ADXL345 3-Axis Accelerometer

Technical Manual Rev 3r0

Features and Specifications

e-Gizmo break-out board for ADXL345 3-Axis Accelerometer requires low power and has a high resolution of 13 bits. It has a programmable range of resolution specifically ±2 g, ±4 g, ±6 g, ±8 g, and ±16 g and requires +3.3V supply voltage.

Pin Assignments:

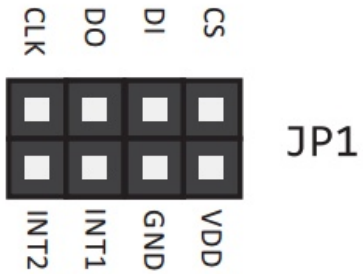


Figure 1. *JP1 Pin I.D. Illustration*

Table 1. *JP1 Pin I.D. & Descriptions*

Pin I.D.	Descriptions
VDD	+3.3V Supply Voltage
GND	Ground
INT1	Interrupt 1 Output
INT2	Interrupt 2 Output
CS	Chip Select
DI	Data Input
DO	Data Output
CLK	Clock

Chip: ADXL345
Power Input: 1.8 V to 3.6 V
Size: 3 × 5 × 1 mm LGA package

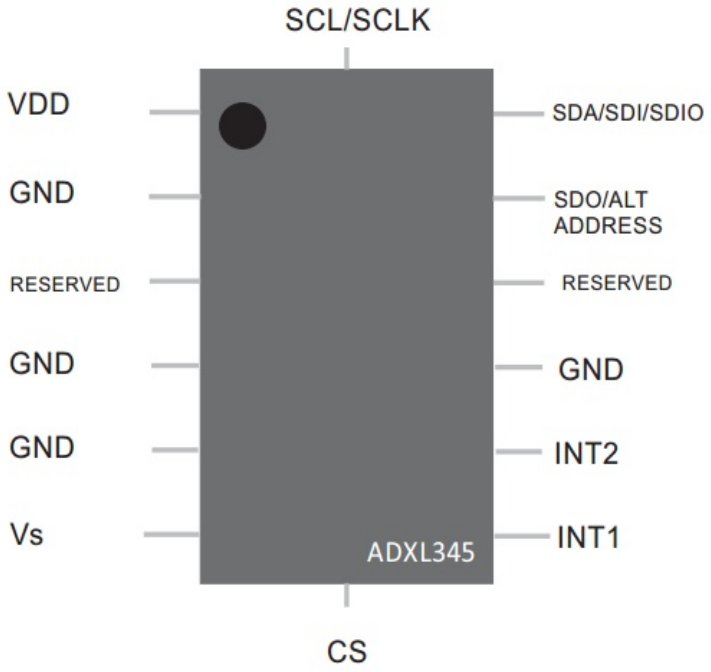


Figure 2. *ADXL345 Pin Illustration*

Table 2. *ADXL345 Pin Function and Descriptions*

Pin No.	Label	Description
1	VDD	Digital Supply Voltage
2	GND	Ground
3	Reserved	Reserved
4	GND	Ground
5	GND	Ground
6	Vs	Supply Voltage
7	CS	Chip Select
8	INT1	Interrupt 1 Output
9	INT2	Interrupt 2 Output
10	GND	Ground
11	Reserved	Reserved
12	SDO/ALT/ Address	Serial Data Out, Alternate I2C Address select
13	SDA/SDI/ SDIO	Serial Data (I2C), Serial Data In (SPI 4-Wire), Serial Data In/Out (SPI 3-Wire)
14	SCL/SCLK	Serial Communications Clock

FEATURES:

- Built in motion detection
- Selectable bandwidth
- Flexible interrupt modes
- Tap/double tap detection
- Free-fall detection

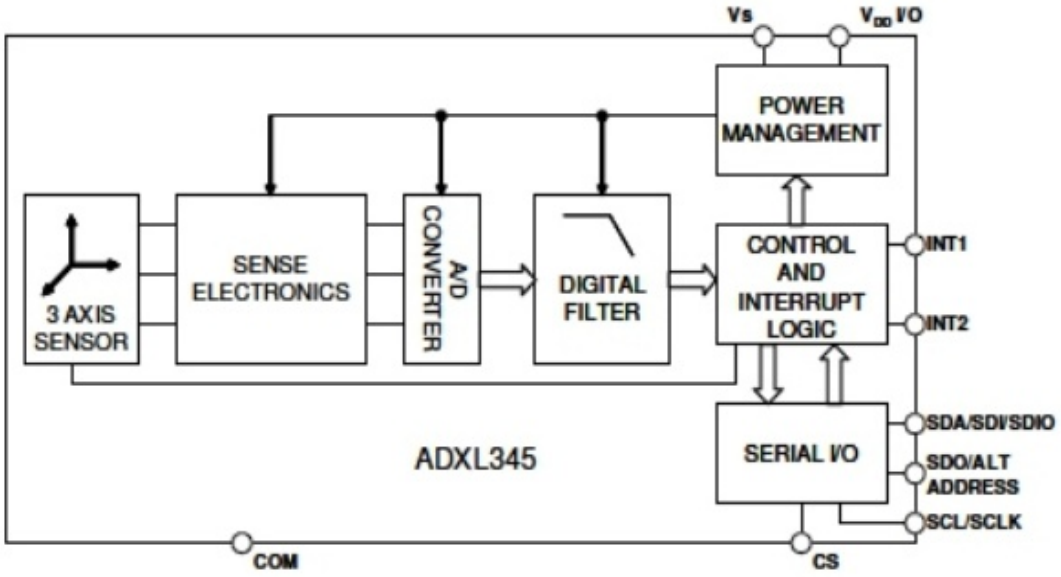


Figure 3. *ADXL345 Functional Block Diagram*

PCB BOARD PRESENTATION

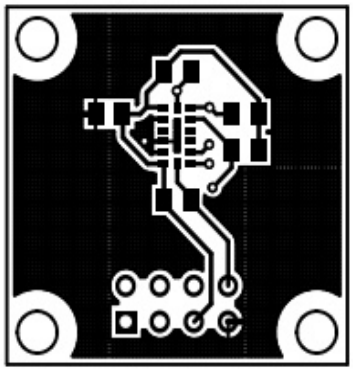


Figure 5. *ADXL345 3-Axis Accelerometer PCB Copper Pattern (Top Layer)*

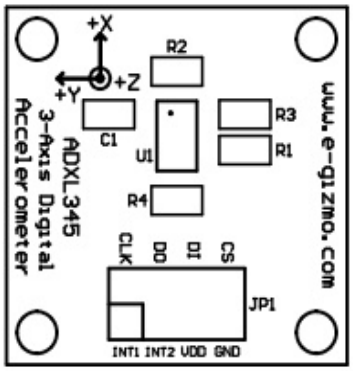


Figure 4. *ADXL345 3-Axis Accelerometer (silkscreen layout)*

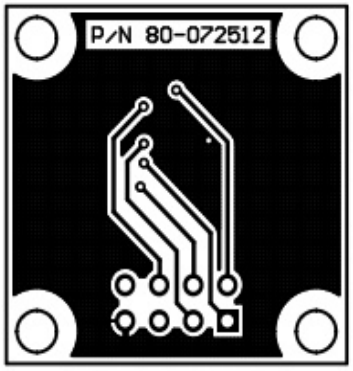
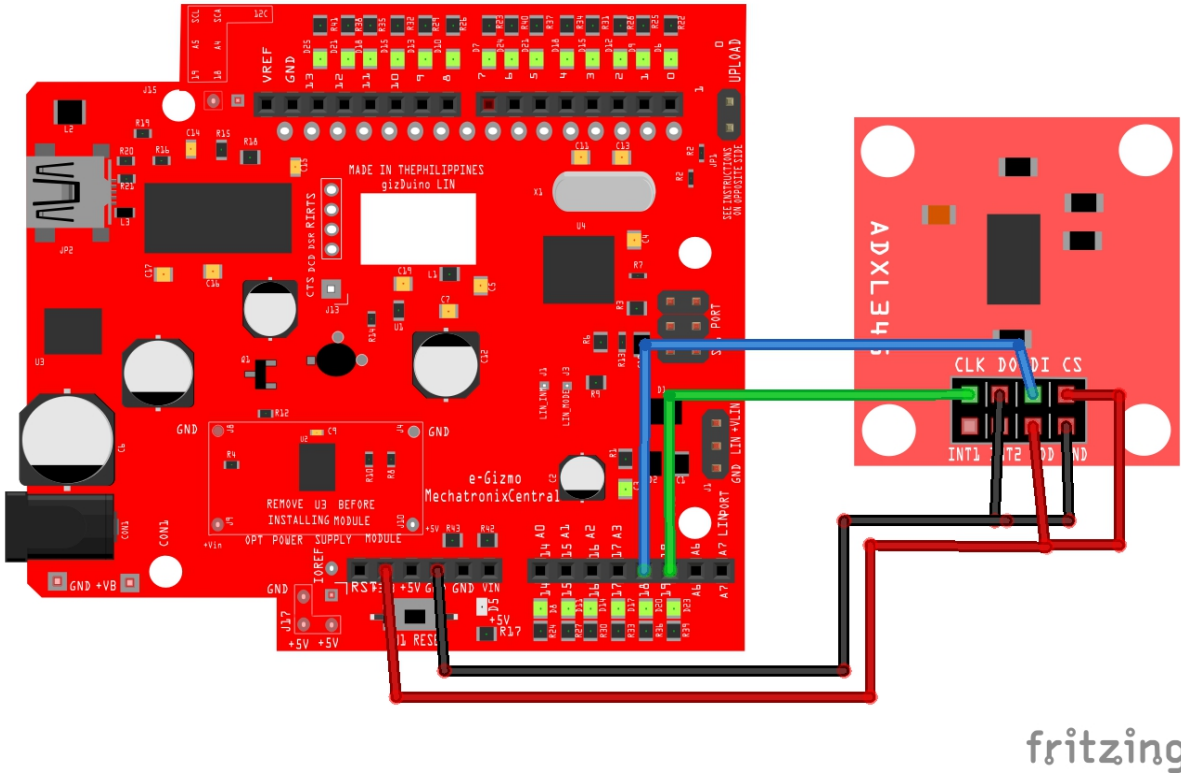


Figure 6. *ADXL345 3-Axis Accelerometer PCB Copper Pattern (Bottom Layer)*

I2C Wiring Connections:

Gizduino to ADXL345 acceleromter

3.3V	**VDD**
3.3V	**CS**
GND	**GND**
GND	**DO**
A4/SDA	**DI**
A5/SCL	**CLK**



```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>

/* Assign a unique ID to this sensor at the same time */
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);

void displaySensorDetails(void)
{
  sensor_t sensor;
  accel.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println(" m/s^2");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println(" m/s^2");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" m/s^2");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}

void displayDataRate(void)
{
  Serial.print  ("Data Rate:    ");

  switch(accel.getDataRate())
  {
    case ADXL345_DATARATE_3200_HZ:
      Serial.print  ("3200 ");
      break;
    case ADXL345_DATARATE_1600_HZ:
      Serial.print  ("1600 ");
      break;
    case ADXL345_DATARATE_800_HZ:
      Serial.print  ("800 ");
      break;
    case ADXL345_DATARATE_400_HZ:
      Serial.print  ("400 ");
      break;
    case ADXL345_DATARATE_200_HZ:
      Serial.print  ("200 ");
      break;
    case ADXL345_DATARATE_100_HZ:
      Serial.print  ("100 ");
      break;
    case ADXL345_DATARATE_50_HZ:
      Serial.print  ("50 ");
      break;
```

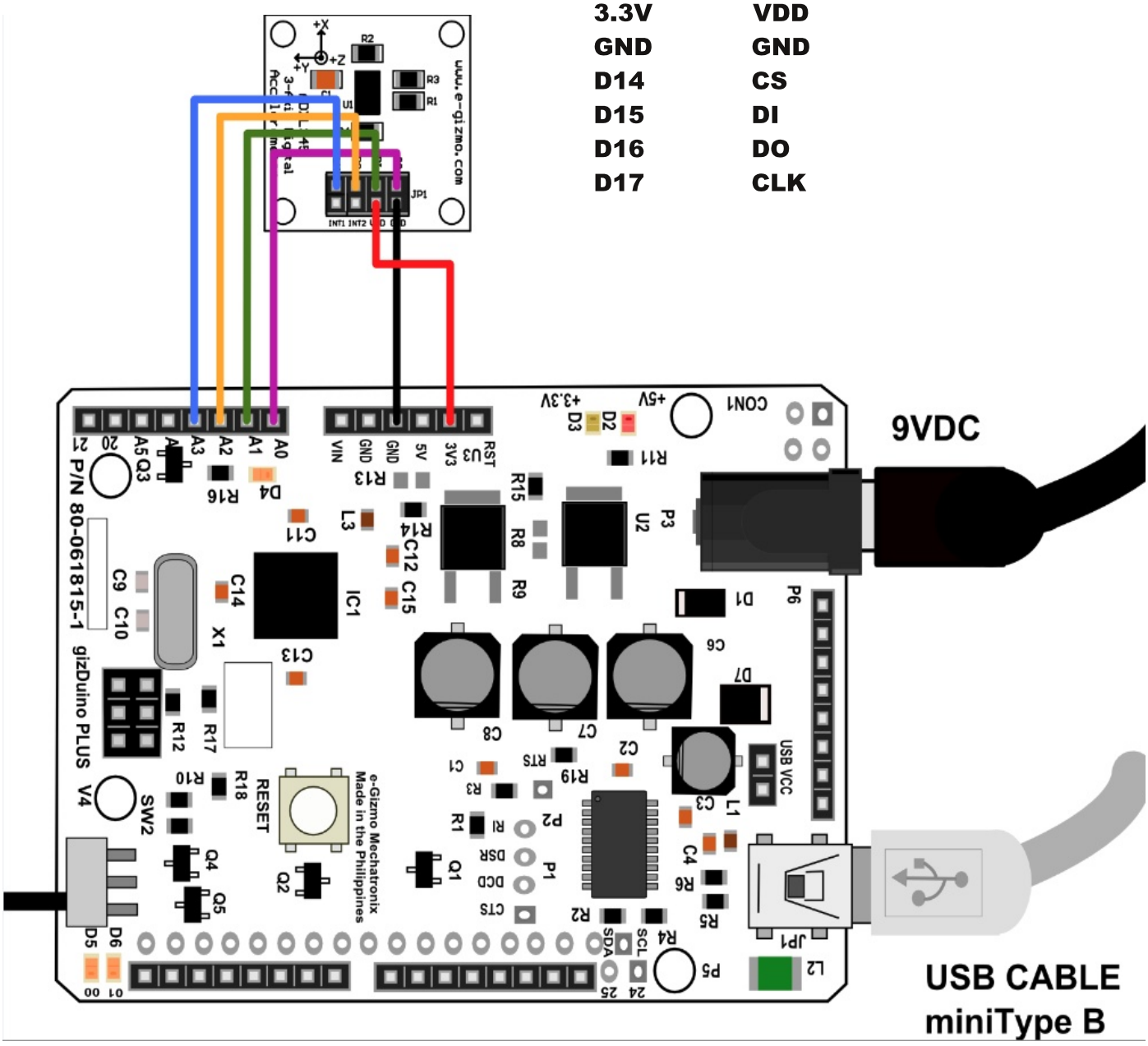
```
    case ADXL345_DATARATE_25_HZ:
      Serial.print  ("25 ");
      break;
    case ADXL345_DATARATE_12_5_HZ:
      Serial.print  ("12.5 ");
      break;
    case ADXL345_DATARATE_6_25HZ:
      Serial.print  ("6.25 ");
      break;
    case ADXL345_DATARATE_3_13_HZ:
      Serial.print  ("3.13 ");
      break;
    case ADXL345_DATARATE_1_56_HZ:
      Serial.print  ("1.56 ");
      break;
    case ADXL345_DATARATE_0_78_HZ:
      Serial.print  ("0.78 ");
      break;
    case ADXL345_DATARATE_0_39_HZ:
      Serial.print  ("0.39 ");
      break;
    case ADXL345_DATARATE_0_20_HZ:
      Serial.print  ("0.20 ");
      break;
    case ADXL345_DATARATE_0_10_HZ:
      Serial.print  ("0.10 ");
      break;
    default:
      Serial.print  ("???? ");
      break;
  }
  Serial.println(" Hz");
}

void displayRange(void)
{
  Serial.print  ("Range:         +/- ");

  switch(accel.getRange())
  {
    case ADXL345_RANGE_16_G:
      Serial.print  ("16 ");
      break;
    case ADXL345_RANGE_8_G:
      Serial.print  ("8 ");
      break;
    case ADXL345_RANGE_4_G:
      Serial.print  ("4 ");
      break;
    case ADXL345_RANGE_2_G:
```

```
      Serial.print  ("2 ");
      break;
    default:
      Serial.print  ("?? ");
      break;
  }
  Serial.println(" g");
}

void setup(void)
{
#ifndef ESP8266
  while (!Serial); // for Leonardo/Micro/Zero
#endif
  Serial.begin(9600);
  Serial.println("Accelerometer Test"); Serial.println("");

  /* Initialise the sensor */
  if(!accel.begin())
  {
    /* There was a problem detecting the ADXL345 ... check your connections */
    Serial.println("Ooops, no ADXL345 detected ... Check your wiring!");
    while(1);
  }

  /* Set the range to whatever is appropriate for your project */
  accel.setRange(ADXL345_RANGE_16_G);
  // accel.setRange(ADXL345_RANGE_8_G);
  // accel.setRange(ADXL345_RANGE_4_G);
  // accel.setRange(ADXL345_RANGE_2_G);

  /* Display some basic information on this sensor */
  displaySensorDetails();

  /* Display additional settings (outside the scope of sensor_t) */
  displayDataRate();
  displayRange();
  Serial.println("");
}

void loop(void)
{
  /* Get a new sensor event */
  sensors_event_t event;
  accel.getEvent(&event);

  /* Display the results (acceleration is measured in m/s^2) */
  Serial.print("X: "); Serial.print(event.acceleration.x); Serial.print("  ");
  Serial.print("Y: "); Serial.print(event.acceleration.y); Serial.print("  ");
  Serial.print("Z: "); Serial.print(event.acceleration.z); Serial.print("  ");Serial.println("m/s^2 ");
  delay(500);
}
```

Wiring Connections:

Gizduino to ADXL345 acceleromter

3.3V	**VDD**
GND	**GND**
D14	**CS**
D15	**DI**
D16	**DO**
D17	**CLK**



```
/*
    ADXL345 Software Interface

  by e-Gizmo Mechatronix Central
    http://www.e-gizmo.com

  This program uses a special software technique
  to allow direct and safe interfacing of a 3.3V ADXL345
  device (e-Gizmo ADXL345 breakout board) with the
  5V logic interface of the gizDuino/Arduino I/O bus

  Usage terms:
  Free, as long as you agree to make us
  not liable for any bad things that may happen
  with the use of this code. Please keep our name
  on the credit.
*/



// Pin usage, change assignment if you want to
const byte  spiclk=17;    // connect to ADXL CLK
const byte  spimiso=16;  // connect to ADXL DO
const byte spimosi=15;  // connect to ADXL DI
const byte spics=14;    // connect to ADXL CS
// Don't forget, connect ADXL VDD-GND to gizDuino/Arduino +3.3V-GND


byte  xyz[8];  // raw data storage
int x,y,z;    // x, y, z accelerometer data

byte spiread;

void setup(void){
  Serial.begin(9600);      // serial i/o for test output
  init_adxl();           // initialize ADXL345

}

void loop(void){

  read_xyz();           // read ADXL345 accelerometer

  // and then send results to serial port
  // view results by using IDE Tools>Serial Monitor

  Serial.print("x = ");
  Serial.print(x);
  Serial.print("     y = ");
  Serial.print(y);
```

```
  Serial.print("     z = ");
  Serial.println(z);

  delay(500);

}

/*   Bit bang SPI function

    All SPI interface pins of the ADXL345 must be provided
    with pull-up resistors (to 3.3V, 3.3Kto 10K ohm) in order
    to work using this code.e-Gizmo ADXL345 breakout board
    already has these parts on board, hence is ready for use
    without any modifications.

    Principle of operation:
    A 3.3V logic 1 output is effected by configuring
    the driving pin as input, letting the pull up resistor
    take the logic level up to 3.3V only. A logic 0 output
    is generated by configuring the driving pin to output.
*/

void spi_out(byte spidat){
  byte bitnum=8;

    spiread=0;
    // start spi bit bang
    while(bitnum>0){

      pinMode(spiclk,OUTPUT);    // SPI CLK =0
      if((spidat & 0x80)!=0)
        pinMode(spimosi,INPUT);  // MOSI = 1 if MSB =1
        else
        pinMode(spimosi,OUTPUT);  // else MOSI = 0

      spidat=spidat<<1;
      pinMode(spiclk,INPUT);  // SPI CLK = 1

      // read spi data
      spiread=spiread<<1;

      if(digitalRead(spimiso)==HIGH) spiread |= 0x01; // shift in a 1 if MISO is 1

      pinMode(spimosi,INPUT);  // reset MOSI to 1
      bitnum--;
    }


}
```

```
/*  Initialize ADXL345 */

void  init_adxl(void){
  delay(250);
  pinMode(spics,OUTPUT);  // CS=0
  //Write to register 0x31, DATA FORMAT
  spi_out(0x31);
  // uncomment your desired range
  spi_out(0x0B); //full resolution, +/- 16g range
  //spi_out(0x0A); //full resolution, +/- 8g range
  //spi_out(0x09); //full resolution, +/- 4g range
 // spi_out(0x08); //full resolution, +/- 2g range

  pinMode(spics,INPUT);  //CS HIGH
  delay(1);
    pinMode(spics,OUTPUT);  // CS=0

  // Write to register 0x2d, POWER_CTL
  spi_out(0x2d);
  //set to measure mode
  spi_out(0x08);
  pinMode(spics,INPUT);  //CS HIGH
  delay(1);
}

/*
   Read all 3 axis x,y,z
*/

void read_xyz(void){
  int i;
    pinMode(spics,OUTPUT);  // CS=0
  //Set start address to 0x32
  //D7= 1 for read and D6=1 for sequential read
  spi_out(0xF2);
  // dump xyz content to array
  for(i=0;i<6;i++){
    spi_out(0x00);
    xyz[i]=spiread;
  }

  // merge to convert to 16 bits
  x=((int)xyz[1]<<8) + xyz[0];
  y=((int)xyz[3]<<8) + xyz[2];
  z=((int)xyz[5]<<8) + xyz[4];

  pinMode(spics,INPUT);  //CS HIGH
}
```