```
// Connor Noddin
// ECE 406
// Computer Engineering Capstone
// main_3050.ino

#include <SPI.h>
#include <avr/pgmspace.h>
#include "ps2mouse.h"
#include "ADNS3050.h"

// Allows device to send packets
int DEVICE_ENABLED = 0;  //Flag for if host sent device enabled signal

/*
Description: Initial setup, runs on boot only once.
Establishes GPIO pins and sensor. Also sends initial packet sequence
*/
void setup() {

  //Initiate Serial communication for debugging
  Serial.begin(SERIAL_RATE);  //9600 bits/second (Baud rate)

  delay(INIT_DELAY);  // 500 ms delay for PS/2 standard

  // Initialize the mouse buttons as inputs:
  pinMode(LEFT, INPUT);
  pinMode(MIDDLE, INPUT);
  pinMode(RIGHT, INPUT);

  // In assignments for reading data and clock bus
  pinMode(DATA_IN, INPUT);
  pinMode(CLK_IN, INPUT);

  // Out assignments for controlling data and clock bus
  pinMode(DATA_OUT, OUTPUT);
  pinMode(CLK_OUT, OUTPUT);

  startup();  // Begin ADNS 3050 sensor

  // Write self test passed
  while (ps2_dwrite(BAT) != 0);
  // Write mouse ID
  while (ps2_dwrite(ID) != 0);
}

/*
Description: Runs indefinitely. Establishes handshake with PS2_host.
Then, reads sensor and buttons. Finally sends 3 data packets to PS2_host.
*/
void loop() {
  byte* data;  //3 data packets
  byte tmp;    //Temporary byte from functions

  // Check if host is trying to send commands
  if (((digitalRead(DATA_IN) == LOW) || (digitalRead(CLK_IN) == LOW)) && DEVICE_ENABLED ==
0) {
    while (ps2_dread(&tmp));  // If this fails it halts the program
    ps2_command(tmp);
    if (tmp == ENABLE) DEVICE_ENABLED = 1;
  }

  /*
  Serial.print("\n");
  Serial.print("Sensor X: ");
  Serial.print(sensor_x, DEC);
  Serial.print("\t Sensor Y: ");
  Serial.print(sensor_y, DEC);
  Serial.print("\n");
```

```
  delay(5);
  */

  // Writes data to host
  if (DEVICE_ENABLED == 1 || FORCE_ENABLE == 1) {

    data = get_bytes();  // Gets all data from sensors

    //Writes to DATA and CLK lines
    ps2_dwrite(data[0]);
    ps2_dwrite(data[1]);
    ps2_dwrite(data[2]);

    delay(DATA_DELAY);  // Delay between bytes. Increases stability

    /*
    Serial.print("Byte 1: 0x");
    Serial.print(byte_1, HEX);
    Serial.print("\t Sensor X: ");
    Serial.print(sensor_x, DEC);
    Serial.print("\t Sensor Y: ");
    Serial.print(sensor_y, DEC);
    Serial.print("\n");
    */
  }
}
```