Pico Python Getting connected. Connec t to a compass module to demonstrate i 2c.

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

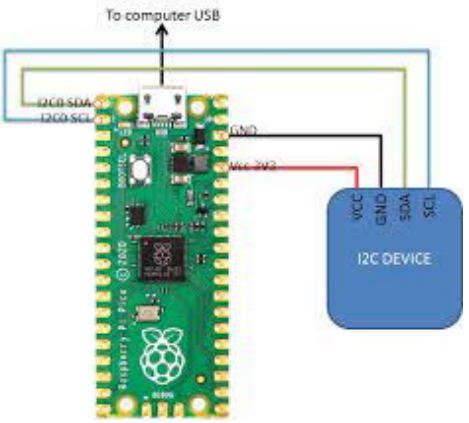
HMC5883L
This is a more available 3 axis device that looks like it is more common.

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

```
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:

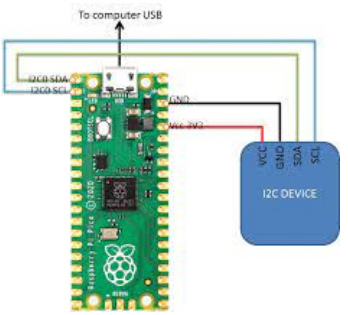
```
i2c.writeto(0x21,'S')
```

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.

I2c connection

I2C connection	

The code in Thonny should look like.

```
"""
I2C Program this one reads a compass
"""
import machine
import time
import binascii

from hmc6352 import Direction as Direction
loop=True
address=[]
devices=[]
#Create a I2C object
#0 is the I2C bus number on the RP2040
#scl is the clock line. We use GP17 for SCL
#sda is the data line. We use GP16 for SDA

HMC6352SlaveAddress = 0x21
HMC6352ReadAddress = 0x41               #"A" in hex, A command is: "Get Data" command
HMC6352SleepAddress = 0x53      #"S" in hex, S command is: "Sleep" command
HMC6352WakeAddress = 0x57               #"W" in hex, W command is: "Wake" command

i2c = machine.I2C(0, scl=machine.Pin(1), sda=machine.Pin(0) )
devices=i2c.scan()

if devices:
    for d in devices:
        address.append(d)
        print('Slave with address found -',hex(d))

print('````````````````')

data = i2c.readfrom(0x21, 2)

devices=i2c.scan()
if devices:
    for d in devices:
```

```
        address.append(d)
        print('Slave with address found -',hex(d))

i2c.writeto(33, 'A')
data = i2c.readfrom(33, 2)
#print(data)
data1=bin(0)
data1=bin(0)

while loop==True:
    i2c.writeto(0x21, 'W')
    time.sleep_us(200)
    i2c.writeto(0x21, 'A')
    (data1, data2) = i2c.readfrom(33, 2)

    time.sleep_us(6010)

    print('Heading =>', Direction.heading(data1,data2)/10,' degrees')


    i2c.writeto(0x21, 'S')
    time.sleep(10)
```

check out the read-outs in the shell, that should give the direction it is pointing in degrees.

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.