Arduino Tutorial 9 – Reading Sensors

Background
- In tutorial 3, with the LCD shield plugged in - when you pressed one of the 5 buttons we were able to read which button on the LCD shield was pressed.
In tutorial 5, using an Infra Red detector - we could tell which button on the remote control was pressed. These are examples of sensors or inputs.

Types
There are 2 types of sensor - a digital sensor and an analog sensor.

Digital - A switch is a digital sensor – it is either on or off.
We can use digitalRead on the Arduino pin to tell us the state of the switch.
All the pins of the Arduino can be used for digital inputs.

Analog – when the output of a sensor is variable, it is called analog.
Arduino pins A0 to A5 are analog input pins, and we use analogRead..
This gives us a value between 0 and 1023. (They have 10 bits resolution).
Examples of analog sensors are temperature; Ultra sonic; light; moisture.

Shield
While you can connect a sensor to the proto-type shield, it is difficult to attach other shields, such as an LCD screen to display the results from the sensor.
A Sensor shield solves this problem, and provides 6 secure sockets (called buckled ports).
Plus it provides a socket for I2C (if jumpers are set to IIC). I2C uses pins A4 and A5.

Sensors: Today we will use 4 sensors.
- DHT11 a temperature and humidity sensor;
- HC-SR501 a PIR or Passive Infra Red sensor
- HC-SR04 an Ultra sonic distance sensor.
- ADXL345 A 3 axis tilt sensor and accelerometer.
The PIR is a digital sensor (either on or off). The other 3 are analog (variable).
CARE – the RGB Led module (Keyes CNT1) needs external resistors.
Sensors

**DHT11 – Temperature and Humidity.**

Reading and displaying the temperature / humidity is a very simply Arduino project.

This DHT11 sensor uses only 3 wires : VCC or +ve (positive 5V) ; GND or -ve (negative) and data.

The DHT11 sensor comes in 2 forms –
- The bare unit with 4 pins , one of which is NC – not connected , or
- The sensor mounted on a board, as a module - with 3 pins and resistors etc, or

The data wire can be connected to any pin on the Arduino.

Your sketch/code must include the DHT library, as this provides a READ function to obtain the values for humidity and temperature.
```
dht.readHumidity();
dht.readTemperature();
```

- Connect a 9V battery to your Arduino as the power source.
- Use the sensor shield, with the DHT11 plugged in to one of the “buckled” sockets.
- Plug the LCD shield on top to display the readings , and you have a mobile weather station.

Refer to the example code.

**PIR – HC-SR05 Passive Infra Red (PIR) or heat sensor.**

All living things give off heat. A human gives off more heat than a beetle.

This sensor can detect when the amount of heat ( infra red) changes.
When it detects a change, it causes the data pin to go high (eg 5V) – eg this is a digital sensor.

You can adjust the sensitivity (or distance). Turn the pot SX fully clockwise, small movement close by (or moderate 7 metes away), will trigger it. Turned fully anti-clockwise only large movement triggers it.

You can also adjust the time delay. Turn the pot marked TX fully clockwise and once triggered, the sensor stays on for over 5 minutes. Turned fully anti-clockwise, it stays on for only 3 seconds then turn off; After triggering, it will remain off for 3 seconds before the sensor will trigger again.

The sensor uses 3 pins -
- VCC or +ve (positive 5V) ; GND or -ve (negative) and Data.

Since it detects changes in the background infra red, it needs about 60 seconds at start-up, to read the average back-ground IR.

Here is a simple way to light a LED, when the sensor triggers :
```
digitalWrite(led, digitalRead(pir));
delay(100);
```
**HC-SR04 – Ultra Sonic sensor.**

Bats, dolphins etc navigate by “echo location”. They send out a high frequency signal, and listen for the echo. The time it takes for the signal to return, lets them calculate the distance to the object.

The HC-SR04 works in the same way. It has one transmitter, and one receiver. It has 4 pins - VCC or +ve (positive 5V); GND or -ve (negative); a Trig (or Send) and an Echo.

Arduino has a special function called pulseIn, that gives the time (duration) from transmission to echo. We know that the speed of sound in air is 340 meters per second, or 29 microseconds per centimetre. Since we know how long it took for the signal to go out and come back, we can calculate the distance.

\[
\text{distance} = \frac{\text{duration}}{29} / 2;
\]

// divide the duration by 29, then divide by 2 since it travels out and back – eg twice the distance.
This gives the distance in centimetres.

```plaintext
// Example code for using HC-SR04 Ultra Sonic distance sensor.

int trigPin = 4; // connect the trig pin to Arduino pin 4
int echoPin = 8; // connect the echo pin to Arduino pin 8.

int duration, distance; // create integer variables for duration and distance

void setup()
{
    Serial.begin(9600);
    pinMode(trigPin, OUTPUT);  // declare the trig pin as output
    pinMode(echoPin, INPUT);    // declare the echo pin as input
}

void loop()
{
    digitalWrite(trigPin, HIGH);  // send out (transmit) the pulse;
    delayMicroseconds(10);       // The sensor needs 10 microseconds of pulse
    digitalWrite(trigPin, LOW);  // stop the pulse.
    duration = pulseIn(echoPin, HIGH);  // listen for the echo, and return the duration or time.
    distance = (duration/2) / 29.1;   // calculate the distance in cm
    Serial.print(distance);        // print the distance
    Serial.println(" cm");        // add the unit
    delay(500);
}
```

**Uses for the HC-SR04**

Range finder – measure distance to objects;
Open a door as you approach – automatic opening doors.
Detect a mouse in your kitchen at night – trigger a camera.
Avoid obstacles - your robot can turn if it detects a wall.
Avoid edges – make the robot/vacuum cleaner turn, if a step or drop is detected.
Create a radar display – see project next page.
**Project – Create a RADAR, using Arduino and PROCESSING.**

**Overview** – Use an Arduino Uno to control a servo motor moving through a 180 deg arc. An ultra sonic sensor HC-SR04 is attached to an arm on the motor shaft. The angle of the motor and the range from the sensor, are sent to the PC by the Arduino. A program written using “Processing” reads this data and displays it on a radar style screen.

The video of the project is on Youtube:
[Youtube – Grook.net](https://grook.net) (or – go to Youtube, and enter “Arduino Radar” in the search bar.).

Main site for code and instructions - [http://www.grook.net/how-to-make-radar-using-arduino-uno](http://www.grook.net/how-to-make-radar-using-arduino-uno)

**Detail:**

As you may know, "Processing" is an IDE (free Open Source Program) to easily enable graphics to be drawn on your computer screen. ( Go to their website – Processing.org to download the IDE. )

You can draw simple graphics – lines; ovals and rectangles, right through to 3d animations and [Mandlebrot fractals](http://www.grook.net). The Arduino IDE is based on "processing" hence they look almost identical.

They connected an Arduino Uno, with an ultra sonic sensor ([HC-SR04 - $2](https://www.grook.net)), and a servo motor ([SG90 - $3](https://www.grook.net)). You could probably use any stepper and just calculate the angle.

Using [Serial.print](https://www.grook.net), the Arduino sends the data (the angle of the servo and the distance the sensor picks up), to the PC. A program written using the “Processing” IDE, reads this data then displays on the PC, a RADAR screen showing all items within the sensor range.

Their website has the code for Arduino and the code for Processing, plus the construction details.

There are similar projects on Youtube. Go to Youtube and enter Arduino Radar in the search bar.
ADXL345 – Tilt Sensor

The ADXL345 is a low-power, 3-axis MEMS accelerometer module. It can connect to your Arduino with either I2C or SPI interface. (Use a 4 wire connector and plug into the I2C on the Sensor shield.) It features 4 sensitivity ranges from +/- 2G to +/- 16G.

You can use it to detect movement in each of three directions, plus single & double tap, & on which side the tap was; plus free fall;

It can measure inclination changes of less than 1.0°. Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion and if the acceleration on any axis exceeds a user-set level.

How it Works: MEMS - Micro Electro-Mechanical Systems

The sensor consists of a micro-machined structure on a silicon wafer. The structure is suspended by polysilicon springs which allow it to deflect smoothly in any direction when subject to acceleration in the X, Y and/or Z axis. Deflection causes a change in capacitance between fixed plates and plates attached to the suspended structure. This change in capacitance on each axis is converted to an output voltage proportional to the acceleration on that axis.

The device is calibrated at the factory, providing sufficient accuracy for most projects.

I2C Wiring:

The ADXL345 Breakout has an I2C address of 0x53. It can share the I2C bus with other I2C devices as long as each device has a unique address. Only 4 connections are required for I2C communication:

- GND->GND
- VIN->+5v
- SDA->SDA (Analog A4 on Arduino Uno)
- SCL->SCL (Analog A5 on Arduino Uno)

Download the ADXL345 library and install it.

Google Arduino ADXL345 for code and projects.
Go to Youtube and type Arduino ADXL345 in the search bar.

See next page for changing the sensitivity of the ADXL345 Tilt Sensor

Our Arduino tutorials are on our U3A Website – here.
Can we change the sensitivity of the ADXL345 Tilt Sensor (accelerometer) device?

Download the specification sheet (in PDF format). It states sensitivity at 2g; 4g; 8g and 16g. This looks promising.

Under THEORY OF OPERATION, it states:

“The ADXL345 is a complete 3-axis acceleration measurement system with a selectable measurement range of ±2 g, ±4 g, ±8 g, or ±16 g.”

Let's look through the PDF and see if we can find how to select the different sensitivities.

As you scroll down the PDF, you will see - REGISTER MAP - Table 19.

A register is simply another name for a memory location.
You may notice that register 0x31 is called “data format”.
This is an 8 bit register (or memory address), and bits D0 and D1 control the range or sensitivity.

Here is the table from the specification sheet.

<table>
<thead>
<tr>
<th>Register 0x31—DATA_FORMAT (Read/Write)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
</tr>
<tr>
<td>SELF_TEST</td>
</tr>
</tbody>
</table>

**Range Bits** (Sensitivity)

These bits (D1 and D0) set the g range as described in Table 21 below.

<table>
<thead>
<tr>
<th>Table 21. g Range Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

From Table 19, we know the default value for 0x31 is 00000000.
Since the last 2 values are 00, we know the default range/sensitivity is 2g.

SO – how do we change the last 2 “bits” of this 0x31 address?

There are 2 commands – bitSet and bitClear we can use.
Each command needs 2 values – the memory address, and the BIT you want to change.
For example – bitSet(0x31, 0) will change the last bit (bit 0) to a 1 - eg it becomes 00000001
To select the 4g sensitivity you would issue 2 commands:

```c
bitSet(0x31, 0);
bitClear(0x31, 1);
```

Here is one function to achieve this: It requires a variable - byte new_range;
void setRange()
{
  // this takes the value in new_range (0,1,2,or 3), and uses bitSet and bitClear
  // table is - 2g = 00; 4g = 01; 8g = 10; 16g = 11; (where 11 is binary for 3)
  if(new_range == 0) { bitClear(range,0); bitClear(range,1); }
  else if(new_range == 1) { bitSet(range,0); bitClear(range,1); }
  else if(new_range == 2) { bitClear(range,0); bitSet(range,1); }
  else if(new_range == 3) { bitSet(range,0); bitSet(range,1); }
  writeTo(DEVICE, 0x31, range);
} // end of setRange