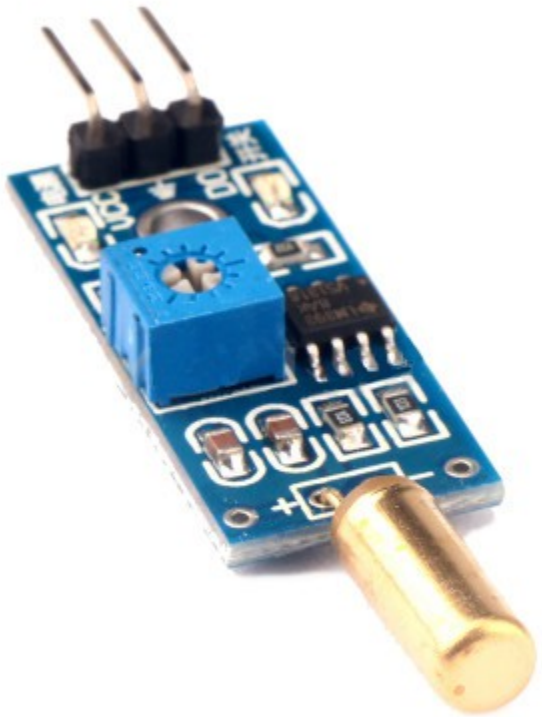
HK Shan Hai Group Limited
Room 620 , Yutian building , songling road , Futian district , Shenzhen

Tilt Sensor Module



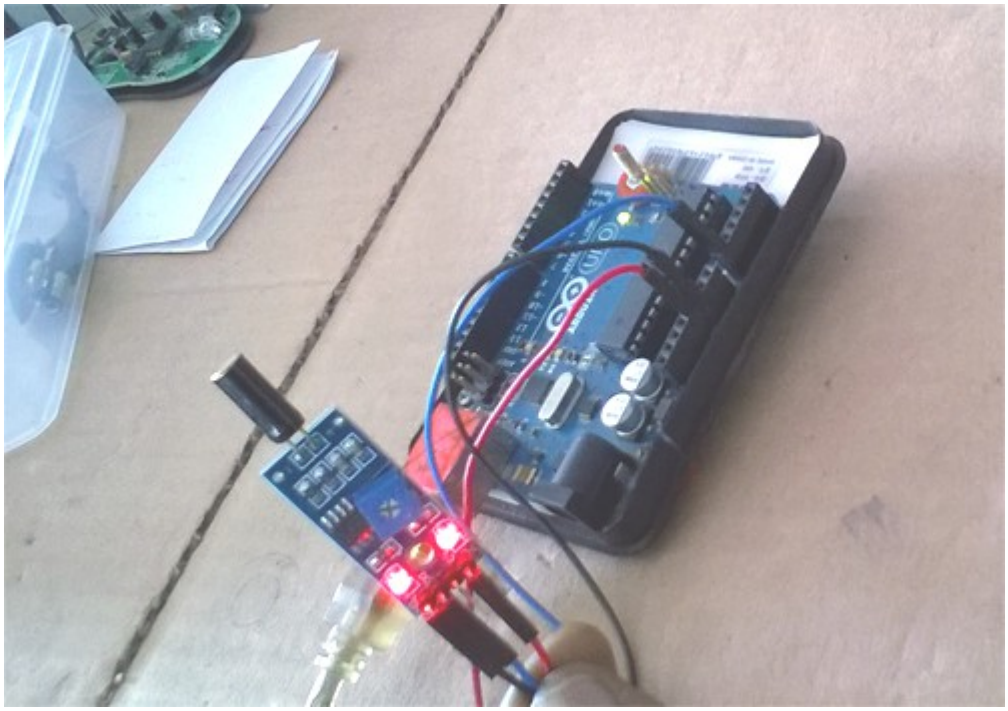
Tilt sensors are essential components in security alarm systems today. Standalone tilt sensors sense tilt angle or movement. Tilt sensors can be implemented using mercury and roller ball technology, and can be mounted using mechanical threading, magnets, or adhesives, depending on what type of surface they are being mounted to.

Recent technological advancements in the manufacturing of tilt sensors have improved accuracy, reduced cost, and increased lifetime. The type SW-520D is a commonly available roller-ball type tilt sensor consists of two conductive elements (poles) and a conductive free mass (rolling ball), encapsulated in the same case. When the tilt sensor is oriented so that that end is downwards, the mass rolls onto the poles and shorts them, acting as a switch stroke. Microcontroller-compatible tilt sensor modules based on SW-520D are also available at affordable costs.

Electronics circuitry behind this tiny module is usually centered around the dual- comparator chip LM393. The module features a tilt sensor, a signal amplifier, a standard 3-pin header, a power indicator that signals that the module is correctly powered, and a status indicator that lights up when a tilt is detected by the tilt sensor.

When the tilt sensor is in its upright position, the ball inside the tilt sensor bridges the two contacts, completing the circuit. When the board is tilted, the ball moves, and the circuit opens. When upright, the module outputs 0V (L) and when it is tilted, it outputs 5V (H) through the digital output (DO) terminal of the 3-pin header. If the analog output (AO) of the module is connected to an analog input

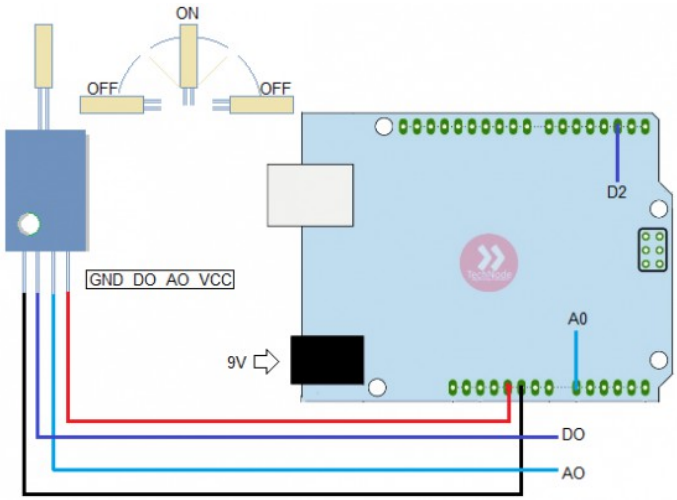
(for example A0) on the Arduino you can expect to read a value of 0 (0V) when in its upright position and 1023 (5V) when it is tilted.



Description:
- Adopts high sensitivity ball switch SW-520D angle sensor
- comparator output,high drive ability,over 15ma current
- working voltage:3.3V-12V
- with screw mounting hole
- PCB size:4.2cm x 1.4cm
- Adopts LM393 comparator

Hook Up

The tilt sensor module can be connected to arduino using suitable jumper wires. First of all connect the power supply lines; VCC and GND of the module to 5V and GND of the Arduino respectively. Next link the digital output (DO) of the module to digital pin 2 (D2) and analog output (AO) to analog input 0 (A0) of the arduino. The whole hardware should be powered by a 9V DC / USB source through the DC IN /USB socket of the Arduino board. Keep the tilt switch position in upright position as indicated in the figure shown below.

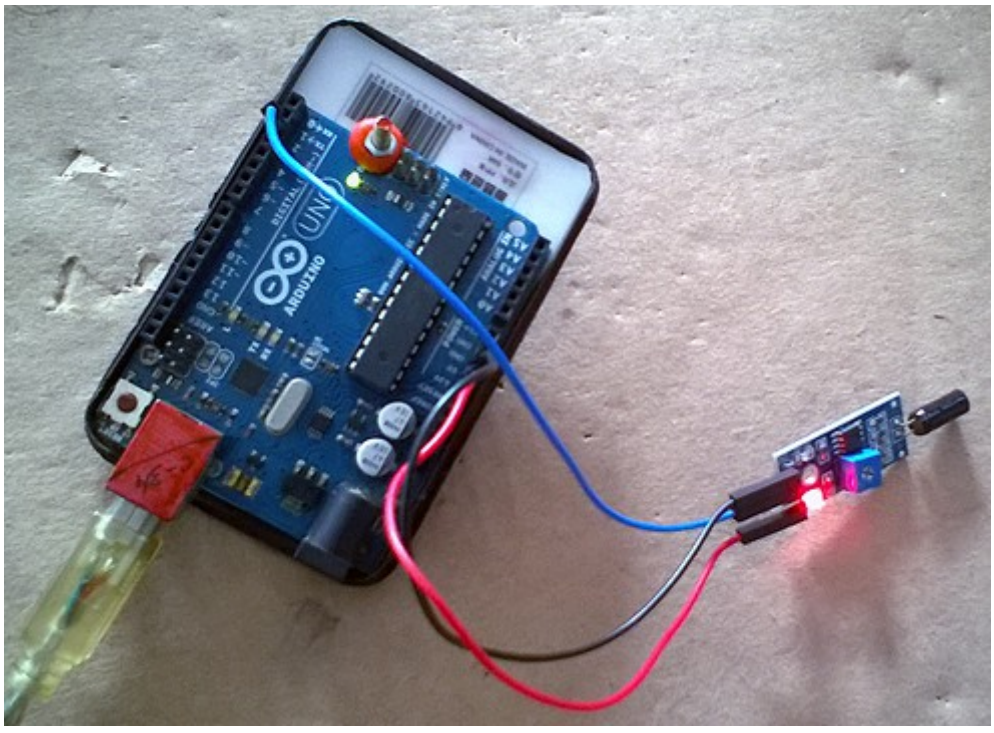


Sketch: Digital

This example code wakes the onboard indicator (LED at D13) of the Arduino when a tilt is inputted by the tilt sensor module through the occupied digital input (D2). Just copy-paste this code into your Arduino IDE, compile, and upload it to your Arduino as usual.

```
const int statusLED = 13;
const int switchTilt = 2;
int val = 0;
void setup(){
   pinMode (statusLED,OUTPUT);
   pinMode (switchTilt,INPUT);
}
void loop(){
   val = digitalRead(switchTilt);
   if (val == HIGH){
      digitalWrite(statusLED,HIGH);
   }
   else {
   digitalWrite(statusLED,LOW);
   }
}
```

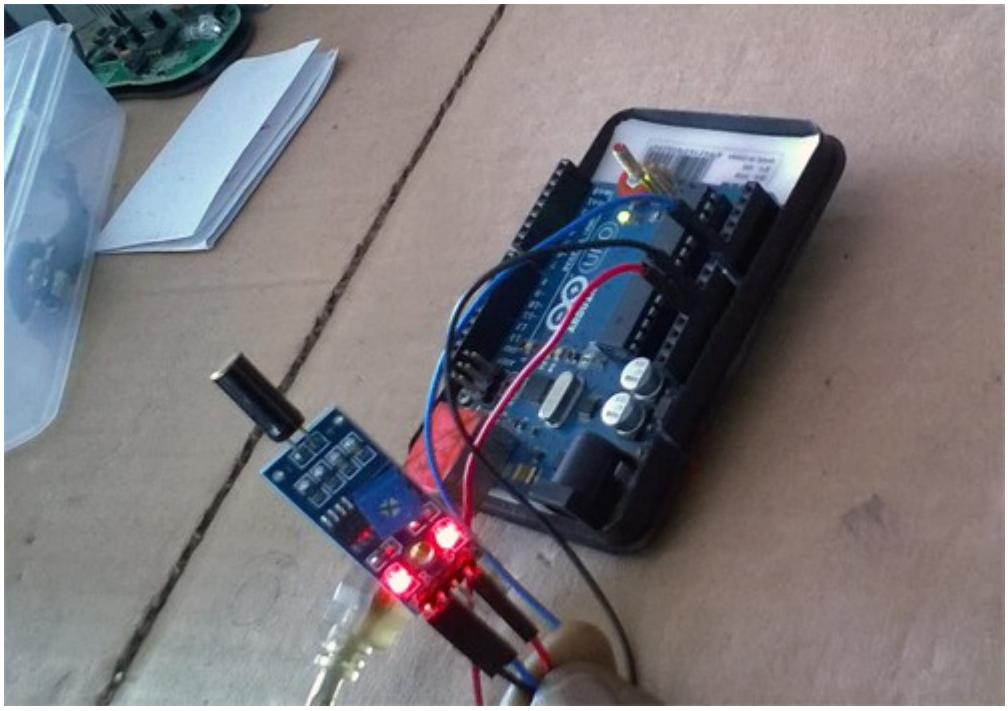
Note that this code does not include a “software-debounce” feature commonly used with button/switch inputs. This is not necessary here because the tilt sensor module have a built-in (1ms) “hardware debounce” arrangement using a simple RC network (R-10K & C-100n).



Sketch: Analog

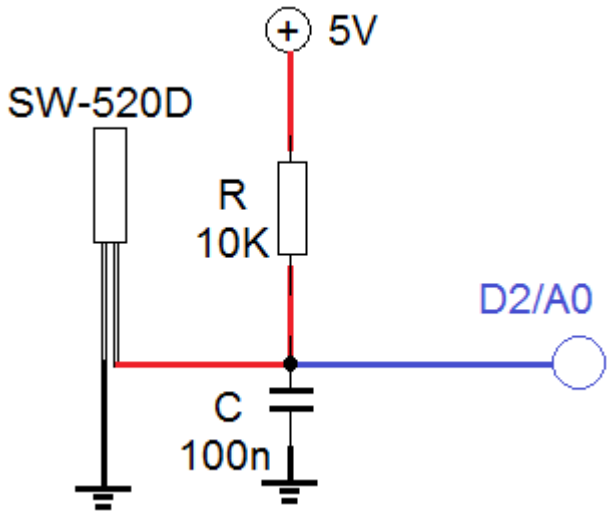
This example code lights up the onboard indicator (LED at D13) of the Arduino when a tilt is inputted by the tilt sensor module through the occupied analog input (A0). Again, copy-paste this code into your Arduino IDE, compile, and upload it to your Arduino as done earlier.

```
1.  int lightPin = 13;
2.  int tiltPin = 0;
3.  void setup() {
4.  }
5.  void loop() {
6.     int analogValue = analogRead(tiltPin);
7.     if (analogValue<512) {
8.        analogWrite(lightPin, 0);
9.     }
10.    else {
11.       analogWrite(lightPin, analogValue);
12.       delay(1000);
13.    }
14. }
```



Many readers might wonder why I opted for somewhat “strange” sketches here. It’s for nothing; first of all test your hardware with the included sketches, and then start the brainstorming. Replace the above described sketches with your own favourite sketches – merely a little home work for you!

Frankly speaking, with the sketches presented here, these experiments can also be conducted using a tilt sensor (SW-520D) only, ie. without the whole tilt sensor module. If you have an independent tilt sensor component at hand, make a try with the following hardware instead of the dedicated module.



Sketch: Interrupts

Interrupt – incorporated since the 0007 version of the Arduino IDE – breaks in in the execution of the main code. On the hardware front, Arduino is equipped with two interrupt ports so Arduino can sense those pins for an event to wake up and resume execution of code. It is even possible to execute special code depending on which pin triggered the wake up (the interrupt). In short, interrupt is a method by which a microcontroller can execute its normal program while continuously monitoring for some kind of interrupt (event). This interrupt can be triggered by some sort of sensor, or input like a switch.

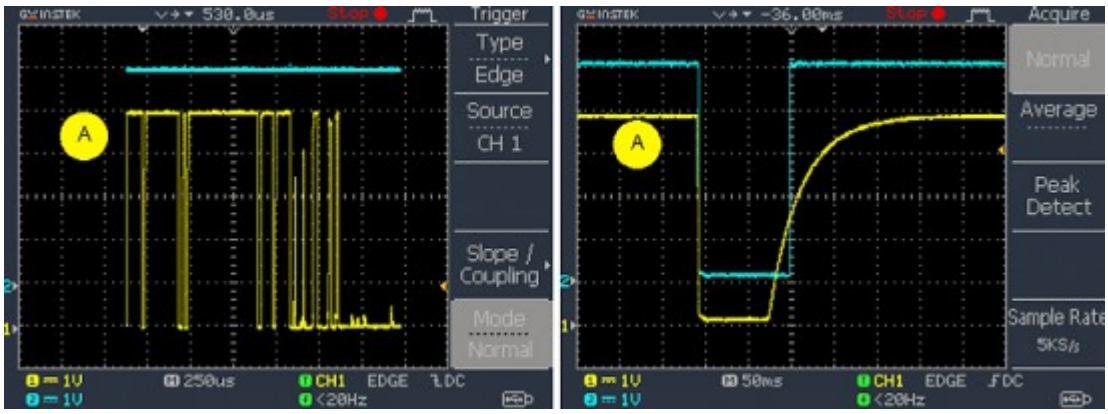
When the interrupt occurs, the microcontroller takes immediate notice, saves its execution state, runs a small chunk of code often called the interrupt handler or interrupt service routine, and then returns back to whatever it was doing before. The set up in the program defines where the microcontroller should start executing code if a particular interrupt occurs. In Arduino, we use a function called "attachInterrupt()" to do this. This function adopts three parameters. The first is the number of the interrupt, which tells the microprocessor which pin to monitor. The second parameter of this function is the location of code we want to execute if this interrupt is triggered. And the third, tells it what type of trigger to look for, a logic high, a logic low or a transition between the two. You can find detailed articles/tutorials on Arduino Interrupts (prepared by me) elsewhere in this website.

Following is a basic example sketch to demonstrate the interrupt function in an Arduino connected with the tilt sensor. This code looks for interrupts on "interrupt pin 0" (D2) of the Arduino to control the onboard indicator at its output (D13).

```
1.  #define LED_PIN 13
2.  #define INTERRUPTPIN 0
3.  volatile boolean state = HIGH;
4.  void setup() {
5.     pinMode(LED_PIN, OUTPUT);
6.     attachInterrupt(INTERRUPTPIN, inputChange, CHANGE);
7.  }
8.  void loop() {
9.     digitalWrite(LED_PIN, state);
10. }
11. void inputChange()
```

```
12.        {
13.           state = !state;
14.        }
```

Unfortunately, this loose sketch is prone to mischief because of the switch-bounce problems. Replacing the 100nF capacitor (C) with a 1uF capacitor might solve this to a certain extent (hardware-debounce). Otherwise opt for a sketch filled with “debounce” code lines (software-debounce).



You can see just how much bouncing occurs in this oscillogram of the input from a switch. Trace “A” is the voltage appearing at the input pin of the Arduino. Instead of the expected smooth transition, a series of pulses lasting over 1 mS are generated. Each one of these pulses would generate an (unnecessary) interrupt. However, if we attach some debouncing hardware to the switch then the bouncing effect is filtered out. Notice how the “A” trace is a smooth curve, with a gradual transition!