface is a slave SMBus/I2C port, through which temperature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

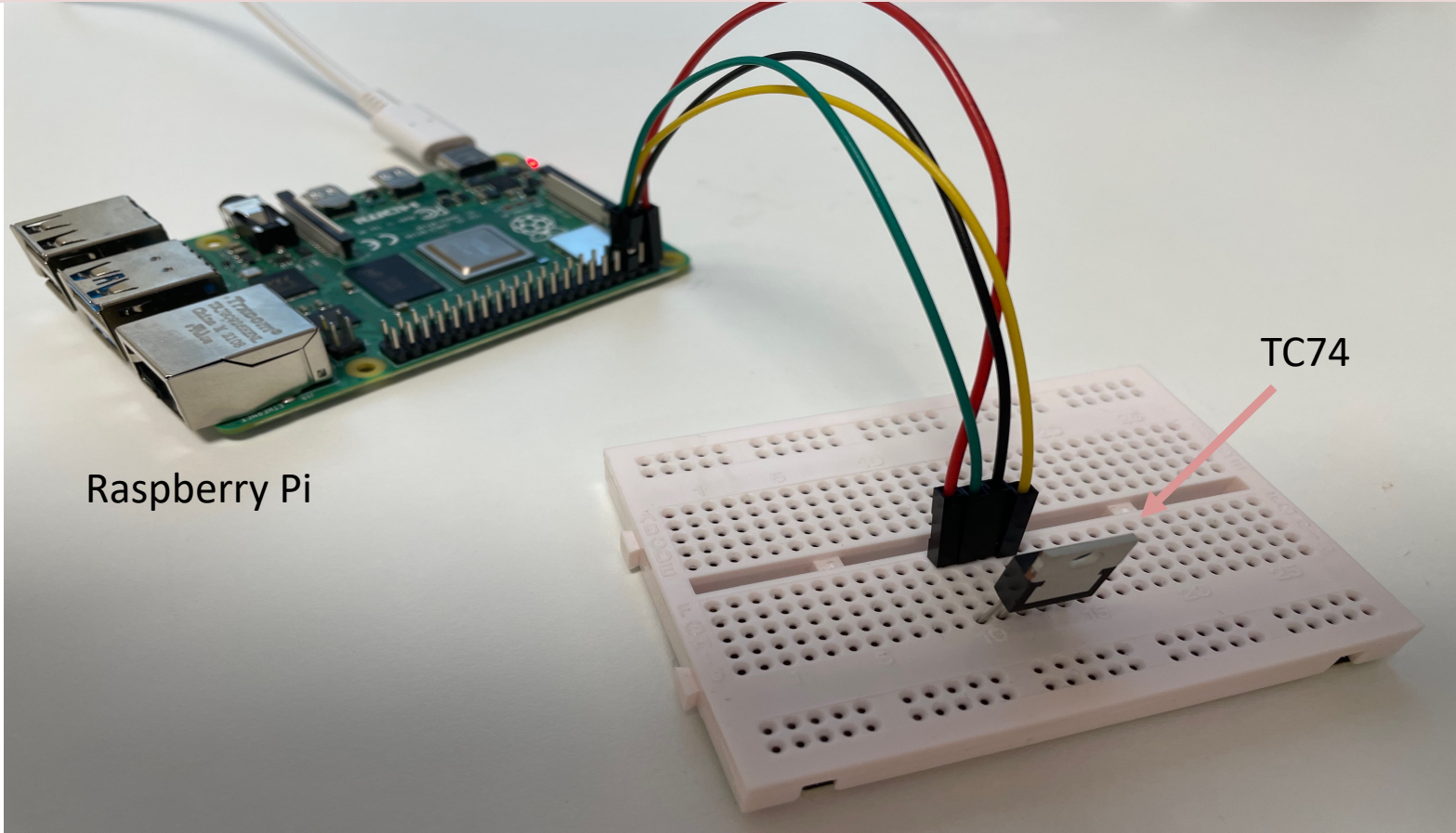
Datasheet: <https://ww1.microchip.com/downloads/en/DeviceDoc/21462D.pdf>

TC74 Wiring

Raspberry Pi GPIO Pins



Wiring



Raspberry Pi

TC74

Read Temperature Example

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

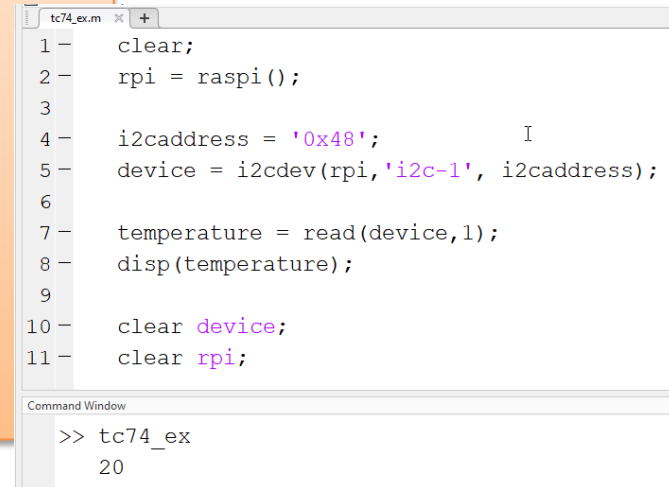
```
clear;
rpi = raspi();

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

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

clear device;
clear rpi;
```

↓
Read 1 Byte



```
tc74_ex.m x +
1 - clear;
2 - rpi = raspi();
3
4 - i2caddress = '0x48';
5 - device = i2cdev(rpi,'i2c-1', i2caddress);
6
7 - temperature = read(device,1);
8 - disp(temperature);
9
10 - clear device;
11 - clear rpi;

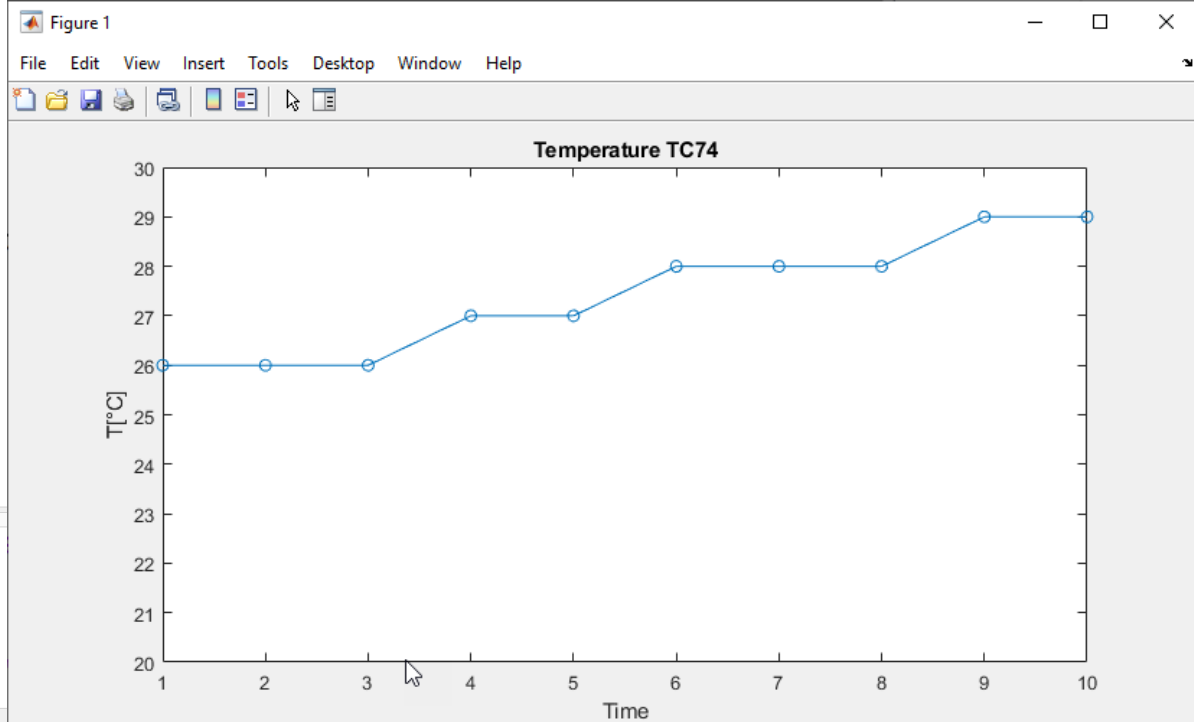
Command Window
>> tc74_ex
20
```

Reading Temperature and Plotting

```
MATLAB R2022a - academic use
HOME PLOTS APPS EDITOR PUBLISH VIEW
New Open Save Print Compare Go To Find Refactor Analyze Profiler Run Section Run and Advance Run Step Stop
FILE NAVIGATE CODE ANALYZE SECTION RUN
C:\Temp\RaspberryPi-MATLAB
Workspace
Name Value
led_builtin.m led_ex.m led_ex2.m tc74_ex.m
1 clear;
2 rpi = raspi();
3
4 disp('Initialization...')
5 i2caddress = '0x48';
6 device = i2cdev(rpi, 'i2c-1', i2caddress);
7
8 N = 10;
9 T = 2;
10 x = 1:N;
11 y = zeros(N,1);
12
13 disp('Start Reading Temperature Data...')
14 for i = 1:N
15     temperature = read(device,1);
16     disp("T=" + temperature + "°C");
17     y(i) = temperature;
18     pause(T);
19 end
20
21 plot(x,y, '-o')
```

Command Window

```
T=27 °C
T=28 °C
T=28 °C
T=28 °C
T=29 °C
T=29 °C
fx >>
```



```
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;
```



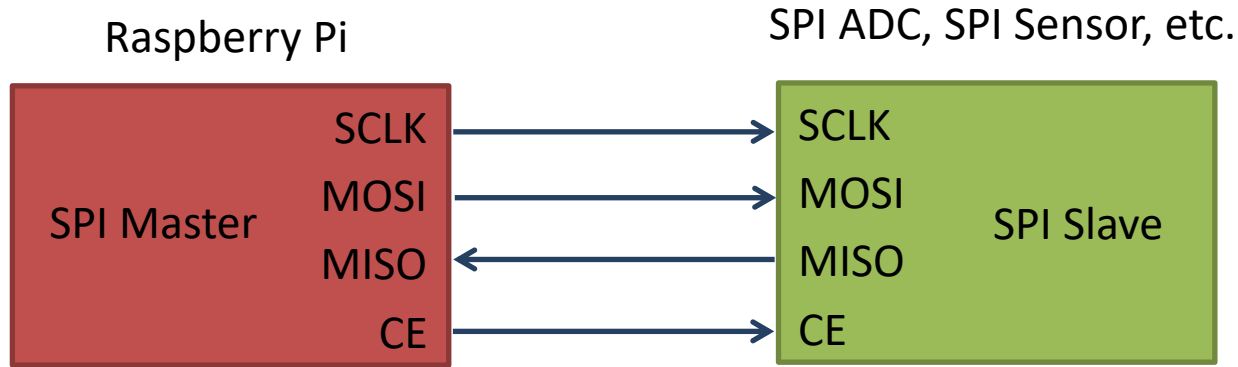
SPI

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

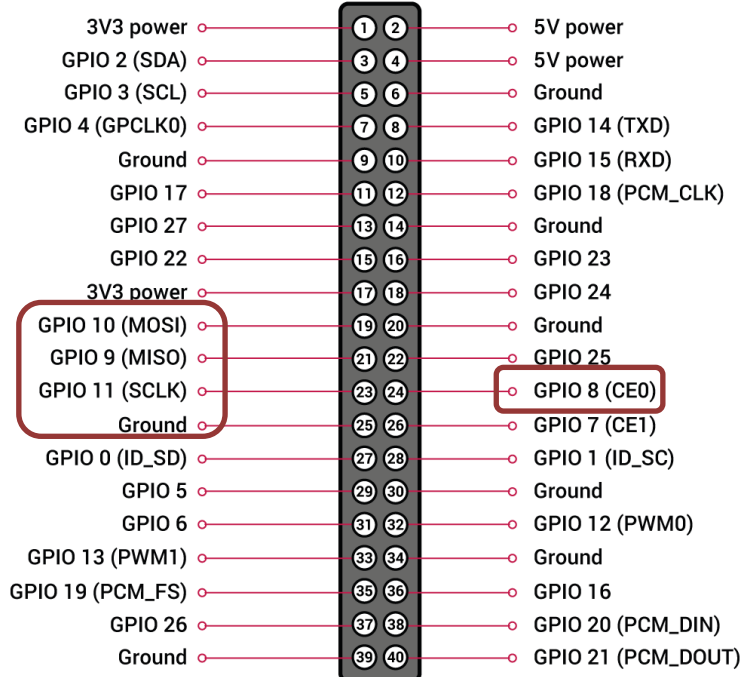
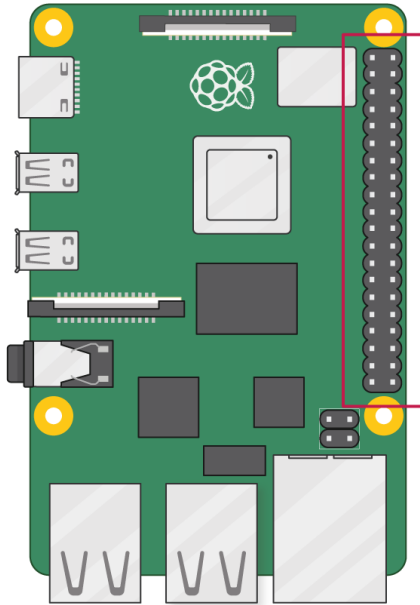


The SPI bus specifies four logic signals:

- **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

SPI Interface

Use the Raspberry Pi™'s SPI interface

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

Objects

<code>raspi</code>	Connection to Raspberry Pi board
<code>spidev</code>	Connection to SPI device on Raspberry Pi hardware

Functions

<code>writeRead</code>	Write data to and read data from SPI device
<code>enableSPI</code>	Enable SPI interface
<code>disableSPI</code>	Disable SPI interface

Find SPI Channels

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

Command Window

```
>> rp_ex
```

```
ans =
```

```
1x2 cell array
```

```
 {'CE0'}
```

```
 {'CE1'}
```

SPI Example

```
clear rpi;  
rpi = raspi();  
  
spidevice = spidev(mypi, 'CE1', 0)  
  
wrrdata = [hex2dec('08') hex2dec('D4')]; %Just an example  
dataout = writeRead(spidevice, wrrdata)
```

Write Data

Read Data



ADC with SPI interface

Analog to Digital Converter

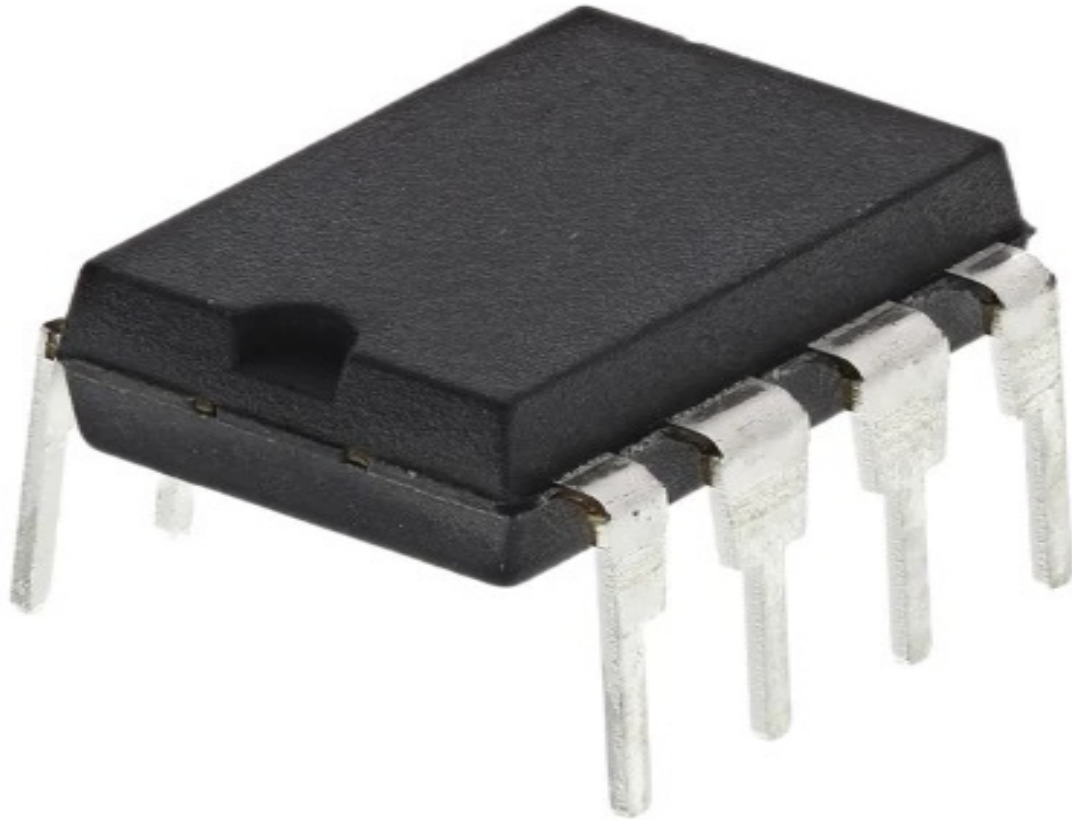
Hans-Petter Halvorsen

[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

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>

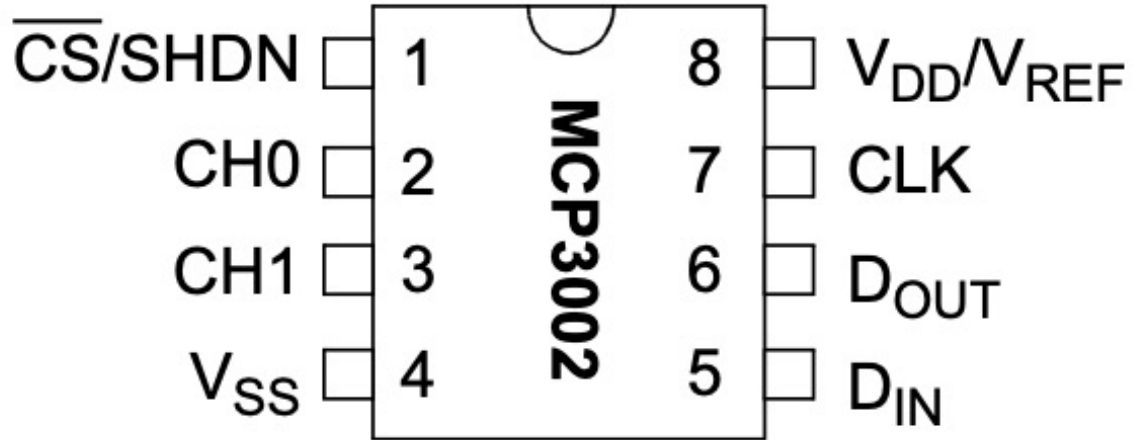
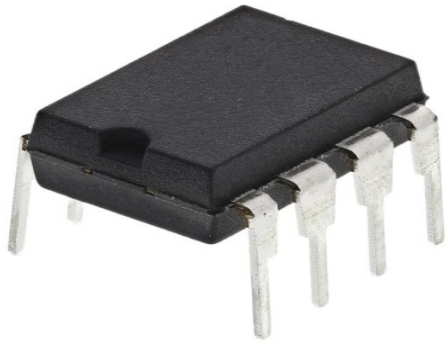
Sensors:

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).

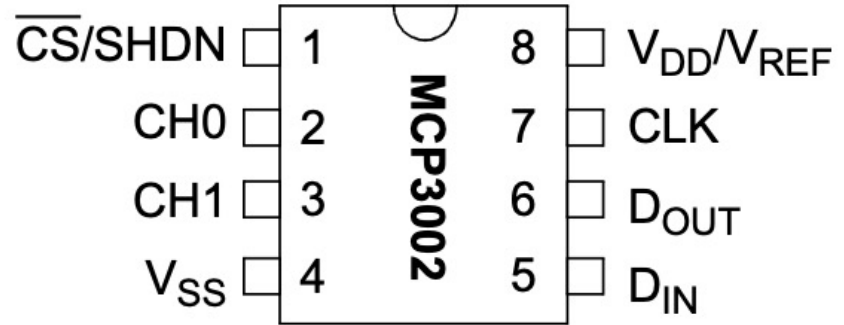
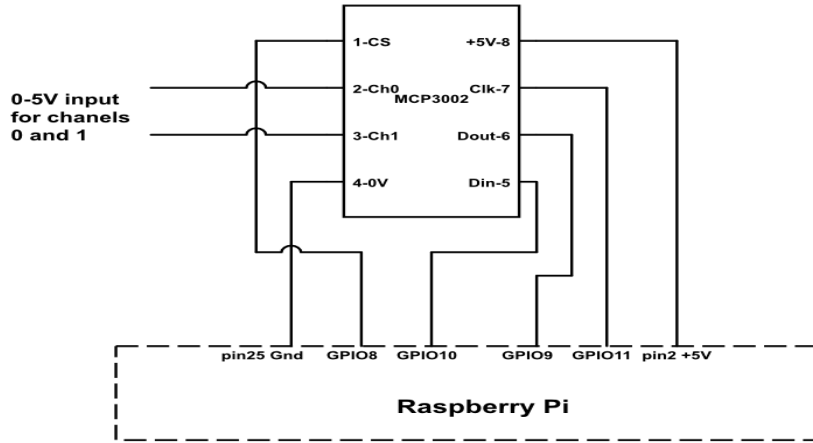
The MCP3002 uses a SPI Interface



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

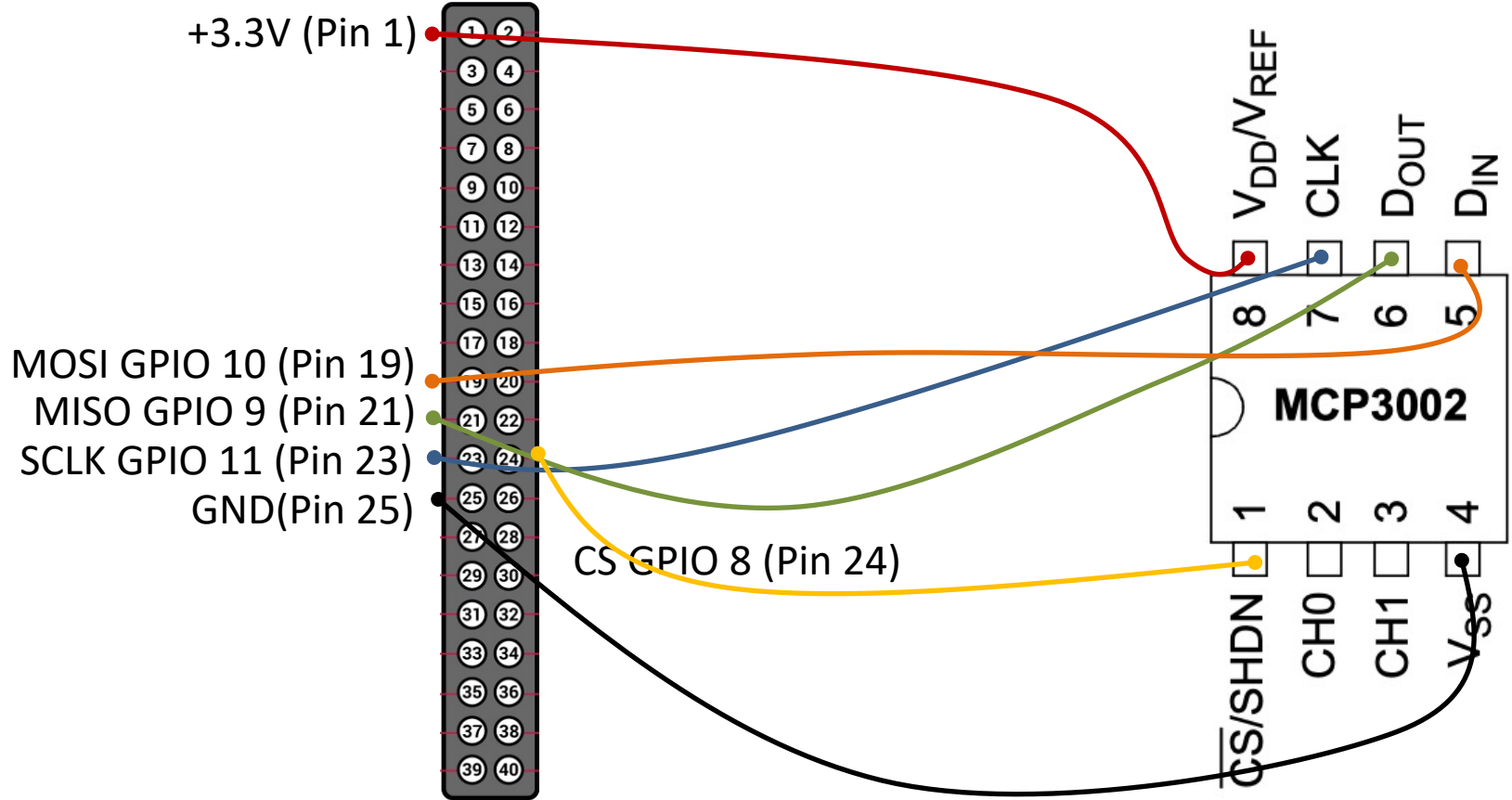
<https://learn.sparkfun.com/tutorials/python-programming-tutorial-getting-started-with-the-raspberry-pi/experiment-3-spi-and-analog-input>

Wiring



Wiring

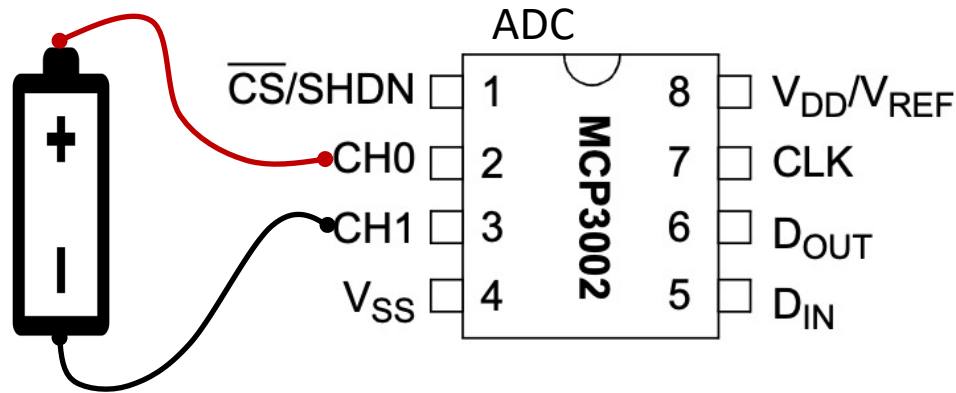
Raspberry Pi GPIO Pins



Read Data from ADC

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

1.5V Battery



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

Channel Configuration

	CONFIG BITS		CHANNEL SELECTION		GND
	SGL/ DIFF	ODD/ SIGN	0	1	
Single-Ended Mode	1	0	+		—
	1	1		+	—
Pseudo-Differential Mode	0	0	IN+	IN-	—
	0	1	IN-	IN+	—

From MCP3002 Datasheet

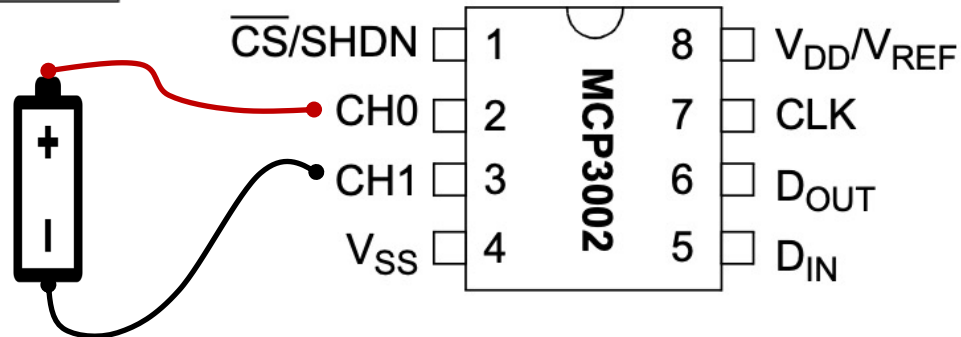
1000

1100

0000

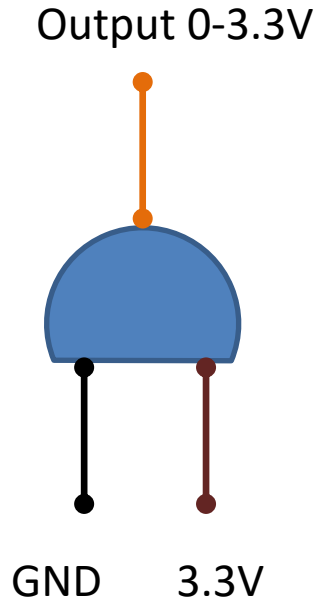
0100

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



Potentiometer

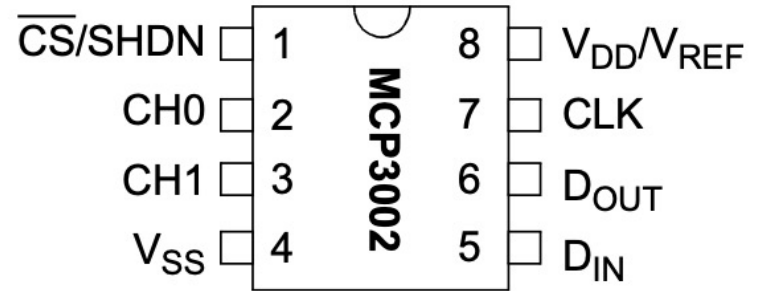
Potentiometer



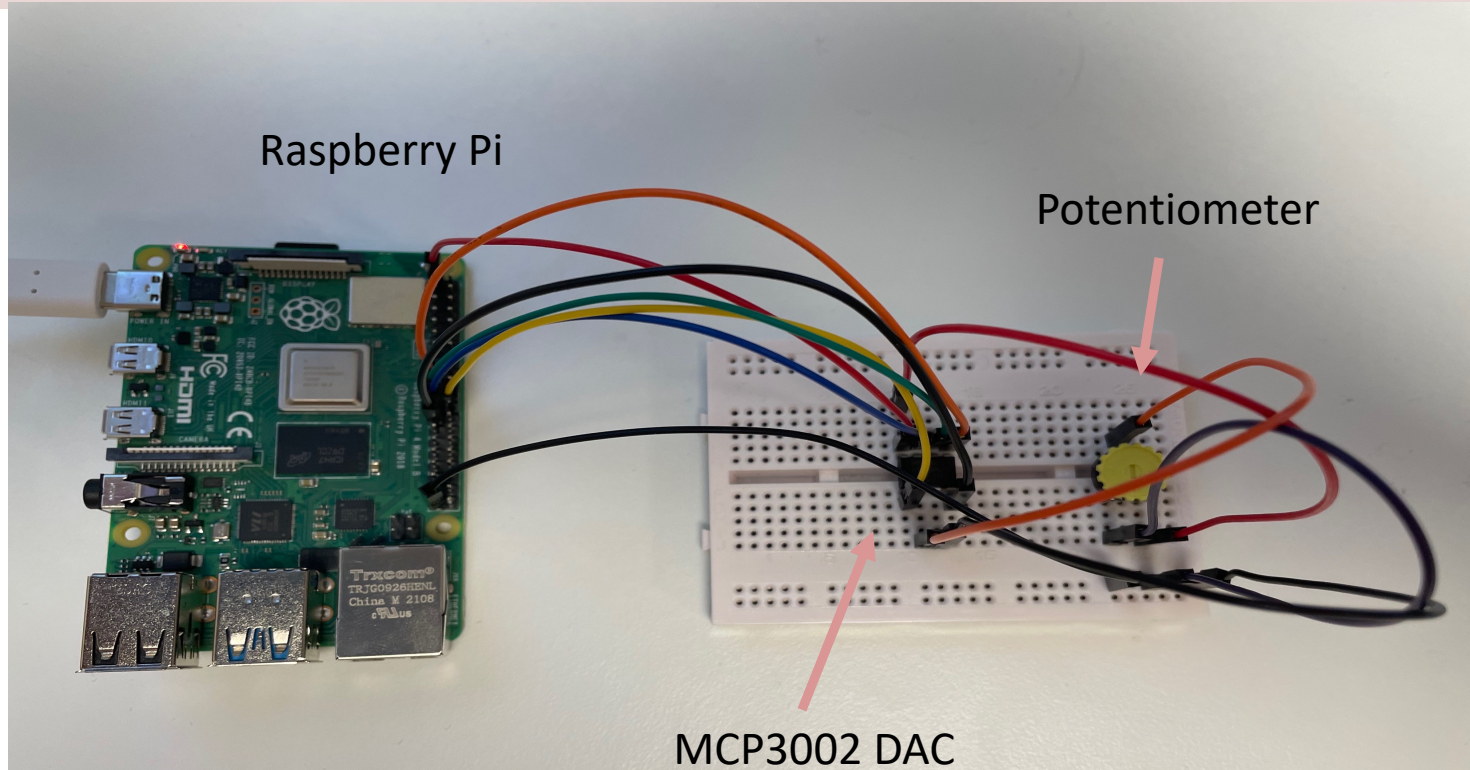
Raspberry Pi



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

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(bitior(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)



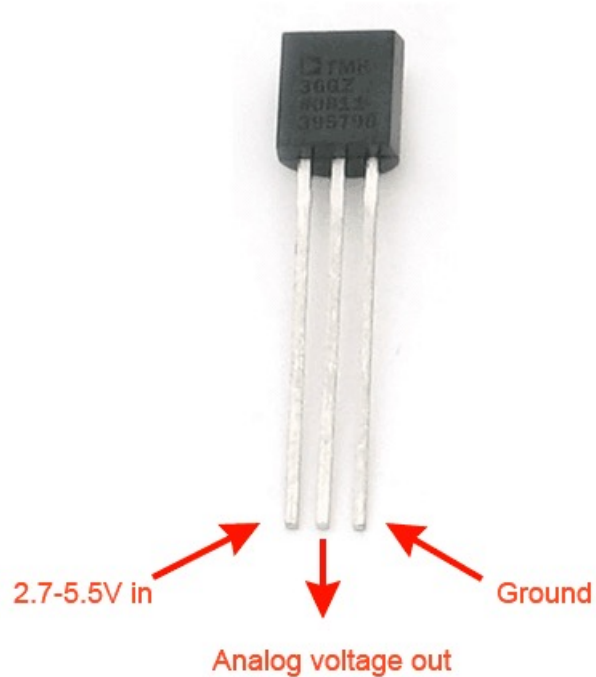
TMP36

Analog Temperature Sensor

Hans-Petter Halvorsen

[Table of Contents](#)

TMP36 Temperature Sensor

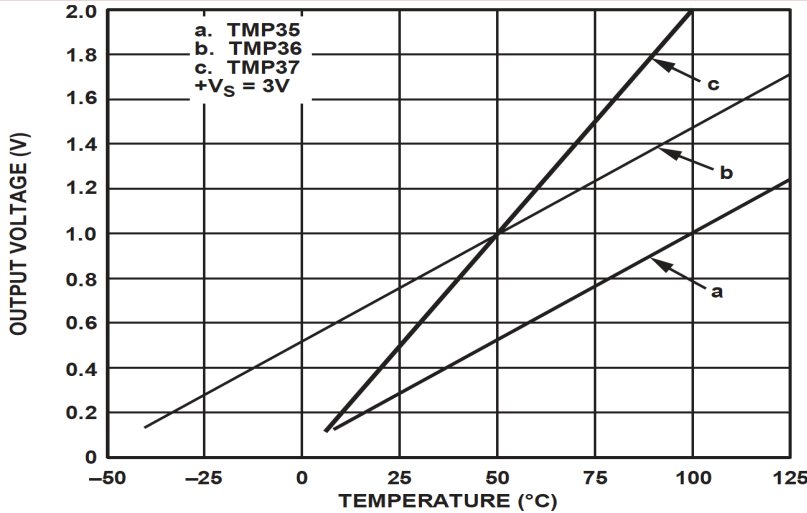


A Temperature sensor like TM36 use a solid-state technique to determine the temperature.

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 from Voltage (V) to degrees Celsius

From the Datasheet we have:

$$(x_1, y_1) = (0.75V, 25^\circ C)$$

$$(x_2, y_2) = (1V, 50^\circ C)$$

There is a linear relationship between Voltage and degrees Celsius:

$$y = ax + b$$

This gives:

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

Then we get the following formula:

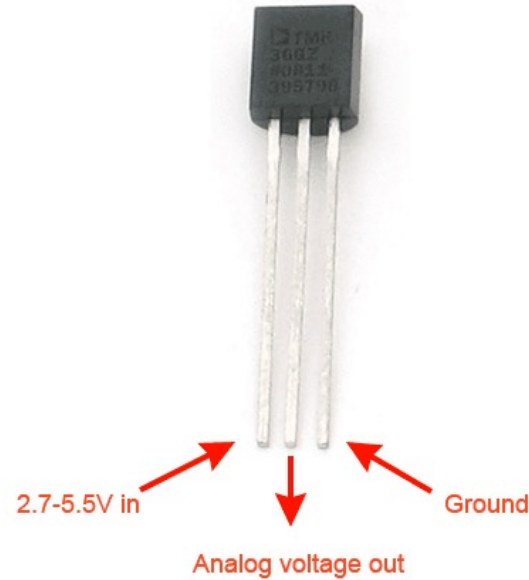
$$y = 100x - 50$$

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

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

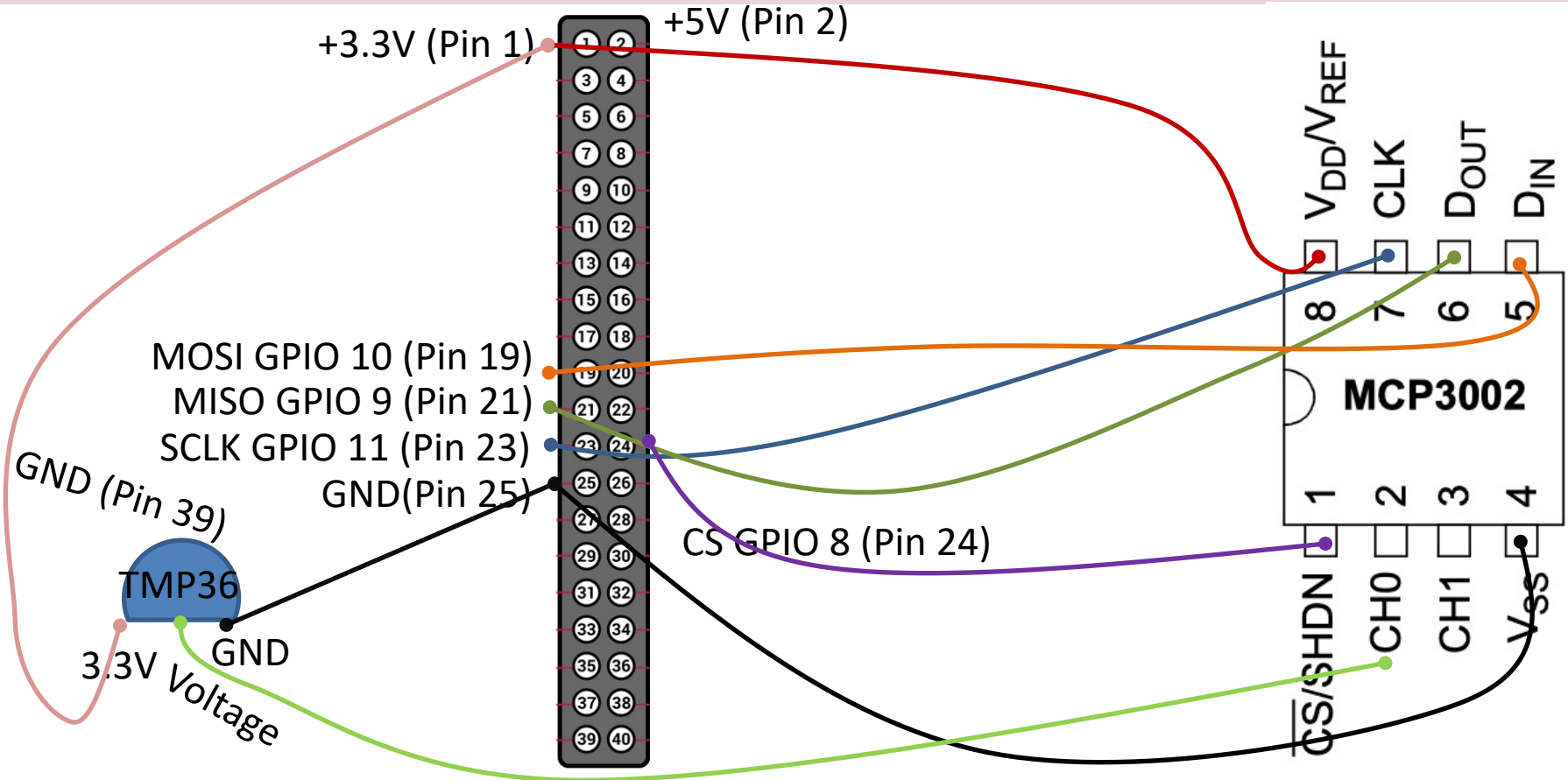
Measure Temperature with an ADC

TMP36 Temperature Sensor

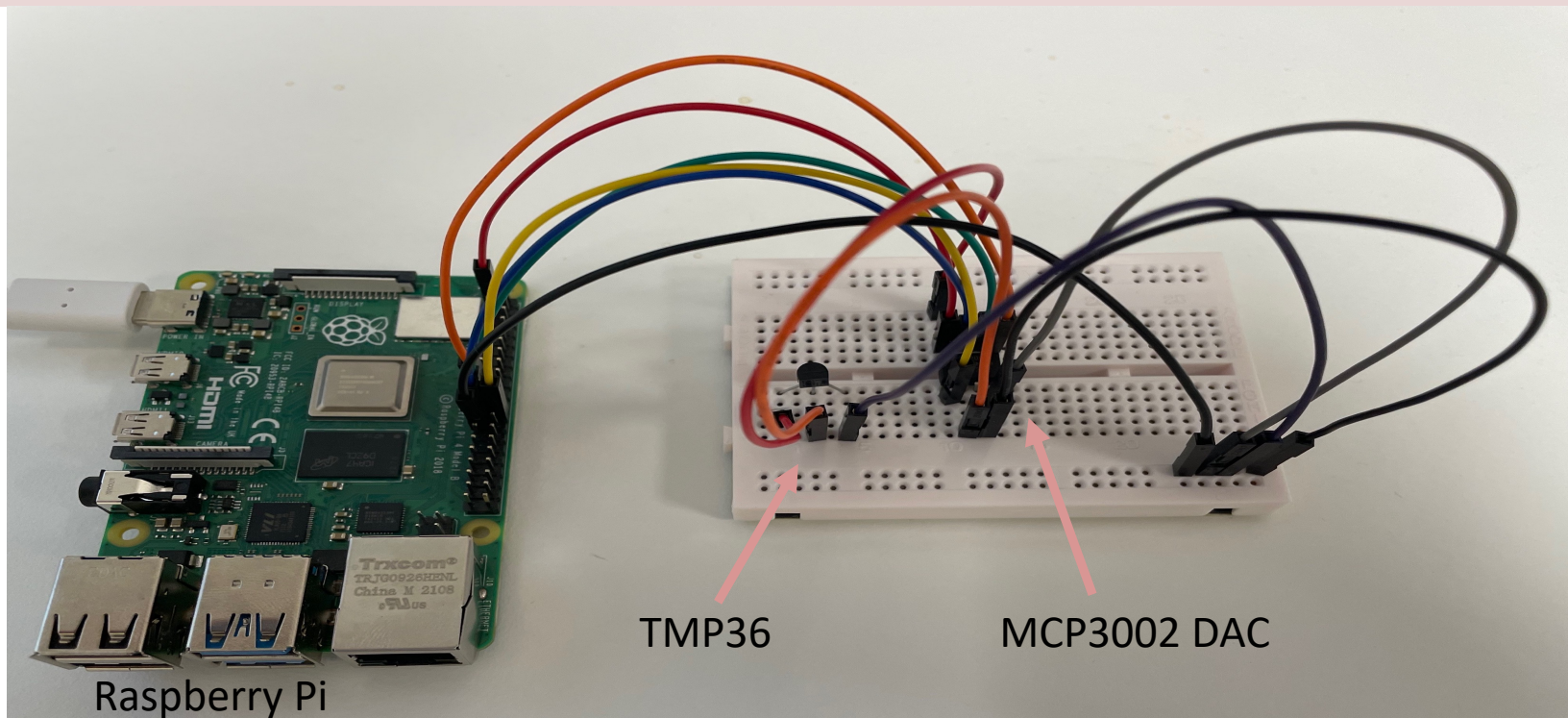


Wire a TMP36 temperature sensor to the first channel of an MCP3002 analog to digital converter and the other pins to +3.3V and GND

Wiring



Wiring



Raspberry Pi

TMP36

MCP3002 DAC

```
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>

