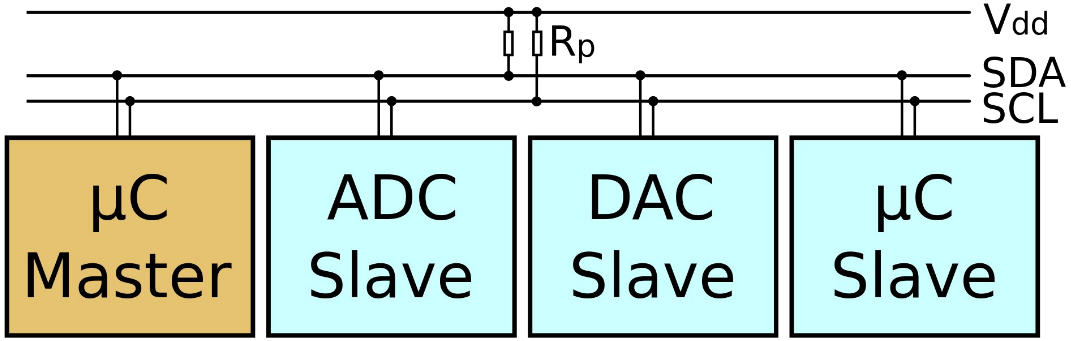
IoT-ALE:
Reading Sensor Data with I2C

Jon Mason

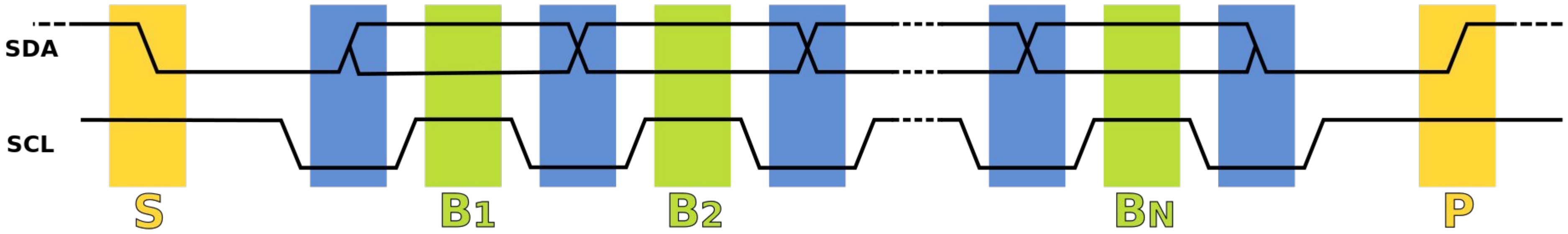
SCaLE 17x - March 2019

I2C Some background

- Released in 1982
- Bus Protocol
- Devices use addresses
- 2-pins needed
  - Clock (scl)
  - Data (sda)
- SCL & SDA pulled-up against voltage to ship (Vdd)
  - Level shifting can be complicated to get right
- Upwards of 3.4Mbps
  - More realistically: ~1Mbps
  - Most devices communicate in Kbps
- SMBus is derived from, but not identical to, I2C
  - Devices may claim to be one or the other not really consistent



What this looks like on the bus:

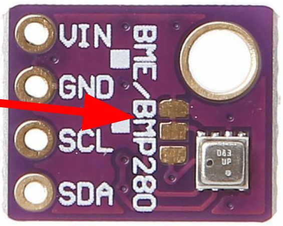


1. Data transfer is initiated with a *start* bit (S) signaled by SDA being pulled low while SCL stays high.
2. SCL is pulled low, and SDA sets the first data bit level while keeping SCL low (during blue bar time).
3. The data are sampled (received) when SCL rises for the first bit (B1). For a bit to be valid, SDA must not change between a rising edge of SCL and the subsequent falling edge (the entire green bar time).
4. This process repeats, SDA transitioning while SCL is low, and the data being read while SCL is high (B2, ...Bn).
5. The final bit is followed by a clock pulse, during which SDA is pulled low in preparation for the *stop* bit.
6. A *stop* bit (P) is signaled when SCL rises, followed by SDA rising.

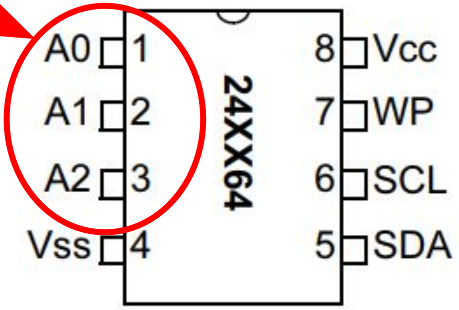
Addresses

- Device specific implementation
- Some devices only provide a single address
  - One device per-bus
- Address on the bus “needs” to be unique
- 7-bit address normal
  - 128 Devices normally
  - 10-bit exists, very little uses it
  - 10-bit gives you 1008 devices (reserved addresses)

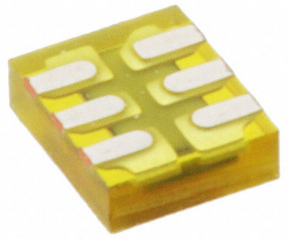
Two Addresses Selectable



Eight Addresses Selectable



TSL2591
No selectable Address



I2Cdetect (Linux)

```
$ i2cdetect -y -r 1
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- 56 -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- —-
```

Devices at:

- 0x56
- 0x68

- I2C is ***NOT*** discoverable, detection is not guaranteed
- Random probing can cause systems to crash - you are warned

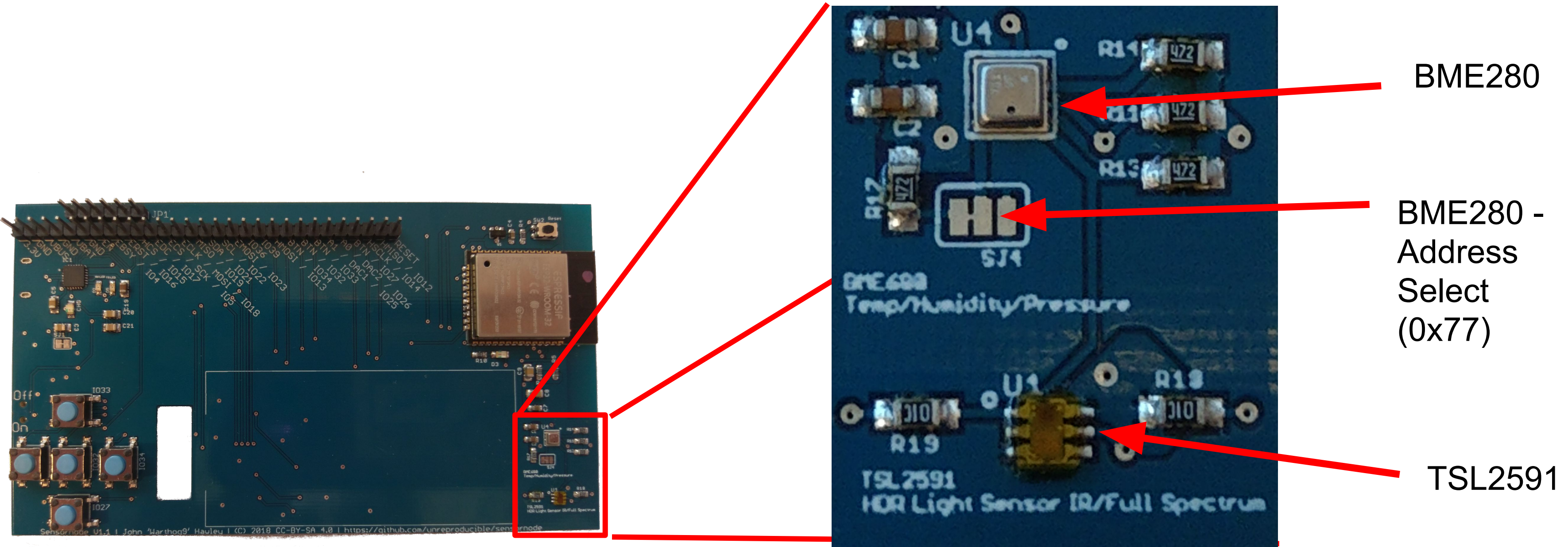
SensorNode has 2 x I2C devices:

BME280

- Temperature
- Humidity
- Relative Pressure

TSL2591

- Full Spectrum Light Sensor
- IR Spectrum Light Sensor

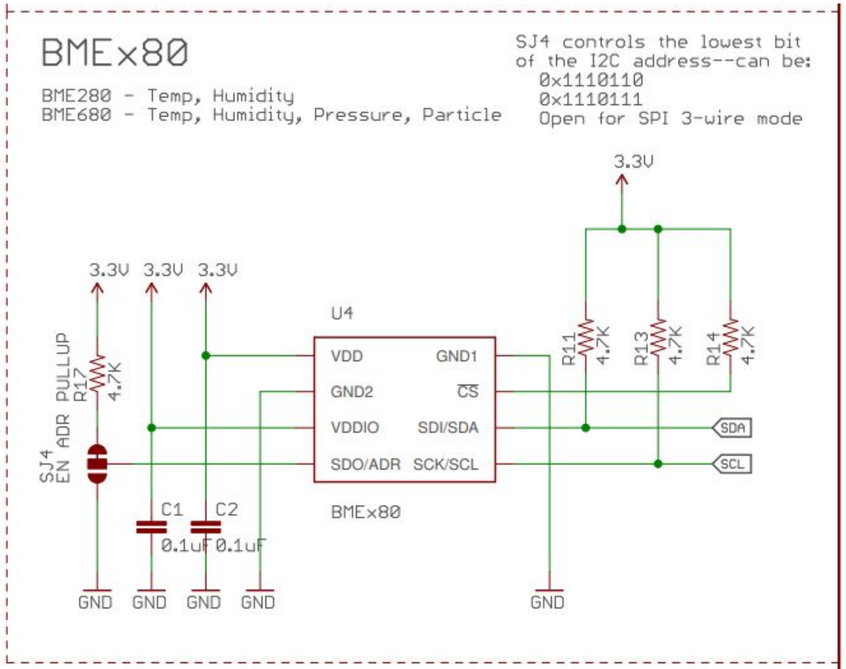


BME280
BME280 - Address Select (0x77)
TSL2591
BME680 Temp/Humidity/Pressure
TSL2591 HDR Light Sensor IR/Full Spectrum

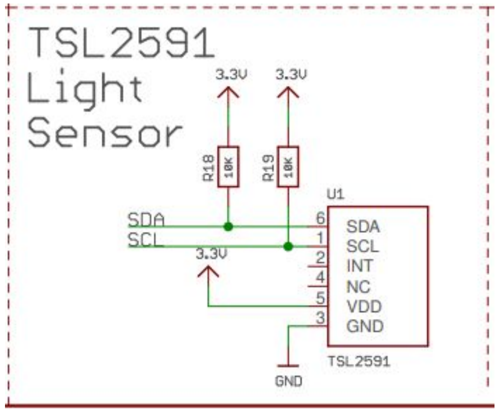
Figuring out Address - See the Schematic(s)

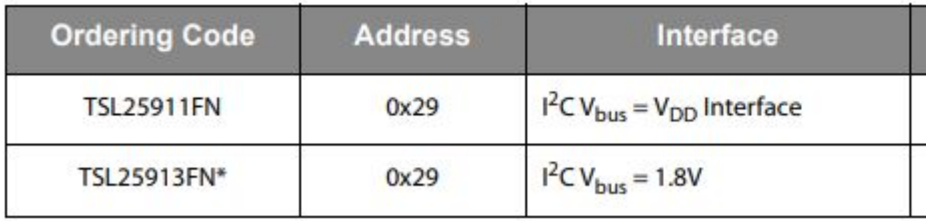
https://github.com/unreproducible/sensornode/blob/master/Schematic%20-%20sensornode.pdf

BME280:



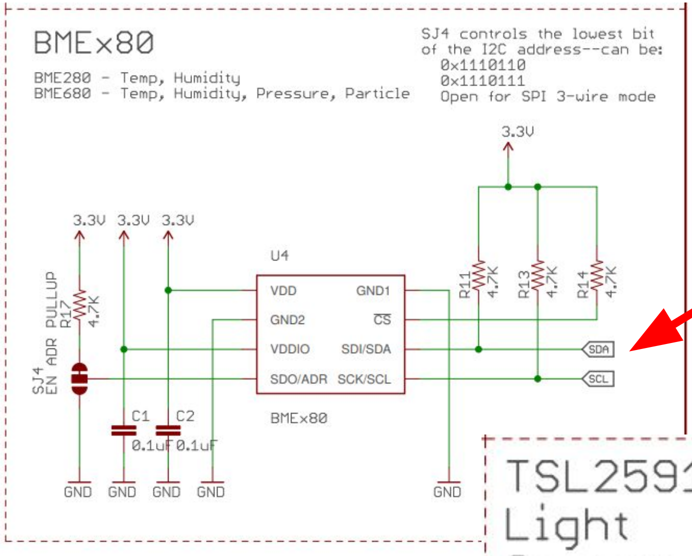
TSL2591

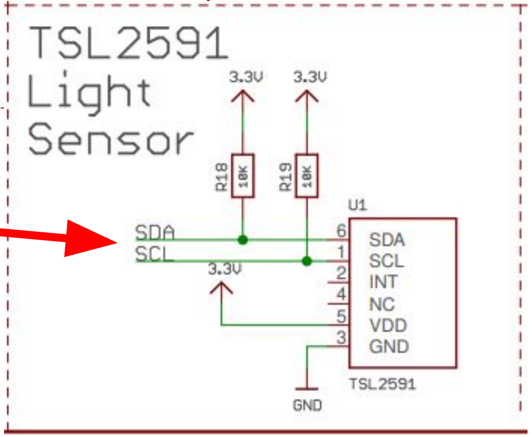


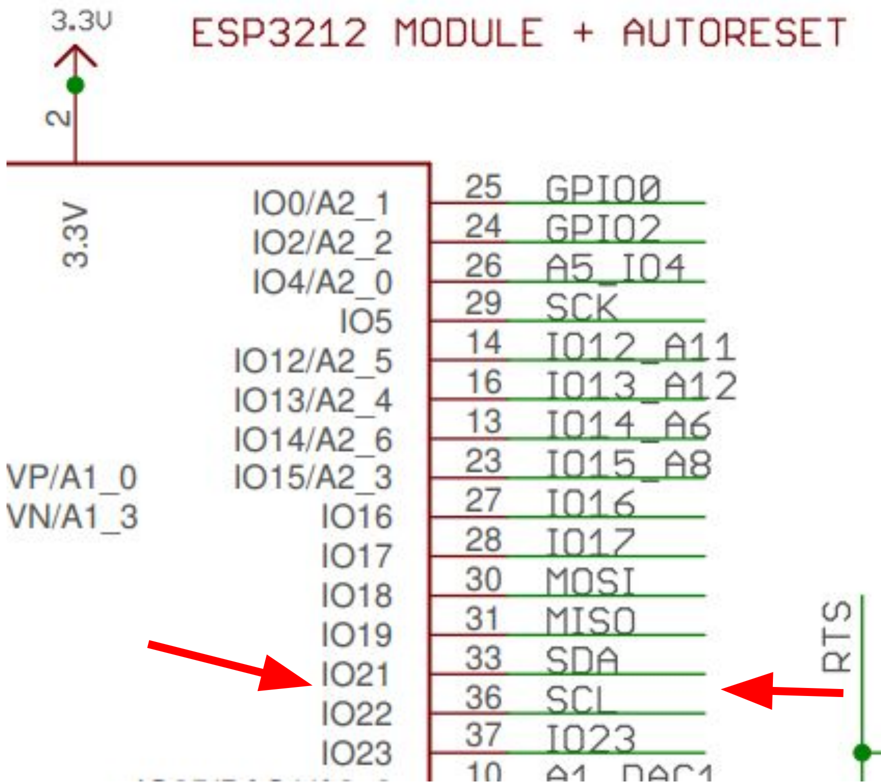


Ordering Code	Address	Interface
TSL25911FN	0x29	I^{2}C V$_{bus}$ = V$_{DD}$ Interface
TSL25913FN*	0x29	I^{2}C V$_{bus}$ = 1.8V

Figuring out MCU pins







Time to read some data

1. Exit screen

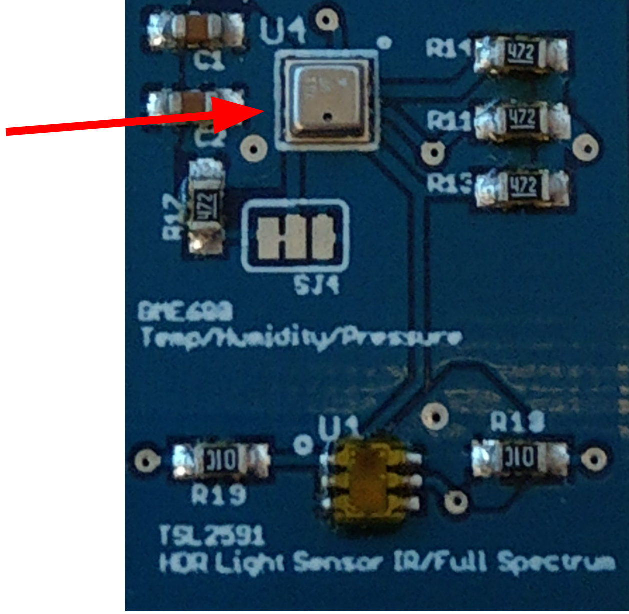
2. Upload the following using ampy:
```
# ampy --port /dev/ttyUSB0 put sensornode-stuff/src/bme280.py
# ampy --port /dev/ttyUSB0 put sensornode-stuff/src/tsl2591.py
# ampy --port /dev/ttyUSB0 put sensornode-stuff/src/usmbus
```
   - Note the last one is a directory

3. Open up the serial port again

Confirm file upload

```
>>> import os
>>> os.listdir()
['boot.py', 'bme280.py', 'tsl2591.py', 'usmbus']
>>>
```

BME280 - Environment Sensor



```
>>> from machine import Pin, I2C
>>> import machine
>>> import bme280

>>> pin_i2c_scl     = 22
>>> pin_i2c_sda      = 21

>>> bme280_address  = 0x77

>>> sensor_i2c = I2C( scl=Pin(pin_i2c_scl), sda=Pin(pin_i2c_sda) )

>>> bme = bme280.BME280( i2c=sensor_i2c, address=bme280_address )

>>> bme.values
('26.84C', '1015.59hPa', '17.71%')
>>>
```

TSL2591 - Light Sensor

```
>>> import tsl2591
>>> tsl = tsl2591.Tsl2591()
>>> tsl.get_full_luminosity()
(58, 14)
>>>
```

The TSL2591 driver is a very setup than the BME280. The I2C bus, and address, are hard
Coded into the driver:

```
    def __init__(self, scl_pinno=22, sda_pinno=21):
        self.i2c = I2C(scl=Pin(scl_pinno, Pin.IN),
                       sda=Pin(sda_pinno, Pin.IN))
```

It also makes use of more SMBus like support (usmbus)

Places to find more information on I2C:

- https://i2c.info/
- https://en.wikipedia.org/wiki/I%C2%B2C
- https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME280-DS002.pdf
- https://cdn-shop.adafruit.com/datasheets/TSL25911_Datasheet_EN_v1.pdf