Report: SPI Communication with ADXL345

Group# 1: Jason Dai(td2593), Yusheng Hu(yh3097), Brian Wu (zw2542)

1. **Hardware Configuration:**
   We first configured the accelerometer with a 4-wire SPI connection to establish a serial communication with Huzzard board. The pins SDA, SDO, SCL on the accelerometer are connected to MOSI, MISO, and SCK on the ESP-8266 board respectively. See Fig.1 for a connection diagram.

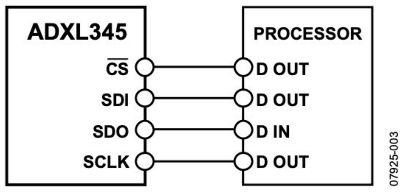

Figure.1 -Wire SPI connection diagram from the spec

2. **Software Implementation:**
   We first initialized the SPI connection by specifying the baud rate (SCK clock rate), polarity (the level the idle clock line sits at) and phase (to sample data on the first or second clock edge respectively.). We set the SCK clock rate to be 2 MHz.

```
//spi initialization:
spi = machine.SPI(1, baudrate=2000000, polarity=1, phase=1)
cs = machine.Pin(15, machine.Pin.OUT)
```

Note: 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. We used pin 15 for cs. Everytime we want to read or write data from the accelerometer, we have to enable cs (set value=0) and then disable it (set value =1). See Fig. 2 for a clock diagram.

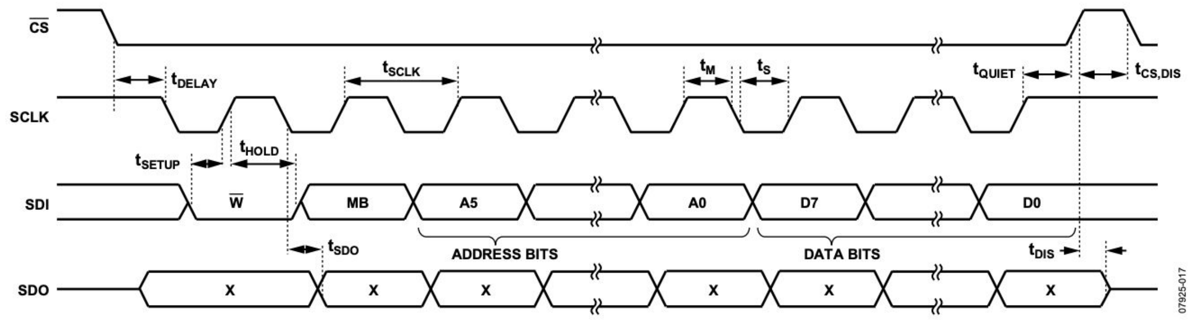

Figure.2 -When CS’ is low, SDI and SDO are able to write and read from the accelerometer.

For reading data from the accelerometer, we have to first initialize SPI's power control and data format control. We issued two write commands to SPI according to ADXL345's data sheet:

```
#0x2D power_control
power_ctl = b'\x2d'
cs.value(0)
spi.write(power_ctl)
cs.value(1)

#0x31 --> data format control
data_format = b'\x31'
cs.value(0)
spi.write(data_format)
cs.value(1)
```

We found that the registers ranging from 0x32 to 0x37 represent the x,y,z axis data. And since the feather is only a 2-dimensional display, we would only need the x, y axis data. We picked 0x33 and 0x35 for this purpose. For the read() function in the SPI protocol, the first argument represents the number of bytes ("nbytes") for reading input. And for the second argument, the first bit of the 8-bit input represents "read/write" ability (0:disable read/write; 1:enable read/write). The second bit represents "multiple bytes" specified by the "nbytes" argument previously. So we need to convert the first 2 bits of register address "0x33" and "0x35" to be both "1"s. In binary, "0x33" is "00110011" and "0x35" is "00110101" Therefore, we used "0xf3" ("00110011") and "0xf5" ("00110101") to be the input register address for SPI.read() function in order to get the corresponding x and y value.