Electronics
Serial Communication-I2C

Terry Sturtevant

Wilfrid Laurier University

February 13, 2019
Serial Communication - $I^2C$

Inter-Integrated Circuit Interface

Master/slave communication
Uses 2 signals (and Ground), SDA and SCL
Many slaves can be on the same bus since each has an address
Device addresses are pre-programmed, but can usually be changed
Synchronous, so master controls clock rate
Serial Communication - $I^2C$

- Inter-Integrated Circuit Interface
Serial Communication -\textsuperscript{I}2C

- Inter-Integrated Circuit Interface
- Master/slave communication
Serial Communication - I\(^2\)C

- Inter-Integrated Circuit Interface
- Master/slave communication
- Uses 2 signals (and Ground), SDA and SCL
Serial Communication - I²C

- Inter-Integrated Circuit Interface
- Master/slave communication
- Uses 2 signals (and Ground), SDA and SCL
- Many slaves can be on the same bus since each has an address
Serial Communication - \( I^2C \)

- Inter-Integrated Circuit Interface
- Master/slave communication
- Uses 2 signals (and Ground), SDA and SCL
- Many slaves can be on the same bus since each has an address

Device addresses are pre-programmed, but can usually be changed
Serial Communication - I²C

- Inter-Integrated Circuit Interface
- Master/slave communication
- Uses 2 signals (and Ground), SDA and SCL
- Many slaves can be on the same bus since each has an address

  *Device addresses are pre-programmed, but can usually be changed*

- Synchronous, so master controls clock rate
Bidirectional Communication on a Single Line
Bidirectional Communication on a Single Line

- How can multiple devices communicate on the same line?
Bidirectional Communication on a Single Line

- How can multiple devices communicate on the same line?
  Each device has an *open-collector* (or open-drain) output.
**Bidirectional Communication on a Single Line**

- How can multiple devices communicate on the same line?
  Each device has an *open-collector* (or open-drain) output. A pull-up resistor on the line allows it to go high when nothing is pulling it low.
Bidirectional Communication on a Single Line

How can multiple devices communicate on the same line?
Each device has an *open-collector* (or open-drain) output.
A pull-up resistor on the line allows it to go high when nothing is pulling it low.
Any device which is not speaking must let their output float.
Bidirectional Communication on a Single Line

How can multiple devices communicate on the same line?
Each device has an open-collector (or open-drain) output.
A pull-up resistor on the line allows it to go high when nothing is pulling it low.
Any device which is not speaking must let their output float.
Only one device can pull the line low at a time.
Bidirectional Communication on a Single Line

- How can multiple devices communicate on the same line?
  - Each device has an *open-collector* (or open-drain) output.
  - A pull-up resistor on the line allows it to go high when nothing is pulling it low.
  - Any device which is not speaking must let their output float.
  - Only one device can pull the line low at a time.

  **On the Raspberry Pi board, there are 1.8kΩ pull-up resistors to 3.3V, so external ones are not normally required.**
Mixing devices with different supply voltages
Mixing devices with different supply voltages

Do you need optoisolators if devices require different supply voltages?
Mixing devices with different supply voltages

Do you need optoisolators if devices require different supply voltages?

Each device has an *open-collector* (or open-drain) output.
Mixing devices with different supply voltages

Do you need optoisolators if devices require different supply voltages?

Each device has an *open-collector* (or open-drain) output.

A pull-up resistor on the line allows it to go high when nothing is pulling it low.
Mixing devices with different supply voltages

Do you need optoisolators if devices require different supply voltages?

Each device has an open-collector (or open-drain) output. A pull-up resistor on the line allows it to go high when nothing is pulling it low. If the pull-up resistors only go to the lowest supply voltage, then the data line will never be too high.
Mixing devices with different supply voltages

Do you need optoisolators if devices require different supply voltages?

Each device has an *open-collector* (or open-drain) output. A pull-up resistor on the line allows it to go high when nothing is pulling it low.

If the pull-up resistors only go to the *lowest* supply voltage, then the data line will never be too high.

Since the Raspberry Pi has 1.8kΩ pull-up resistors to 3.3V, connecting to an \( I^2C \) device with a 5V supply will not cause a problem for the Pi.
Mixing devices with different supply voltages

Do you need optoisolators if devices require different supply voltages?

Each device has an open-collector (or open-drain) output. A pull-up resistor on the line allows it to go high when nothing is pulling it low.

If the pull-up resistors only go to the lowest supply voltage, then the data line will never be too high.

Since the Raspberry Pi has 1.8kΩ pull-up resistors to 3.3V, connecting to an I²C device with a 5V supply will not cause a problem for the Pi.

Note, however, if 3.3V inputs are not high enough for the device, then it might not operate correctly.
All floating; SDA is HIGH.
One output LOW; SDA is LOW.
I^2C ; bits are read when SCL is HIGH

ACK is sent by receiver if OK

sender must release SDA after LSB
- I²C; bits are read when SCL is HIGH
- NACK is sent by master-receiver if OK
- sender must release SDA after LSB
• $I^2C$ write to slave register
Slave address

START  W

$^{2}$C write to slave register
Slave address

START   WA

- $I^2C$ write to slave register
I²C write to slave register
- I²C write to slave register
- I²C write to slave register
I²C write to slave register
I²C write to slave register
I²C read from slave register
I²C read from slave register
- I²C read from slave register
Serial Communication - I2C

Raspberry Pi Python I2C Library

Slave address
Register number
START WA

- I²C read from slave register
• I^2C read from slave register
I²C read from slave register
I²C read from slave register
I²C read from slave register
I²C read from slave register
- **I²C read from slave register**
Smbus

Serial Communication - I2C
Raspberry Pi Python I2C Library

```python
bus = smbus.SMBus(1)  # create object

# read from device
bus.read_i2c_block_data(addr, cmd)

# write to device
bus.write_i2c_block_data(addr, cmd, vals)
```
Smbus

- `bus = smbus.SMBus(1)`
  - create object
**Smbus**

- `bus = smbus.SMBus(1)`
  - create object
- `bus.read_i2c_block_data(addr, cmd)`
  - read from device
Smbus

- `bus = smbus.SMBus(1)`
  create object
- `bus.read_i2c_block_data(addr, cmd)`
  read from device
- `bus.write_i2c_block_data(addr, cmd, vals)`
  write to device
Smbus

```python
#!/usr/bin/python
import smbus
bus = smbus.SMBus(1)
DEVICE_ADDRESS = 0x15
DEVICE_REG_MODE1 = 0x00
DEVICE_REG_LEDOUT0 = 0x1d
#Write a single register
bus.write_byte_data(DEVICE_ADDRESS, DEVICE_REG_MODE1, 0x80)
#Write an array of registers
led_out_values = [0xff, 0xff, 0xff, 0xff, 0xff, 0xff]
bus.write_i2c_block_data(DEVICE_ADDRESS, DEVICE_REG_LEDOUT0, led_out_values)
```
#!/usr/bin/python
import smbus
bus = smbus.SMBus(1)
DEVICE_ADDRESS = 0x15
DEVICE_REG_MODE1 = 0x00
DEVICE_REG_LEDOUT0 = 0x1d

#Write a single register
bus.write_byte_data(DEVICE_ADDRESS,
                     DEVICE_REG_MODE1, 0x80)

#Write an array of registers
ledout_values = [0xff, 0xff,
                  0xff, 0xff, 0xff, 0xff]
bus.write_i2c_block_data(DEVICE_ADDRESS,
                         DEVICE_REG_LEDOUT0, ledout_values)