Cyber-Physical Systems

Communication

UNIVERSITY AT ALBANY
State University of New York

ICEN 553/453– Fall 2018

Prof. Dola Saha

Why do we need Communication?

- Connect different systems together
  - Two embedded systems
  - A desktop and an embedded system
- Connect different chips together in the same embedded system
  - MCU to peripheral
  - MCU to MCU

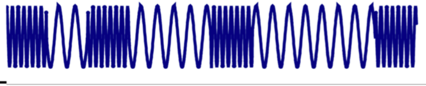
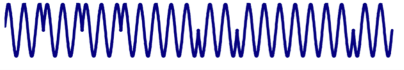
What determines how much we can transmit?

$$C = W log_2\left(\frac{S+N}{N}\right)$$

- Shannon's noisy channel coding theorem
  - Says you can achieve error-free communicate at any
- Rate up to the *channel capacity*, and can't do any better
  - C: channel capacity, in bits / s
  - W: bandwidth amount of frequency "real estate", in Hz (cycles / s)
  - S: Signal power
  - N: Noise power

Communication Methods

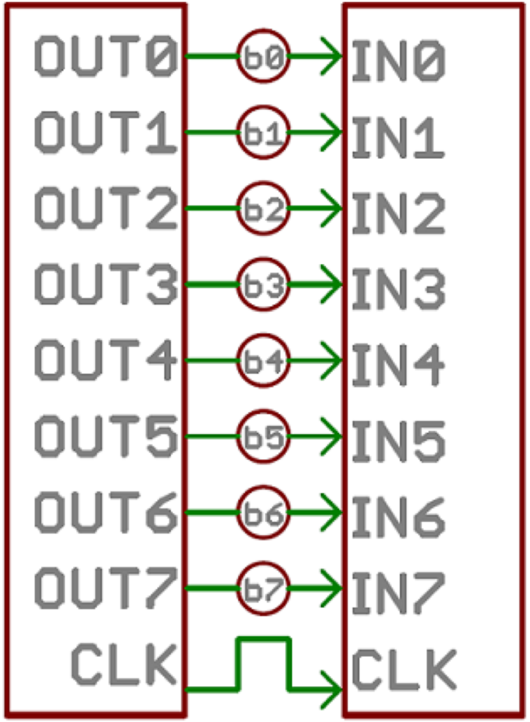
- Different physical layers methods: wires, radio frequency (RF), optical (IR)
- Different encoding schemes: amplitude, frequency, and pulse-width modulation

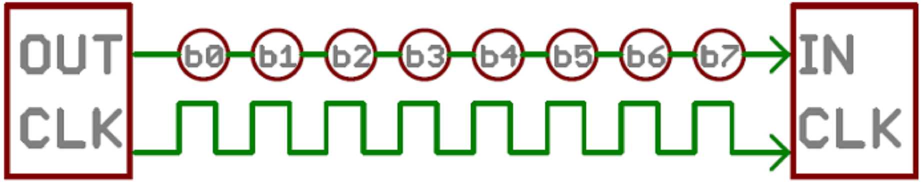
Modulation Technique	Waveform
No modulation (Baseband)	
On-Off Keying (OOK)	
Amplitude Modulation	
Frequency Shift Keying (FSK)	
Binary Phase Shift Keying (BPSK)	
Direct Sequence Spread Spectrum (DSSS), etc	

Dimensions to consider

- bandwidth – number of wires – serial/parallel
- speed – bits/bytes/words per second
- timing methodology – synchronous or asynchronous
- number of destinations/sources
- arbitration scheme – daisy-chain, centralized, distributed
- protocols – provide some guarantees as to correct communication

Parallel and Serial Bus





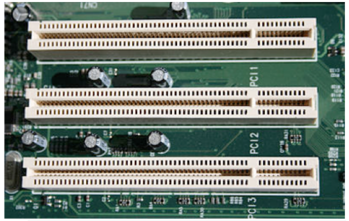
Parallel and Serial Communication

- Serial
  - Single wire or channel to transmit information one bit at a time
  - Requires synchronization between sender and receiver
  - Sometimes includes extra wires for clock and/or handshaking
  - Good for inexpensive connections (e.g.,terminals)
  - Good for long-distance connections (e.g.,LANs)
- Parallel
  - Multiple wires to transmit information one byte or word at a time
  - Good for high-bandwidth requirements (CPU to disk)
  - Crosstalk creates interference between multiple wires
  - Length of link increases crosstalk
  - More expensive wiring/connectors/current requirements

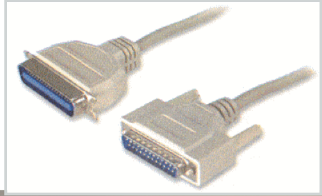
Parallel vs. Serial Digital Interfaces

- Parallel (one wire per bit)
  - ATA: Advanced Technology Attachment
  - PCI: Peripheral Component Interface
  - SCSI: Small Computer System Interface
  - Serial (one wire per direction)
  - RS-232
  - SPI: Serial Peripheral Interface bus
  - I2C: Inter-Integrated Circuit
  - USB: Universal Serial Bus
  - SATA: Serial ATA
  - Ethernet, IrDA, Firewire, Bluetooth, DVI, HDMI
- Mixed (one or more “lanes”)
  - PCIe: PCI Express

PCI



SCSI



USB

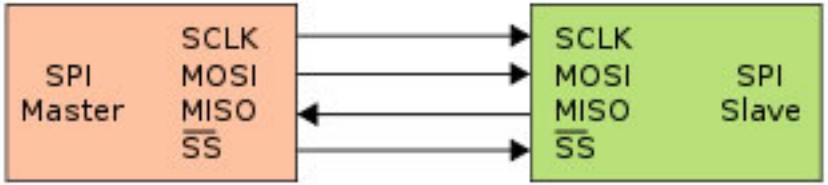
RS-232

Parallel vs Serial Digital Interfaces

- Parallel connectors have been replaced by Serial
  - Significant crosstalk/inter-wire interference for parallel connectors
  - Maintaining synchrony across the multiple wires
  - Serial connection speeds can be increased by increasing transmission freq, but parallel crosstalk gets worse at increased freq

Serial Peripheral Interface (SPI)

- Synchronous full-duplex communication
- Can have multiple slave devices
- No flow control or acknowledgment
- Slave cannot communicate with slave directly.



Serial Peripheral Interface

http://upload.wikimedia.org/wikipedia/commons/thumb/e/ed/SPI_single_slave.svg/350px-SPI_single_slave.svg.png

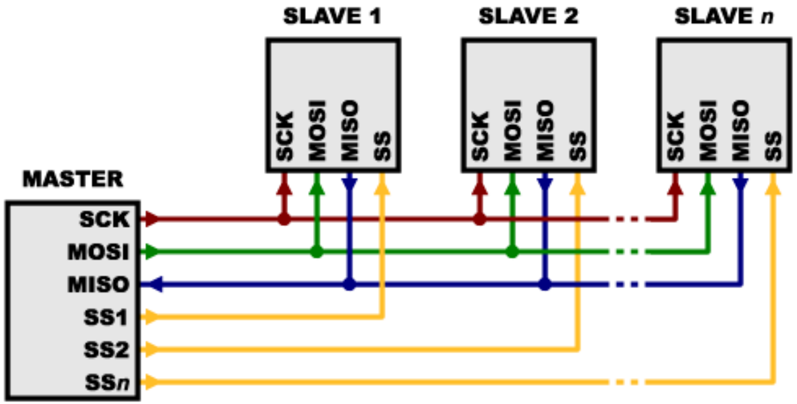
SCLK: serial clock

SS: slave select (active low)

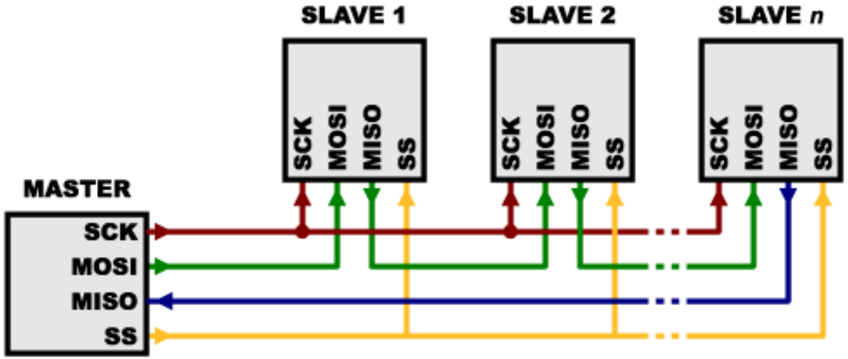
MOSI: master out slave in

MISO: master in slave out

SPI – Point-to-point and Daisy Chain



Point-to-point



Daisy Chain

SCLK: serial clock

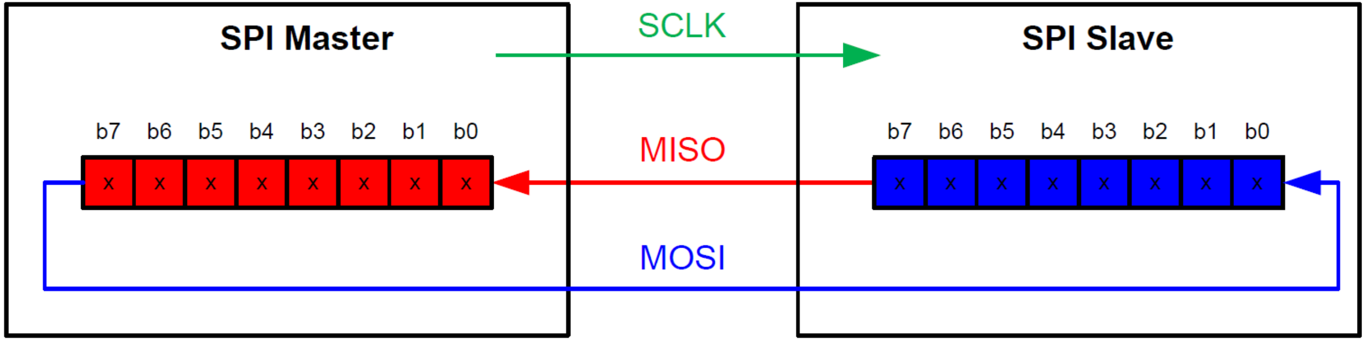
SS: slave select (active low)

MOSI: master out slave in

MISO: master in slave out

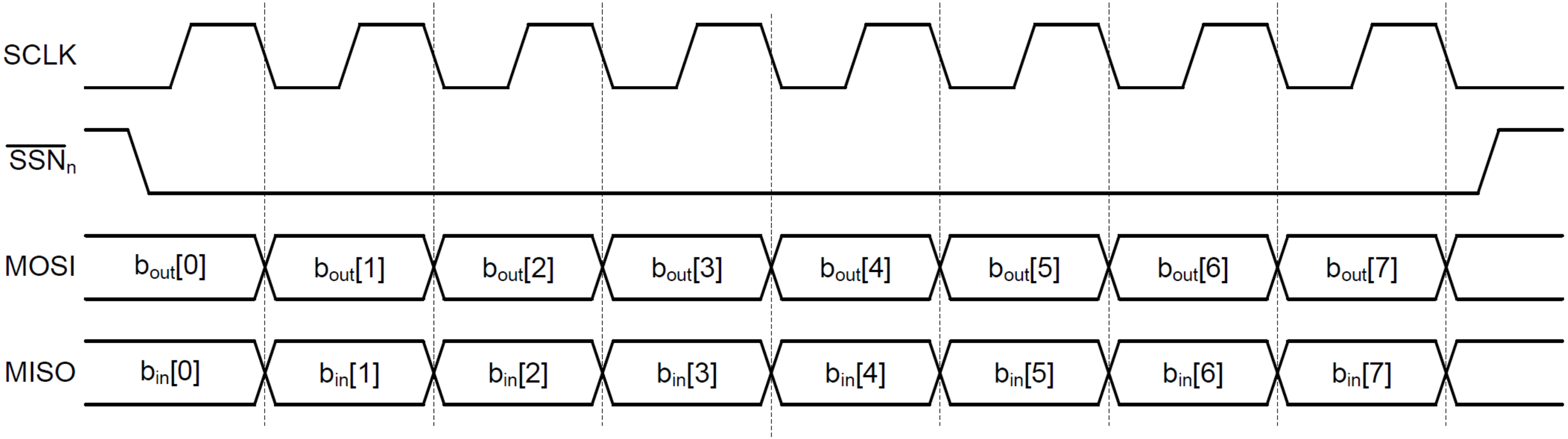
Pictures: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/

Data Exchange



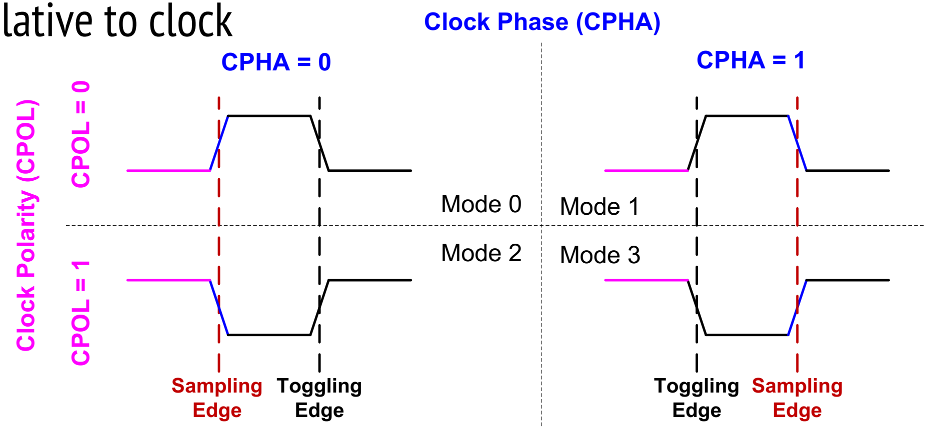
- Master has to provide clock to slave
- Synchronous exchange: for each clock pulse, a bit is shifted out and another bit is shifted in at the same time. This process stops when all bits are swapped.
- Only master can start the data transfer

Clock


SCLK
SSNn
MOSI
bout[0]
bout[1]
bout[2]
bout[3]
bout[4]
bout[5]
bout[6]
bout[7]
MISO
bin[0]
bin[1]
bin[2]
bin[3]
bin[4]
bin[5]
bin[6]
bin[7]

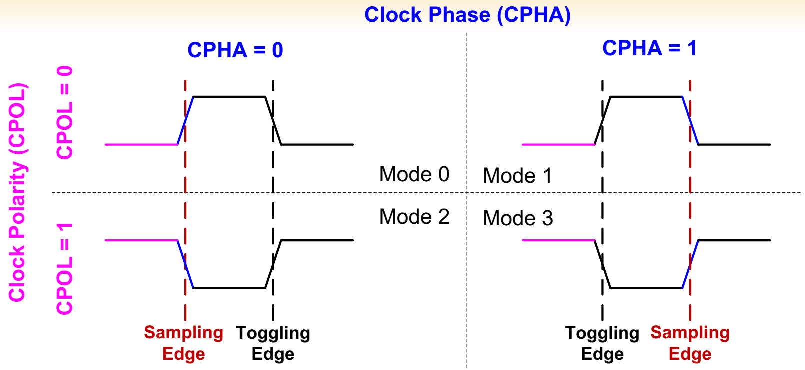
Clock Phase and Polarity

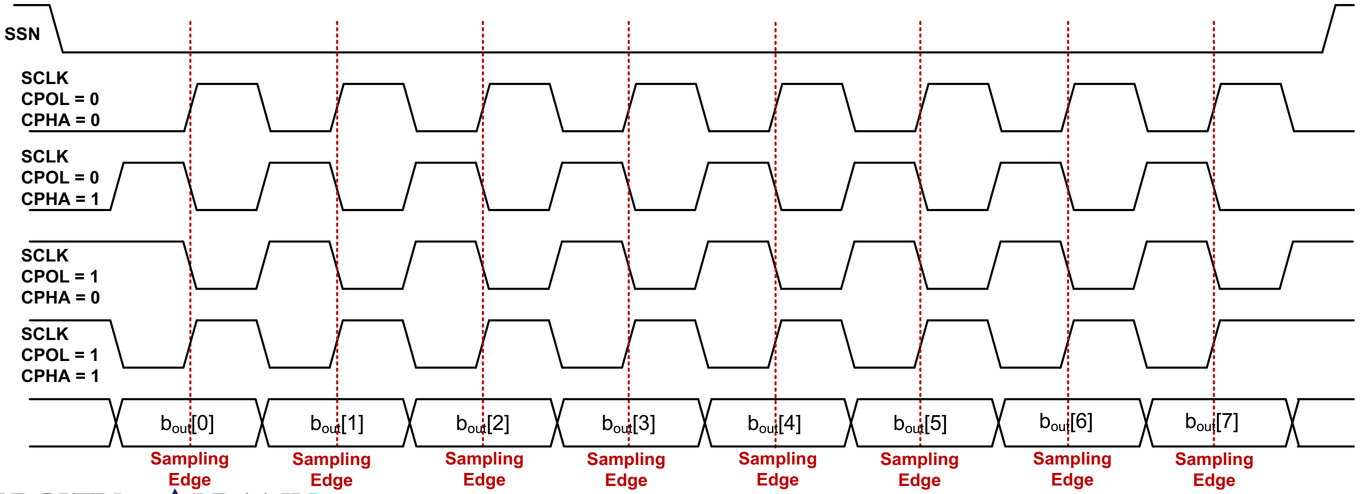
- CPHA (Clock PHase)
  - = 0 or =1, determines when data goes on bus relative to clock
- CPOL (Clock POLarity)
  - =0 clock idles low between transfers
  - =1 clock idles high between transfers



- CPOL = 0 ⟶ SCLK is pushed to low during idle. Otherwise, pulled to high during idle.
- CPHA = 0 ⟶ the first clock transition (either rising or falling) is the first data capture edge. Otherwise, the second clock transition is the first data capture edge.
- Combination of CPOL and CPHA determines the clock edge for transmitting and receiving.

Clock Phase and Polarity





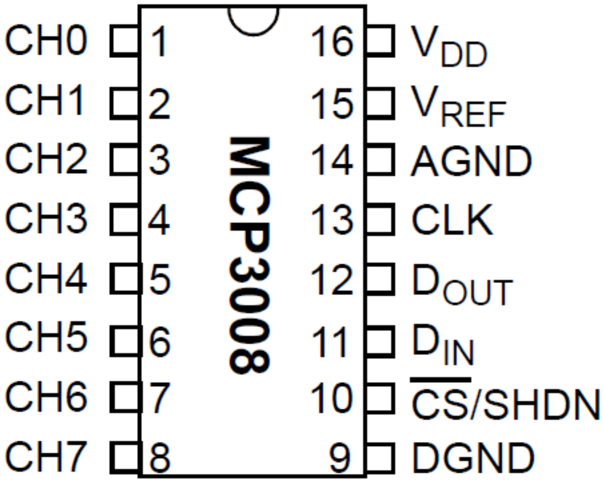
UNIVERSITY AT ALBANY
State University of New York

SPI: Pros and Cons

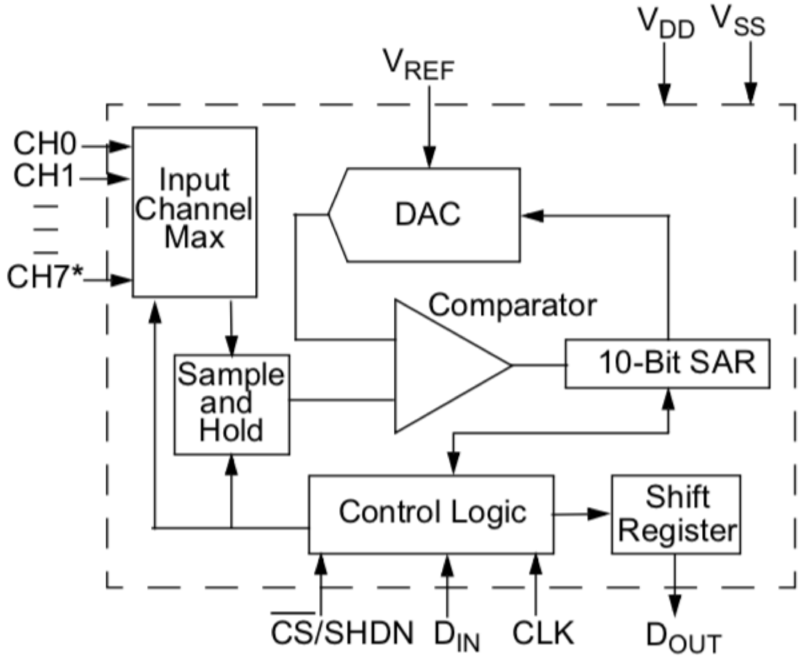
- Pros
  - Simplest way to connect 1 peripheral to a micro
  - Fast (10s of Mbits/s, not on MSP) because all lines actively driven, unlike I2C
  - Clock does not need to be precise
  - Nice for connecting 1 slave
- Cons
  - No built-in acknowledgement of data
  - Not very good for multiple slaves
  - Requires 4 wires
  - 3 wire variants exist…some get rid of full duplex and share a data line, some get rid of slave select

Analog to Digital Converter

- DGND : digital ground pin for the chip
- CS : chip select.
- DIN : data in from the MC itself.
- DOUT: data out pin.
- CLK: clock pin.
- AGND: analog ground and obviously connects to ground.
- VREF: analog reference voltage. You can change this if you want to change the scale. You probably want to keep it the same so keep this as 3.3v.
- VDD: positive power pin for the chip.



MCP 3008



* **Note:** Channels 4-7 are available on MCP3008 Only

$$Digital\ Output\ Code = \frac{1024 \times V_{IN}}{V_{REF}}$$

Where:

V_{IN} = analog input voltage

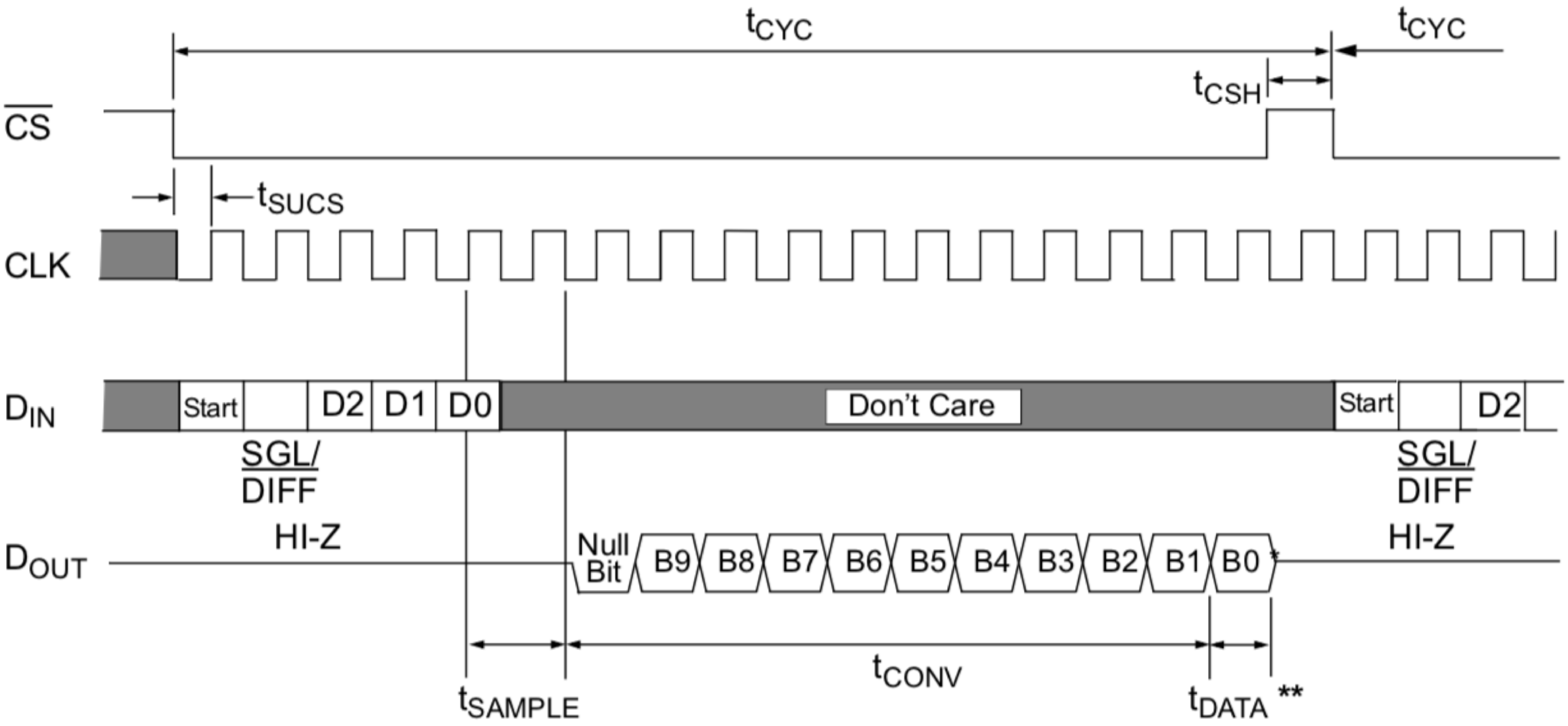
V_{REF} = analog input voltage

Configuration Bits

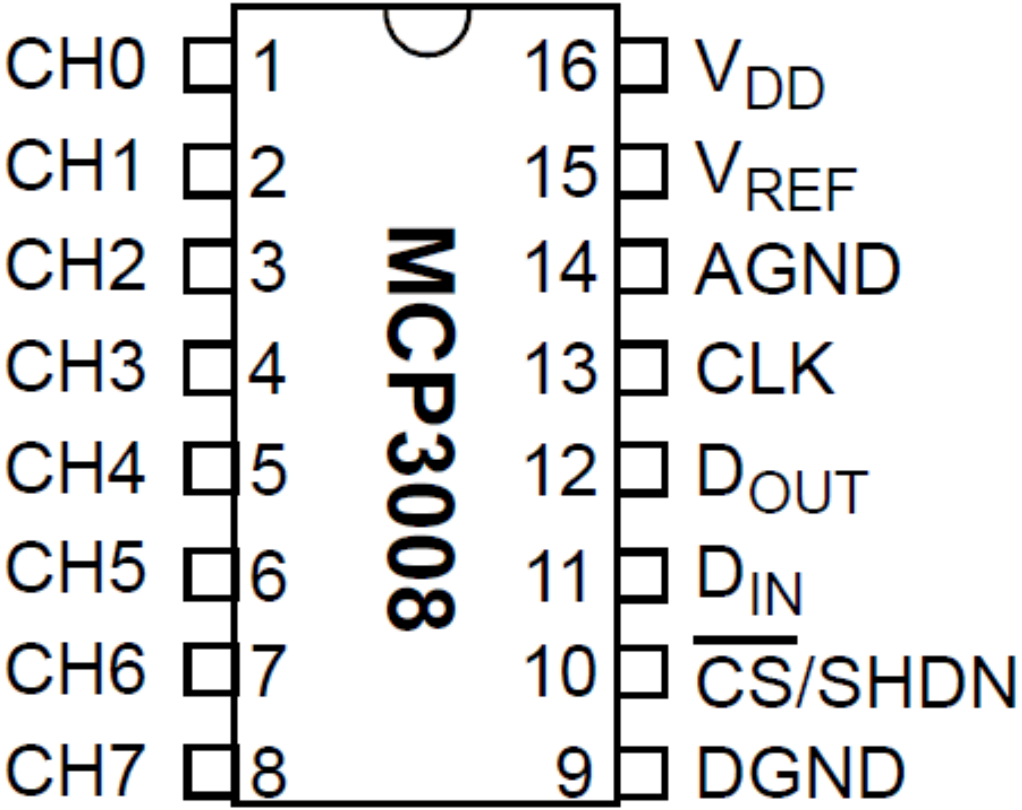
- Single or Differential
- D2, D1, D0

Control Bit Selections				Input Configuration	Channel Selection
Single/Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7
0	0	0	0	differential	CH0 = IN+ CH1 = IN-
0	0	0	1	differential	CH0 = IN- CH1 = IN+
0	0	1	0	differential	CH2 = IN+ CH3 = IN-
0	0	1	1	differential	CH2 = IN- CH3 = IN+
0	1	0	0	differential	CH4 = IN+ CH5 = IN-
0	1	0	1	differential	CH4 = IN- CH5 = IN+
0	1	1	0	differential	CH6 = IN+ CH7 = IN-
0	1	1	1	differential	CH6 = IN- CH7 = IN+

Communication

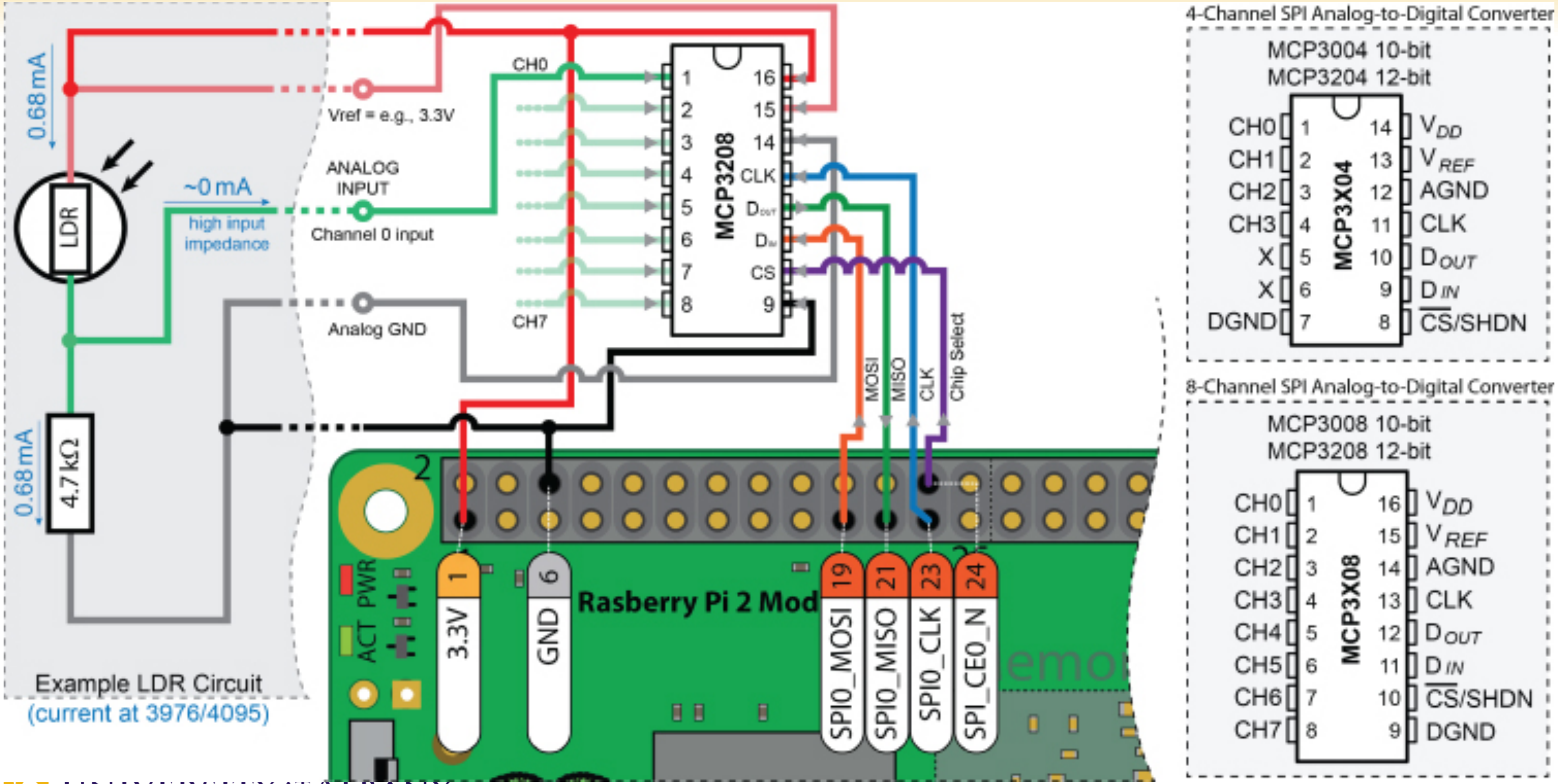


Analog to Digital Converter

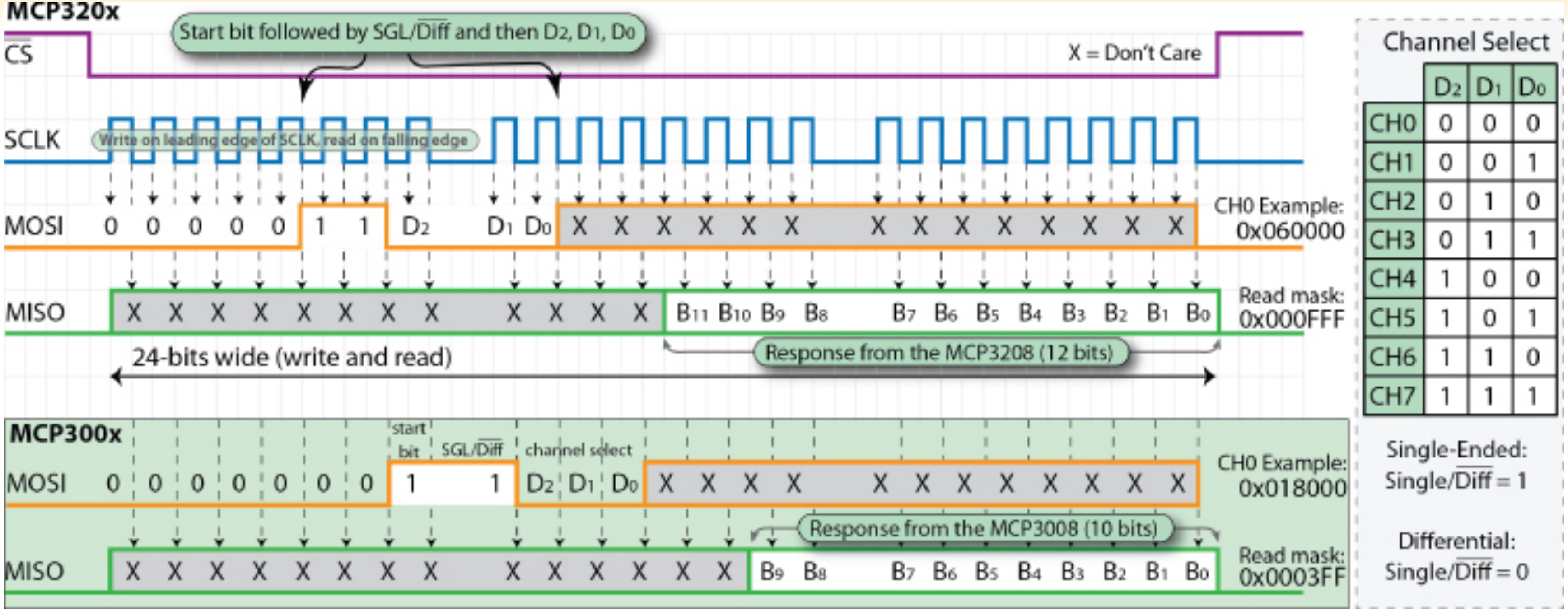


RPi 3.3V
RPi 3.3V
RPi GND
RPi SClk
RPi MISO
RPi MOSI
RPi CE0
RPi GND

Connect a Sensor



Channel Select



- The device will begin to sample the analog input on the fourth rising edge of the clock after the start bit has been received. The sample period will end on the falling edge of the fifth clock following the start bit.

Enable SPI in Raspberry PI

- sudo raspi-config
- 5 Interfacing Options
- P4 SPI
- Would you like the SPI interface to be enabled?
  - Select Yes
- The SPI interface is enabled
  - Select OK
- Finish

Has SPI been really enabled?

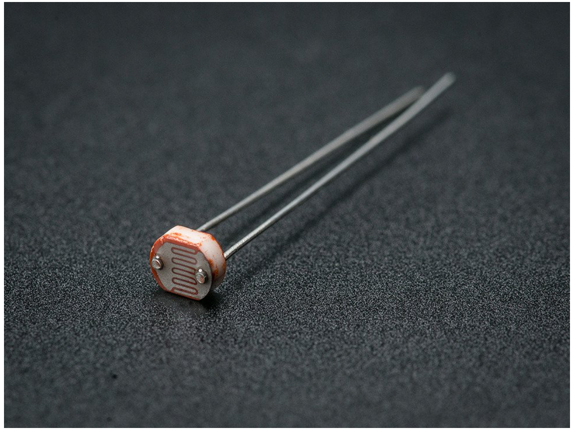
- sudo ls /dev/spi*
- /dev/spidev0.0   /dev/spidev0.1

Install Adafruit MCP 3008 Python Library

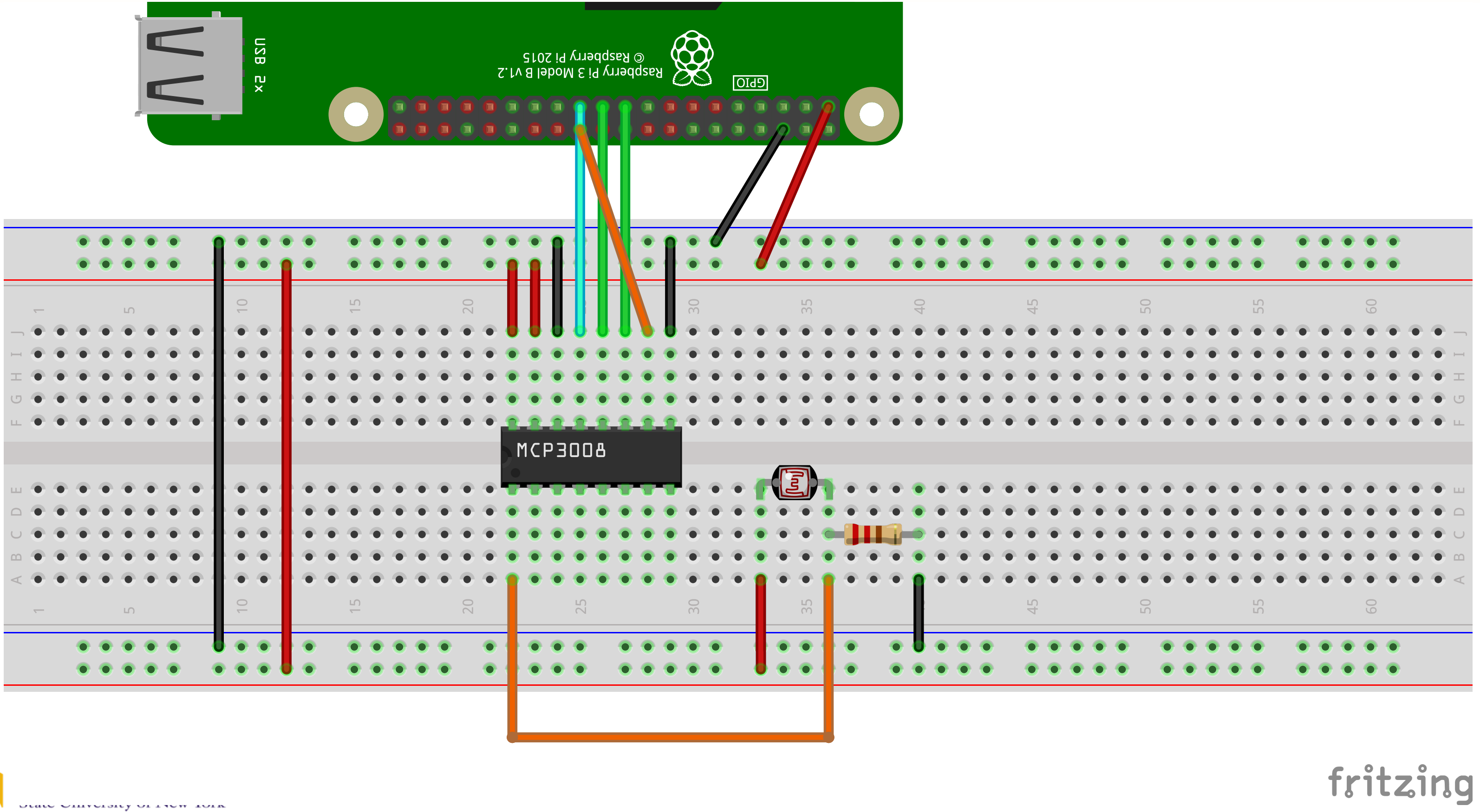
- sudo apt-get update
- sudo apt-get install build-essential python-dev python-smbus git
- cd ~
- git clone https://github.com/adafruit/Adafruit_Python_MCP3008.git
- cd Adafruit_Python_MCP3008
- sudo python setup.py install

Photocell

- Measure Voltage drop depending on Lux
- **Resistance range:** 200KΩ (dark) to 10KΩ (10 lux brightness)
- **Sensitivity range:** Respond to light between 400nm (violet) and 600nm (orange) wavelengths, peaking at about 520nm (green).
- **Power supply:** Up to 100V
  - uses less than 1mA of current on average



Connect Photocell



Run existing code

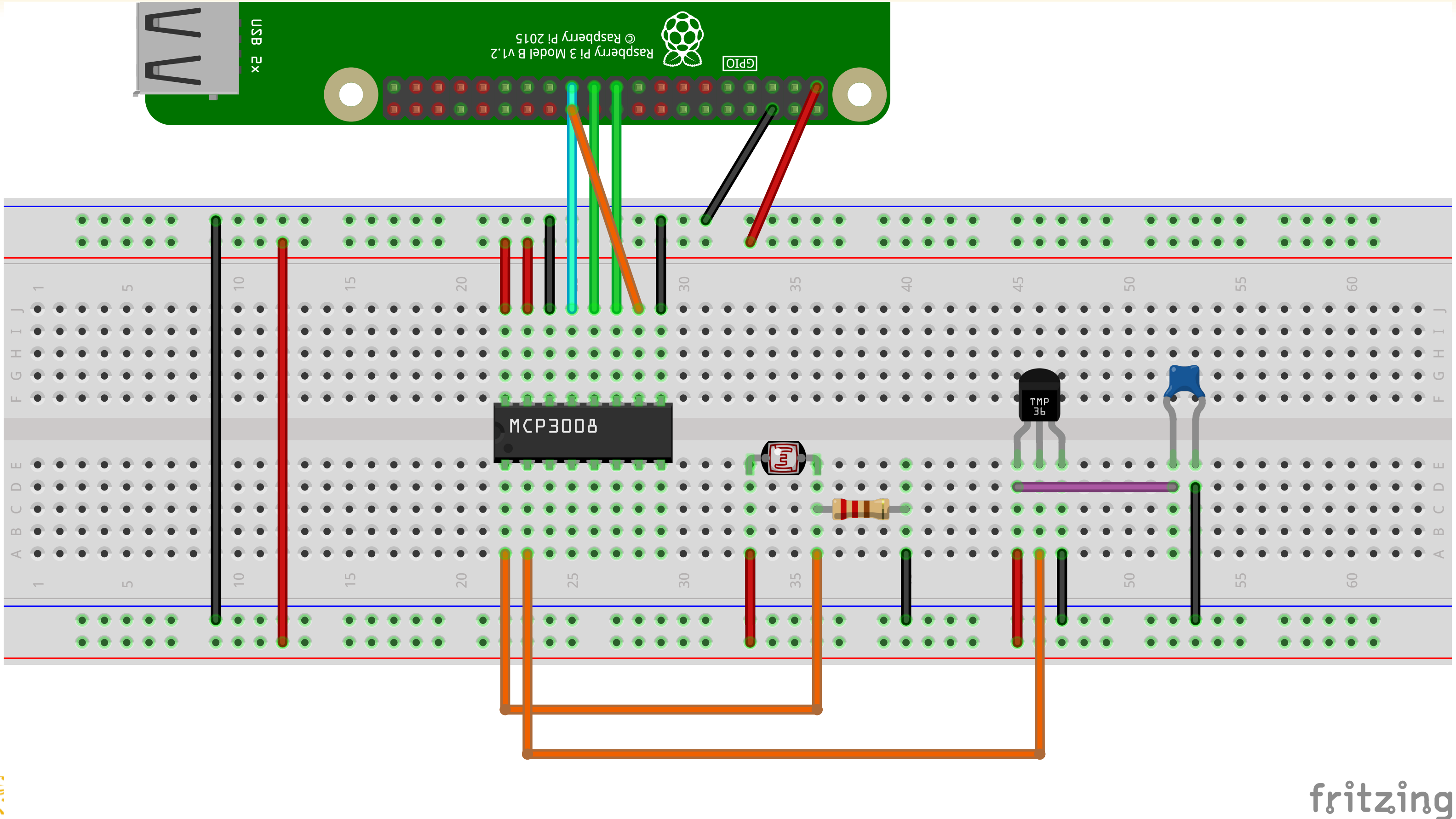
- cd ~/Adafruit_Python_MCP3008/examples/
- Change the simpletest.py code to enable Hardware SPI and disable Software SPI
- Then run the simpletest.py

Output of simpletest.py

```
Reading MCP3008 values, press Ctrl-C to quit...
|    0 |    1 |    2 |    3 |    4 |    5 |    6 |    7 |
---------------------------------------------------------
|  891 |  263 |  116 |  106 |  106 |  115 |  132 |  183 |
|  894 |  309 |  139 |  129 |  127 |  138 |  157 |  214 |
|  894 |  320 |  146 |  134 |  134 |  142 |  158 |  213 |
|  893 |  274 |  128 |  117 |  114 |  121 |  132 |  174 |
|  895 |  201 |   96 |   87 |   84 |   87 |   95 |  123 |
|  895 |   99 |   49 |   44 |   39 |   37 |   35 |   37 |
|  894 |    0 |    2 |    0 |    0 |    0 |    0 |    0 |
```

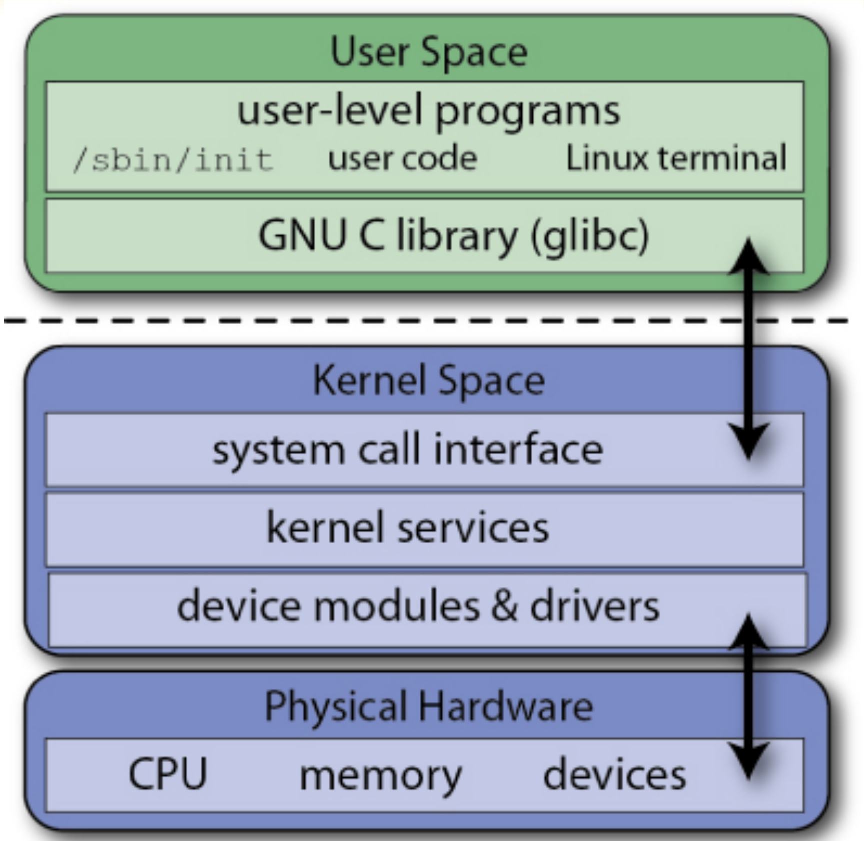
- Why are there values in unused channels?
- What can be the range of values in the used channel?

Connect Temperature Sensor



Kernel and User Space

ioctl()



SPI Bus on Linux

- lsmod | grep spi
- modprobe spidev
- modprobe spi_bcm2835
- dmesg | grep spi

SPI Using User->Kernel Modules

- ioctl
  - /usr/include/asm-generic/ioctl.h
- spidev
  - /usr/include/linux/spi/spidev.h
- Kernel Module
  - https://github.com/raspberrypi/linux/blob/rpi-3.12.y/drivers/spi/spi-bcm2835.c

ioctl() – Input/Output Control

- **int ioctl(int *fd*, unsigned long *request*, ...);**
- The **ioctl**() system call manipulates the underlying device parameters of special files.
- Input Arguments
  - fd – File Descriptor
  - request – Device dependent request code
  - Third Argument – Integer value of a pointer to data for transfer
- Return
  - 0 on success.
  - -1 on error.

spi_ioc_transfer structure

```
struct spi_ioc_transfer {
    __u64       tx_buf;
    __u64       rx_buf;

    __u32       len;
    __u32       speed_hz;

    __u16       delay_usecs;
    __u8        bits_per_word;
    __u8        cs_change;
    __u8        tx_nbits;
    __u8        rx_nbits;
    __u16       pad;

    /* If the contents of 'struct spi_ioc_transfer' ever change
     * incompatibly, then the ioctl number (currently 0) must change;
     * ioctls with constant size fields get a bit more in the way of
     * error checking than ones (like this) where that field varies.
     *
     * NOTE: struct layout is the same in 64bit and 32bit userspace.
     */
};
```

SPI Dev Interface

- https://www.kernel.org/doc/Documentation/spi/spidev

- /dev/spidevB.C (B=bus, C=slave number).
  - On RPi it is /dev/spidev0.0

- To open the device:
  - fd=open("/dev/spidev0.0",O_RDWR);

SPI Dev Interface

- To set the mode:
  - int mode=SPI_MODE_0;
  - result = ioctl(spi_fd , SPI_IOC_WR_MODE , &mode);
- To set the bit order:
  - int lsb_mode =0;
  - result = ioctl(spi_fd, SPI_IOC_WR_LSB_FIRST, &lsb_mode);

SPI Dev Interface

- To transfer:
  - ret = ioctl(fd, SPI_IOC_MESSAGE(1), &tr);
- To close:
  - close(fd);

MCP 3008 Data Transfer

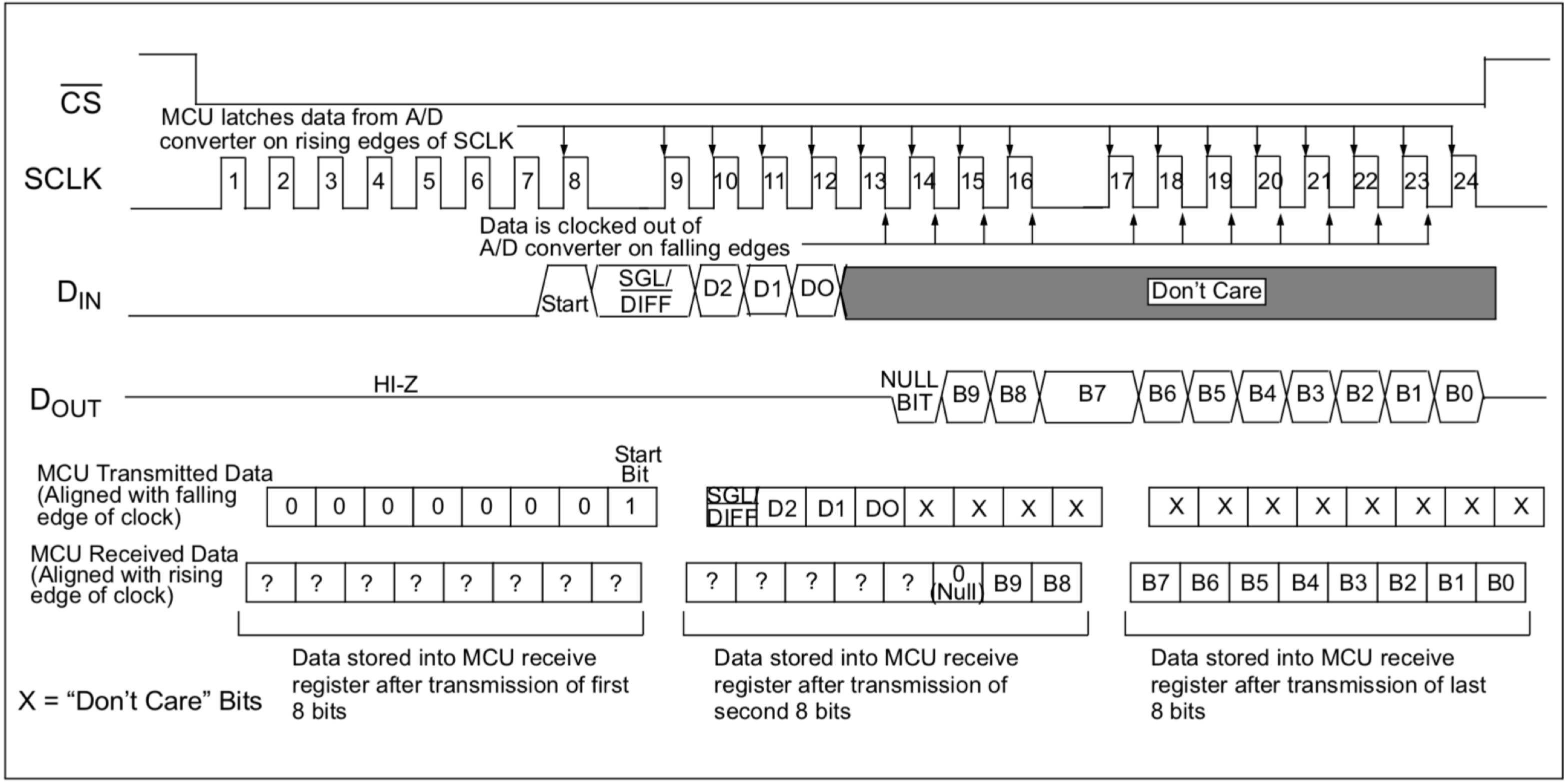


FIGURE 6-1: *SPI Communication with the MCP3004/3008 using 8-bit segments (Mode 0,0: SCLK idles low).*