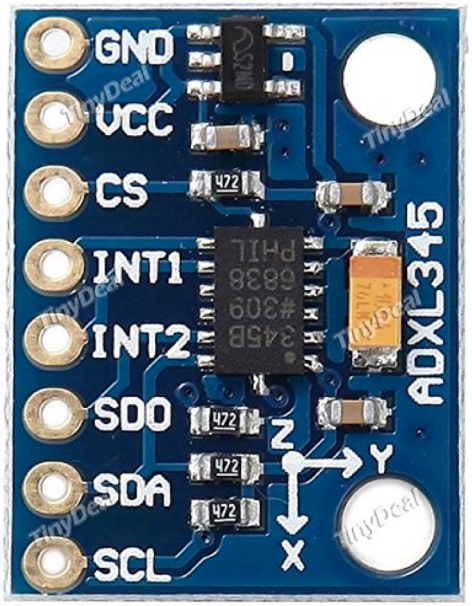
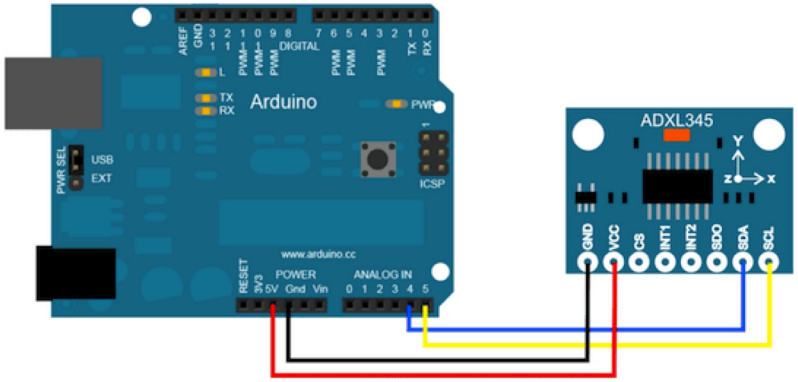
ADXL345 accelerometer

ADXL345 Accelerometer



Accelerometer sensor – ADXL345



ADXL345 I2C Wired connection

- Accelerometers measure acceleration, which is the rate of change of the velocity of an object, in meters per second squared (m/s2) or in G-forces (g). A single G-force on planet Earth is equivalent to 9.8 m/s2, but this will vary due to elevation and gravitational pull variations.
- The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g.
- Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface.

Accelerometer ADXL345 – How it works

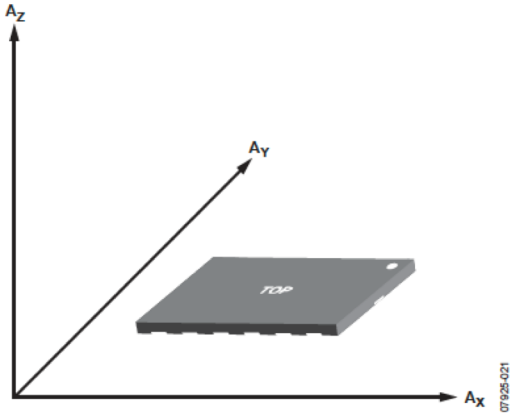
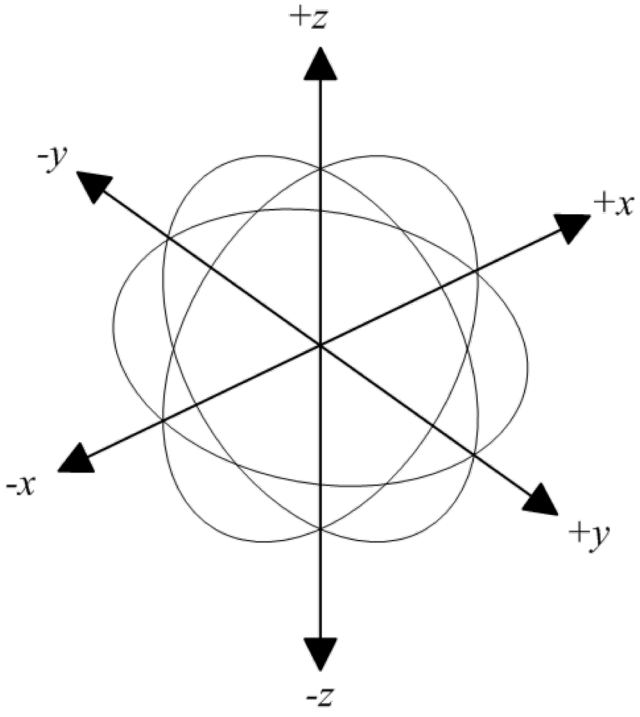


Figure 16. Axes of Acceleration Sensitivity (Corresponding Output Voltage Increases When Accelerated Along the Sensitive Axis)



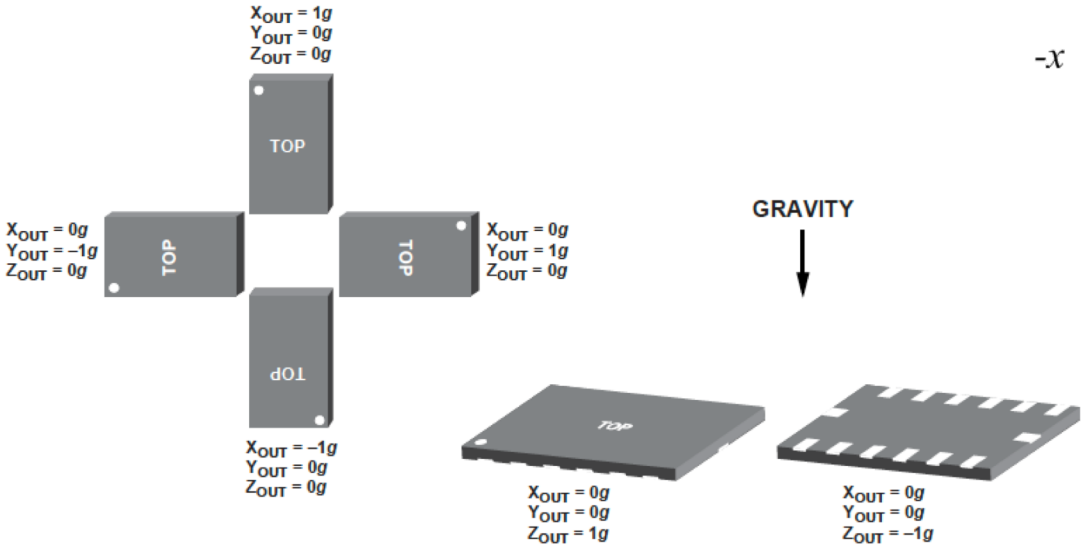


Figure 17. Output Response vs. Orientation to Gravity

Accelerometer sensor – ADXL345 Registers

REGISTER MAP

Table 16. Register Map

Address					
Hex	Dec	Name	Type	Reset Value	Description
0x00	0	DEVID	R	11100101	Device ID.
0x01 to 0x01C	1 to 28	Reserved			Reserved. Do not access.
0x1D	29	THRESH_TAP	R/$\overline{W}$	00000000	Tap threshold.
0x1E	30	OFSX	R/$\overline{W}$	00000000	X-axis offset.
0x1F	31	OFSY	R/$\overline{W}$	00000000	Y-axis offset.
0x20	32	OFSZ	R/$\overline{W}$	00000000	Z-axis offset.
0x21	33	DUR	R/$\overline{W}$	00000000	Tap duration.
0x22	34	Latent	R/$\overline{W}$	00000000	Tap latency.
0x23	35	Window	R/$\overline{W}$	00000000	Tap window.
0x24	36	THRESH_ACT	R/$\overline{W}$	00000000	Activity threshold.
0x25	37	THRESH_INACT	R/$\overline{W}$	00000000	Inactivity threshold.
0x26	38	TIME_INACT	R/$\overline{W}$	00000000	Inactivity time.
0x27	39	ACT_INACT_CTL	R/$\overline{W}$	00000000	Axis enable control for activity and inactivity detection.
0x28	40	THRESH_FF	R/$\overline{W}$	00000000	Free-fall threshold.
0x29	41	TIME_FF	R/$\overline{W}$	00000000	Free-fall time.
0x2A	42	TAP_AXES	R/$\overline{W}$	00000000	Axis control for tap/double tap.
0x2B	43	ACT_TAP_STATUS	R	00000000	Source of tap/double tap.
0x2C	44	BW_RATE	R/$\overline{W}$	00001010	Data rate and power mode control.
0x2D	45	POWER_CTL	R/$\overline{W}$	00000000	Power-saving features control.
0x2E	46	INT_ENABLE	R/$\overline{W}$	00000000	Interrupt enable control.
0x2F	47	INT_MAP	R/$\overline{W}$	00000000	Interrupt mapping control.
0x30	48	INT_SOURCE	R	00000010	Source of interrupts.
0x31	49	DATA_FORMAT	R/$\overline{W}$	00000000	Data format control.
0x32	50	DATAX0	R	00000000	X-Axis Data 0.
0x33	51	DATAX1	R	00000000	X-Axis Data 1.
0x34	52	DATAY0	R	00000000	Y-Axis Data 0.
0x35	53	DATAY1	R	00000000	Y-Axis Data 1.
0x36	54	DATAZ0	R	00000000	Z-Axis Data 0.
0x37	55	DATAZ1	R	00000000	Z-Axis Data 1.
0x38	56	FIFO_CTL	R/$\overline{W}$	00000000	FIFO control.
0x39	57	FIFO_STATUS	R	00000000	FIFO status.

Accelerometer sensor – ADXL345 access from Arduino

To access to the ADXL registers using the I2C bus:

- Initiate transmission to device (using write address*)
- Write the address of the register we want to acess
- Write the data we want to push into the register
- Finish transmission

```
//Writes val to address register on device
void writeTo(int device, byte address, byte val) {
   Wire.beginTransmission(device); // start transmission to device
   Wire.write(address);            // send register address
   Wire.write(val);                // send value to write
   Wire.endTransmission();         //end transmission
}
```

Register 0x2D—POWER_CTL (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	Link	AUTO_SLEEP	Measure	Sleep	Wakeup	

```
//Turning on the ADXL345
writeTo(0x53, 0x2D, 0);  // Reset the power control
writeTo(0x53, 0x2D, 16); // Sensor in standby mode
writeTo(0x53, 0x2D, 8);  // Sensor in measure mode
```

Accelerometer sensor – ADXL345 Access from Arduino

To read the measured values from the the ADXL345 registers using the I2C bus:

- Initiate transmission to device (using write address*)
- Write the address of the register we want to start** reading from
- Initiate transmission to device, again! (using read address*)
- Read bytes one after another
- Finish transmission

```
//reads num bytes starting from address register on device in to buff array
void readFrom(int device, byte address, int num, byte buff[]) {
  Wire.beginTransmission(device); //start transmission to device
  Wire.write(address);    //sends address to read from
  Wire.endTransmission(); //end transmission

  Wire.beginTransmission(device); //start transmission to device
  Wire.requestFrom(device, num);  // request 6 bytes from device

  int i = 0;
  while(Wire.available())  //device may send less than requested (abnormal)
  {
    buff[i] = Wire.read(); // receive a byte
    i++;
  }
  Wire.endTransmission(); //end transmission
}
```

```
byte buff[6];    //6 bytes buffer for saving data read from the device
readFrom(0x53, regAddress, 6, buff); //read the acceleration data from the ADXL345
```

Accelerometer sensor - ADXL345 XYZ axis

The measured values are stored in three registers (6bytes), starting from the 0x32 address

Register 0x31—DATA_FORMAT (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range	

Table 18. *g* Range Setting

Setting		
D1	D0	*g* Range
0	0	±2 *g*
0	1	±4 *g*
1	0	±8 *g*
1	1	±16 *g*

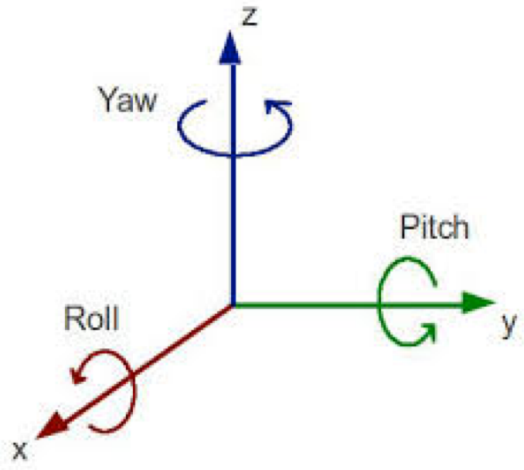
Register 0x32 to Register 0x37—DATAX0, DATAX1, DATAY0, DATAY1, DATAZ0, DATAZ1 (Read Only)

These six bytes (Register 0x32 to Register 0x37) are eight bits each and hold the output data for each axis. Register 0x32 and Register 0x33 hold the output data for the x-axis, Register 0x34 and Register 0x35 hold the output data for the y-axis, and Register 0x36 and Register 0x37 hold the output data for the z-axis. The output data is twos complement, with DATAx0 as the least significant byte and DATAx1 as the most significant byte, where x represent X, Y, or Z. The DATA_FORMAT register (Address 0x31) controls the format of the data. It is recommended that a multiple-byte read of all registers be performed to prevent a change in data between reads of sequential registers.

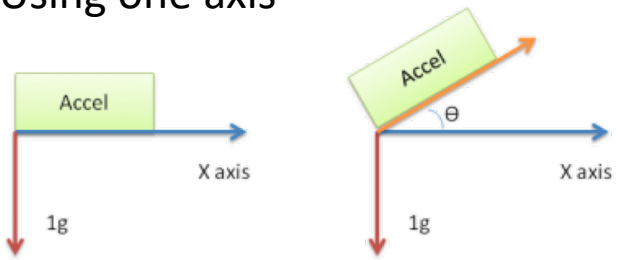
```
//Range: 0-2g, 1-4g, 2-8g, 3-16g
writeTo(0x53, 0x31, 0);

//read the acceleration data from the ADXL345
readFrom(0x53, 0x32, 6, buff);

//each axis reading comes in 10 bit resolution, ie 2 bytes.
//Least Significat Byte first!!
//thus we are converting both bytes in to one int
x = (((int)buff[1]) << 8) | buff[0];
y = (((int)buff[3])<< 8) | buff[2];
z = (((int)buff[5]) << 8) | buff[4];
```

Accelerometer sensor - ADXL345 XYZ angles

- It is possible to measure roll, yaw and pitch evaluating the X Y Z values

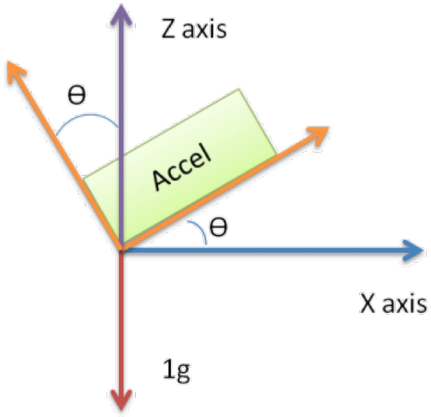


- Using one axis



$$\Theta = \sin^{-1}(x)$$

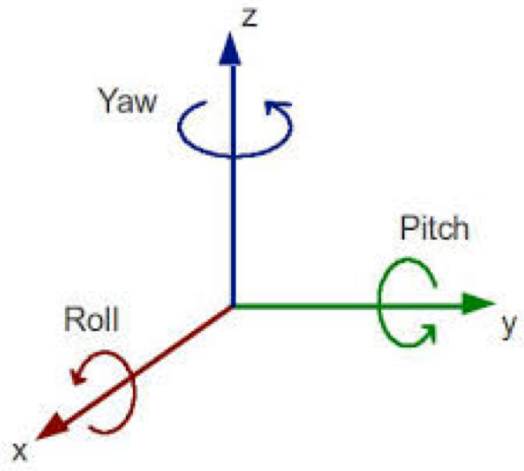
- Using two axes



$$\Theta = \arctan\left(\frac{x}{z}\right)$$

Accelerometer sensor - ADXL345 XYZ angles

- It is possible to measure roll, yaw and pitch evaluating the X Y Z values



- Using all three axis

$$\Theta = \arctan\left(\frac{x}{\sqrt{y^2 + z^2}}\right)$$

Remember *atan2* from the math.h library returns values in radians range