Implementation Of Soil Moisture Sensor

FC-28 soil moisture sensor

How Does Soil Moisture Sensor Work?

The fork-shaped probe with two exposed conductors, acts as a variable resistor (just like a potentiometer) whose resistance varies according to the water content in the soil.

This resistance is inversely proportional to the soil moisture:

- The more water in the soil means better conductivity and will result in a lower resistance.
- The less water in the soil means poor conductivity and will result in a higher resistance.

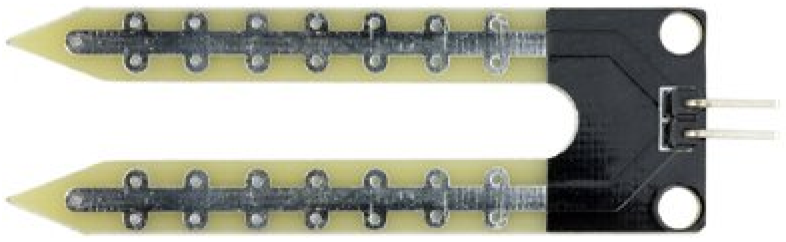
The sensor produces an output voltage according to the resistance, which by measuring we can determine the moisture level.

Hardware Overview :

A typical soil moisture sensor has two components.

The Probe :

The sensor contains a fork-shaped probe with two exposed conductors that goes into the soil or anywhere else where the water content is to be measured.

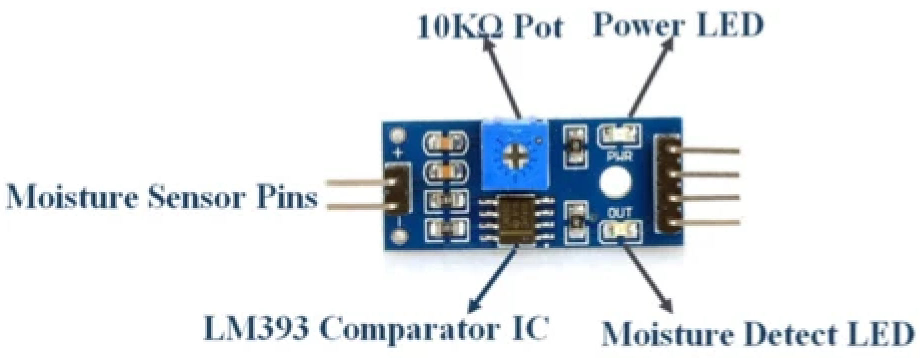


The Module :

The sensor also contains an electronic module that connects the probe to the Arduino/ Raspberry Pi.

The module produces an output voltage according to the resistance of the probe and is made available at an Analog Output (AO) pin / Digital Output (DO) pin.

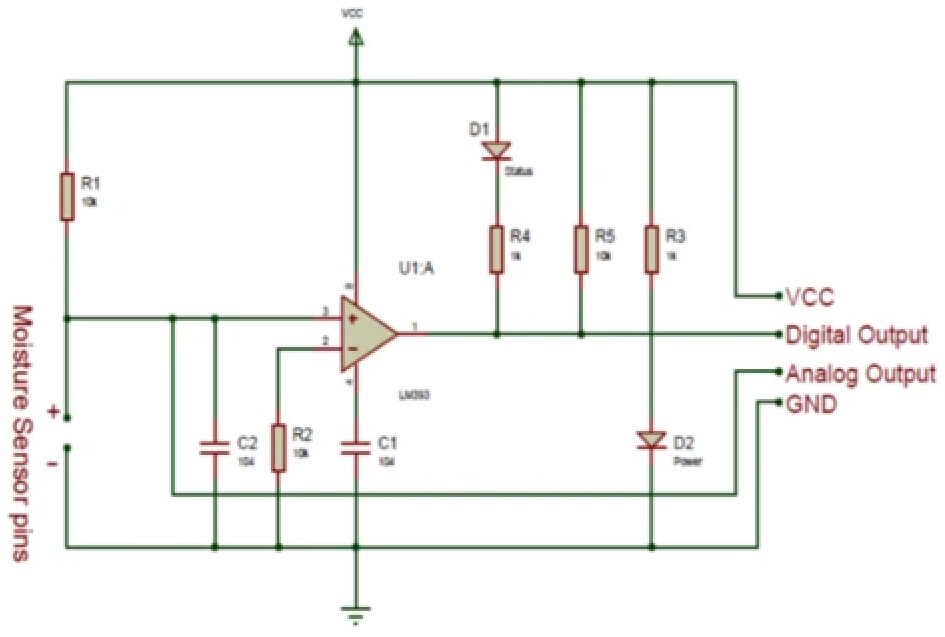
The module also contains a potentiometer, which will set the threshold value. This threshold value will be compared by the LM393 comparator. The output LED will light up and down according to this threshold value. For example, when you turn the screw clockwise to its limit, the pin will always return a 0. This is what will be returned when the soil moisture sensor is immersed in a glass of water.



LM393 Comparator IC is used as a voltage comparator in this Moisture sensor module. Pin 2 of LM393 is connected to Preset (10KΩ Pot) while pin 3 is connected to Moisture sensor pin. The comparator IC will compare the threshold voltage set using the preset (pin2) and the sensor pin (pin3).

Moisture sensor module consists of four pins i.e. VCC, GND, DO, AO. Digital out pin is connected to the output pin of LM393 comparator IC while the analog pin is connected to the Moisture sensor. The internal Circuit diagram of the Moisture sensor module is given below.

The following circuit will be helpful in understanding the working of a typical soil moisture sensor.



As you can see, one input of the comparator is connected to a 10KΩ Potentiometer while the other input is connected to a voltage divider network formed by a 10KΩ Resistor and the Soil Moisture Probe.

Using a Moisture sensor module with a microcontroller is very easy. Connect the Analog/Digital Output pin of the module to the Analog/Digital pin of Microcontroller. Connect VCC and GND pins to 5V and GND pins of the Microcontroller, after that insert the probes inside the soil. When there is more water present in the soil, it will conduct more electricity that means resistance will be low and the moisture level will be high.

Soil Moisture Sensor Using Raspberry Pi 4

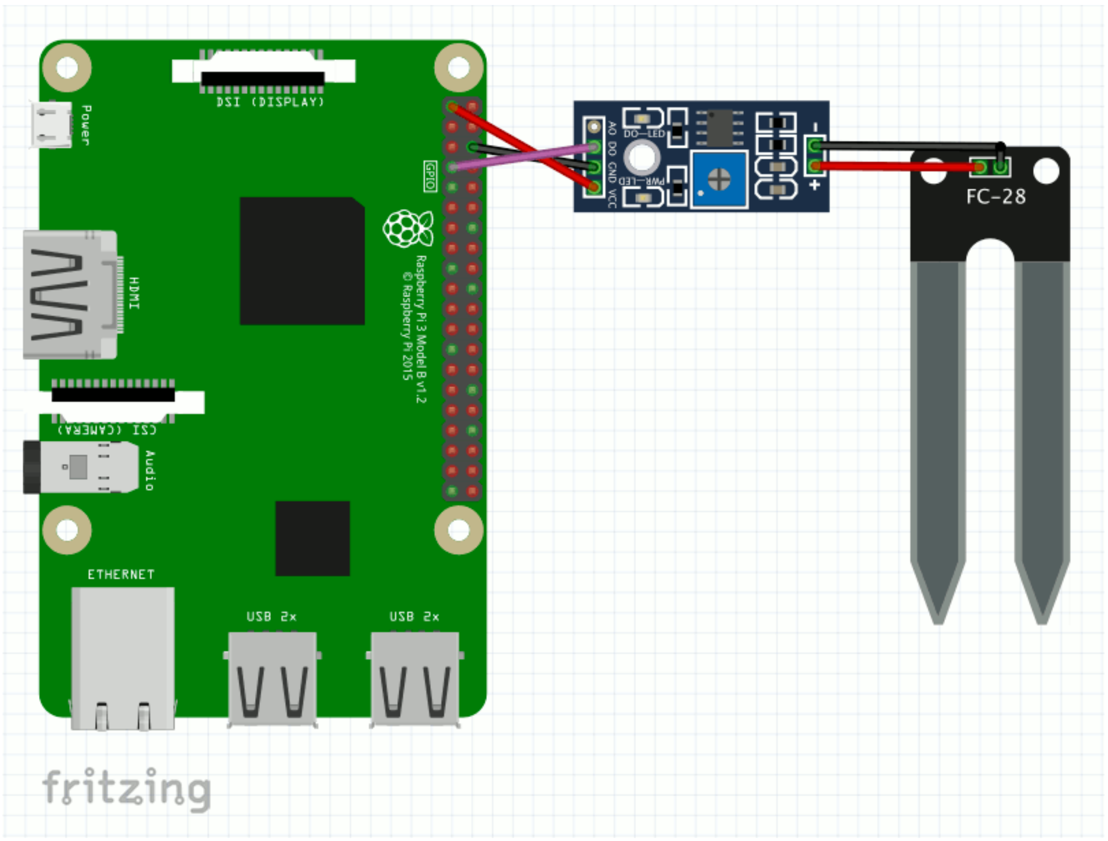
1. Reading from the DO pin of the soil moisture sensor with the Raspberry Pi

The VCC pin will need to be connected to a power source of 3.3V while the GND pin to the ground.

The **DO** pin can be connected to one of Raspberry Pi GPIO pins directly. However, the value that can be read is either a 0 or 1. I.e If you only want to be notified when your soil moisture has reached a threshold that you had set with the potentiometer, then you can read the **DO** pin directly from your Raspberry Pi.

Connecting the DO pin of the soil moisture sensor directly to your Raspberry Pi

The following diagram depicts how you can wire your soil moisture sensor directly to your Raspberry Pi:



We had connected the **DO** pin to the GPIO4 pin of our Raspberry Pi. So, the code will be

```
import RPi.GPIO as GPIO
import time

#GPIO SETUP
channel = 4
GPIO.setmode(GPIO.BCM)
GPIO.setup(channel, GPIO.IN)

while True:
     if GPIO.input(channel):
          # if input is LOW
          print("No Water Detected")
     else:
          # if input is HIGH
          print("Water Detected")
     time.sleep(1)

GPIO.cleanup()
```

In order for our Raspberry Pi to read from the **AO** pin, we need to use an anolog to digital converter like the MCP3008 chip. When connected to an analog to digital converter, our Raspberry Pi can get a wider range of values from the **AO** pin. Therefore, readings from this pin are more useful as compared to readings from the **DO** pin.

2. Reading the soil moisture sensor through a MCP3008 chip.

Since the AO pin can give a wider spectrum of readings, we will be able to get a percentage of how wet our soil is.

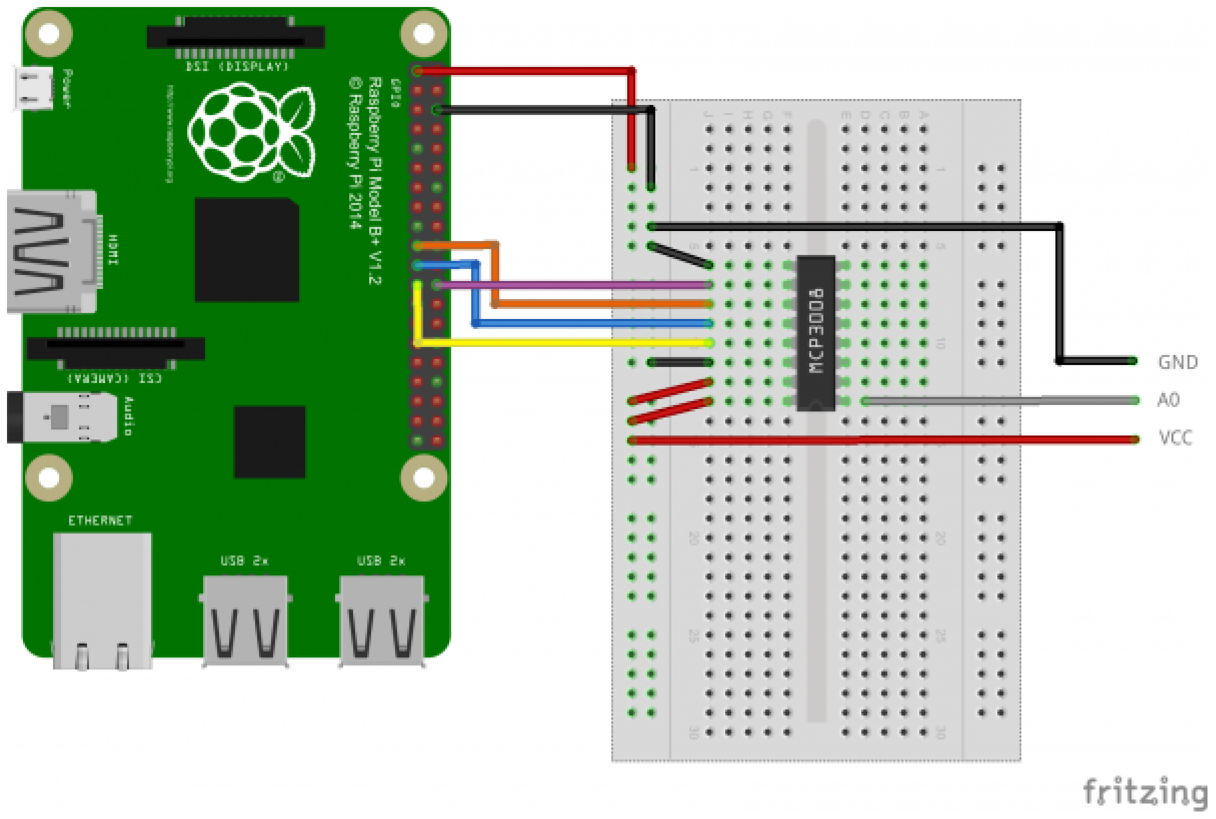
Connecting the soil moisture sensor to the Raspberry Pi through a MCP3008 chip.

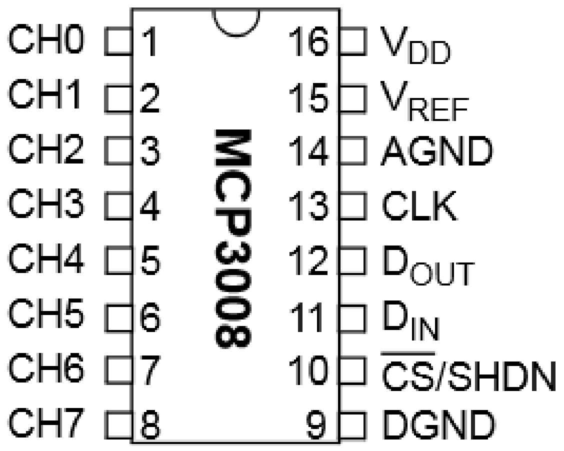
To enable the sensor analog mode, we'll need an analog pin to produce output. Which poses an issue, since the Raspberry Pi doesn't include an analog pin.

We'll be using an MCP3008 IC, a 10-bit, 8-channel ADC (analog to digital converter) to resolve the analog pin problem.
The MCP3008 uses the SPI bus protocol to receive analog input values from the Raspberry Pi.

It produces output values from a range of 0-1023 (Note: 0 represent OV and 1023 represents 3.3V)

Given that, this is how we can connecting the soil moisture sensor to the Raspberry Pi through a MCP3008 chip:





The 8 pins to the left take the analog inputs that we wish to read with our Raspberry Pi. You will connect the analog output of the sensor that you wish to read to one of these pins.

The 8 pins to the right are what we will connect to our Raspberry Pi:

- **VDD** (Pin 16) to **3.3V** pin on Raspberry Pi
- **VREF** (Pin 15) to **3.3V** pin on Raspberry Pi
- **AGND** (Pin 14) to **GND** pin on Raspberry Pi
- **CLK** (Pin 13) to **Pin 23/SCLK** on Raspberry Pi
- **DOUT** (Pin 12) to **Pin 21/MISO** on Raspberry Pi
- **DIN** (Pin 11) to **Pin 19/MOSI** on Raspberry Pi
- **CS** (Pin 10) to **Pin 24/CE0** on Raspberry Pi
- **DGND** (Pin 9) to **GND** on Raspberry Pi

Enabling the SPI Interface with the Raspberry Pi

Follow these steps to enable the SPI interface with the Raspberry Pi:

1. Start by launching the terminal and typing in the below command:
   sudo raspi-config
2. Navigate to the interfacing options.
3. Enable the SPI interface.
4. Reboot your Raspberry Pi.

Now you can install the spidev library if you have not already done so:

```
sudo apt-get install git python-dev
git clone git://github.com/doceme/py-spidev
cd py-spidev/
sudo python setup.py install
```

With the following script you can then address the sensor

```
import spidev
import time
import RPi.GPIO as GPIO

# create SPI object and start SPI connection
spi = spidev.SpiDev()
spi.open(0,0)

# Read MCP3008 data
def analogInput(adc_pin):
     spi.max_speed_hz = 1350000
     adc = spi.xfer2([1, (8 + adc_pin)<< 4, 0])
     data = ((adc[1] & 3) << 8) + adc[2]
     return data

while True:
     #Reading from CH0
     Output = analogInput(0)
     Output = interp(output, [0, 1023], [100, 0])
     Output = int(Output)
     print("------------------------------")
     print("Soil Moisture is  : ", Output)
     time.sleep(0.001)
```

Moisture Sensor with Raspberry Pi (Using MCP 3008 IC Analog to Digital)

Understanding the code :

Follow this Documentation to Understand the Spidev library.

```
adc = spi.xfer2([1,(8+channel)<<4,0])
data = ((adc[1]&3) << 8) + adc[2]
```

Line that starts with adc

Let's say you are looking at channel 1.

8+**1** = 9

9 << (bit shift to left) 4 (places) = 144 or (10010000 in binary)

If we look at page 19 of the datasheet for the MCP3008 table 5-1 shows that we need to send 1001 to read channel 1 in single ended mode. and that these should be the upper 4 bits of a byte.
This matches what the calculation gave us 10010000.

The rest of the command is just to send the number we calculated above to the SPI device and sequentially read the result back into adc.

Line that starts with data

Refer to page 21, figure 6-1 of the datasheet.
The ADC returns 3 bytes (0, 1 and 2)
Byte 0 we don't care about
Byte 1 contains only 2 bits of data from the ADC, the bits of interest are in bits 0 and 1
Byte 2 contains 8 bits of data.

To turn the data into a valid number for us to use we ignore all but the first 2 bits of byte 0 **(adc[1]&3)** using bit masking.
We shift that number 8 bits to the left (**<< 8**) and add the result from **adc[2]** to the number.

To summarise,
1. Calculate what to send to the ADC in order to get the correct channels data back
2. Send this to the ADC

3. Receive data from the ADC
4. Use bit masking and bit shifting operations to figure out what our result is.

Additional Explanation of spi.xfer2 (Refer to the Datasheet of MCP3008)

Lots of people have asked about the spi.xfer2 line. This sends 3 bytes to the device. The first byte is 1 which is equal to 00000001 in binary.

“8+channel” is 00001000 in binary (where channel is 0). “<<4” shifts those bits to the left which gives 10000000. The last byte is 0 which is 00000000 in binary.

So “spi.xfer2([1,(8+channel)<<4,0])” sends 00000001 10000000 00000000 to the device. The device then sends back 3 bytes in response. The “data=” line extracts 10 bits from that response and this represents the measurement.

Additional Information

The spidev library uses the SpiDev() method to communicate with the SPI device on the Raspberry Pi. The open() method enables communication, and the xfer2() method both sends and receives a message with the SPI device. The trick is in telling the MCP3008 that you want to take a sample reading, and then reading that value from the MCP3008. According to the MCP3008 specifications, to activate a reading the MCP must receive a three-byte message:

00000001 1ddd0000 xxxxxxxx

The first byte must be a 1, and the second byte uses three bits to indicate which channel to read. For channel 0, we’ll use the value 10000000. The value of the last byte doesn’t matter.

When the MCP3008 chip receives the command, it takes a sample reading from the designated analog channel, and converts the analog value into a 10-bit digital value based on the relationship of the voltage to the reference voltage applied to the Vref pin (pin 15). The resulting 10-bit digital value is sent back to the requesting host as a three-byte value:

???????? ?????0bb *bbbbbbbb*

where *bbbbbbbbbb* is the 10-bit value.

So, to send the data to activate a reading, we use the following line:

result = spi.xfer2([1, (8 + channel) << 4, 0])

This command sends three byte values down the communication line. The first byte is the 1 value. The second byte sets the high bit to a 1, adds the channel value, and then shifts the four-bit value to the high part of the byte. Thus, for channel 0, we send the value 10000000, or 128. The third byte is irrelevant, so we just send a 0 value.

The xfer2() method waits for a response from the MCP3008, and returns the value as a list datatype:

adcValue = ((result[1] & 3) << 8) + result[2]

You can use this to Convert the values into Voltage too.

```
def Volts(data):
  volts = (data * 3.3) / float(1023)
  # Round off to 2 decimal places
  volts = round(volts, 2)
  return volts
```

Some Important links Found during the process :

https://lastminuteengineers.com/soil-moisture-sensor-arduino-tutorial/

https://www.hnhcart.com/blogs/learn/soil-moisture-sensor-module

https://maker.pro/raspberry-pi/tutorial/interfacing-soil-moisture-sensor-with-raspberry-pi

https://tutorials-raspberrypi.com/measuring-soil-moisture-with-raspberry-pi/

https://iot4beginners.com/a-smart-home-sensor-raspberry-pi/

https://www.instructables.com/Measuring-Soil-Moisture-Using-Raspberry-Pi/

https://www.raspberrypi.org/forums/viewtopic.php?t=238890

https://www.techcoil.com/blog/how-to-read-soil-moisture-level-with-raspberry-pi-and-a-yl-69-fc-28-moisture-sensor/

https://www.raspberrypi.org/forums/viewtopic.php?t=235158

https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi#programming-for-spi

https://forum.dronebotworkshop.com/wp-content/uploads/wpforo/attachments/30/1079-SpiDevDoc.pdf

https://www.electroniclinic.com/soil-moisture-sensor-with-raspberry-pi-circuit-diagram-and-python-code/

https://www.raspberrypi.org/forums/viewtopic.php?t=199793#p1246000

https://www.raspberrypi-spy.co.uk/2013/10/analogue-sensors-on-the-raspberry-pi-using-an-mcp3008/

https://realpython.com/python-bitwise-operators/#bitmasks

https://www.informit.com/articles/article.aspx?p=2491680

https://www.youtube.com/watch?v=QHIWV904jNw

https://www.youtube.com/watch?v=IS_3aAW-wqc

https://www.youtube.com/watch?v=bHpnu1te0uU

https://www.pantechsolutions.net/plant-watering-system-based-on-soil-moisture-using-raspberry-pi

https://www.youtube.com/watch?v=iKGYbMD3NT8