# PLANT ENVIRONMENT MONITORING WITH ARDUINO

**Aim**: To monitor the environmental conditions of plant using arduino.

**Components required**:

* DHT-11
* Soil moisture sensor(SMT150T)
* ARDUINO UNO
* Servo motor
* Connecting wires
* 1N4007

**Apparatus**:

* A potted plant
* Water bottle
* Saline pipe

**Working:**

By using data logging techniques we’ll continuously monitor the values of the temperature and humidity.

We have to be clear about the temperatures and environmental conditions in which it can survive.

We’ll write a sketch in such a way that if the values of temperature, moisture and humidity of the soil in which the plant was plotted increases then ATMEGA328(MCU) present in ARDUINO UNO will send the required commands to servo motors such that those values are controlled to those levels that plant can survive.

**PROGRAM:**

#include DHT.h

Int safehumidity=70;

Int safetemp=43;

Int safemoisture=59;

Int humiditytemp=A0;

Int moisture=13;

Servo servo1;

Servo servo2;

Servo servo3;

Int servo1pos=0;

Int servo2pos=0;

Int servo3pos=0;

Void setup()

{

pinMode(humiditytemp,INPUT);

pinMode(moisture,INPUT);

pinMode(servo1,OUTPUT);

pinMode(servo2,OUTPUT);

pinMode(servo3,OUTPUT);

servo1.Attach(9);

servo2.Attach(10);

servo3.Attach(11);

Serial.begin(9600);

}

Void loop()

{

Int sensorval1=analogRead(humiditytemp);

Serial.println(sensorval1”-humidity,temperature”);

Int sensorval2=digitalRead(moisture);

Serial.println(sensorval2”-soil moisture”);

If(sensorval1>safehumidity||temperature)

{

Servo1pos= servo1pos++;

servo1.write(servo1pos);

servo2pos=servo2pos++;

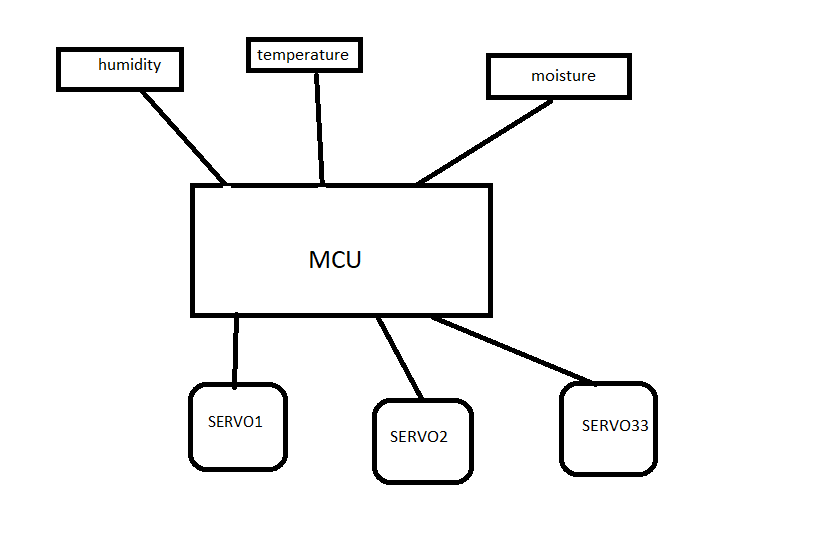
Servo2.write(servo2pos);

}

If (sensorval2>safemoisture)

{servo3pos=servo3pos++;  
servo3.write(servo3pos);

}

**Algorithm :**

**Details about components**:

[**DHT11**](http://www.adafruit.com/products/386)**:**

* Ultra low cost
* 3 to 5V power and I/O
* 2.5mA max current use during conversion (while requesting data)
* Good for 20-80% humidity readings with 5% accuracy
* Good for 0-50°C temperature readings ±2°C accuracy
* No more than 1 Hz sampling rate (once every second)
* Body size 15.5mm x 12mm x 5.5mm
* 4 pins with 0.1" spacing

**SOILMOISTURE SENSOR:**

The SparkFun Soil Moisture Sensor is a simple breakout for measuring the moisture in soil and similar materials. The soil moisture sensor is pretty straight forward to use. The two large exposed pads function as probes for the sensor, together acting as a variable resistor. The more water that is in the soil means the better the conductivity between the pads will be and will result in a lower resistance, and a higher SIG out.

To get the SparkFun Soil Moisture Sensor functioning all you will need is to connect the VCC and GND pins to your Arduino-based device (or compatible development board) and you will receive a SIG out which will depend on the amount of water in the soil. One commonly known issue with soil moisture senors is their short lifespan when exposed to a moist environment. To combat this, we’ve had the PCB coated in Gold Finishing (ENIG or Electroless Nickel Immersion Gold). We recommend either a simple 3-pin screw pin terminal or a 3-pin jumper wire assembly to be soldered onto the sensor for easy wiring.

**SERVO MOTOR**

A **servomotor** is a [rotary actuator](https://en.wikipedia.org/wiki/Rotary_actuator) or [linear actuator](https://en.wikipedia.org/wiki/Linear_actuator) that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Servomotors are not a specific class of motor although the term *servomotor* is often used to refer to a motor suitable for use in a [closed-loop control](https://en.wikipedia.org/wiki/Closed-loop_control) system.

Servomotors are used in applications such as [robotics](https://en.wikipedia.org/wiki/Robotics), [CNC machinery](https://en.wikipedia.org/wiki/CNC_machine) or [automated manufacturing](https://en.wikipedia.org/w/index.php?title=Automated_manufacturing&action=edit&redlink=1).

A servomotor is a [closed-loop](https://en.wikipedia.org/wiki/Closed-loop_controller) [servomechanism](https://en.wikipedia.org/wiki/Servomechanism) that uses position feedback to control its motion and final position. The input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft.

The motor is paired with some type of [encoder](https://en.wikipedia.org/wiki/Encoder) to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an [error signal](https://en.wikipedia.org/wiki/Error_signal) is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.

The very simplest servomotors use position-only sensing via a [potentiometer](https://en.wikipedia.org/wiki/Potentiometer) and [bang-bang control](https://en.wikipedia.org/wiki/Bang-bang_control) of their motor; the motor always rotates at full speed (or is stopped). This type of servomotor is not widely used in industrial [motion control](https://en.wikipedia.org/wiki/Motion_control), but it forms the basis of the simple and cheap [servos](https://en.wikipedia.org/wiki/Servo_(radio_control)) used for [radio-controlled models](https://en.wikipedia.org/wiki/Radio-controlled_model).

More sophisticated servomotors use optical [rotary encoders](https://en.wikipedia.org/wiki/Rotary_encoder) to measure the speed of the output shaft and a variable-speed drive to control the motor speed. Both of these enhancements, usually in combination with a [PID control](https://en.wikipedia.org/wiki/PID_controller) algorithm, allow the servomotor to be brought to its commanded position more quickly and more precisely, with less [overshooting](https://en.wikipedia.org/wiki/Overshoot_(signal)).

**Cost of estimation**:1000/-