Introduction:
Now we have covered simple input and outputs with buttons and LEDs, let’s start using the sensors! The sensors in the kit can be used to interact with your circuits through vibration, movement, light or temperature.

Goals

- Understand the difference between digital and analog sensors.
- Explore knock sensor circuits.
- Explore temperature sensor circuit.
- Explore tilt switch circuits.
The Sensors

In your kit you will find sensors that all create outcomes dependent on their input into the Arduino.

Tilt Switch

The Tilt switch is used to sense movement, breaking the circuit when it is tilted.

This is a Digital Input (it can only return on or off, no in-between states).

Thermister

The Thermistor is a temperature sensor used to pick changes in temperature.

This is an Analog Input.

Piezo Transducer

The Piezo Transducer is used to sense vibrations, a lot like a microphone.

This is an Analog Input.
For this circuit, we are going to create a knock sensor. This will detect vibration and convert this into an output for an LED. Our Piezo Transducer will use an Analog input, this means its value can be more than just High or Low. So we will use the Analog In pins.

For this, we will need:

1x Arduino Uno
1x USB Cable
1x LED
1x Piezo Transducer
1x 220 Ohm Resistor (Red, Red, Brown, Gold)
1x 1 MegaOhm resistor (Brown, Black, Green, Gold)
6x Jumper Wires

Our Piezo transducer is used with a 1 MegaOhm resistor, this allows the transducer to be within range for the Arduino Analog inputs to read its value.
Make a new sketch in the Arduino IDE and name this ‘Knock sensor’.

The program is going to use a new software interface called Serial. This is a way for the Arduino to communicate with your computer, making it very good for understand input sensor data. Copy the code below.

```cpp
//Global variable for piezo pin
int knock_sensor = 0;
//variable for store of piezo data
int knock_sensor_data = 0;

void setup(){
    //put your setup code here, to run once;
    //begin Serial communication between Arduino and the computer
    Serial.begin(9600);
}

void loop(){
    //put your main code here, to run repeatedly;
    //read piezo analog pin
    knock_sensor_data = analogRead(knock_sensor);

    //print piezo data to Serial
    Serial.println(knock_sensor_data);

    //delay to make readings every 50 milliseconds
    delay(50);
}
```

Once you have copied the code, press (compile) and if no errors appear, press (upload) and watch the results!

My code won't compile!

Is everything spelt correctly?

Are all your lines ending in a semi-colon?
Do you have the correct capital letters?

Did you close the curly brackets?
Test the circuit

Now we have our Arduino communicating with our Computer, we want to be able to view this data. We do this by opening the Serial Monitor. This is situated in the top right hand corner of the Arduino IDE. Once this is opened, you will see a new screen appear. This will display your data.

Click ![Serial Monitor](image) to open the serial monitor.

Now you have that running, you will notice that the value changes when you tap or knock the sensor. The value range of Analog In Pins is between 0 and 1023. This means you can use these values to control various things – such as an LED!
Let's go through the code understand what every part is doing.

**Global Variables**

These are our Global Variables that are initialized outside both void setup and void loop so they can be used in both.

```c
int knock_sensor = 0;
int knock_sensor_data = 0;
```

**void setup()**

This is our void setup() function. This is called once at the start of the program. In here we are simply initializing our Serial Communication. This is how we communicate data from the Arduino to our computer. The value set to 9600 is the baud rate. This is the speed of transmission and how the Arduino and Computer know how fast to communicate with each other.

```c
void setup(){
    Serial.begin(9600);
}
```

**void loop()**

This is the start of our void loop() function. Notice the open curly brackets. Everything within the open and closed bracket are within the loop function, and will be repeated one after the other infinitely.

```c
void loop(){

```

**analogRead**

This is how we read our piezo data using an Arduino function called analogRead(). We put knock_sensor within the brackets to signal to look at analog In 0, as 0 is the value of knock_sensor. By calling this, it is then stored in the variable called knock_sensor_data.

```c
knock_sensor_data = analogRead(knock_sensor);
```

**Serial.print**

Once we have stored the data, we then want to view it. This is why we call Serial.println(). This function transmits the data to the computer through Serial communication.

```c
Serial.println(knock_sensor_data);
```

**delay**

Finally, we call delay() to put a pause in our loop for 50 milliseconds. This means that our reading of the piezo data will only happen once every 50 milliseconds.

```c
delay(50);
```
This example uses the same circuit as the previous example (Circuit 3.0)

For this, we will need:

- 1x Arduino Uno
- 1x USB Cable
- 1x LED
- 1x Piezo Transducer
- 1x 220 Ohm Resistor (Red, Red, Brown, Gold)
- 1x 1 MegaOhm resistor (Brown, Black, Green, Gold)
- 6x Jumper Wires

Our Piezo transducer is used with a 1 MegaOhm resistor, this allows the transducer to be within range for the Arduino Analog inputs to read its value.
As we have already wired up our LED with our Piezo transducer, let’s edit the code to make the sensor value control the brightness of the LED.

Make a new sketch in the Arduino IDE and name this ‘KnockSensor_LED’.

```cpp
//Global variable for piezo pin
int knock_sensor = 0;
//variable for store of piezo data
int knock_sensor_data = 0;
//global variable for LED pin
int led = 11;

void setup(){
  //put your setup code here, to run once;
  //begin Serial communication between Arduino and the computer
  Serial.begin(9600);

  //set LED pin to output
  pinMode(led, OUTPUT);
}

void loop(){
  //put your main code here, to run repeatedly;

  //Read piezo analog pin
  knock_sensor_data = analogRead(knock_sensor);

  //change brightness of LED using sensor data and analogWrite()
  analogWrite(led, knock_sensor_data);

  //Print piezo data to Serial
  Serial.println(knock_sensor_data);

  //delay to make readings every 50 milliseconds
  delay(50);
}
```

Once you have copied the code, press (compile) and if no errors appear, press (upload) and watch the results!
My code won’t compile!

Is everything spelt correctly?

Are all your lines ending in a semi-colon?
Do you have the correct capital letters?

Did you close the curly brackets?

What is it doing?
In this program, we have only added 3 extra lines of code. All of which we’ve used before;

Global Variables
This is to set the led pin so we can use it in both void setup and void loop.

pinMode
This is set in void setup as it only needs to be called once. This sets the the LED pin to an output pin.

analogWrite
This is how we control the LED with the piezo vibration data.

Circuit 3.1 - Knock Sensor + LED Challenge

Can you make map the piezo sensor data (0 – 1023) to fit between the analogWrite brightness scale (0 – 255)?

Could you invert the sensor data so the LED is bright when the sensor data is low?
Now we are going to look at the thermistor. This is another type of sensor that changes the resistance dependent on the temperature.

For this, we will need:

1x Arduino Uno
1x USB Cable
1x Thermistor
1x 10k Ohm resistor
(Brown Black Orange Gold)
5x Jumper Wires

Note: There is no wrong way to put the thermistor in!

This circuit is called a voltage divider. This is how we make sure our sensor value is in range for our Arduino. We use a 10k Ohm resistor (Brown Black Orange Gold) as the maximum resistance of the thermistor is 10k Ohm, so we are dividing the voltage in half by matching the resistance.
Make a new sketch in the Arduino IDE and name this ‘Temperature Sensor’.

```cpp
// set Global variables
int thermistor = 0;
int thermistor_data = 0;

void setup() {
    // put all your setup code here to run once:
    // begin Serial communication between Arduino and the computer
    Serial.begin(9600);
}

void loop() {
    // read thermistor data through analog pin 0
    thermistor_data = analogRead(thermistor);

    // send thermistor data to computer using Serial communication
    Serial.println(thermistor_data);

    // delay in loop so readings happen every 50 milliseconds
    delay(50);
}
```

Once you have copied the code, press (compile) and if no errors appear, press (upload) and watch the results!
My code won't compile!

Is everything spelt correctly?

Are all your lines ending in a semi-colon?
Do you have the correct capital letters?

Did you close the curly brackets?

If you notice, this program and the knock_sensor program are almost identical, with the only differences being their variable names.

As like the knock sensor, we use the Serial Monitor to view the incoming data.

**Circuit 3.2 - Temperature Sensor**

Can you see a difference in pattern of the Thermistor’s sensor data and the Piezo transducers?

How could you use the Thermistor data to make a LED brighter when it is hotter?

Could you make a Temperature meter? (3 LEDs in a row, in which the adjacent LED turns on when it gets hotter).

**Circuit 3.2 - Temperature Sensor**

Arduino along with making the hardware and software also offer amazing resources to help create quick and easy projects. One of these offered is for the thermistor. Arduino have worked with the community to create a look-up table to convert Thermistor analog values to temperature values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Temperature (°C)</th>
<th>Value</th>
<th>Temperature (°C)</th>
<th>Value</th>
<th>Temperature (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>1.4</td>
<td>525</td>
<td>26.7</td>
<td>784</td>
<td>55.0</td>
</tr>
<tr>
<td>275</td>
<td>4.0</td>
<td>550</td>
<td>29.0</td>
<td>825</td>
<td>61.5</td>
</tr>
<tr>
<td>300</td>
<td>6.4</td>
<td>575</td>
<td>31.3</td>
<td>850</td>
<td>66.2</td>
</tr>
<tr>
<td>325</td>
<td>8.8</td>
<td>600,</td>
<td>33.7</td>
<td>875</td>
<td>71.5</td>
</tr>
<tr>
<td>350</td>
<td>11.1</td>
<td>625</td>
<td>36.1</td>
<td>900</td>
<td>77.9</td>
</tr>
<tr>
<td>375</td>
<td>13.4</td>
<td>650</td>
<td>38.7</td>
<td>925</td>
<td>85.7</td>
</tr>
<tr>
<td>400</td>
<td>15.6</td>
<td>675</td>
<td>41.3</td>
<td>937</td>
<td>90.3</td>
</tr>
<tr>
<td>425</td>
<td>17.8</td>
<td>700</td>
<td>44.1</td>
<td>950</td>
<td>96.0</td>
</tr>
<tr>
<td>450</td>
<td>20.0</td>
<td>725</td>
<td>47.1</td>
<td>975</td>
<td>111.2</td>
</tr>
<tr>
<td>475</td>
<td>22.2</td>
<td>750</td>
<td>50.2</td>
<td>1000</td>
<td>139.5</td>
</tr>
<tr>
<td>500</td>
<td>24.4</td>
<td>775</td>
<td>53.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The value on the left is the Analog Reading, with the temperature in celsius next to it. Roughly, what temperature is it where you are now?
Our final sensor we will use is the Tilt switch. This component, unlike the other two sensors we looked at, only has two states, on or off. Although this is the case, this makes it a lot like our push buttons.

The tilt switch works by using a metal ball-bearing inside the switch (shake it and you’ll hear it rattle!). This means that the ball-bearing is conductive, allowing electricity to flow through and close the circuit if it is touching the legs of the switch.

As this is a physical open and close of the switch (like our pushbuttons!) we can use it in two different ways.

For this, we will need:

1x Arduino Uno
1x USB cable
1x Tilt switch
1x LED
1x 220 Ohm Resistor (Red Red Brown Gold)
5x Jumper wires

(The tilt switch can be used either way!)

**Code**

This circuit does not require any programming, as the tilt switch is physically opening and closing the circuit with the LED. Test your circuit by moving the tilt switch.
To make it possible to control different outcomes using the tilt switch, we will need to use digitalRead() to sense the state of the tilt switch. To achieve this we will have to change our circuit.

For this, we will need:

- 1x Arduino Uno
- 1x USB cable
- 1x Tilt switch
- 1x LED
- 1x 10k Ohm Resistor (Brown Black Orange Gold)
- 1x 220 Ohm Resistor (Red Red Brown Gold)
- 7x Jumper Wires

(The tilt switch can be used either way!)
Make a new sketch in the Arduino IDE and name this ‘Tilt_Switch_Sensor’.

You will use two if statements to check if the tilt switch is an open or closed circuit (HIGH or LOW in digitalRead()). These are then used to turn the LED on or off.

```cpp
//Global variables
int tilt_switch = 4;
int led = 11;
int tilt_switch_state = 0;

void setup() {
  //put your setup code here, to run once:
  //begin Serial communication between Arduino and the computer
  Serial.begin(9600);
}

void loop(){
  //read the state of the tilt switch and store in the variable tilt_switch_state
  tilt_switch_state = digitalRead(tilt_switch);

  //check to see if the tilt switch is closed
  if(tilt_switch_state == HIGH){
    digitalWrite(led,HIGH);
  }
  if(tilt_switch_state == LOW){
    digitalWrite(led,LOW);
  }
}
```

The code above is purposely missed one part. Do you know what it is and why? Add it!
Once you have copied the code, press (compile) and if no errors appear, press (upload) and watch the results!

**My code won’t compile!**

Have you made sure your have enough curly brackets?

Is your spelling correct?

Have you missed anything? (secolons!)

Check the jump wire from the tilt switch to pin 4 is between the tilt switch & resistor.