# Pico Python Getting connected. Connect to a compass module to demonstrate i2c.

This note explains how to connect a Raspberry PI and a PICO serially and to programme the PICO.
The PICO has an I2C compass module connected to it.
The PICO is programmed via a raspberry Pi using the Thonny Python Editor.

In this example, The PI is connected to via a VNC server and the PICO is attached to the PI, this note explains how to get this done.

The Data Sheet for the compass module

1683374.pdf 437 kB

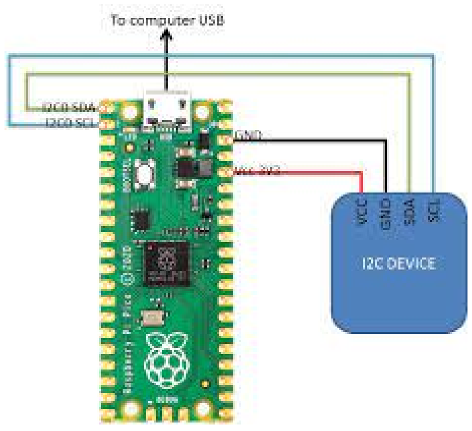
HMC5883L
This is a more available 3 axis device that looks like it is more common.

Python sdk guide. A very good guide.

raspberry-pi-pico-python-sdk.pdf 2 MB

**To connect to an I2C device:**

first check the you have it connected correctly, I had the SCL and SDA the wrong way around so, check your connections.
Here is the the connections used for this example.



Example code (Use the code on Github rather)

```
import machine
import time

i2c=machine.I2C(0, scl=machine.Pin(1), sda=machine.Pin(0), frequency=100000)

The 0 is the i2c port 0 or 1 as the pico has two build in hardware i2c ports.
pins 1 and 0 are the pins that have ben connected, and the frequency is the top bus
speed.

i2c.scan()

if you print i2c.scan() you will get the slave address of devices that are on the bus.

You have to possibly do it  the following way if you have more than one device on the
bus.
devices=[]
addresses=[]

if devices:
        for  items in devices:
                addresses.append(items)
                print ('Devices address found', items)

In our case we just have the one.
The factory address is 0x42

devices = i2c.scan()
```

We get 0x21 which is not what you might expect.
The I2C by default is only 7 bits and the first bit is used for indicating if the command is a read or write.

so instead of the expected  1000010 =0x42 we read                                   0100001=0x21

We use 0x21 when addressing the device in 7 bit mode. Some I2c devices can apparently have a 10bit mode, that works differently.

The documentation indicates if we write a 'A' character to the address the device will return a 2 byte heading data.

```
i2c.writeto(0x21,'A')
time.sleep_us(6001)                              #indicated time to produce output from
the device datasheet
(data1, data2) = i2c.readfrom(0x21,2)      #read two bytes of data from the address
and stores the results.
```

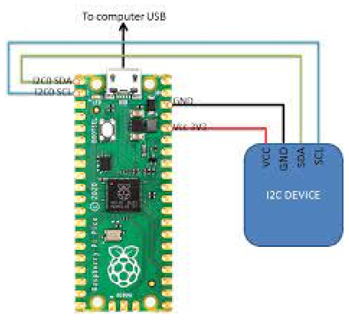
The data indicates a few more functions and things that you can do on the device. Like put the device to sleep:

```
i2c.writeto(0x21,'S')
```

or calibrate the device, i2c.writeto(0x21,'C') see the device sheet for more specific details.
It is important to note that you might need to observe some minimum times to wait after issuing an instruction, these times are on the data sheet.

To connect to the  PICO
Open Thonny editor on the PI.
The compass module is wired as in the diagram below.

I2c connection

|  |  |
|---|---|
| I2C connection |   To computer USB<br>I2C0 SDA<br>I2C0 SCL<br>GND<br>Vcc 3V3<br>VCC<br>GND<br>SDA<br>SCL<br>I2C DEVICE |

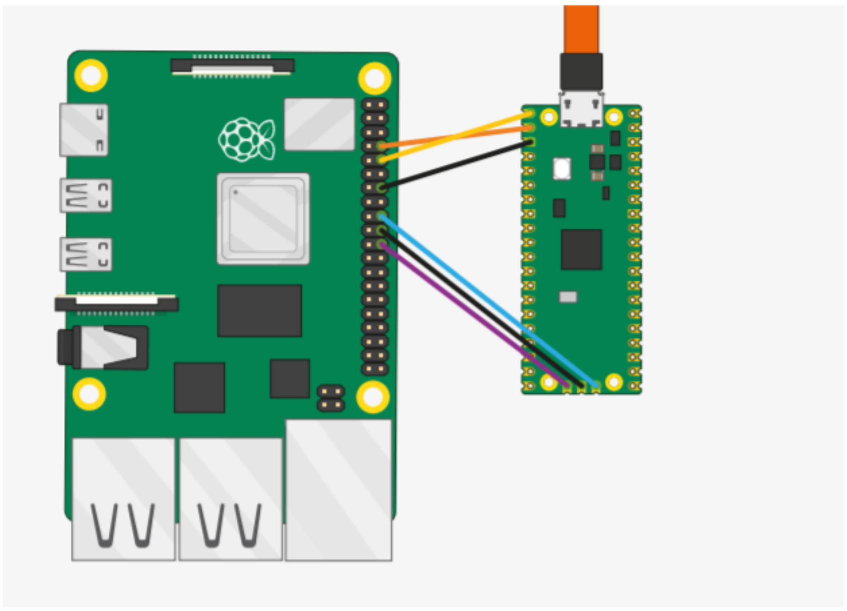
The code in Thonny should look like.
https://github.com/optimho/PICO_Compass

When you make a change to the code and save it, it saves it to the PICO.
There are brackets around the tabs, these are the current files that are being worked on and are on the PICO I think.

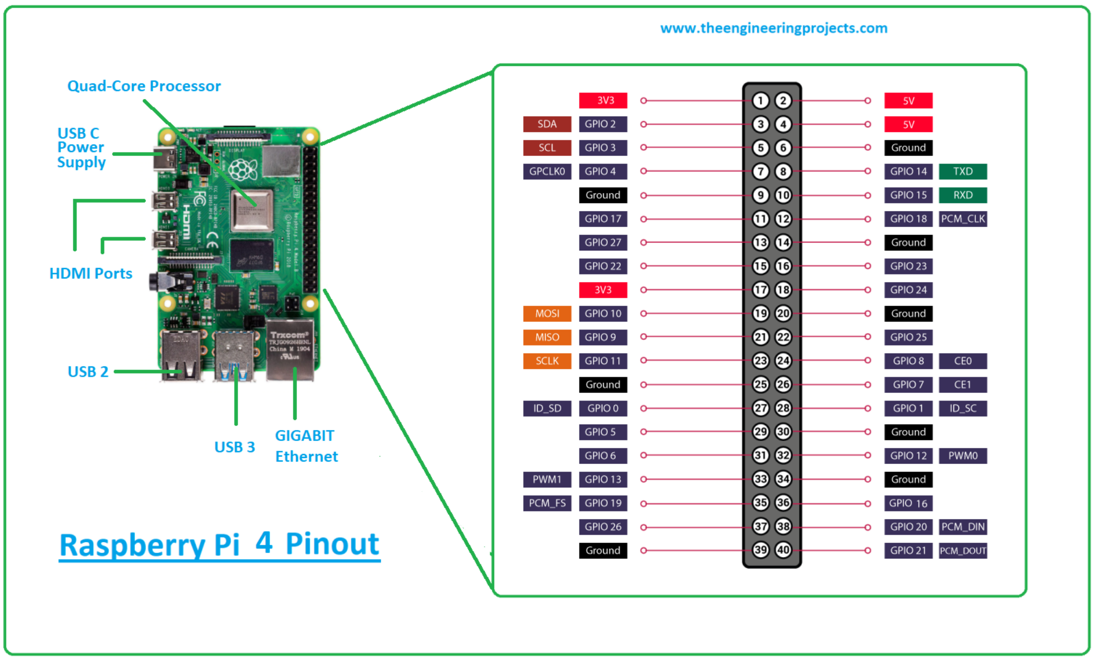
## Connect and program the PICO

After a few struggles, I am trying a raspberry Pi 4 via a VNC connection.
Wire the PICO to the pi so that It can make use of the Serial Wire Debug (SWD)

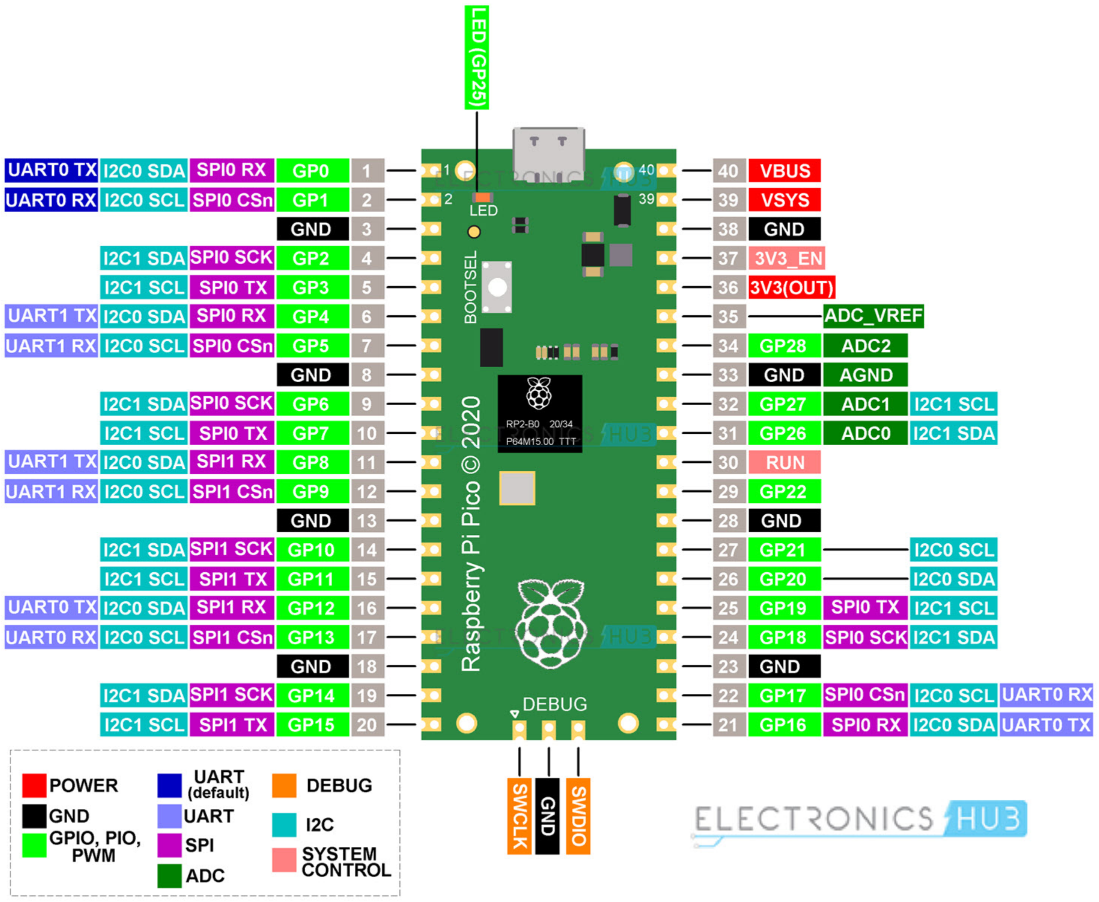
This is the connection to the Raspberry PI



To power the pico, one can use 3.3v and gnd, see RP4 pinout below.



Pasppberry Pico Pinout VSYS and GND is the power in should be between 1.8v and 5.5v
VBUS is the power from a plugged in USB supply.



To get VNC working you need to enable VNC and SSH on the raspberry Pi

The PI I am using has a host name of greenmamba.
a user name of pi
and a password is

● ● ● ● ● ● ●

Use Putty to connect to the PI with the host name.
The PI connects to the WIFI automatically, so you should not have a problem.

once connected to the PI, type vncserver to start the server.
Check out the IP address and port that is returned, you will need this to connect with the VNC client.

download and install VNCserver client on your remote machine -not the PI it is already sorted.

https://www.realvnc.com/en/connect/download/viewer/

Run the VNC client program on the computer and enter the PI's IP address and vnc port, it should look something like this format 192.160.0.1:1 - This should allow you to connect to the PI, once this is done you can close the Putty connection if you like.

Once you have a connection.
Run the following script that will do all the setting up of the required software on the PI.

If you are developing for Raspberry Pi Pico on the Raspberry Pi 4B, or the Raspberry Pi 400, most of the installation steps in this Getting Started guide can be skipped by running the setup script.

NOTE This setup script requires approximately 2.5GB of disk space on your SD card, so make sure you have enough free space before running it. You can check how much free disk space you have with the df -h command.

You can get this script by running the following command in a terminal:

```
$       wget https://raw.githubusercontent.com/raspberrypi/pico-setup/master/pico_setup.sh
```

You should first sudo apt install wget if you don’t have wget already installed. Then make the script executable with,

```
$ chmod +x pico_setup.sh
```

and run it with,

```
$ ./pico_setup.sh
```

The script will:

• Create a directory called pico
• Install required dependencies
• Download the pico-sdk, pico-examples, pico-extras, and pico-playground repositories
• Define PICO_SDK_PATH, PICO_EXAMPLES_PATH, PICO_EXTRAS_PATH, and PICO_PLAYGROUND_PATH in your ~/.bashrc
• Build the blink and hello_world examples in pico-examples/build/blink and pico-examples/build/hello_world
• Download and build picotool (see Appendix B), and copy it to /usr/local/bin.
• Download and build picoprobe (see Appendix A).
• Download and compile OpenOCD (for debug support)
• Download and install Visual Studio Code
• Install the required Visual Studio Code extensions (see Chapter 7 for more details)
• Configure the Raspberry Pi UART for use with Raspberry Pi Pico