#include "Wire.h" // This library allows you to communicate with I2C devices.

const int MPU\_ADDR = 0x68; // I2C address of the MPU-6050. If AD0 pin is set to HIGH, the I2C address will be 0x69.

int16\_t accelerometer\_x, accelerometer\_y, accelerometer\_z; // variables for accelerometer raw data

char tmp\_str[7]; // temporary variable used in convert function

char\* convert\_int16\_to\_str(int16\_t i) { // converts int16 to string. Moreover, resulting strings will have the same length in the debug monitor.

sprintf(tmp\_str, "%6d", i);

return tmp\_str;

}

void setup() {

Serial.begin(9600);

Wire.begin();

Wire.beginTransmission(MPU\_ADDR); // Begins a transmission to the I2C slave (GY-521 board)

Wire.write(0x6B); // PWR\_MGMT\_1 register

Wire.write(0); // set to zero (wakes up the MPU-6050)

Wire.endTransmission(true);

}

void loop() {

Wire.beginTransmission(MPU\_ADDR);

Wire.write(0x3B); // starting with register 0x3B (ACCEL\_XOUT\_H) [MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2, p.40]

Wire.endTransmission(false); // the parameter indicates that the Arduino will send a restart. As a result, the connection is kept active.

Wire.requestFrom(MPU\_ADDR, 7\*2, true); // request a total of 7\*2=14 registers

// "Wire.read()<<8 | Wire.read();" means two registers are read and stored in the same variable

accelerometer\_x = Wire.read()<<8 | Wire.read(); // reading registers: 0x3B (ACCEL\_XOUT\_H) and 0x3C (ACCEL\_XOUT\_L)

accelerometer\_y = Wire.read()<<8 | Wire.read(); // reading registers: 0x3D (ACCEL\_YOUT\_H) and 0x3E (ACCEL\_YOUT\_L)

accelerometer\_z = Wire.read()<<8 | Wire.read(); // reading registers: 0x3F (ACCEL\_ZOUT\_H) and 0x40 (ACCEL\_ZOUT\_L)

// print out data

Serial.print("aX = "); Serial.print(convert\_int16\_to\_str(accelerometer\_x));

Serial.print(" | aY = "); Serial.print(convert\_int16\_to\_str(accelerometer\_y));

Serial.print(" | aZ = "); Serial.print(convert\_int16\_to\_str(accelerometer\_z));

Serial.println();

// delay

delay(1000);

}