**Introduction**

In this lesson, we will learn how to use the acceleration sensor ADXL345.

**Components**

- 1 \* SunFounder Uno board

- 1 \* Breadboard

- 1 \* ADXL345 module

- 1 \* USB cable

- Jumper wires

**Principle**

**Accelerometer**

An accelerometer is used to measure the force generated during the acceleration. The most fundamental is the commonly-known acceleration of gravity which is 1g.

By measuring the acceleration caused by gravity, you can calculate the tilt angle of the device to the level surface. Through analyzing the dynamic acceleration, you can tell the way how the device is moving. For example, self-balancing board or hoverboard applies the acceleration sensor and gyroscope for Kalman filter and posture correction.

**ADXL345**

The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g. Digital output data is formatted as 16-bit two’s complement and is accessible through either an SPI (3- or 4-wire) or I2C digital interface. In this experiment, the I2C digital interface is used.

It is well suited to measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables the inclination change measurement by less than 1.0°. And the excellent sensitivity (3.9mg/LSB @2g) provides a high-precision output of up to ±16g.

**How ADXL345 works**

The ADXL345 detects the acceleration with the sensing component at the front, and then the electric signal sensing component changes it into electric signal, which is analog. Next, the AD adapter integrated on the module will convert the analog signal into digital one.

The X\_OUT, Y\_OUT and Z\_OUT are the values at the X, Y, and Z axis respectively. Place the module face up: Z\_OUT can reach +1g at most, the minimum of X\_OUT is -1g toward the Ax direction, and the minimum of Y\_OUT is -1g toward the Ay direction. On the other hand, turn the module upside down: the minimum of Z\_OUT is -1g, the maximum of X\_OUT is +1g toward the Ax direction, and the maximum of Y\_OUT is +1g toward the Ay direction. , as shown below. Rotate the ADXL345 module and you'll see the change of three values.

**Relationship between output and gravity direction**

Pin Function of ADXL345 Module

|  |  |
| --- | --- |
| **Name** | **Description** |
| VS | Supply Voltage |
| CS | Chip SelectI2C mode is enabled if the CS pin is tied high to VDD I/O(VDD I/O = 1.8V), so the I2C mode is set as default for the module. |
| SD0 | Serial Data Out, Alternate I2C Address Select |
| INT1 | Interrupt 1 Output |
| INT2 | Interrupt 2 Output |
| 3.3V | 3.3V |
| SDA | Serial Data (I2C), Serial Data In (SPI 4-Wire), Serial Data In/Out (SPI 3-Wire) |
| SCL | Serial Communications Clock |
| GND | GND |

See the following figure for the schematic diagram. There is a 3.3v voltage regulator chip in the circuit, so you can power the module with 5V or 3.3V.

Since SDO has been connected to GND, the I2C address of the ADXL345 is 0x53, 0xA6 for write, 0xA7 for read.

**Experimental Procedures**

**Step 1:**Build the circuit

|  |  |
| --- | --- |
| **ADXL345** | **Uno/Mega2560** |
| 3.3V | 3.3V |
| GND | GND |
| SCL | A5 Uno/ pin 21 Mega2560 |
| SDA | A4 Uno/pin 20 Mega2560 |

**Step 2**: Open the code file

**Step 3**: Select correct Board and Port

**Step 4**: Upload the sketch to the SunFounder Uno board

After uploading, open Serial Monitor, where you can see the data detected. When the acceleration of the module changes, the figure will change accordingly on the window.

**Code Analysis**

The code for ADXL345 experiment includes 3 parts: initialize each port and device, acquire and store data sent from the sensors, and convert the data.

**Code**

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Name: ADXL345 Description: Print the X, Y, and Z values on Serial Monitor  Website: www.sunfoundder.coom Email: serice@sunfounder.com Date: 14/12/2016 Author: Daisy \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/#include <wire.h> //Call the I2C library built in Arduino//Set the address of the register#define Register\_ID 0#define Register\_2D 0x2D //standby,measurement,sleep,wake\_up mode set#define Register\_X0 0x32 //DATAX0, store the value of X0#define Register\_X1 0x33 //DATAX1, store the value of X1#define Register\_Y0 0x34 //DATAY0, store the value of Y0#define Register\_Y1 0x35 //DATAY1, store the value of Y1#define Register\_Z0 0x36 //DATAZ0, store the value of Z0#define Register\_Z1 0x37 //DATAZ1, store the value of Z1int ADXAddress = 0x53; //I2C addressint reading = 0;int val = 0;int X0, X1, X\_out;int Y0, Y1, Y\_out;int Z1, Z0, Z\_out;double Xg, Yg, Zg;void setup(){ Serial.begin(9600);//Set the baud rate of serial monitor as 9600bps delay(100); Wire.begin(); //Initialize I2C delay(100); Wire.beginTransmission(ADXAddress); //transmit to device ADXAddress 0x53  Wire.write(Register\_2D); // Wire.write(8); //measuring enable  Wire.endTransmission(); //end transmitting Serial.println("Accelerometer Test ");}void loop(){ Wire.beginTransmission(ADXAddress); //transmit to device ADXAddress 0x53  Wire.write(Register\_X0); //request the X0 value Wire.write(Register\_X1); //request the X1 value Wire.endTransmission(); //stop transmitting Wire.requestFrom(ADXAddress, 2); //request 2 bytes from device 0x53 if (Wire.available()  |