物聯網與微處理機系統設計
Internet of Things and Microprocessor System Design

Lecture 05 – I2C, SPI, Accelerometer

Lecturer: 陳彥安 Chen, Yan-Ann

YZU CSE

Outline

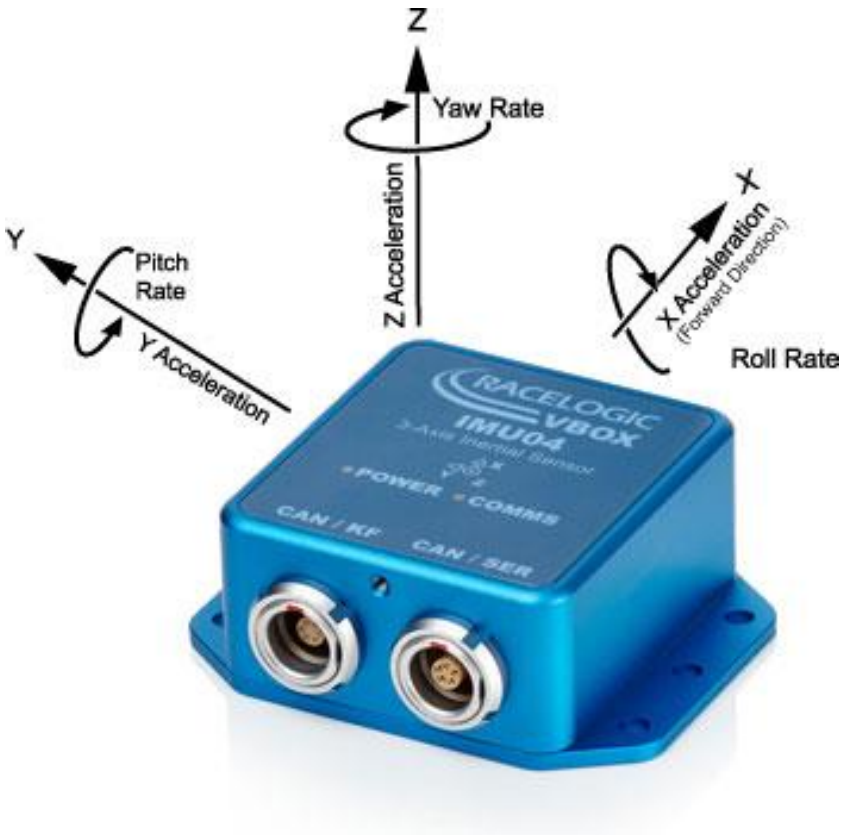
- IMU
- SPI
- I2C
- Lab

Outline

- IMU
- SPI
- I2C
- Lab

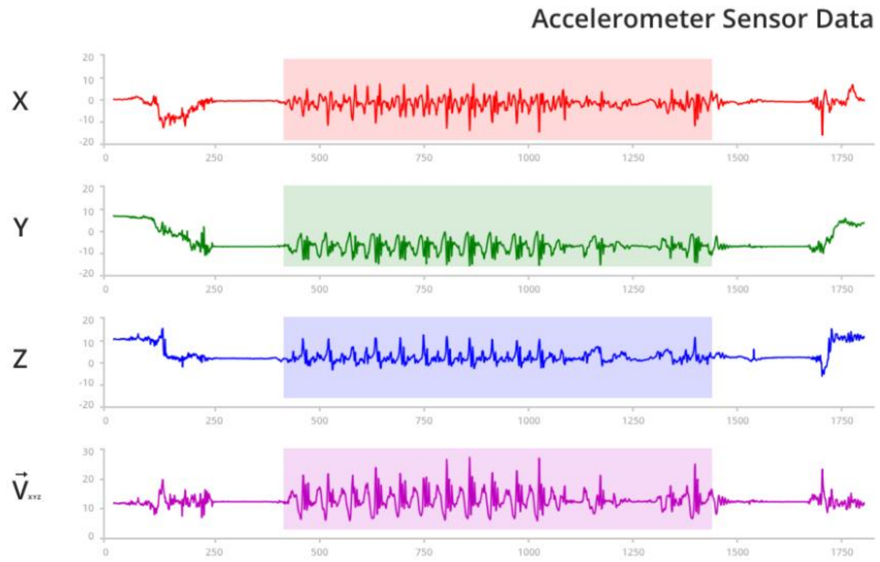
IMU

- Inertial Measurement Unit (IMU)
  - 3-axis accelerometer
  - 3-axis gyroscope
  - 3-axis magnetometer

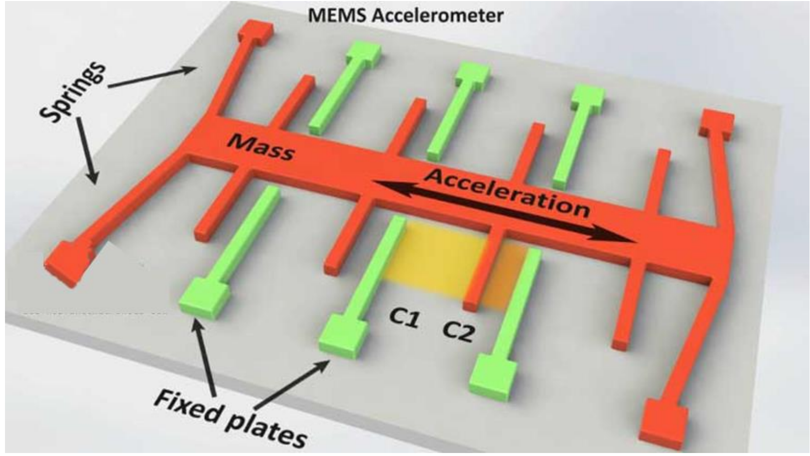


Accelerometer

- Accelerations in 3-axis
  - Including gravity

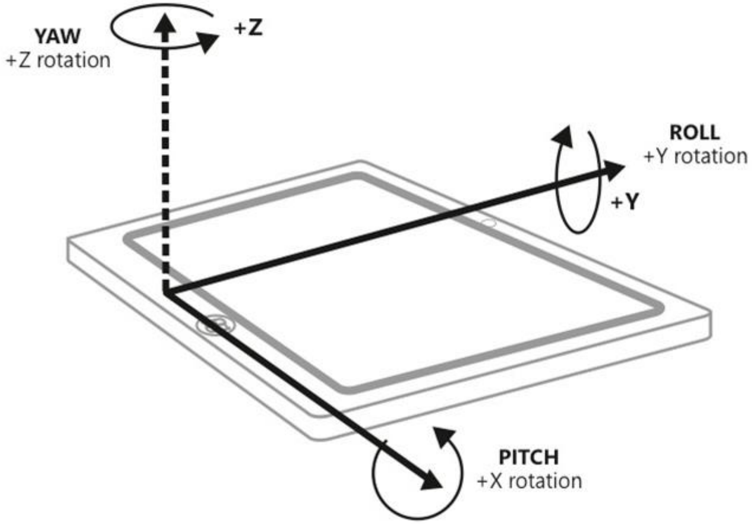


- Measure acceleration by measuring change in capacitance.

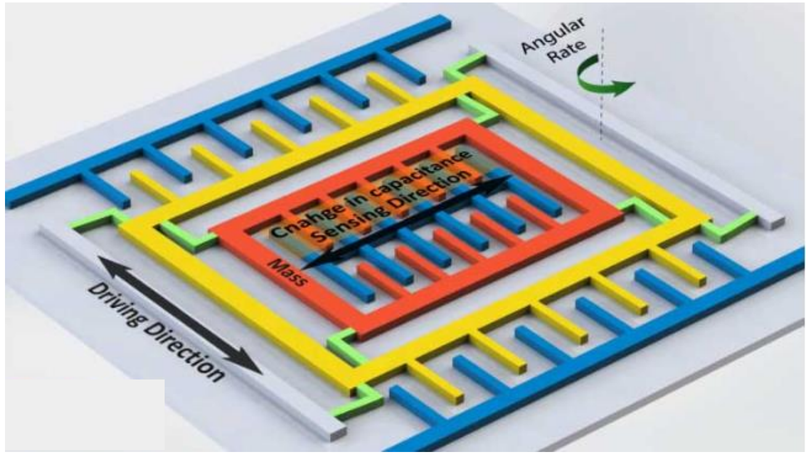


Gyroscope

- Measure angular velocity



- External angular rate will be applied a flexible part of the mass would move and make the perpendicular displacement.



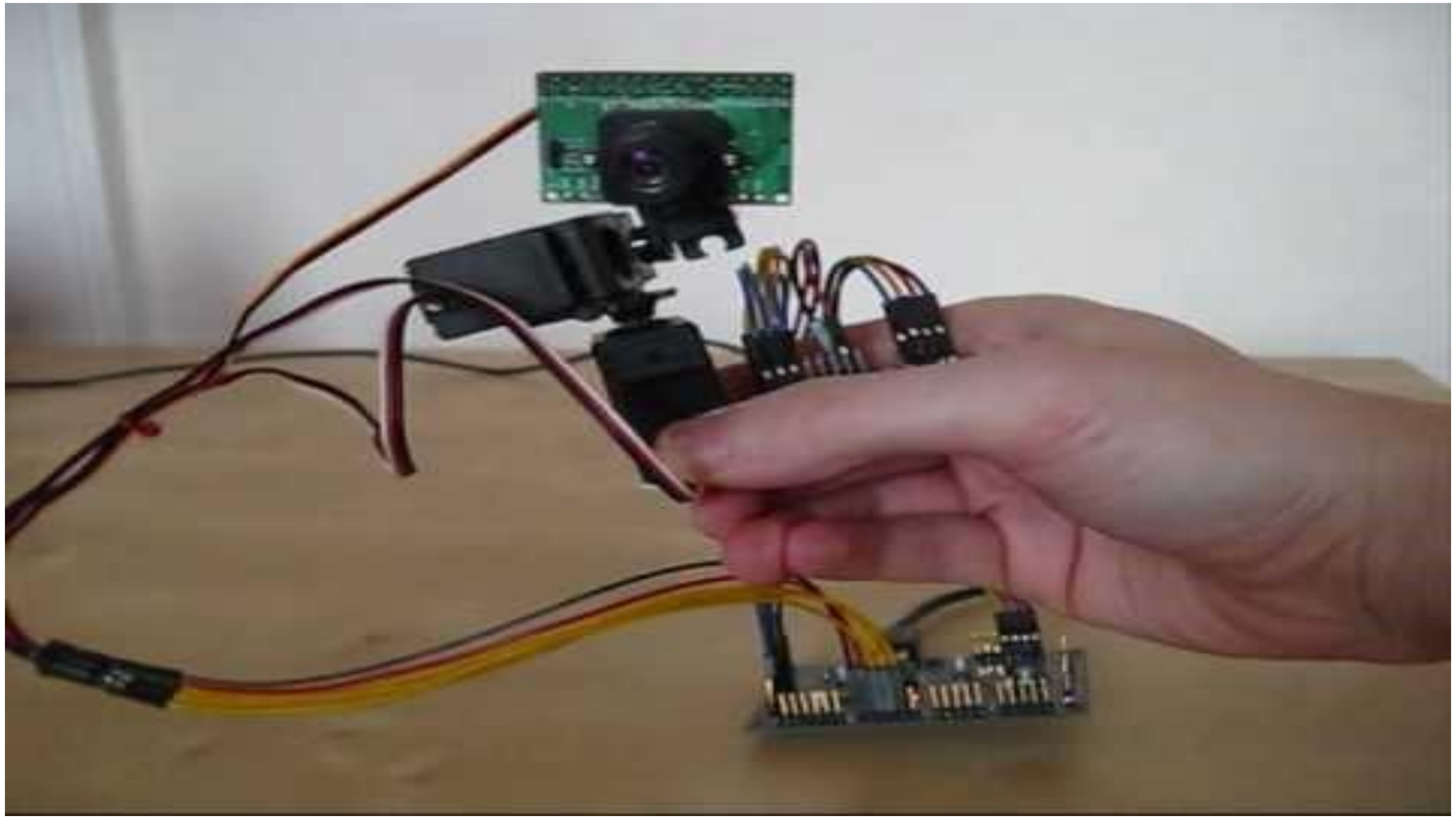
Applications (1/5)

- How Do Smartphones Know Which Is The Right Side Up?
  - https://www.scienceabc.com/innovation/smartphones-change-orientation-horizontal-landscape-gravity-sensor-accelerometer.html

- https://youtu.be/KZVgKu6v808

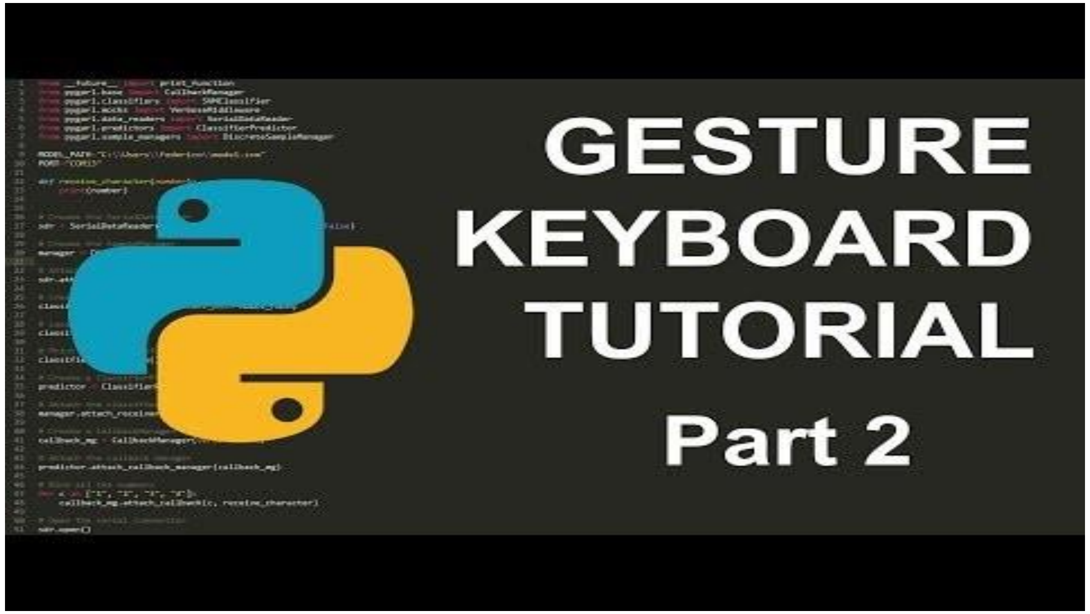
Applications (2/5)

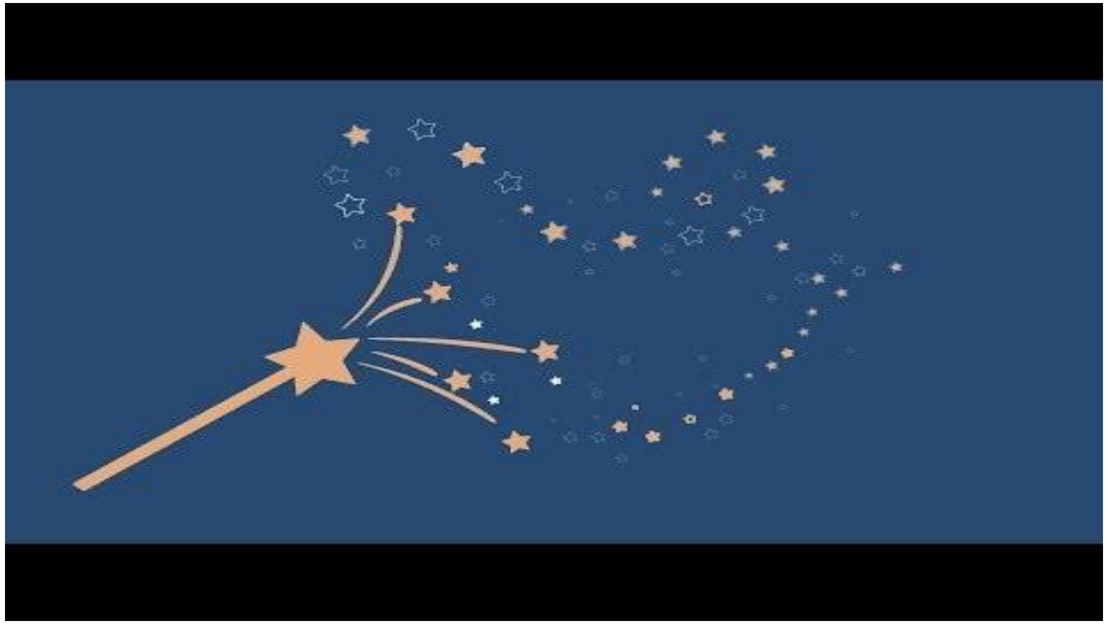
- Camera control / stabilization
  - https://youtu.be/7GVXqNLLH7Q



Applications (3/5)

- Gesture recognition
  - https://youtu.be/EvaaAxwY0nc?t=323
- Magic wand
  - https://youtu.be/gw4NCvRSzGo?t=96





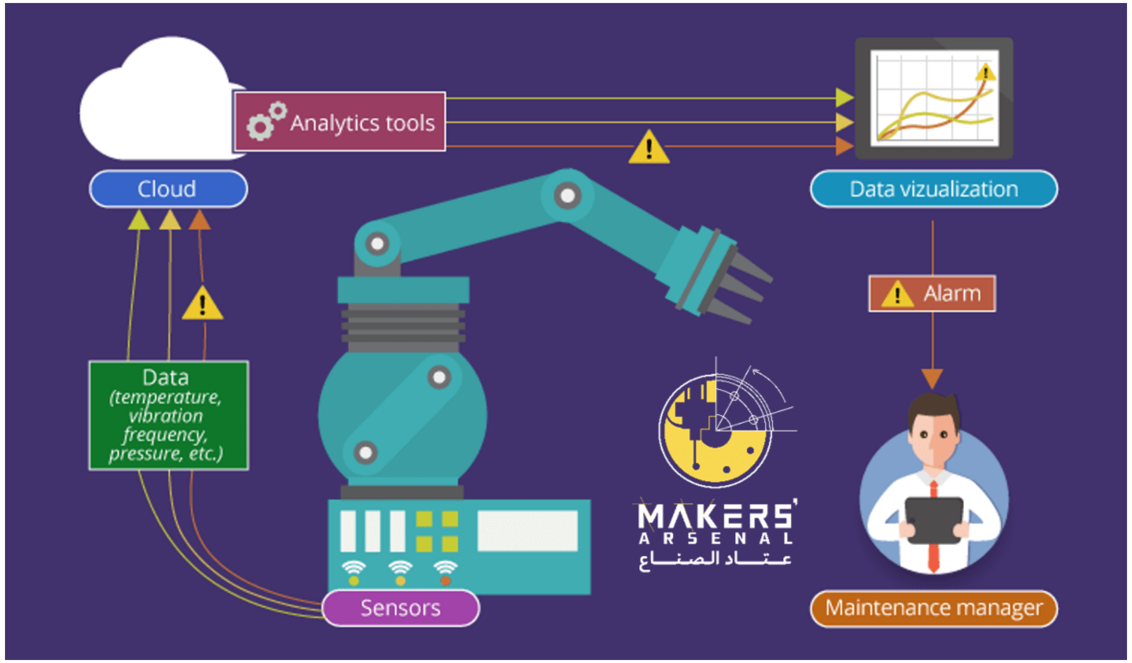
Applications (4/5)

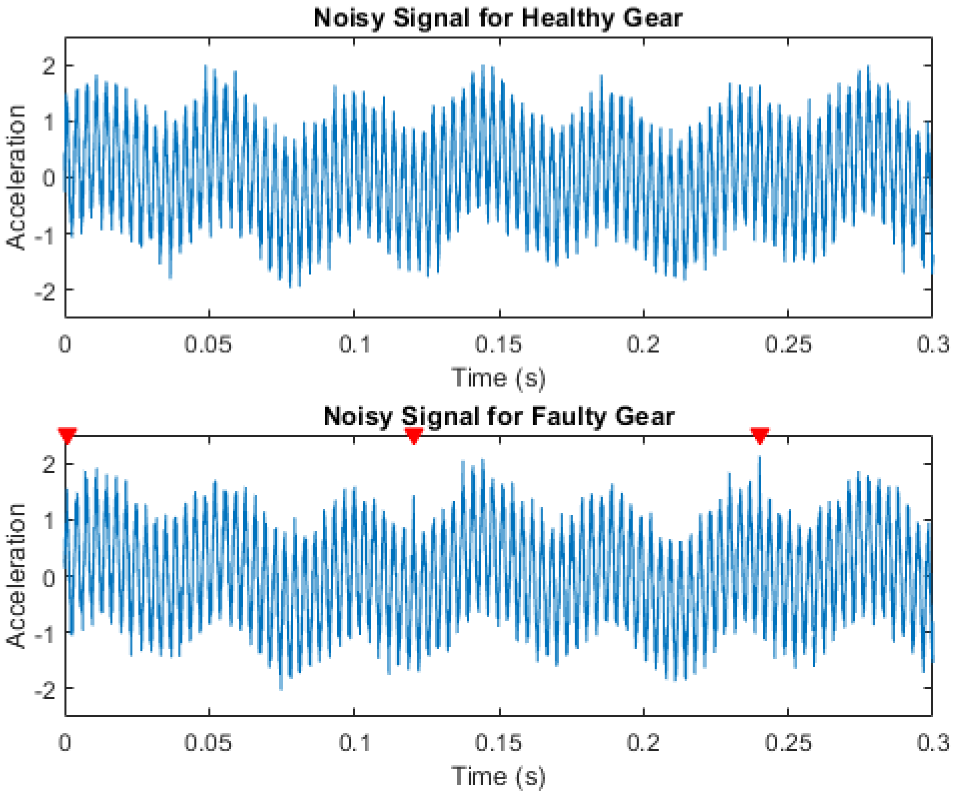
- Motion capture system
  - https://youtu.be/KqKa2Gc7lh8?t=36

- Nintendo Joy-Con
  - https://en.wikipedia.org/wiki/Joy-Con
  - Each Joy-Con contains an accelerometer and gyroscope, which can be used for motion tracking.

Application (5/5)

- Industry 4.0

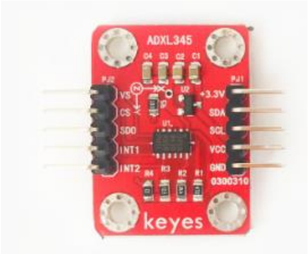




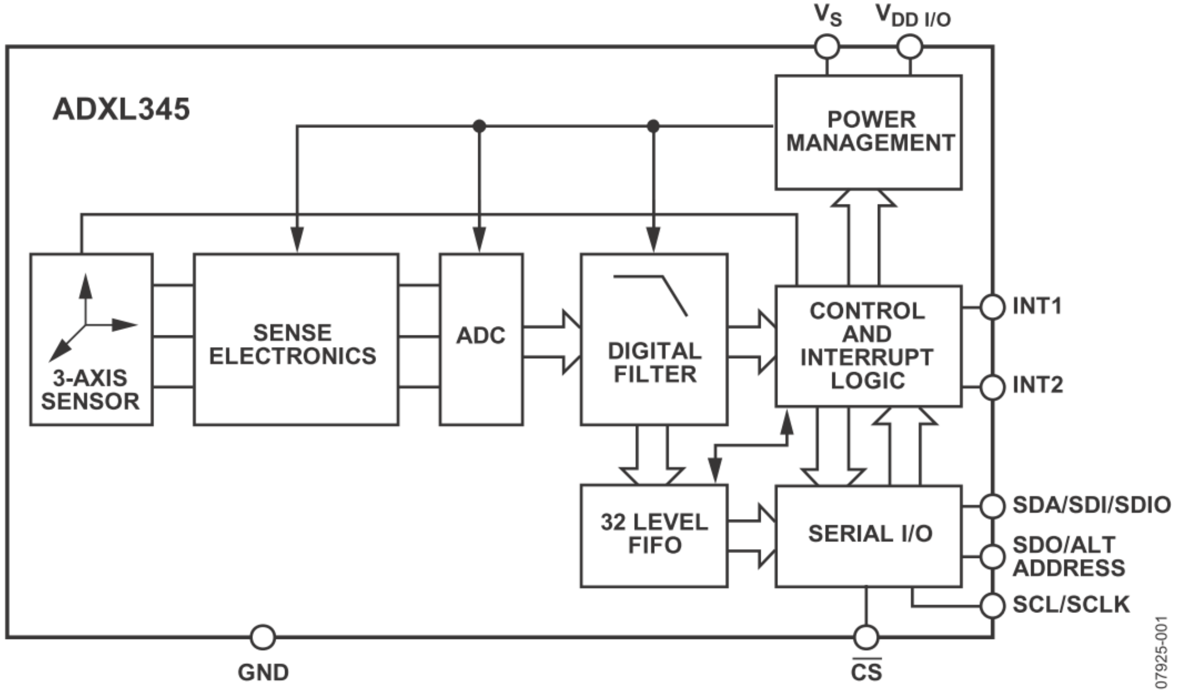
Ref: https://www.makersarsenal.com/case-study/maintenance-vibration-analysis/

ADXL345 (1/2)

- A small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g.
- SPI (3- and 4-wire) and I^2C digital interfaces
- https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf



FUNCTIONAL BLOCK DIAGRAM

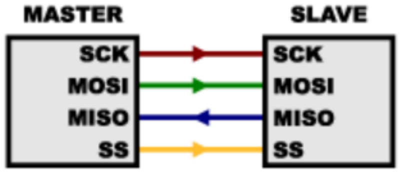


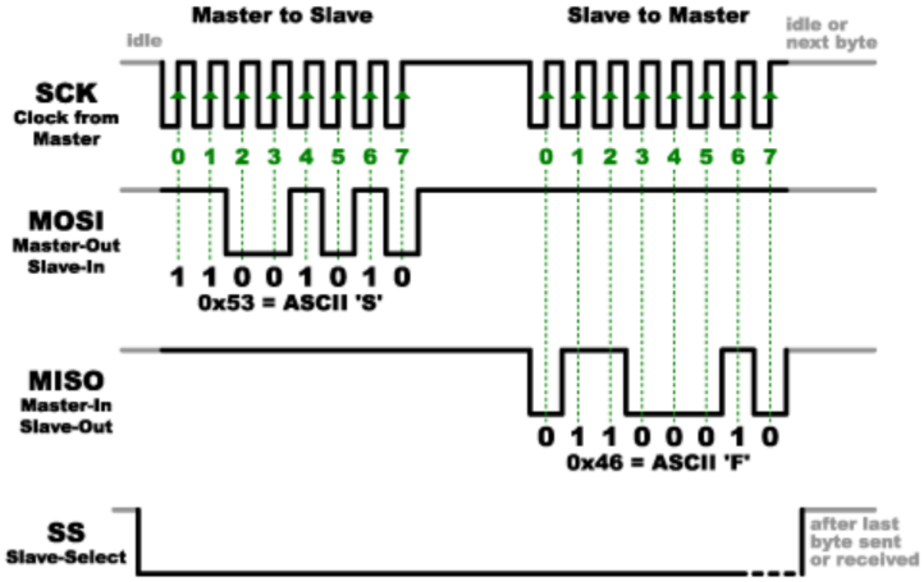
Outline

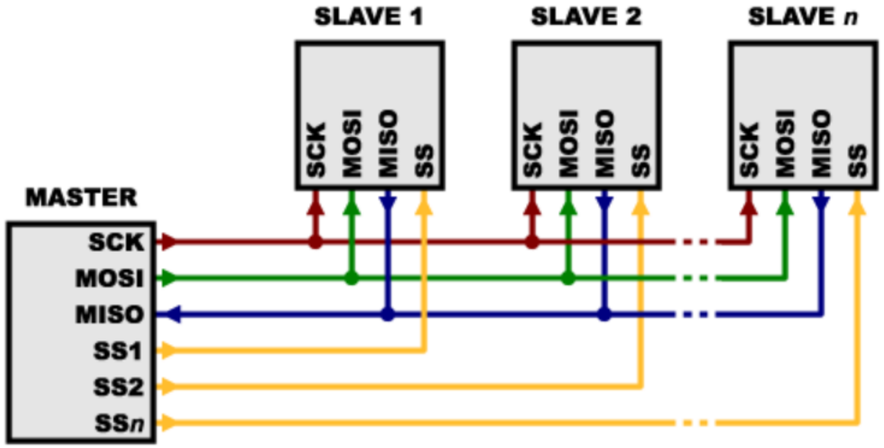
- IMU
- SPI
- I2C
- Lab

SPI

- Serial Peripheral Interface (SPI)
  - four-wire serial bus
  - full duplex
  - higher speed

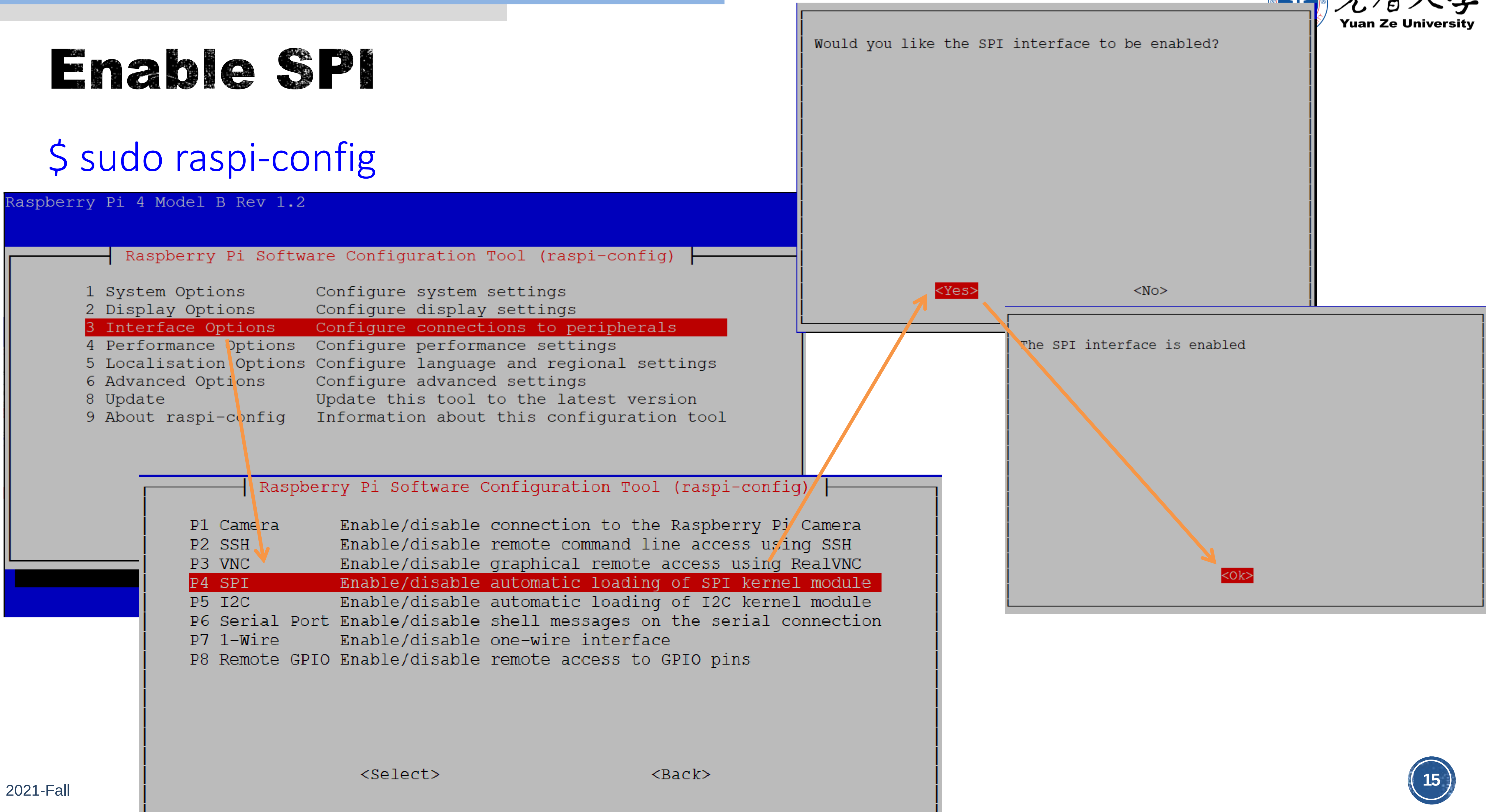






Enable SPI

$ sudo raspi-config



SPI Device

- SPI Device

$ ls -l /dev/spi*

```
pi@rpi4-A00:~/iot/lec05 $ ls -l /dev/spi*
crw-rw---- 1 root spi 153, 0 Oct 18 01:01 /dev/spidev0.0
crw-rw---- 1 root spi 153, 1 Oct 18 01:01 /dev/spidev0.1
```

3v3 Power	1	2	5v Power
GPIO 2 (I2C1 SDA)	3	4	5v Power
GPIO 3 (I2C1 SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (UART TX)
Ground	9	10	GPIO 15 (UART RX)
GPIO 17 (SPI1 CE1)	11	12	GPIO 18 (SPI1 CE0)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3v3 Power	17	18	GPIO 24
GPIO 10 (SPI0 MOSI)	19	20	Ground
GPIO 9 (SPI0 MISO)	21	22	GPIO 25
GPIO 11 (SPI0 SCLK)	23	24	GPIO 8 (SPI0 CE0)
Ground	25	26	GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM SDA)	27	28	GPIO 1 (EEPROM SCL)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (SPI1 MISO)	35	36	GPIO 16 (SPI1 CE2)
GPIO 26	37	38	GPIO 20 (SPI1 MOSI)
Ground	39	40	GPIO 21 (SPI1 SCLK)

SPI

For SPI, either 3- or 4-wire configuration is possible, as shown in the connection diagrams in Figure 34 and Figure 35. Clearing the SPI bit (Bit D6) in the DATA_FORMAT register (Address 0x31) selects 4-wire mode, whereas setting the SPI bit selects 3-wire mode. The maximum SPI clock speed is 5 MHz with 100 pF maximum loading, and the timing scheme follows clock polarity (CPOL) = 1 and clock phase (CPHA) = 1. If power is applied to the ADXL345 before the clock polarity and phase of the host processor are configured, the $\overline{CS}$ pin should be brought high before changing the clock polarity and phase. When using 3-wire SPI, it is recommended that the SDO pin be either pulled up to $V_{DD\ I/O}$ or pulled down to GND via a 10 kΩ resistor.

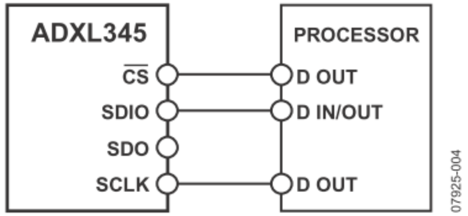


Figure 34. 3-Wire SPI Connection Diagram

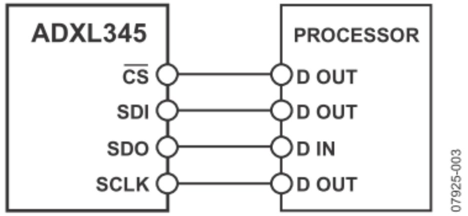


Figure 35. 4-Wire SPI Connection Diagram

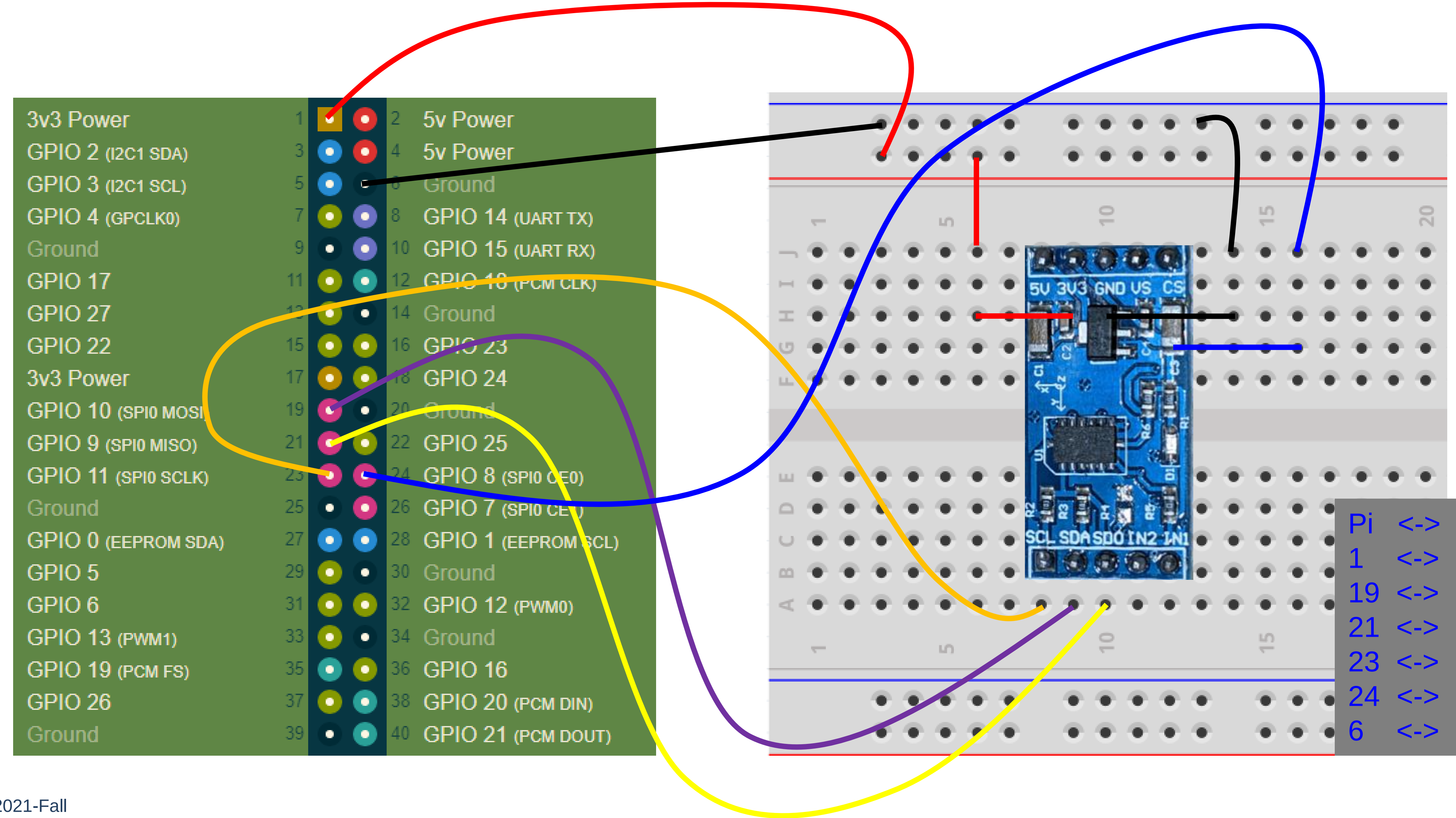
$\overline{CS}$ is the serial port enable line and is controlled by the SPI master. This line must go low at the start of a transmission and high at the end of a transmission, as shown in Figure 37. SCLK is the serial port clock and is supplied by the SPI master. SCLK should idle high during a period of no transmission. SDI and SDO are the serial data input and output, respectively. Data is updated on the falling edge of SCLK and should be sampled on the rising edge of SCLK.

To read or write multiple bytes in a single transmission, the multiple-byte bit, located after the R/$\overline{W}$ bit in the first byte transfer (MB in Figure 37 to Figure 39), must be set. After the register addressing and the first byte of data, each subsequent set of clock pulses (eight clock pulses) causes the ADXL345 to point to the next register for a read or write. This shifting continues until the clock pulses cease and $\overline{CS}$ is deasserted. To perform reads or writes on different, nonsequential registers, $\overline{CS}$ must be deasserted between transmissions and the new register must be addressed separately.

The timing diagram for 3-wire SPI reads or writes is shown in Figure 39. The 4-wire equivalents for SPI writes and reads are shown in Figure 37 and Figure 38, respectively. For correct operation of the part, the logic thresholds and timing parameters in Table 9 and Table 10 must be met at all times.

Use of the 3200 Hz and 1600 Hz output data rates is only recommended with SPI communication rates greater than or equal to 2 MHz. The 800 Hz output data rate is recommended only for communication speeds greater than or equal to 400 kHz, and the remaining data rates scale proportionally. For example, the minimum recommended communication speed for a 200 Hz output data rate is 100 kHz. Operation at an output data rate above the recommended maximum may result in undesirable effects on the acceleration data, including missing samples or additional noise.

Circuit



Pi	<->	ADXL345
1	<->	3V3
19	<->	SDA
21	<->	SDO
23	<->	SCL
24	<->	CS
6	<->	GND

adxl345_spi.py

```python
import time
from enum import Enum
import RPi.GPIO as GPIO
import spidev
import struct

spi = spidev.SpiDev()
spi.open(0, 0)
spi.mode = 0b11
spi.max_speed_hz = 2000000

def writeByte(reg, val):
    spi.xfer2([reg, val])
def writeRegBytes(reg, vals):
    packet = [0] * (len(vals) + 1)
    packet[0] = reg | 0x40
    packet[1:(len(vals)+1)] = vals
    spi.xfer2(packet)
def readByte(reg):
    packet = [0] * 2
    packet[0] = reg | 0x80
    reply = spi.xfer2(packet)
    return reply[1]
deviceID = readByte(0x00)
print("ID: %x" % deviceID)
# Select power control register, 0x2D(45)
# 0x08(08) Auto Sleep disable
writeByte(0x2D, 0x00)
time.sleep(0.1)
writeByte(0x2D, 0x08)

# Select data format register, 0x31(49)
# 0x08(08) Self test disabled, 4-wire interface
# Full resolution, Range = +/-2g
writeByte(0x31, 0x08)
time.sleep(0.5)
```

```python
try:
    while True:
        # Read data back from 0x32(50), 2 bytes
        accel = {'x' : 0, 'y' : 0, 'z': 0}
        # X-Axis LSB, X-Axis MSB
        data0 = readByte(0x32)
        data1 = readByte(0x33)
        # Convert the data to 10-bits
        xAccl = struct.unpack('<h', bytes([data0, data1]))[0]
        accel['x'] = xAccl / 256

        # Read data back from 0x34(52), 2 bytes
        # Y-Axis LSB, Y-Axis MSB
        data0 = readByte(0x34)
        data1 = readByte(0x35)
        # Convert the data to 10-bits
        yAccl = struct.unpack('<h', bytes([data0, data1]))[0]
        accel['y'] = yAccl / 256

        # Read data back from 0x36(54), 2 bytes
        # Z-Axis LSB, Z-Axis MSB
        data0 = readByte(0x36)
        data1 = readByte(0x37)
        # Convert the data to 10-bits
        zAccl = struct.unpack('<h', bytes([data0, data1]))[0]
        accel['z'] = zAccl / 256

        # Output data to screen
        print ("Ax Ay Az: %.3f %.3f %.3f" % (accel['x'], accel['y'], accel['z']))
        time.sleep(0.1)
except KeyboardInterrupt:
    print("Ctrl+C Break")
    spi.close()
```

Results

$ wget https://raw.githubusercontent.com/yachentw/yzucseiot/main/lec05/adxl345_spi.py

```
pi@raspberry:~ $ wget https://raw.githubusercontent.com/yachentw/yzucseiot/main/
lec05/adxl345_spi.py
--2021-11-10 18:24:42--  https://raw.githubusercontent.com/yachentw/yzucseiot/ma
in/lec05/adxl345_spi.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.1
33, 185.199.108.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.
133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2100 (2.1K) [text/plain]
Saving to: 'adxl345_spi.py'

adxl345_spi.py      100%[===================>]   2.05K  --.-KB/s    in 0s

2021-11-10 18:24:42 (26.0 MB/s) - 'adxl345_spi.py' saved [2100/2100]
```

$ python3 adxl345_spi.py

```
pi@rpi4-A00:~/iot/lec05 $ python3 adxl345_spi.py
ID: e5
Ax Ay Az: 0.055 0.059 0.969
Ax Ay Az: 0.047 0.066 0.969
Ax Ay Az: 0.055 0.059 0.973
Ax Ay Az: 0.059 0.059 0.973
Ax Ay Az: 0.055 0.059 0.973
Ax Ay Az: 0.051 0.059 0.973
Ax Ay Az: 0.051 0.062 0.969
Ax Ay Az: 0.055 0.062 0.969
Ax Ay Az: 0.055 0.062 0.973
Ax Ay Az: 0.051 0.059 0.973
Ax Ay Az: 0.059 0.055 0.969
Ax Ay Az: 0.055 0.055 0.969
Ax Ay Az: 0.055 0.059 0.973
Ax Ay Az: 0.059 0.059 0.973
Ax Ay Az: 0.055 0.059 0.977
Ax Ay Az: 0.055 0.062 0.973
```

Operations

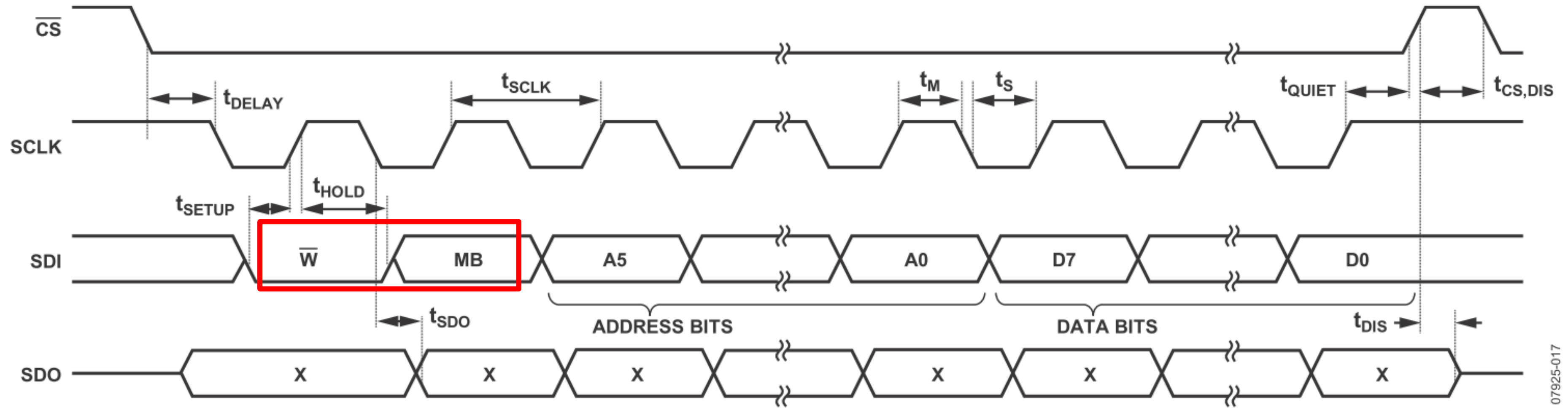
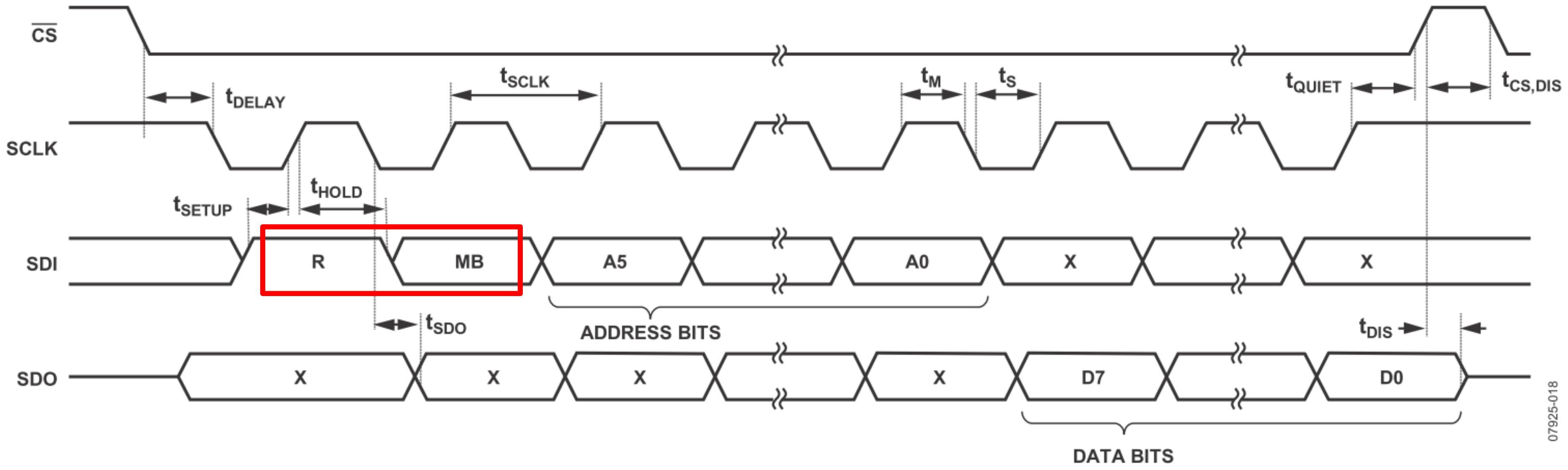


Figure 37. SPI 4-Wire Write



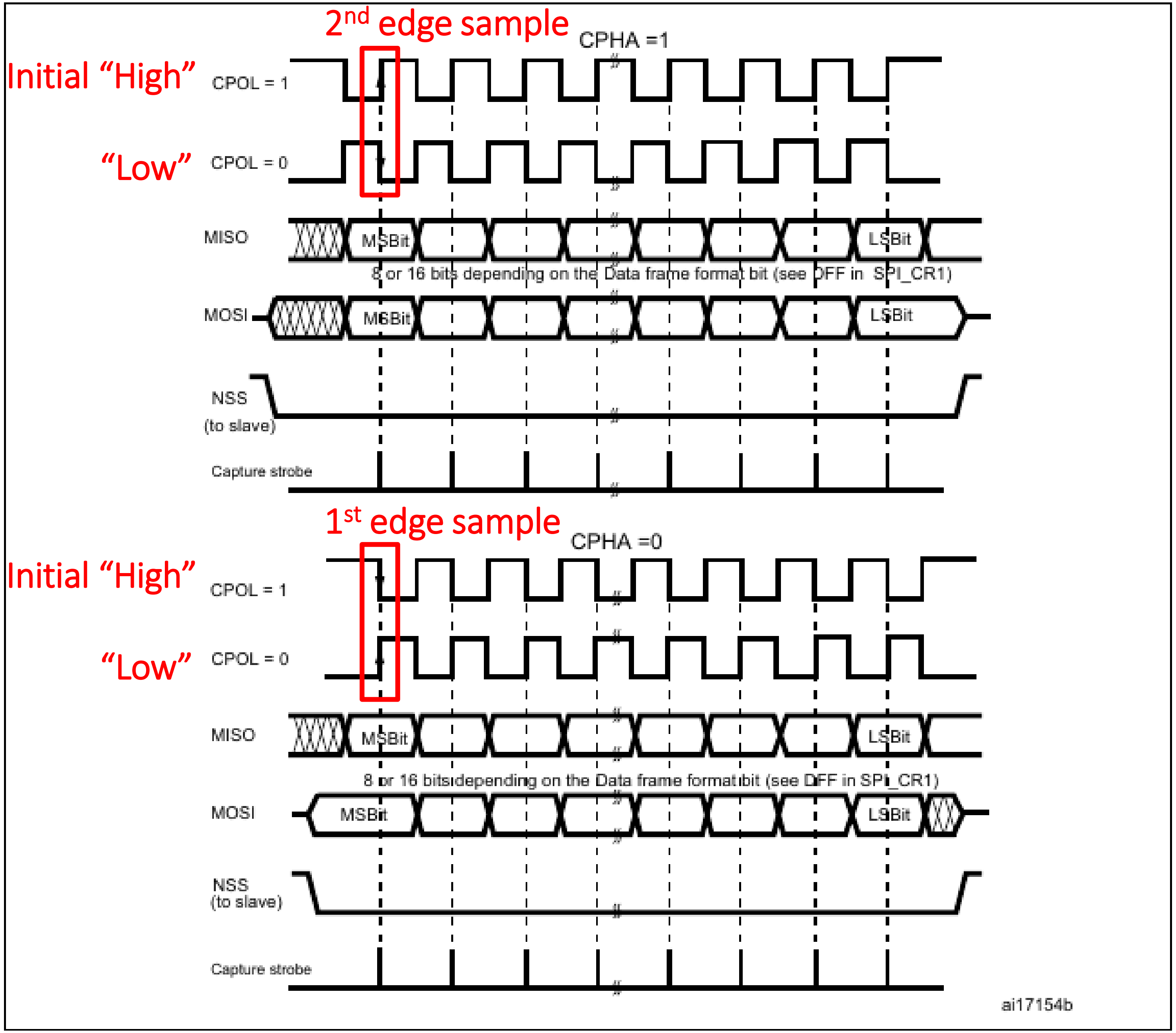
Register Map

Address		Name	Type	Reset Value	Description
Hex	Dec				
0x00	0	DEVID	R	11100101	Device ID
0x01 to 0x1C	1 to 28	Reserved			Reserved; do not access
0x1D	29	THRESH_TAP	R/$\overline{W}$	00000000	Tap threshold
0x1E	30	OFSX	R/$\overline{W}$	00000000	X-axis offset
0x1F	31	OFSY	R/$\overline{W}$	00000000	Y-axis offset
0x20	32	OFSZ	R/$\overline{W}$	00000000	Z-axis offset
0x21	33	DUR	R/$\overline{W}$	00000000	Tap duration
0x22	34	Latent	R/$\overline{W}$	00000000	Tap latency
0x23	35	Window	R/$\overline{W}$	00000000	Tap window
0x24	36	THRESH_ACT	R/$\overline{W}$	00000000	Activity threshold
0x25	37	THRESH_INACT	R/$\overline{W}$	00000000	Inactivity threshold
0x26	38	TIME_INACT	R/$\overline{W}$	00000000	Inactivity time
0x27	39	ACT_INACT_CTL	R/$\overline{W}$	00000000	Axis enable control for activity and inactivity detection
0x28	40	THRESH_FF	R/$\overline{W}$	00000000	Free-fall threshold
0x29	41	TIME_FF	R/$\overline{W}$	00000000	Free-fall time
0x2A	42	TAP_AXES	R/$\overline{W}$	00000000	Axis control for single tap/double tap
0x2B	43	ACT_TAP_STATUS	R	00000000	Source of single tap/double tap
0x2C	44	BW_RATE	R/$\overline{W}$	00001010	Data rate and power mode control
0x2D	45	POWER_CTL	R/$\overline{W}$	00000000	Power-saving features control
0x2E	46	INT_ENABLE	R/$\overline{W}$	00000000	Interrupt enable control
0x2F	47	INT_MAP	R/$\overline{W}$	00000000	Interrupt mapping control
0x30	48	INT_SOURCE	R	00000010	Source of interrupts
0x31	49	DATA_FORMAT	R/$\overline{W}$	00000000	Data format control
0x32	50	DATAX0	R	00000000	X-Axis Data 0
0x33	51	DATAX1	R	00000000	X-Axis Data 1
0x34	52	DATAY0	R	00000000	Y-Axis Data 0
0x35	53	DATAY1	R	00000000	Y-Axis Data 1
0x36	54	DATAZ0	R	00000000	Z-Axis Data 0
0x37	55	DATAZ1	R	00000000	Z-Axis Data 1
0x38	56	FIFO_CTL	R/$\overline{W}$	00000000	FIFO control
0x39	57	FIFO_STATUS	R	00000000	FIFO status

Parameters

- Datasheet
  - https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf
  - The maximum SPI clock speed is 5 MHz
  - Clock polarity (CPOL) = 1 and clock phase (CPHA) = 1

Figure 272. Data clock timing diagram

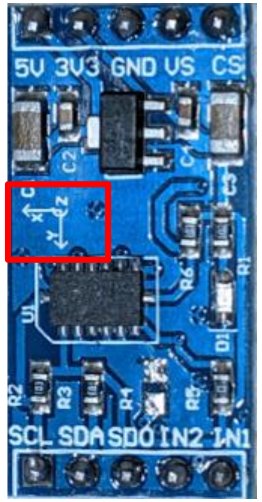


Unit Conversion

Parameter	Test Conditions	Min	Typ[1]	Max	Unit
SENSITIVITY	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	All *g*-ranges, full resolution	230	256	282	LSB/*g*
	±2 *g*, 10-bit resolution	230	256	282	LSB/*g*
	±4 *g*, 10-bit resolution	115	128	141	LSB/*g*
	±8 *g*, 10-bit resolution	57	64	71	LSB/*g*
	±16 *g*, 10-bit resolution	29	32	35	LSB/*g*
Sensitivity Deviation from Ideal	All *g*-ranges		±1.0		%
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	All *g*-ranges, full resolution	3.5	3.9	4.3	m*g*/LSB
	±2 *g*, 10-bit resolution	3.5	3.9	4.3	m*g*/LSB
	±4 *g*, 10-bit resolution	7.1	7.8	8.7	m*g*/LSB
	±8 *g*, 10-bit resolution	14.1	15.6	17.5	m*g*/LSB
	±16 *g*, 10-bit resolution	28.6	31.2	34.5	m*g*/LSB
Sensitivity Change Due to Temperature			±0.01		%/°C

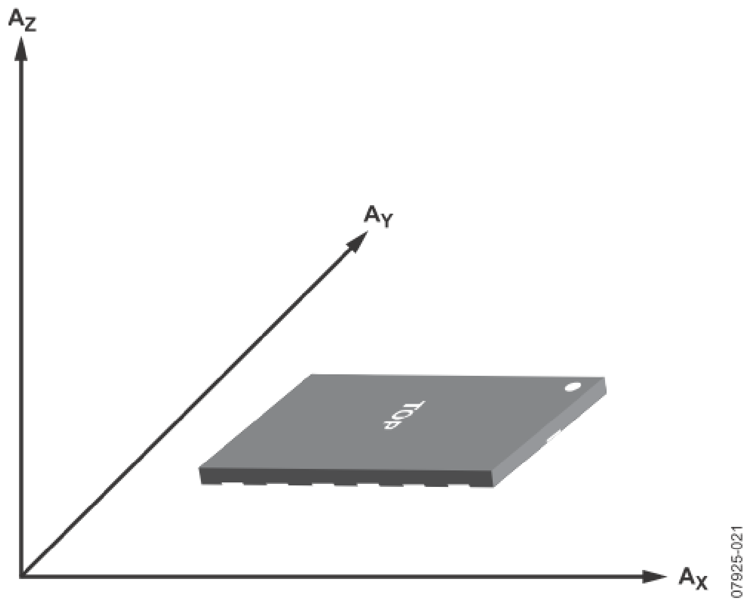
Observation

- Observe the changes of values in accelerations.
- Let X, Y, Z axes point to the sky and stop for a while respectively.



- Repeat the observation by pointing to the ground instead.

Gravity



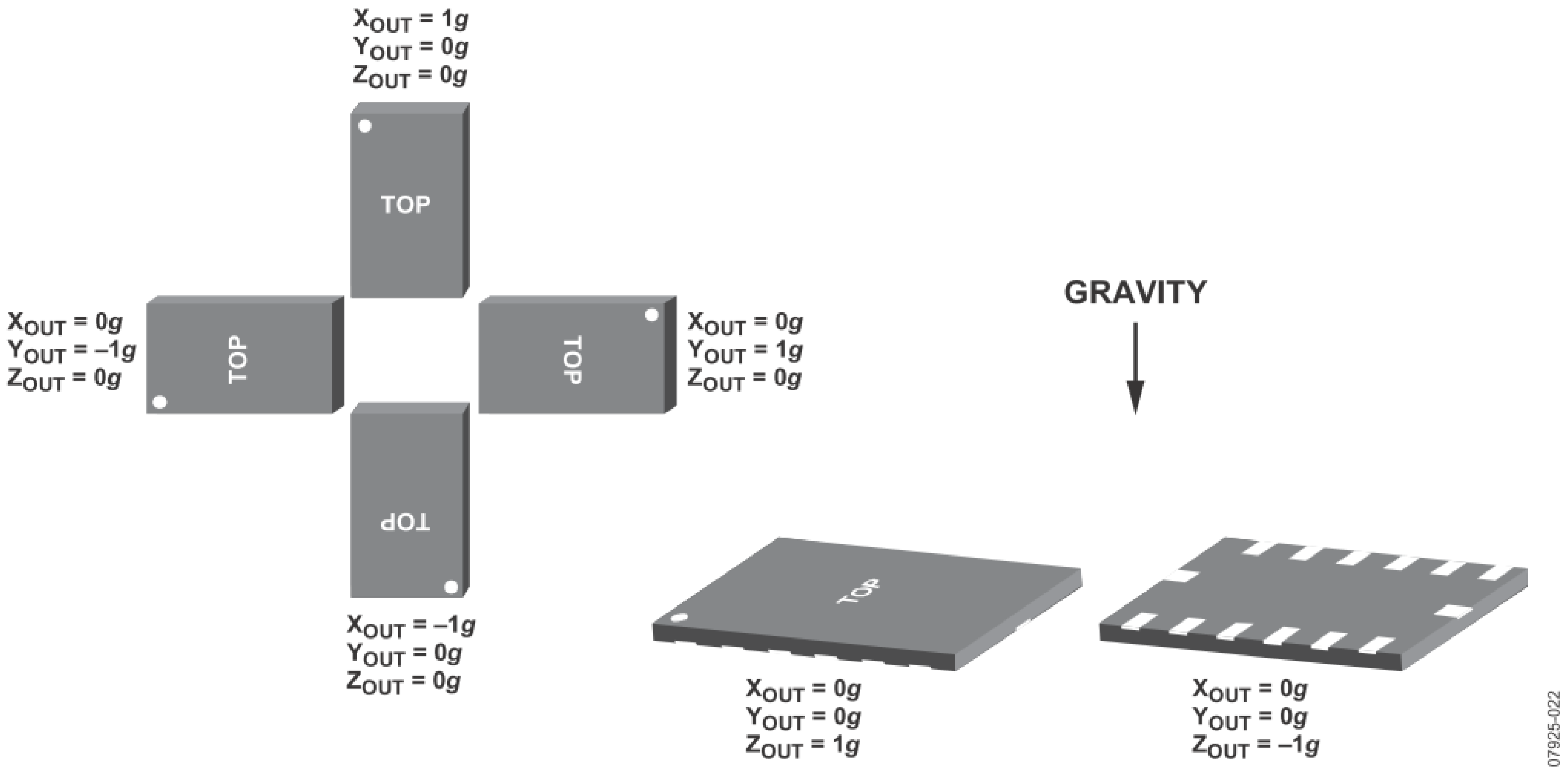


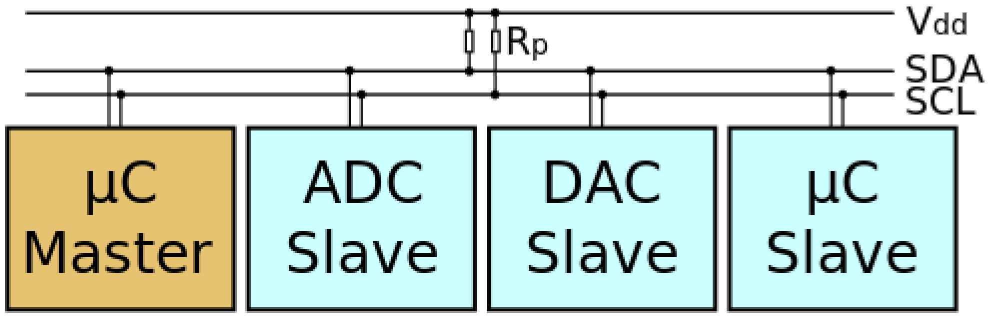
Figure 58. Output Response vs. Orientation to Gravity

Outline

- IMU
- SPI
- I2C
- Lab

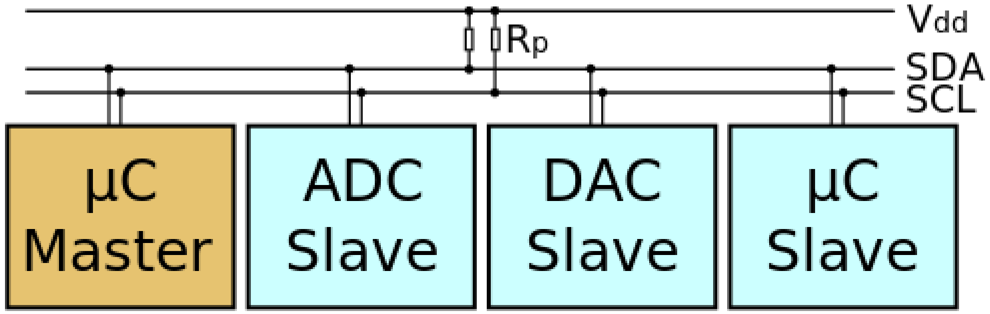
I²C-Bus

- I²C (Inter-Integrated Circuit) Bus
  - Synchronous
  - Multi-master and multi-slave
  - Packet switched
  - Invented in 1982 by Philips Semiconductor (NXP Semiconductors)
  - For lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication



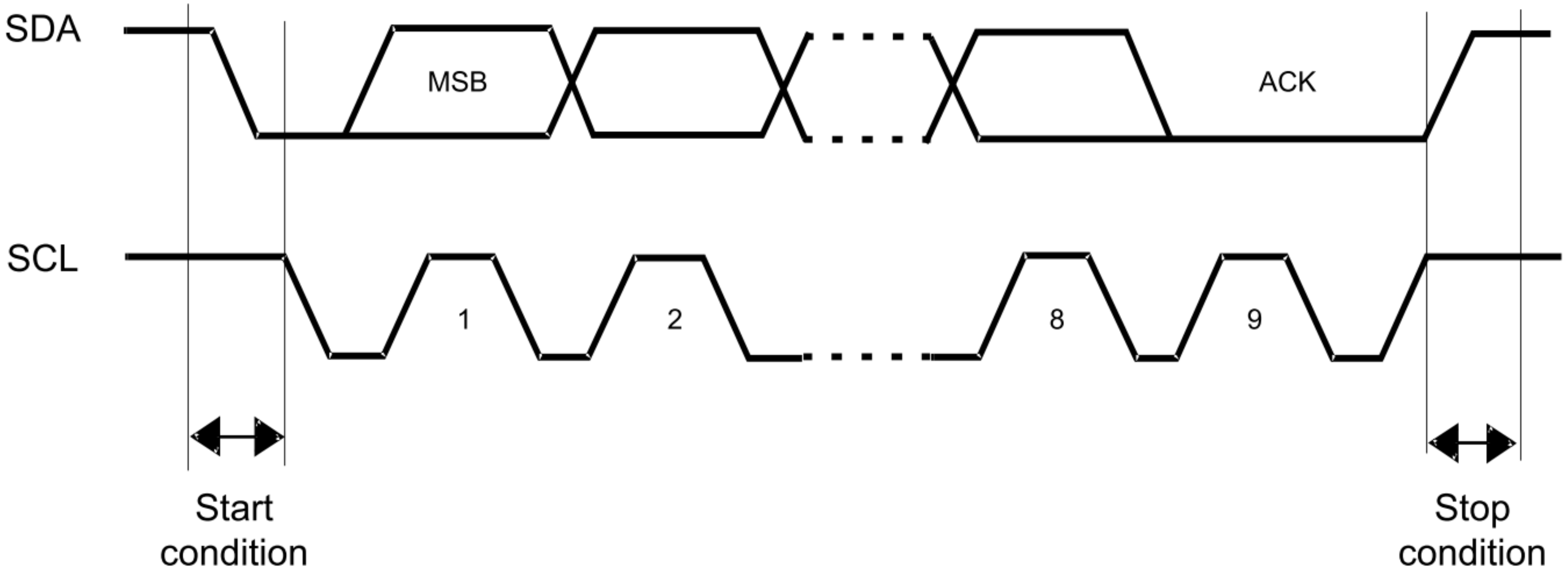
I²C-Bus

- Features
  - Only two bus lines are required; a serial data line (SDA) and a serial clock line (SCL).
  - Each device connected to the bus is software addressable by a unique address and simple master/slave relationships exist at all times.
  - Standard Speed (up to 100 kHz)
  - Fast Speed (up to 400 kHz)
  - The I2C bus frequency can be increased up to 1 MHz.



I²C-Bus

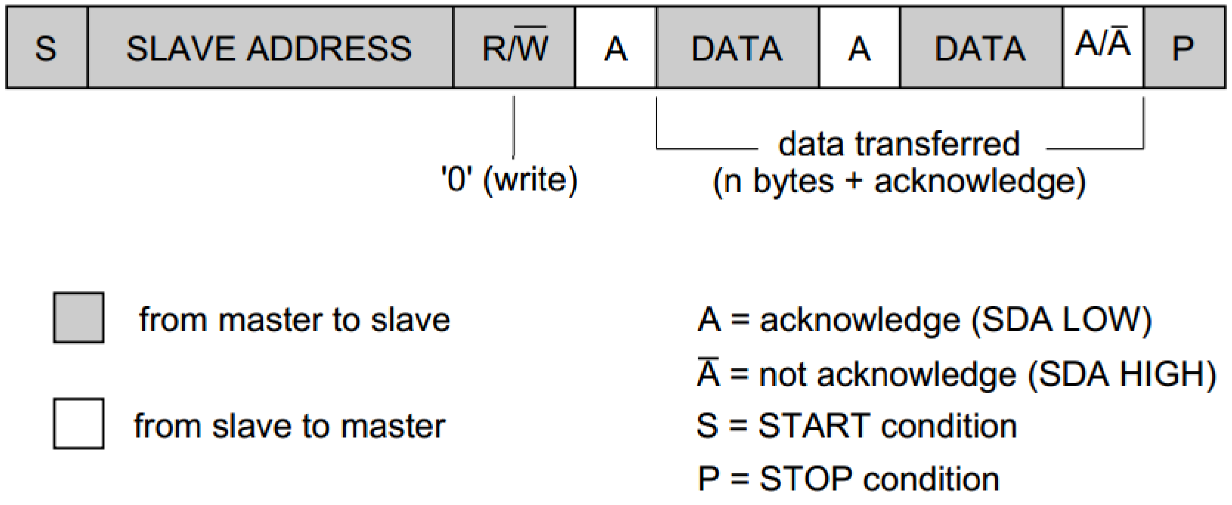
- START, STOP conditions



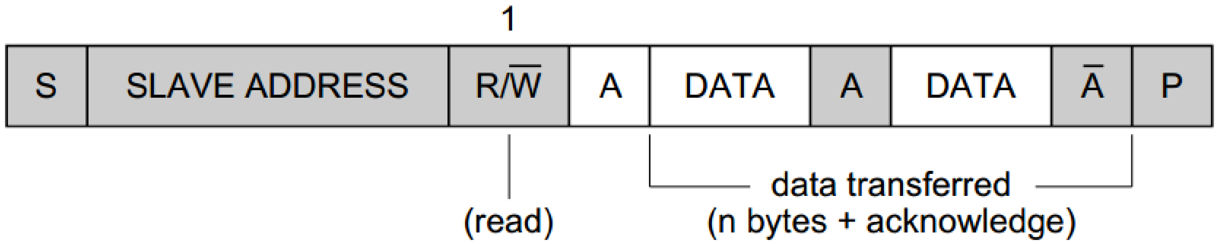
Start by a master.

I²C-Bus

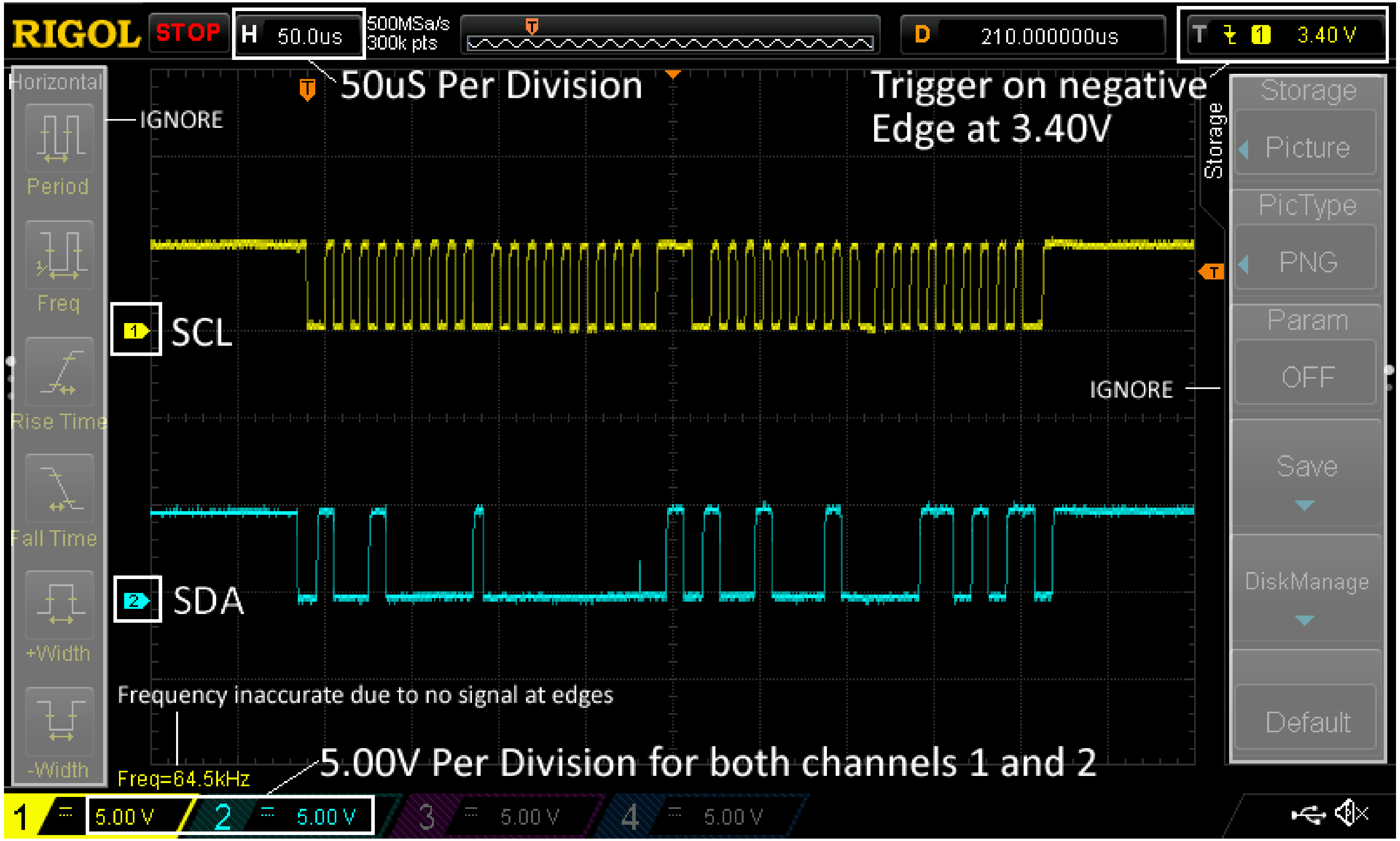
- Master transmits data to a slave



- Master receives data from slave



I²C-Bus



RIGOL
STOP
H 50.0us
500MSa/s
300k pts
D 210.000000us
T 1 3.40 V
50uS Per Division
Trigger on negative Edge at 3.40V
IGNORE
SCL
SDA
IGNORE
Frequency inaccurate due to no signal at edges
Freq=64.5kHz
5.00V Per Division for both channels 1 and 2
Horizontal
Period
Freq
Rise Time
Fall Time
+Width
-Width
Storage
Picture
PicType
PNG
Param
OFF
Save
DiskManage
Default
1 5.00 V
2 5.00 V
3 5.00 V
4 5.00 V

Enable I²C

$ sudo raspi-config

```
Raspberry Pi 4 Model B Rev 1.2

        Raspberry Pi Software Configuration Tool (raspi-config)

 1 System Options          Configure system settings
 2 Display Options         Configure display settings
 3 Interface Options       Configure connections to peripherals
 4 Performance Options     Configure performance settings
 5 Localisation Options    Configure language and regional settings
 6 Advanced Options        Configure advanced settings
 8 Update                  Update this tool to the latest version
 9 About raspi-config      Information about this configuration tool
```

```
        Raspberry Pi Software Configuration Tool (raspi-config)

 P1 Camera      Enable/disable connection to the Raspberry Pi Camera
 P2 SSH         Enable/disable remote command line access using SSH
 P3 VNC         Enable/disable graphical remote access using RealVNC
 P4 SPI         Enable/disable automatic loading of SPI kernel module
 P5 I2C         Enable/disable automatic loading of I2C kernel module
 P6 Serial Port Enable/disable shell messages on the serial connection
 P7 1-Wire      Enable/disable one-wire interface
 P8 Remote GPIO Enable/disable remote access to GPIO pins

          <Select>                    <Back>
```

```
Would you like the ARM I2C interface to be enabled?

                 <Yes>
```

```
The ARM I2C interface is enabled

                 <Ok>
```

I²C Device

$ ls -l /dev/i2c*

```
pi@rpi4-A00:~ $ ls -l /dev/i2c*
crw-rw---- 1 root i2c 89, 1 Oct 17 17:54 /dev/i2c-1
```

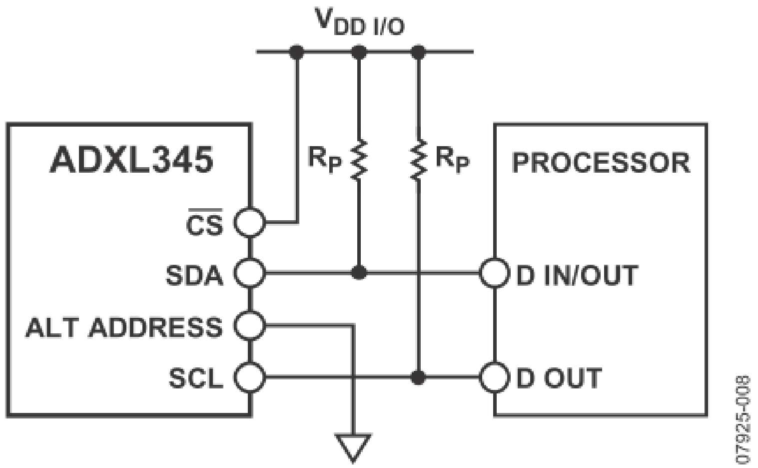
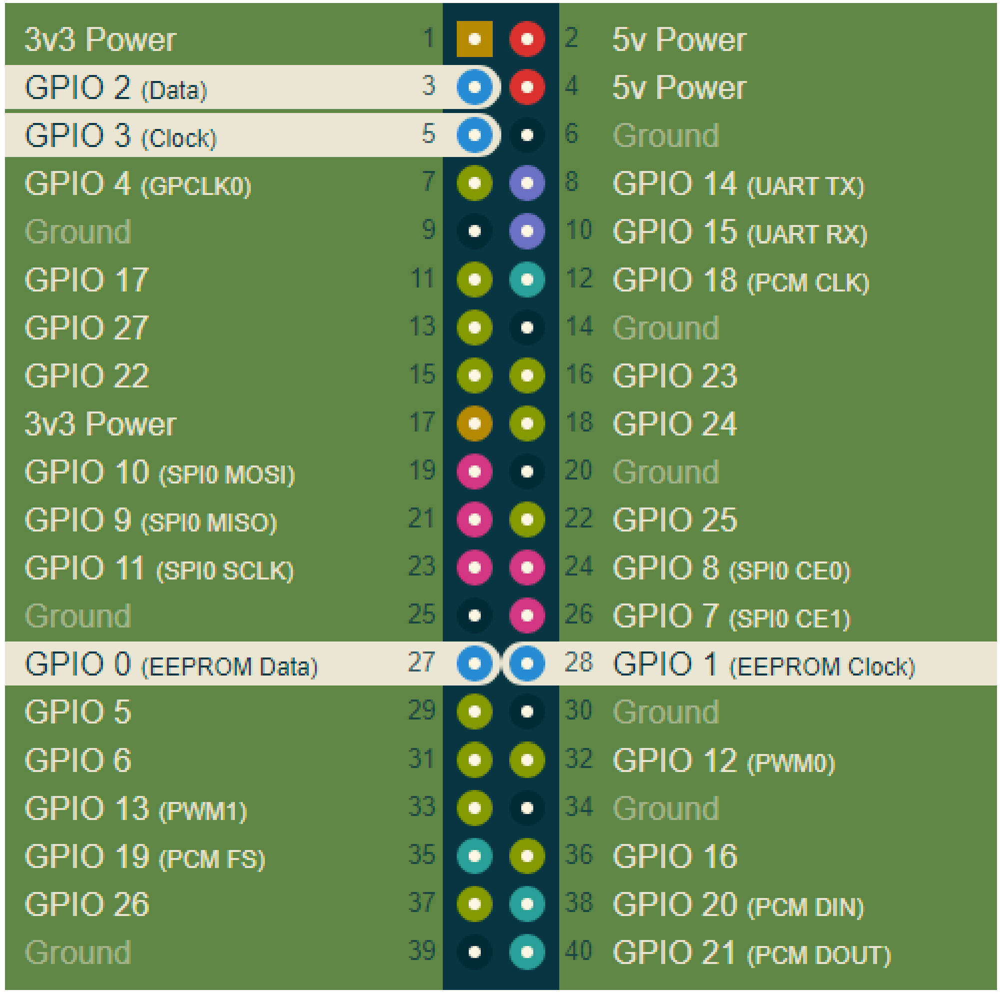
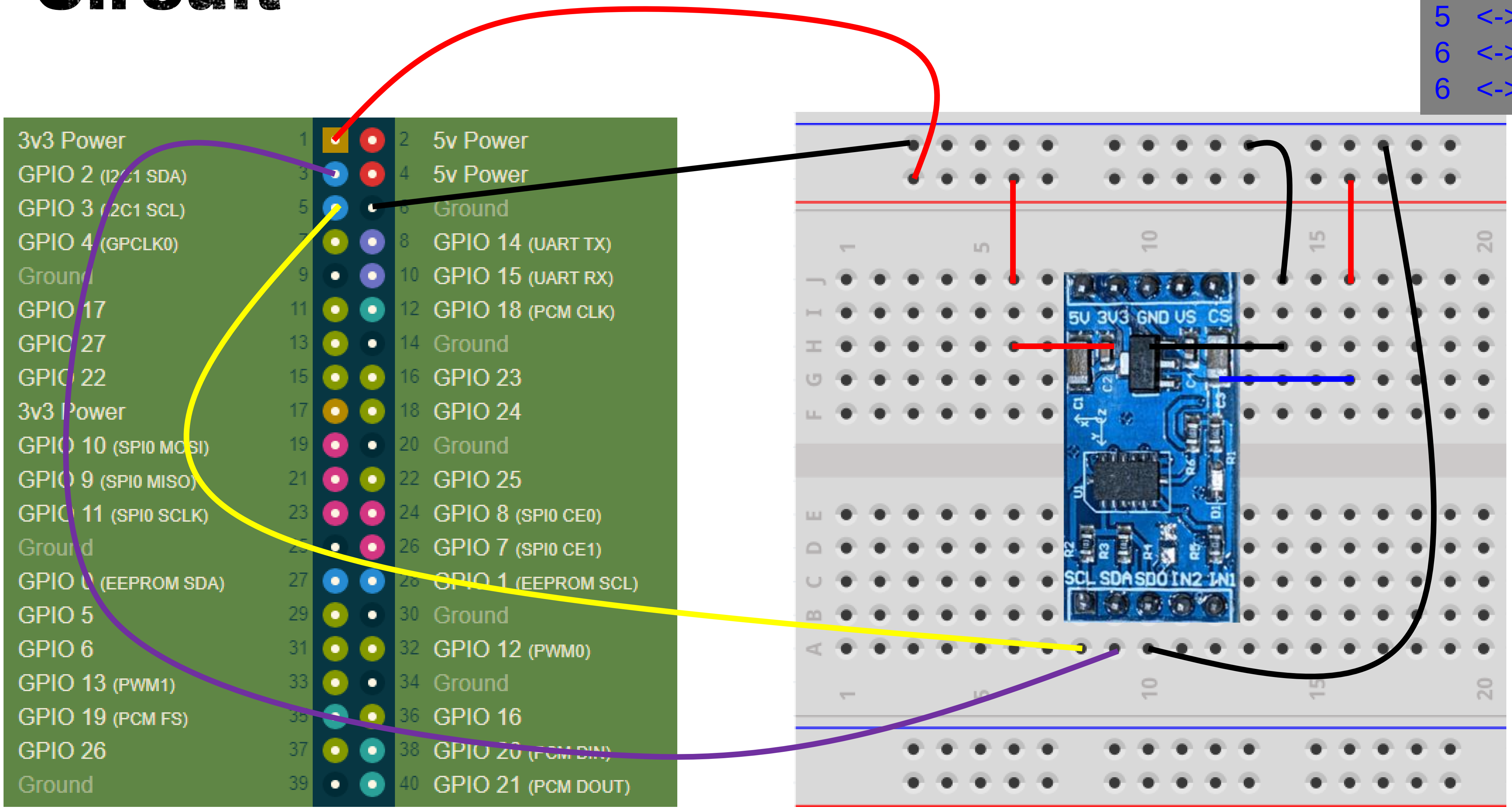


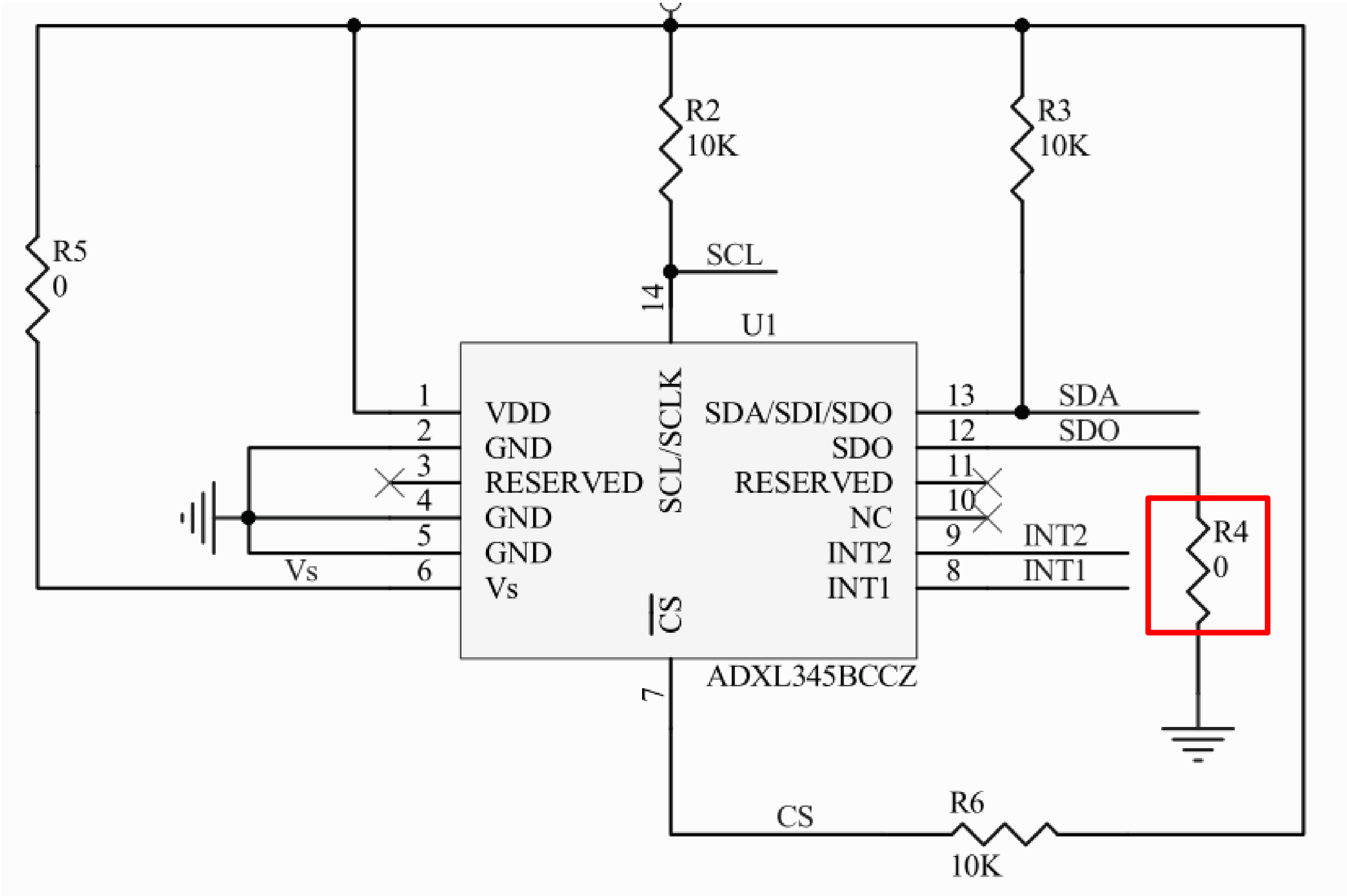
Figure 40. I²C Connection Diagram (Address 0x53)



Circuit

Pi	<->	ADXL345
1	<->	3V3
3	<->	SDA
5	<->	SCL
6	<->	GND
6	<->	SDO




R2
10K
R3
10K
R5
0
SCL
14
U1
1
2
3
4
5
6
VDD
GND
RESERVED
GND
GND
Vs
SCL/SCLK
SDA/SDI/SDO
SDO
RESERVED
NC
INT2
INT1
13
12
11
10
9
8
SDA
SDO
INT2
INT1
R4
0
Vs
CS
ADXL345BCCZ
7
CS
R6
10K

I²C Test

- Detect devices on I2C Bus

$ i2cdetect -y 1

```
pi@rpi4-A00:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- 53 -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

adxl345_i2c.py

```python
import smbus
import time
import struct

# Get I2C bus
bus = smbus.SMBus(1)

deviceID = bus.read_byte_data(0x53, 0x00)
print("ID: %x" % deviceID)

# ADXL345 address, 0x53(83)
# Select power control register, 0x2D(45)
# 0x08(08) Auto Sleep disable
bus.write_byte_data(0x53, 0x2D, 0x00)
bus.write_byte_data(0x53, 0x2D, 0x08)

# ADXL345 address, 0x53(83)
# Select data format register, 0x31(49)
# 0x08(08) Self test disabled, 4-wire interface
# Full resolution, Range = +/-2g
bus.write_byte_data(0x53, 0x31, 0x08)

time.sleep(0.5)
```

```python
try:
    while True:
        # ADXL345 address, 0x53(83)
        # Read data back from 0x32(50), 2 bytes
        accel = {'x' : 0, 'y' : 0, 'z': 0}
        # X-Axis LSB, X-Axis MSB
        data0 = bus.read_byte_data(0x53, 0x32)
        data1 = bus.read_byte_data(0x53, 0x33)
        # Convert the data to 10-bits
        xAccl = struct.unpack('<h', bytes([data0, data1]))[0]
        accel['x'] = xAccl / 256

        # ADXL345 address, 0x53(83)
        # Read data back from 0x34(52), 2 bytes
        # Y-Axis LSB, Y-Axis MSB
        data0 = bus.read_byte_data(0x53, 0x34)
        data1 = bus.read_byte_data(0x53, 0x35)
        # Convert the data to 10-bits
        yAccl = struct.unpack('<h', bytes([data0, data1]))[0]
        accel['y'] = yAccl / 256

        # ADXL345 address, 0x53(83)
        # Read data back from 0x36(54), 2 bytes
        # Z-Axis LSB, Z-Axis MSB
        data0 = bus.read_byte_data(0x53, 0x36)
        data1 = bus.read_byte_data(0x53, 0x37)
        # Convert the data to 10-bits
        zAccl = struct.unpack('<h', bytes([data0, data1]))[0]
        accel['z'] = zAccl / 256

        # Output data to screen
        print ("Ax Ay Az: %.3f %.3f %.3f" % (accel['x'], accel['y'], accel['z']))
        time.sleep(0.1)
except KeyboardInterrupt:
    print("Ctrl+C Break")
```

Results

$ wget https://raw.githubusercontent.com/yachentw/yzucseiot/main/lec05/adxl345_i2c.py

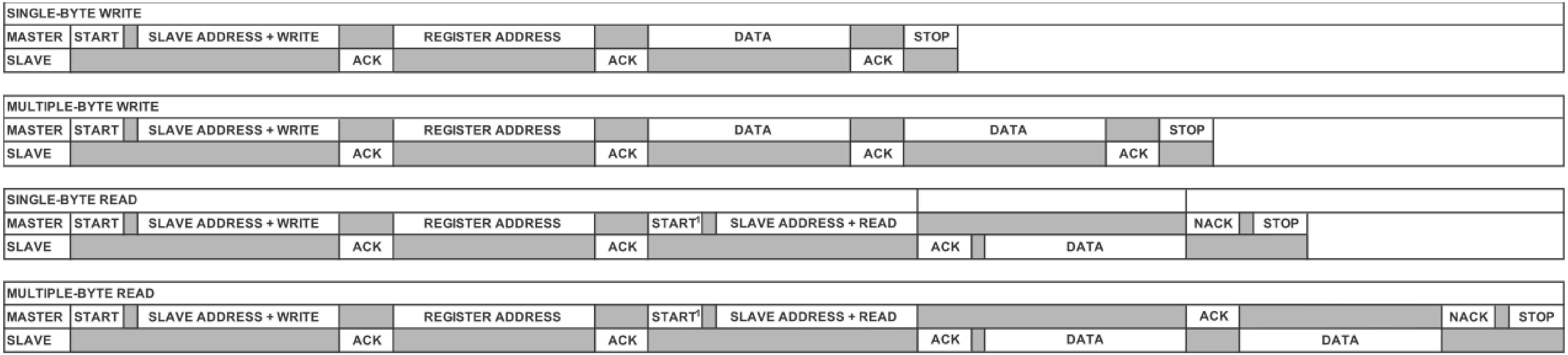
```
pi@raspberry:~ $ wget https://raw.githubusercontent.com/yachentw/yzucseiot/main/
lec05/adxl345_i2c.py
--2021-11-10 18:23:31--  https://raw.githubusercontent.com/yachentw/yzucseiot/ma
in/lec05/adxl345_i2c.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.1
33, 185.199.110.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.
133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2227 (2.2K) [text/plain]
Saving to: 'adxl345_i2c.py'

adxl345_i2c.py      100%[===================>]   2.17K  --.-KB/s    in 0s

2021-11-10 18:23:31 (29.6 MB/s) - 'adxl345_i2c.py' saved [2227/2227]
```

$ python3 adxl345_i2c.py

```
pi@rpi4-A00:~ $ python3 adxl345_i2c.py
ID: e5
Ax Ay Az: 0.043 0.016 1.000
Ax Ay Az: 0.043 0.016 0.996
Ax Ay Az: 0.039 0.012 0.996
Ax Ay Az: 0.039 0.016 0.996
Ax Ay Az: 0.039 0.012 1.000
Ax Ay Az: 0.035 0.012 1.000
Ax Ay Az: 0.043 0.012 0.996
Ax Ay Az: 0.043 0.012 1.004
Ax Ay Az: 0.039 0.012 1.000
Ax Ay Az: 0.035 0.012 0.992
```

Register Map

Address		Name	Type	Reset Value	Description
Hex	Dec				
0x00	0	DEVID	R	11100101	Device ID
0x01 to 0x1C	1 to 28	Reserved			Reserved; do not access
0x1D	29	THRESH_TAP	R/$\overline{W}$	00000000	Tap threshold
0x1E	30	OFSX	R/$\overline{W}$	00000000	X-axis offset
0x1F	31	OFSY	R/$\overline{W}$	00000000	Y-axis offset
0x20	32	OFSZ	R/$\overline{W}$	00000000	Z-axis offset
0x21	33	DUR	R/$\overline{W}$	00000000	Tap duration
0x22	34	Latent	R/$\overline{W}$	00000000	Tap latency
0x23	35	Window	R/$\overline{W}$	00000000	Tap window
0x24	36	THRESH_ACT	R/$\overline{W}$	00000000	Activity threshold
0x25	37	THRESH_INACT	R/$\overline{W}$	00000000	Inactivity threshold
0x26	38	TIME_INACT	R/$\overline{W}$	00000000	Inactivity time
0x27	39	ACT_INACT_CTL	R/$\overline{W}$	00000000	Axis enable control for activity and inactivity detection
0x28	40	THRESH_FF	R/$\overline{W}$	00000000	Free-fall threshold
0x29	41	TIME_FF	R/$\overline{W}$	00000000	Free-fall time
0x2A	42	TAP_AXES	R/$\overline{W}$	00000000	Axis control for single tap/double tap
0x2B	43	ACT_TAP_STATUS	R	00000000	Source of single tap/double tap
0x2C	44	BW_RATE	R/$\overline{W}$	00001010	Data rate and power mode control
0x2D	45	POWER_CTL	R/$\overline{W}$	00000000	Power-saving features control
0x2E	46	INT_ENABLE	R/$\overline{W}$	00000000	Interrupt enable control
0x2F	47	INT_MAP	R/$\overline{W}$	00000000	Interrupt mapping control
0x30	48	INT_SOURCE	R	00000010	Source of interrupts
0x31	49	DATA_FORMAT	R/$\overline{W}$	00000000	Data format control
0x32	50	DATAX0	R	00000000	X-Axis Data 0
0x33	51	DATAX1	R	00000000	X-Axis Data 1
0x34	52	DATAY0	R	00000000	Y-Axis Data 0
0x35	53	DATAY1	R	00000000	Y-Axis Data 1
0x36	54	DATAZ0	R	00000000	Z-Axis Data 0
0x37	55	DATAZ1	R	00000000	Z-Axis Data 1
0x38	56	FIFO_CTL	R/$\overline{W}$	00000000	FIFO control
0x39	57	FIFO_STATUS	R	00000000	FIFO status



NOTES
1. THIS START IS EITHER A RESTART OR A STOP FOLLOWED BY A START.
2. THE SHADED AREAS REPRESENT WHEN THE DEVICE IS LISTENING.

07925-033

Figure 41. I²C Device Addressing

Using ADXL345 Library

$ sudo pip3 install adafruit-circuitpython-adxl34x

$ wget https://github.com/yachentw/yzucseiot/blob/main/lec05/ada_i2c.py

```
import time
import board
import busio
import adafruit_adxl34x
i2c = busio.I2C(board.SCL, board.SDA)
accelerometer = adafruit_adxl34x.ADXL345(i2c)
while True:
    print("%f %f %f" % accelerometer.acceleration)
    time.sleep(1)
```

$ python3 ada_i2c.py

```
pi@rpi4-A00:~/iot/lec05 $ python3 ada_i2c.py
0.431493 0.117680 10.081236
0.470719 0.117680 10.081236
0.431493 0.117680 10.042010
0.392266 0.117680 10.081236
0.000000 -3.961887 9.257478
0.000000 -8.119906 5.295591
0.000000 -8.237586 5.021005
0.470719 0.156906 10.081236
0.470719 0.156906 10.081236
```

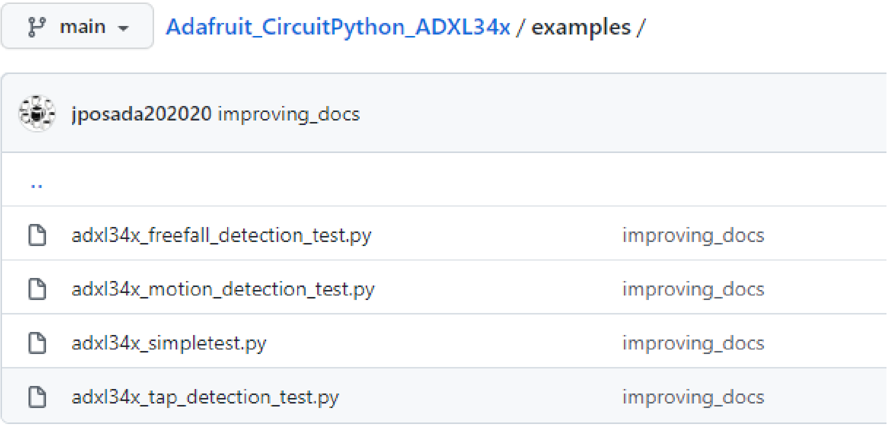
Open Source Driver

- https://github.com/adafruit/Adafruit_CircuitPython_ADXL34x/blob/master/adafruit_adxl34x.py

```
# SPDX-FileCopyrightText: 2018 Bryan Siepert for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
`adafruit_adxl34x`
====================================================

A driver for the ADXL34x 3-axis accelerometer family

* Author(s): Bryan Siepert

Based on drivers by K. Townsend and Tony DiCola


Implementation Notes
--------------------

**Hardware:**

* Adafruit `ADXL345 Digital Accelerometer
  <https://www.adafruit.com/product/1231>`_ (Product ID: 1231)

**Software and Dependencies:**

* Adafruit CircuitPython firmware for the supported boards:
  https://circuitpython.org/downloads

* Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice
"""
```

Examples

- https://github.com/adafruit/Adafruit_CircuitPython_ADXL34x/tree/main/examples



- You can run the provided examples.

$ wget https://raw.githubusercontent.com/adafruit/Adafruit_CircuitPython_ADXL34x/main/examples/adxl34x_freefall_detection_test.py

$ wget https://raw.githubusercontent.com/adafruit/Adafruit_CircuitPython_ADXL34x/main/examples/adxl34x_motion_detection_test.py

$ wget https://raw.githubusercontent.com/adafruit/Adafruit_CircuitPython_ADXL34x/main/examples/adxl34x_simpletest.py

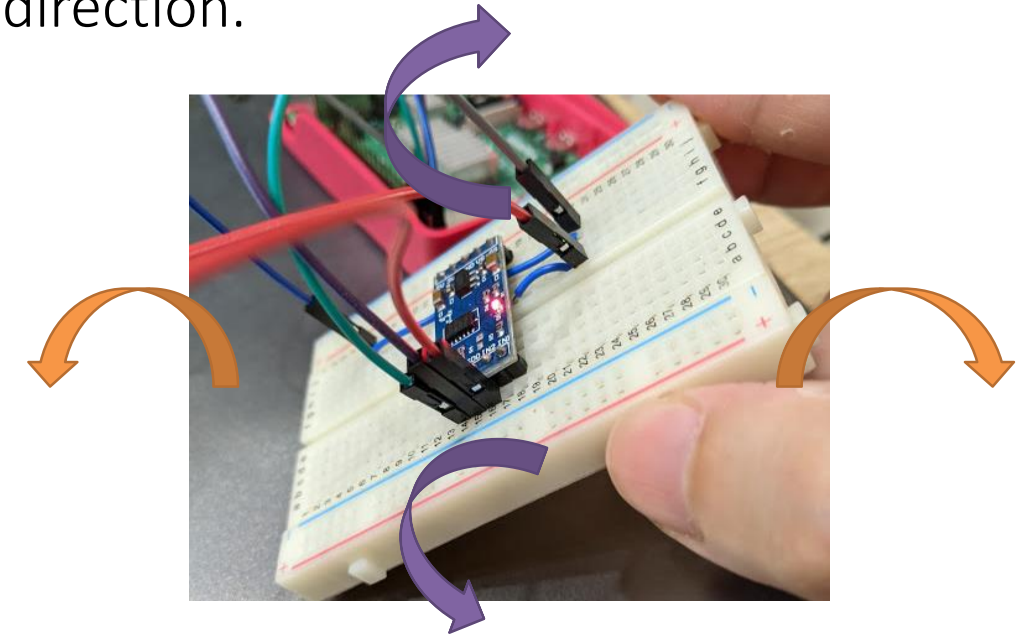
$ wget https://raw.githubusercontent.com/adafruit/Adafruit_CircuitPython_ADXL34x/main/examples/adxl34x_tap_detection_test.py

Outline

- IMU
- SPI
- I2C
- Lab

CS348A

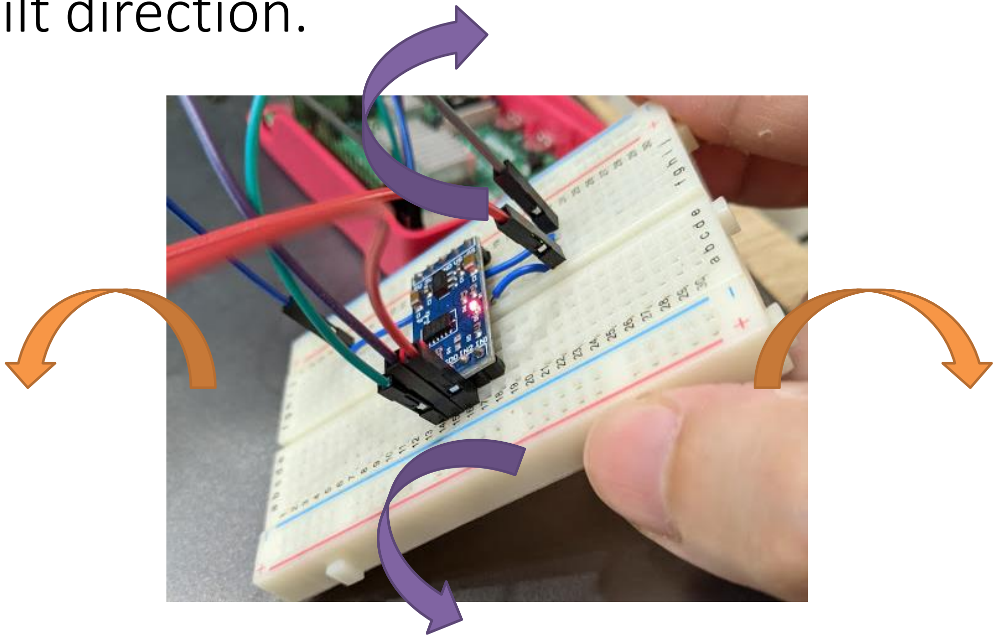
- Change the sound pitch from the buzzer according to the tilt angle of the accelerometer where 0 degree is pitch C and 90 degree is pitch B.
  - e.g., 45 degree is pitch F.
- You can choose your tilt direction.



- You can use I2C or SPI to communicate with ADXL345 by any library.

CS348B

- Change the brightness of the LED according to the tilt angle of the accelerometer where 0 degree is 0% brightness and 90 degree is 100% brightness.
  - e.g., 45 degree is about 50% brightness.
- You can choose your tilt direction.



- You can use I2C or SPI to communicate with ADXL345 by any library.