

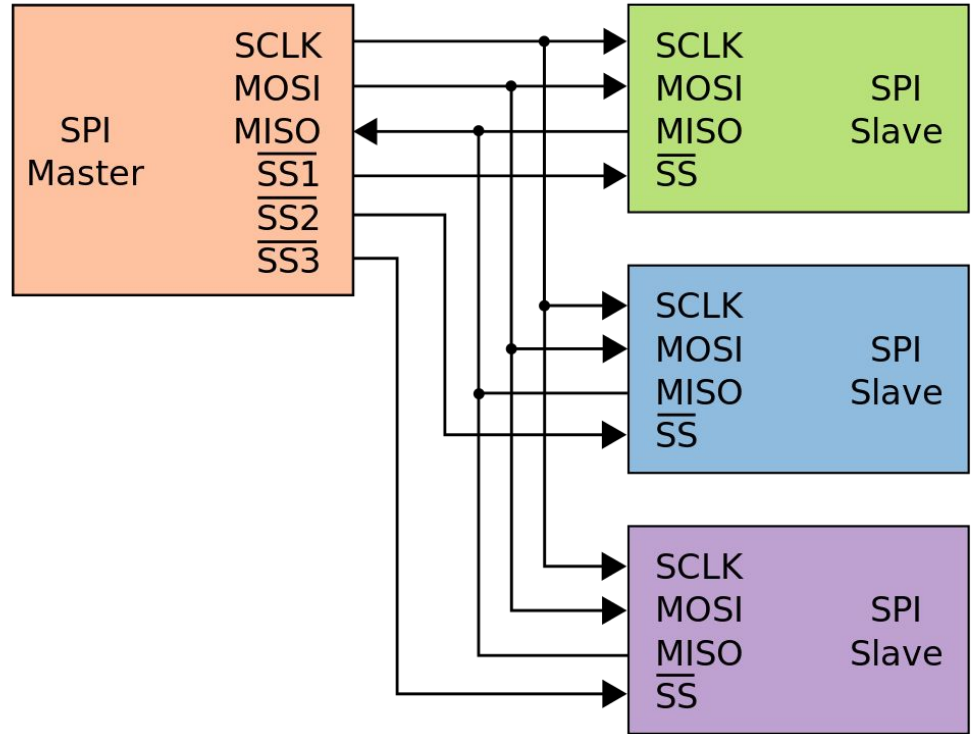
# IoT-ALE: Reading and Writing to SPI SDcards

Nisha Kumar

SCaLE 17x - March 2019

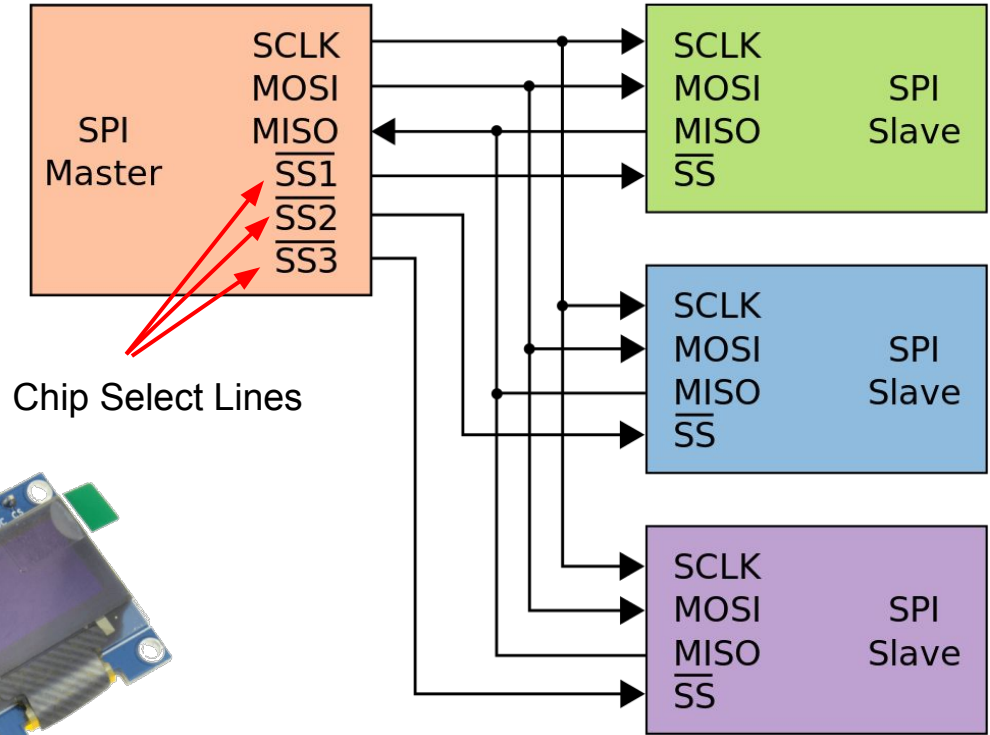
## SPI Background

- Not a hard defined standard like I2C
  - Ubiquitous despite no hard standard
  - Data on the bus is effectively device unique
  - Quad SPI can add 2 more data lines, uncommonly used
- Faster than I2C
  - Possible to go >10Mbps
- Duplex communications
  - Master Out Slave In (MOSI)
  - Master In Slave Out (MISO)
- Hardwired device selection



## Where this gets messy...

- While fast, it's not easy to implement
- Chip select lines can get very expensive, very quickly
- Some devices need more than the minimum 4\* wires



\* Minimum is based on duplex operation, some devices are write or read only and you only need 3 wires then

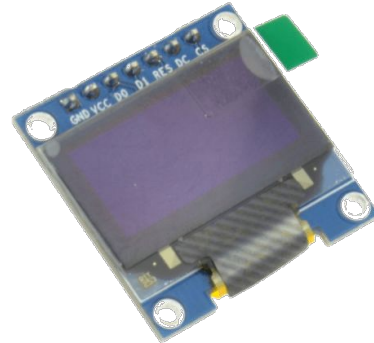
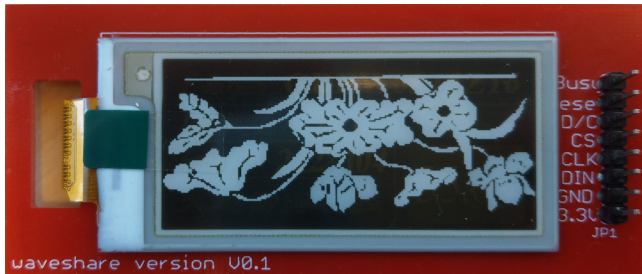
## SPI Screens, cases in point as “odd”

### E-link

- SPI Like Interface
- Busy pin
- Reset pin
- Data/Command (DC) pin
- Write-only device (MOSI)
- 8-pins (including Vcc & GND)

### OLED Screen

- SPI Like interface
- Write-only device (MOSI)
- Reset pin
- Data/Command (DC) pin
- 7-pins (including Vcc & GND)



## Normal SPI Device

### BME280 (SPI mode)

- CSB - Chip Select
- SCL - Clock
- SDA - MOSI (serial data in)
- SDO - MISO (serial data out)
- GND - Ground
- VCC - Power



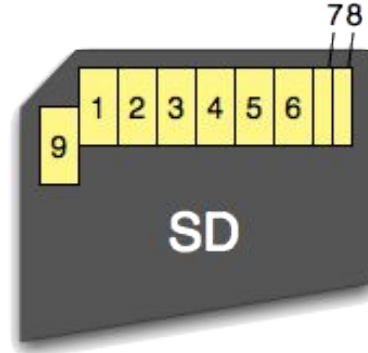
I2C



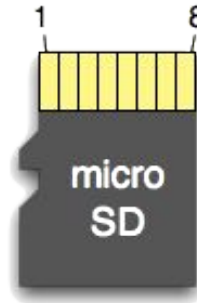
SPI

## SDcards and SPI

- SDcards have two basic modes:
  - SD mode
  - SPI mode
- SPI mode disadvantages:
  - Slower transfers (no parallel data)
  - 'U' modes aren't supported
- SPI mode advantages:
  - Easier to implement
  - Less hardware needed
  - Simpler interface



Pin	SD	SPI
1	CD/DAT3	CS
2	CMD	DI
3	VSS1	VSS1
4	VDD	VDD
5	CLK	SCLK
6	VSS2	VSS2
7	DAT0	DO
8	DAT1	X
9	DAT2	X



Pin	SD	SPI
1	DAT2	X
2	CD/DAT3	CS
3	CMD	DI
4	VDD	VDD
5	CLK	SCLK
6	VSS	VSS
7	DAT0	DO
8	DAT1	X

# Hardware vs. Software Implementation

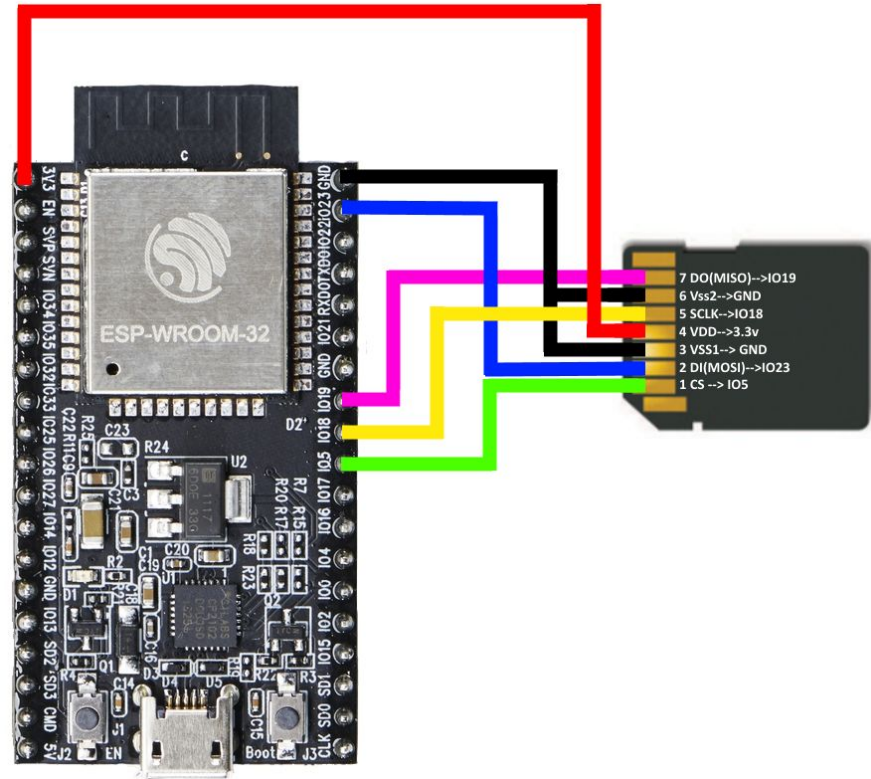
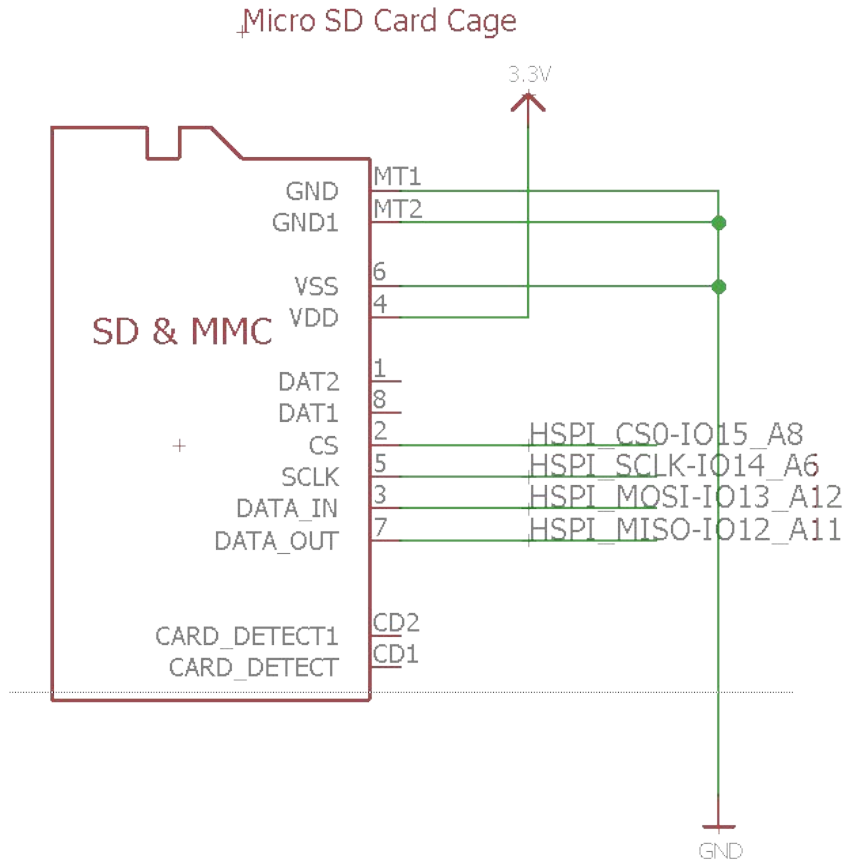
## Hardware:

- 4 SPI Busses
  - SPI0 - typically dedicated to Flash
  - SPI1 - tied to same pins as SPI0
  - HSPI (SPI2)
    - CS: 15
    - SCLK: 14
    - MISO: 12
    - MOSI: 13
    - QUADWP: 2
    - QUADHD: 4
  - VSPI (SPI3)
    - CS: 5
    - SCLK: 18
    - MISO: 19
    - MOSI: 23
    - QUADWP: 22
    - QUADHD: 21

## Software

- Any pins will do
- Bitbanged in software / timers
- SensorNode uses:
  - CS: 15
  - SCLK: 14
  - MISO: 12
  - MOSI: 13
  - QUADWP: -
  - QUADHD: -

# Wiring up an SDcard to an MCU





## Prep work for using the SDcard

1. Exit screen
2. Upload the following using ampy:  
`# ampy --port /dev/ttyUSB0 put sensornode-stuff/src/sdcard.py`
3. Open up the serial port again

## Lets look at some code - Setup the SPI Interface

Software (use this on SensorNode)

```
>>> from machine import Pin, SPI
>>> cs = Pin(15, Pin.OUT)
>>> mosi = Pin(13, Pin.OUT)
>>> miso = Pin(12, Pin.IN)
>>> sck = Pin(14, Pin.OUT)
>>> spi_bus = SPI(sck = sck,
mosi = mosi, miso = miso)
```

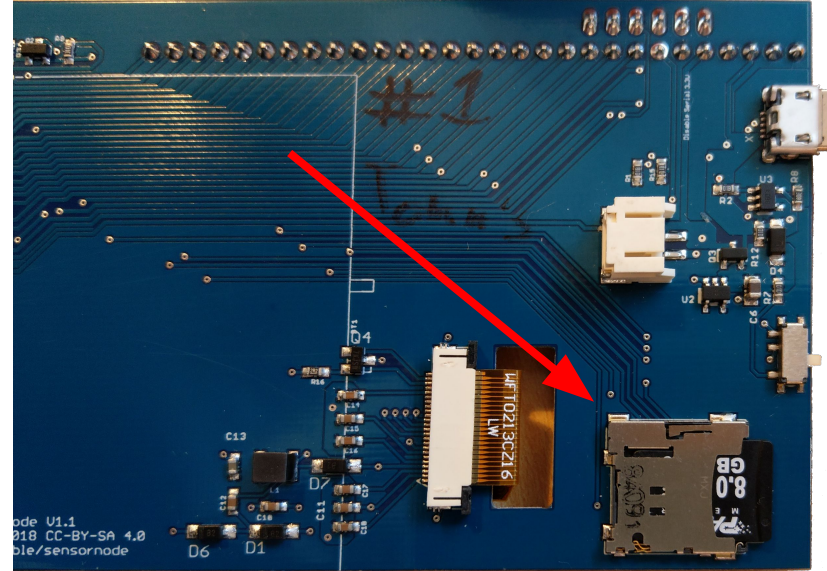
Hardware (for comparison only)

```
>>> from machine import Pin, SPI
>>> cs = Pin(15, Pin.OUT)
>>> spi_bus = SPI(2)
```

## Adding the SD card to the mix

1. Plug in the SD card
  - SD Card is on the back behind the buttons
2. Add the following:

```
>>> import sdcards
>>> sd = sdcards.SDCard( spi_bus, cs)
>>>
```



What this looks like, without the SD card in place:

```
>>> sd = sdcards.SDCard( spi_bus, cs)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "sdcards.py", line 54, in __init__
  File "sdcards.py", line 82, in init_card
OSError: no SD card
>>>
```

## Mounting the SDCard

- You mount it to the filesystem like Unix / Linux

- ```
>>> import os
```

```
>>> os.mount(sd, '/sd')
```


```
>>> os.listdir('/')
```

```
['sd', 'boot.py', 'bme280.py', 'sdcard.py', 'tsl2591.py', 'usmbus']
```

```
>>> os.listdir('/sd')
```

```
['MISC', 'DCIM', 'old']
```

```
>>>
```



Contents here will likely be empty unless you've  
Put things on the card already

## Reading & Writing to the SD card

```
>>> f = open("/sd/demofile.txt", "a")
>>> f.write("Hello World!")
12
>>> f.close()
>>> f = open("/sd/demofile.txt", "r")
>>> f.read()
'Hello World!'
>>>
```