

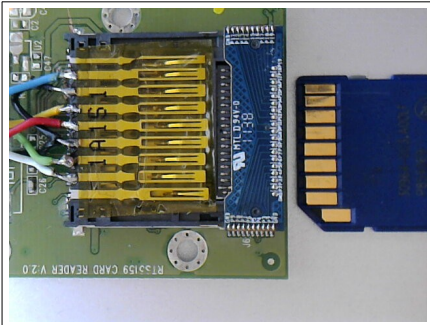
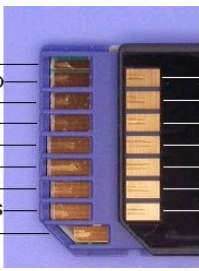
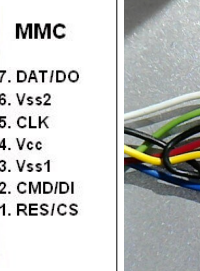
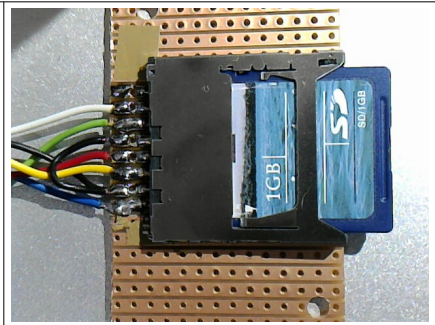
Connecting an SD card to the Pico

jean-claude.feltes@education.lu

SD card adaptor

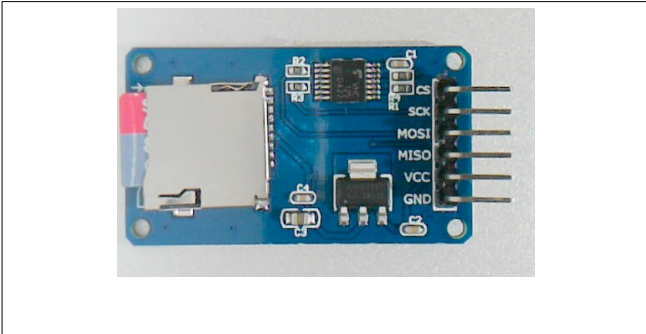
Connecting an SD card adaptor can be somewhat confusing (at least for me), as some sockets are connected from above and some from beneath. And there are some extra pins that we don't need. And the signal description is not always clear, as the data direction depends on the point of view.

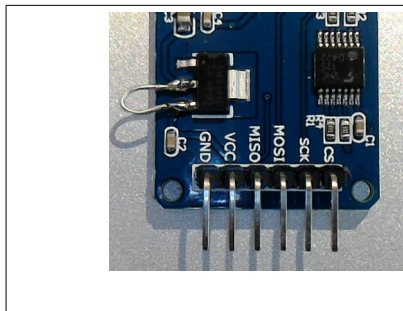
Examples of connections for “big” SD cards or micro SD with adaptor:

	<div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>SDC</p>  </div> <div style="text-align: center;"> <p>MMC</p>  </div> </div>	
<p>SD contacts to the top</p>		<p>SD contacts to the bottom</p>
<p>In both pictures, the same colours are used:</p> <ul style="list-style-type: none"> 7 = DO (data out), blue → 6 = GND, black 5 = CLK, yellow ← 4 = VCC = 3.3V, red 3 = GND (black) 2 = DI (data in), green ← 1 = CS (chip select), white 		<p>The data direction is seen from the point of view of the SD card.</p>

Take care: the supply voltage of SD cards is 3.3V!

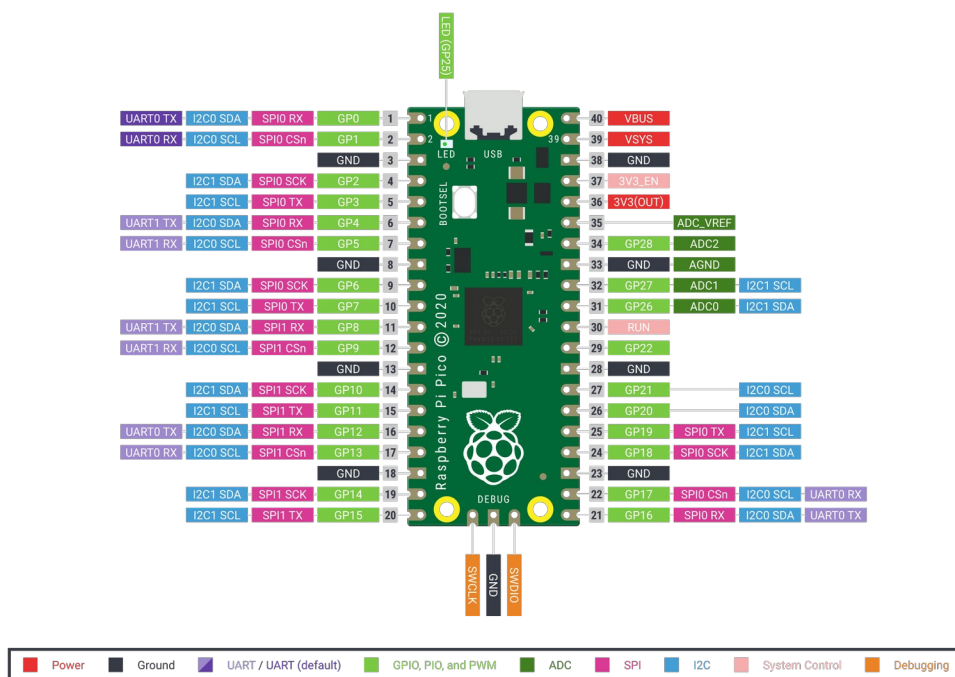
Ready-made adaptor for micro SD cards:

	<p>These integrate a 3.3V regulator, so they can (and must) be operated with +5V on VCC (VBUS on the Pico).</p> <p>The pin description corresponds to the Pico pins! So here the data direction is seen from the Pico viewpoint, as he is the master.</p> <p>MOSI = Master out, slave in = DI (green) MISO = Master in, slave out = DO (blue)</p>
---	---



In cases where this module should be operated with +3.3V, an easy modification is to short pins 2 and 3 of the regulator.

Connecting the SD card to the Pico



The communication goes over the SPI bus.

There are two SPI busses on the Pico, SPI0 and SPI1.

These can be mapped to most pins, as the pinout shows.

The spibus could be defined on the default pins with

```
from machine import SPI, Pin
spibus = SPI(0)
```

but which pins are used with this definition? You would have to look them up in the documentation .

It is better to define the pins explicitly, defining the chip select by the same occasion:

```
from machine import SPI, Pin
SD_CS = Pin(5) # Chip select for SD (may be any GP pin)
spibus = SPI(0, sck=Pin(6), mosi=Pin(7), miso=Pin(4))
```

This gives the same result, but it is clear which pins are used.

With these definitions we have to connect

```
DO (blue)    → MISO = GPIO4
DI (green)   ← MOSI = GPIO7
SCK (yellow) ← SCK = GPIO6
CS (white)   ← GPIO5
```

and of course +3.3V and GND

Then the Sdcard object can be defined:

```
sd = sdcard.SDCard(spibus, Pin(SD_CS))
```

When this can be done without error, the SD card is ready to be mounted.

If there is an error message, verify that the card is formatted as FAT32.

Mounting the SD card to the file system

Thanks to the os module it is easy to mount the card so that files can be read or written.

```
import os
vfs = os.VfsFat(sd)
os.mount(vfs, "/SD")
```

This mounts the SD card to a folder “/SD”. All files on the SD card can now be accessed as if they would reside in the folder “/SD” of the internal file system.

The folder “/SD” is eventually visible in the Thonny file list. I found no reproducible behavior however.

Now we can use normal Python file functions:

List files on SD card:

```
files = os.listdir("/SD")
for fi in files:
    print(fi)
```

Write a file to the card:

```
f = open ("/SD/test.txt", "w")
f.write("Hello world\n")
f.close()
```

Using the flush() function ensures that the file is written to disk and the buffer is cleared. This is done automatically by the close() function. I found no necessity to use it.

Read a file from the card:

```
f = open ("/SD/test.txt", "r")
t = f.read()
f.close()
print(t)
```

Take care to include the “/SD” folder specification into the file name, otherwise it would mean files of the internal Pico file system!

Unmounting the SD card:

```
os.umount("/SD")
```

Sd card tools

Partly to avoid forgetting to include the folder “/SD” into the file name, and partly to simplify writing and reading a file, I wrote a module “sdcard_tools_01.py” available here:

https://github.com/jean-claudeF/MyMicroPythonLibs/blob/main/sdcardtools_01.py

It can be used like this:

```
SD_CS = 5                                # Chip select for SD (may be any GP pin)
spibus = SPI(0, sck=Pin(6), mosi=Pin(7), miso=Pin(4)) #

sd = SD(spibus, SD_CS)

if sd.error:
    print(sd.error)
else:

    print("SD folder: ", sd.sdfolder)

    l = sd.list_all()
    print("Files and folders on SD: ", l)

    l = sd.listfolders()
    print("Folders on SD: ", l)

    l = sd.listfiles()
    print("Files on SD: ", l)

    t = 'Blahblah ghjghg pi'
    sd.writefile('blah.txt', t)

    t = "This is appended to the file"
    sd.print('blah.txt', t)

    t = sd.readfile('blah.txt')
    print(t)

    sd.unmount()
```