

SMART IRRIGATION SYSTEM

Moses Makaka

INDEX

- Introduction
- Problem Statement
- Project Overview
- Hardware Specifications
- System analysis and design
- Circuit diagram
- Block Diagram
- Arduino code
- Server Side Code
- Database Schema
- Programming Languages Used
- User Interface
- Practical Implementation
- Result
- Conclusion
- Future Aspects
- Bibliography

INTRODUCTION

In Kenya, agriculture is the need of most of the Kenyans' livelihood and one of the main sources of income. Also it has a major impact on the economy of our country. But as per the current scenario, consumption of water in the processes of agriculture is increasing day by day that may lead to the problem of water scarcity. Nowadays, farmers are struggling hard in the agriculture departments and the task of irrigation is becoming quite difficult for them. The lack of regularity in their work and negligence lead to an unacceptable amount of wastage of water. Similarly, if they even forget to switch ON the irrigation system, it again leads to the damage of crops. The effects of the applied amount of irrigation water, irrigation frequency and water use are particularly important. To improve water efficiency there must be a proper irrigation scheduling strategy.

Looking at these aspects, we have developed an IOT based smart irrigation system which measures the moisture of the soil and automatically turns on or off the water supply system. In this project we have a soil moisture sensor which is used for sensing the moisture level, whether the soil is dry or wet. The moisture sensor is interfaced with nodeMCU, an open source IOT platform, which in turn sends the data to the webserver where some logic function decides the state of the motor.

IOT : The **Internet of Things (IoT)** is the network of physical devices embedded with electronics, software, sensors, actuators, and connectivity which enables these objects to connect and exchange data. The IoT allows objects to be sensed or controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit in addition to reduced human intervention. The ability to network embed the devices with limited CPU, memory and power resources means that IoT finds applications in nearly every field.

PROBLEM STATEMENT

Agriculture is one of those areas that consume a lot of water; while water is a very precious resource that must be utilized properly. Again, irrigation is a time consuming yet mandatory process and must be done on timely basis. However this has now become one of the main reasons of the damage of crops. In the case of traditional irrigation system water saving is not considered. Since, the water is irrigated directly in the land, plants under go high stress from variation in soil moisture, therefore plant appearance is reduced. The absence of automatic controlling of the system result in improper water management. At present there is emerging global water crisis where managing scarcity of water has become a serious job. So we want to design a Smart Irrigation System which is based on wireless control using nodeMCU that operates automatically by sensing the moisture content of the soil and turns ON/OFF the pump with the chosen intervention of farmer and hence save water.

PROJECT OVERVIEW

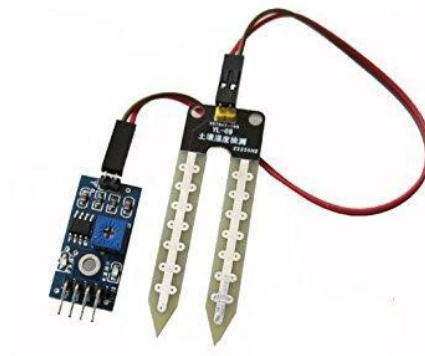
The Project 'Smart Irrigation System' is used for the optimizing use of water in agricultural field with the chosen intervention of farmer by using soil moisture sensor that senses the moisture content of the Soil using NodeMCU that turn ON/OFF the pump automatically according to the need of water for irrigation and hence helpful in saving water. This system is quite affordable and feasible. This system of irrigation is also helpful in the regions where there is scarcity of water and improves their sustainability and can also be adjusted according to the need of varieties of crop to be irrigated by varying the threshold values respectively.

Two modes namely manual and automated are made available to the user. In the manual mode, the farmer can remotely operate the motor and monitor the moisture sensor value while in automated mode, the server takes the decision for turning the motor ON/OFF according to the logical comparison of moisture sensor value with the threshold value.

HARDWARE SPECIFICATION

1. Soil Moisture Sensor:

The moisture sensor is used to measure the water content (moisture) of soil. The Soil Moisture Sensor uses capacitance to measure dielectric permittivity of the surrounding medium. In soil, dielectric permittivity is a function of the water content. The sensor creates a voltage proportional to the dielectric permittivity, and therefore the water content of the soil. The sensor averages the water content over the entire length of the sensor. There is a 2 cm zone of influence with respect to the flat surface of the sensor, but it has little or no sensitivity at the extreme edges. The Soil Moisture Sensor is used to measure the loss of moisture over time due to evaporation and plant uptake, evaluate optimum soil moisture contents for various species of plants, monitor soil moisture content to control irrigation in greenhouses and enhance bottle biology experiments.



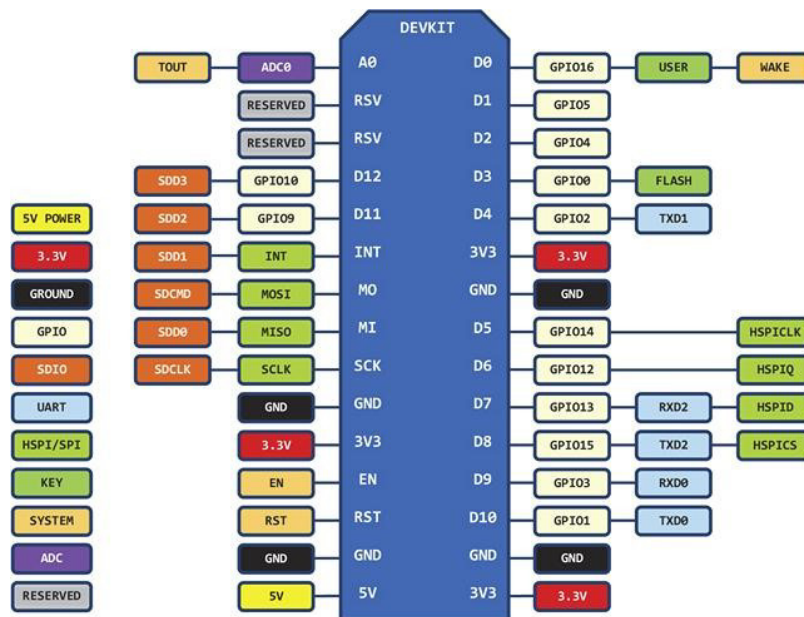
Product Features:

- Operating Voltage: 3.3V ~ 5V
- Sensing Probe Dimensions: 60x30mm
- Panel PCB Dimensions: 30 x 60mm
- On-board LM393 comparator
- On-board power indicator LED
- On-board digital switching indicator LED.

2. **NodeMCU:** NodeMCU is an eLua based firmware for the ESP8266 WiFi SOC from Espressif. The NodeMCU *firmware* is a companion project to the popular NodeMCU dev kits, ready-made open source development boards with ESP8266-12E chips.

Some of the specifications and features are:

- Wi-Fi Module – ESP-12E module similar to ESP-12 module but with 6 extra GPIOs.
- USB – micro USB port for power, programming and debugging
- Headers – 2x 2.54mm 15-pin header with access to GPIOs, SPI, UART, ADC, and power pins
- Misc – Reset and Flash buttons
- Power – 5V via micro USB port
- Dimensions – 49 x 24.5 x 13mm



NodeMCU Pinout

Pins of NodeMCU

NodeMCU provides access to the [GPIO](#) (General Purpose Input/Output) and for developing purposes below pin mapping table should be referenced.

IO index	ESP8266 pin	IO index	ESP8266 pin
0[*]	GPIO16	7	GPIO13
1	GPIO5	8	GPIO15
2	GPIO4	9	GPIO3
3	GPIO0	10	GPIO1
4	GPIO2	11	GPIO9
5	GPIO14	12	GPIO10
6	GPIO12		

3. DC Pump :

This is a small size Submersible Pump Motor which can be operated from a 2.5 ~ 6V power supply. It can take up to 120 liters per hour with very low current consumption of 220mA. Just connect tube pipe to the motor outlet, submerge it in water and power it by making sure that the water level is always higher than the motor.



Specifications:

Operating Voltage : 12V

Flow Rate : 80 ~ 120 L/H

Maximum Lift : 40 ~ 110 mm

Continuous Working Life : 500 hours

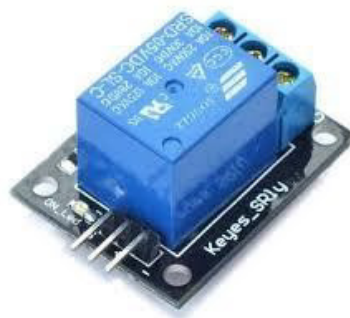
Driving Mode : DC

Material : Engineering Plastic

Outlet Outside Diameter : 7.5 mm

Outlet Inside Diameter : 5 mm

4. **Relay:** 5 x 12 Volts, 7 Amps PCB Mount SPDT Relay. Relays can be used to control both AC and DC appliance which comply with the output rating. Relays are used in all cases when you need to control high current AC / DC using sensor outputs or microcontroller outputs subject to the need of relay driver circuits(in some cases).
- 5 V relay can be used for home automation configured with micro-controller
 - This module complies with international safety standards
 - High voltage handling capacity of 220v ac with full protection □ Led indication of relay functioning



5. **Transistor**

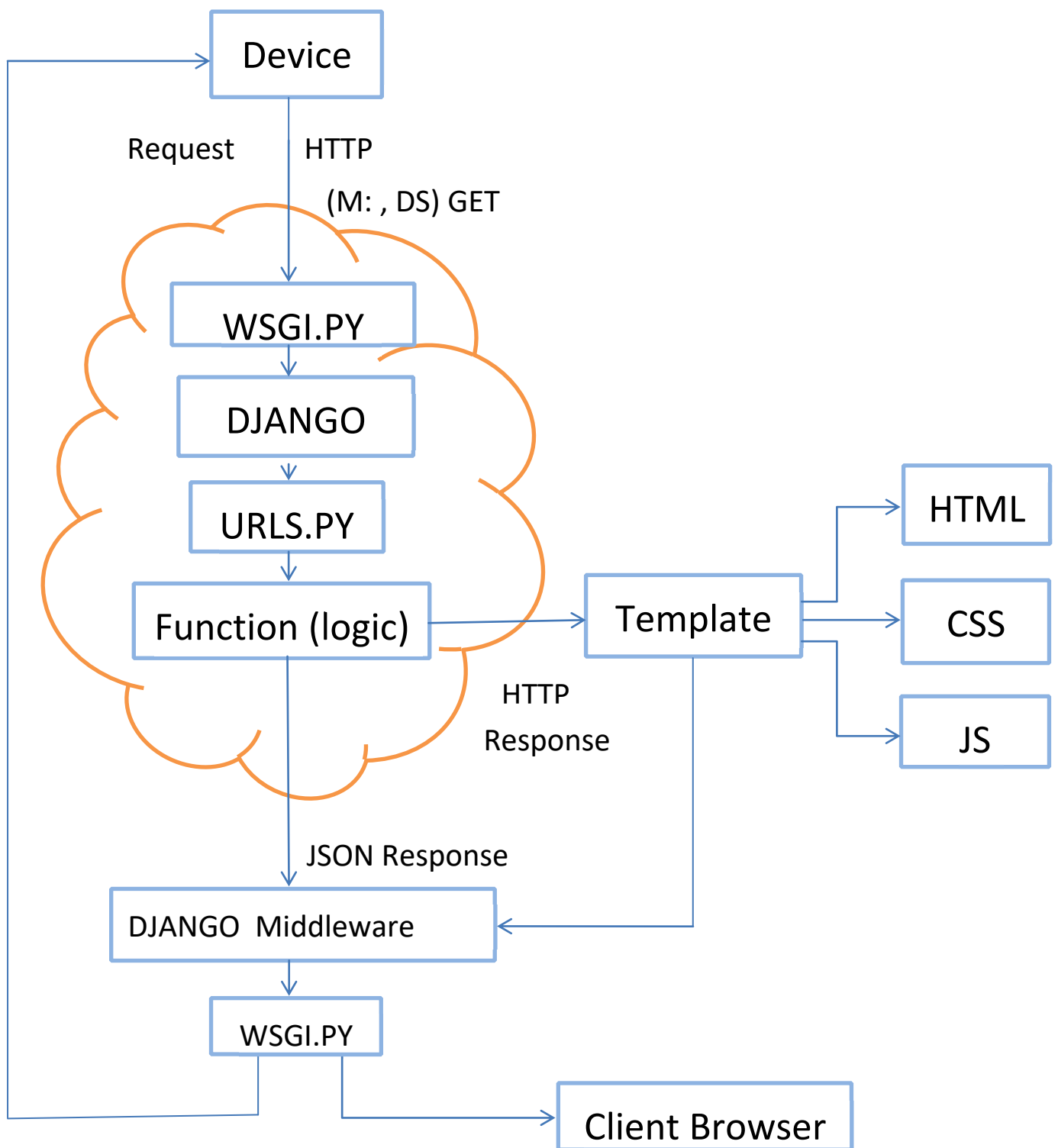
A **transistor** is a semiconductor device used to amplify or switch electronic signals and electrical power. It is composed of semiconductor material usually with at least three terminals for connection to an external circuit. A **transistor** is a device that regulates current or voltage flow and acts as a switch or gate for electronic signals. **Transistors** consist of three layers of a semiconductor material, each capable of carrying a current.



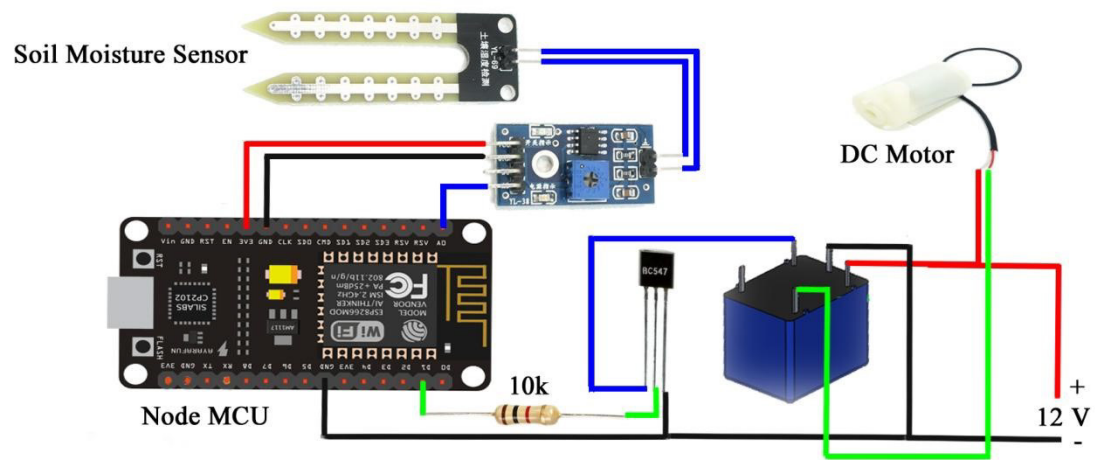
SYSTEM ANALYSIS AND DESIGN

Flowchart

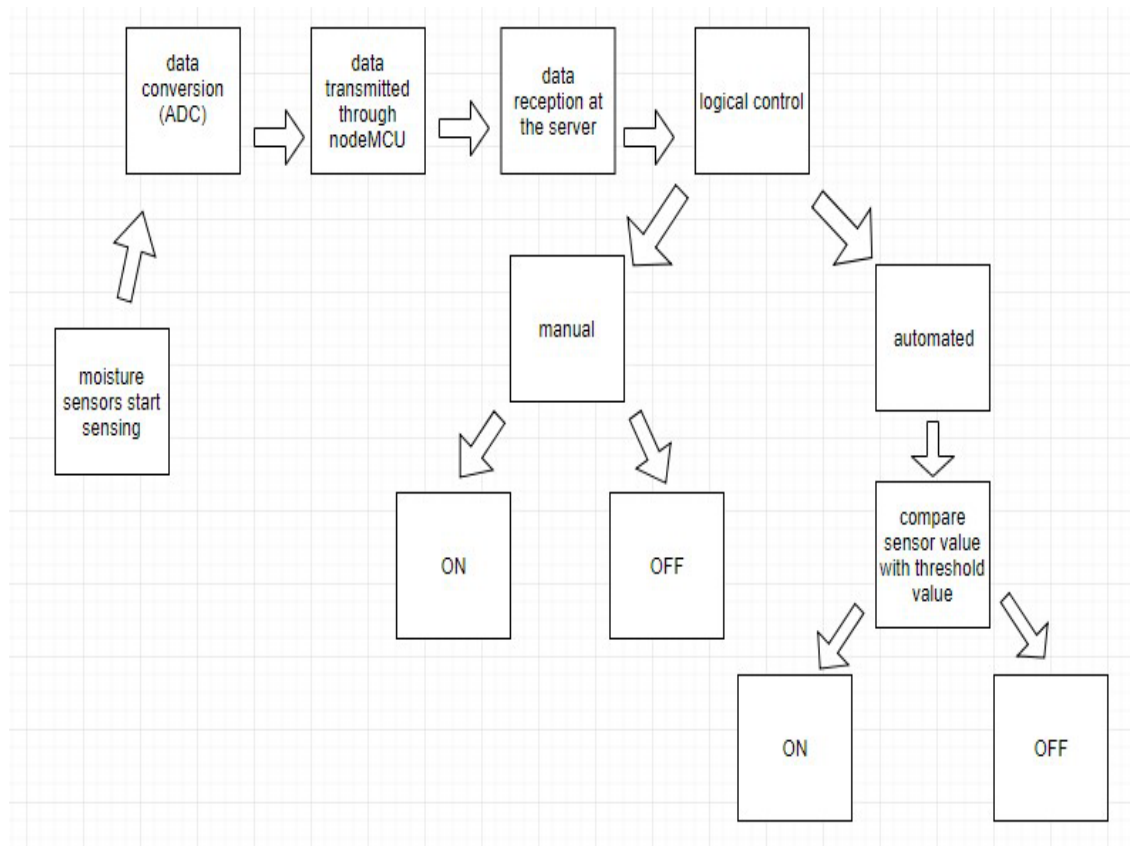
Client interface



CIRCUIT DIAGRAM



FLOW CHART



ARDUINO CODE

```
#include "ESP8266WiFi.h"

#include "ESP8266HTTPClient.h"

#include "ArduinoJson.h"

const int AnalogIn = A0;  const int Motor_Pin = D1;

const char* ssid = "JioFi3_1E8B14";  const char*
password = "9d1u40ntfbb";  const char* host =
"mohitkh7.pythonanywhere.com";

int value = 0;

int state;

void setup() {
Serial.begin(115200);

  delay(10);

  Serial.println();

  Serial.println();

  Serial.print("Connecting to ");

  Serial.println(ssid);

  WiFi.begin(ssid, password);      //WiFi Connection

  while (WiFi.status() != WL_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("");

  Serial.println("WiFi connected");

  Serial.println("IP Address:");

  Serial.println(WiFi.localIP());
```

```

    }

    void loop() {    delay(1000);
value = analogRead(AnalogIn);
state = digitalRead(Motor_Pin);

    Serial.print("Connecting to ");

    Serial.println(host);

    WiFiClient client;    //WiFiClient to create TCP connections
const int httpPort = 80;    if (!client.connect(host, httpPort)) {
Serial.println("Connection failed");    return;

    }

    String url = "/api/create/";
url += "?";    url +=
"moisture";    url += "=";
url += value;    url += "&";
url += "status";    url += "=";
url += state;

    Serial.print("Requesting URL:");

    Serial.println(url);

    client.print(String("GET ") + url + " HTTP/1.1\r\n" + "Host:" + host + "\r\n" + "Connection:
close\r\n\r\n");    int timeout = millis() + 5000;    while (client.available() == 0) {    if (timeout -
millis() < 0) {

        Serial.println(">>> Client Timeout !");
client.stop();

    }

    }
}

```

```

while (client.available()) {

    String line = client.readStringUntil('\r');

    Serial.print(line);

}

Serial.println();

HTTPClient http;

http.begin("http://mohitkh7.pythonanywhere.com/api/read/");

int httpCode = http.GET();


    if (httpCode > 0) {          // Get the request response payload

        String payload = http.getString(); //to do parsing

        Serial.println(payload);    const size_t bufferSize =

        JSON_OBJECT_SIZE(1) + 20;    DynamicJsonBuffer

        jsonBuffer(bufferSize);

        JsonObject& root = jsonBuffer.parseObject(payload);

        int set_state = root["set_state"];

        Serial.println(set_state);    if (set_state == 1) {

            digitalWrite(Motor_Pin, HIGH);

            Serial.println("Motor turned ON");

        }

    else {

        digitalWrite(Motor_Pin, LOW);

        Serial.println("Motor turned OFF");

    }

}

http.end();

Serial.println("Closing connection");

delay(60000);

}

```

SERVER SIDE CODE

urls.py

```
from
django.contrib
import admin

        from django.urls import path, include

        urlpatterns = [

            path('admin/', admin.site.urls),

path('monitor/', include('monitor.urls')),                path('api/',
include('monitor.urls_api')),

        ]
```


view_api.py

```
import json    from django.http import
JsonResponse      from django.core import
serializers

    # from rest_framework import viewsets

    # from .serializers import DeviceControlSerializer, DeviceStateSerializer,
DeviceStateListSerializer      from .models import DeviceState,
DeviceControl

    def check_device_moisture(request):          last =
last_status(request)          last_state_of_device =
json.loads(last.content.decode('utf-8'))          current_moisture =
last_state_of_device['moisture']

        state, lower_threshold_moisture, upper_threshold_moisture, mode =
get_device_detail()

        # It is following inverse value logic
        # To check whether mode is automated if not then skip

    print(mode)          if mode == 'automated':

if current_moisture < lower_threshold_moisture:

        turn_device(1)          print("Device
ON")          if current_moisture >
upper_threshold_moisture:

        turn_device(0)

print("Device OFF")

    """

    value == 1:ON ; 0:OFF

    """

    def turn_device(value):          obj =
DeviceControl.objects.get(id=1)

    obj.set_state = value          obj.save()

    def last_status(request):
```

```

        res = {}

        q = DeviceState.objects.last()

    res['moisture'] = q.moisture

    res['status'] = q.status        return

    JsonResponse(res)    def

all_status(request):

        query = DeviceState.objects.all()

    s = serializers.serialize("json", query)        return

    JsonResponse({'res': s})    def

create_status(request):

        moisture = request.GET['moisture']        status =

    request.GET['status']        device =

    DeviceState(moisture=moisture, status=status)        device.save()

        check_device_moisture(request)        res = {'status': 1,

    'message': 'device status updated'}        return

    JsonResponse(res)    def read_instruction(request):

        device_control = DeviceControl.objects.get(id=1)

        set_state = device_control.set_state        return

    JsonResponse({'set_state': set_state})    def

get_device_detail():

        try:

            device = DeviceControl.objects.first()

        lower_threshold = device.lower_threshold

        upper_threshold = device.upper_threshold        state

    = device.set_state        mode = device.mode

except:

```

```

        print("There is an Exception")








lower_threshold = 15                upper_threshold = 50










state = False                      mode = 'automated'        return

(state, lower_threshold, upper_threshold, mode)

```

DATABASE SCHEMA

DeviceState		
moisture	integer	 
status	varchar	 
time	datetime	 
 Add field		

DeviceControl		
set_state	varchar	 
lower_threshold	integer	 
upper_threshold	integer	 
mode	varchar	 
 Add field		

PROGRAMMING LANGUAGES USED

- C++ : It is used for the programming at the device end. The **Arduino language** is merely a set of C/C++ functions that can be called from the code.
- Python Django : It is used for the server backend code. **Django** is a high-level, free and open source web application framework, written in **Python**. A web framework is a set of components that helps you to develop websites
- SQL Lite : It is used for database. It is a standard language for storing, manipulating and retrieving data in databases. It stores the data i.e. the status of the motor as well as the values returned by the moisture sensor for future analysis.
- HTML : It is used to render webpage. It is the standard mark up language for creating web pages and web applications.
- CSS : It is used for designing and styling webpage using bootstrap as responsive framework. Bootstrap is one of the most widely used framework.
- JavaScript : It is used for adding functionality to front end. It is a "client-side" programming language. This means JavaScript scripts are read, interpreted and executed in the client i.e. Web browser.

1) Sensor technology:

This is basically the device end which senses the moisture value with the help of the soil moisture sensor which acts as the basis for the further action to be taken regarding the turning ON/OFF of the motor as per the logical analysis and comparison of the sensor and threshold value provided.

2) Cloud/Server :

It possesses the detailed records of the device, its current status and the history as well in the database. Each time the state of the motor or the value returned by the moisture sensor changes, it gets recorded for future use and data analysis.

3) User Interface:

This serves as the means for the user interaction. Also it displays all the essential information about the device such as current soil moisture value, current state of the motor and gives various options to the user regarding manual control of the motor and analysis of the history of the device.

4) Protocol:

The protocol used for interaction between the device, server and user interface is http. The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems.^[1] HTTP is the foundation of data communication for the World Wide Web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

USER INTERFACE

The user interface basically comprises of three sections:-

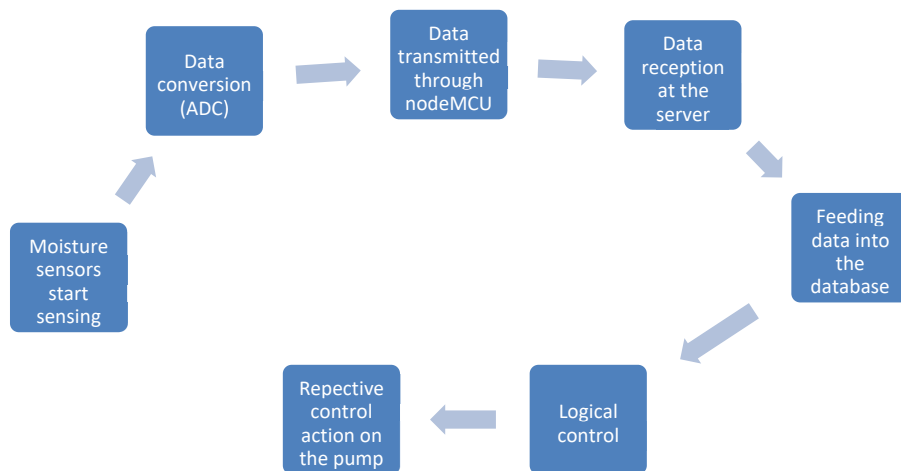
- 1. Status:** It shows the current status of the motor and the moisture content in the soil.

2. **Control:** This section is further divided into two parts hence providing various options for the controlling the motor.
 - a) Manual: It gives the option to the user to operate the motor as per his/her will regardless of the moisture content in the soil.
 - b) Automated: It gives the option to the user to rely on the feedback given, according to the moisture content in the soil, to drive the motor.
3. **Data log/History:** This section gives the user access to the history of the operations of the device through graphical as well as tabular data which may help in the analysis.

Smart Irrigation System			Home	History	Master Admin
History					
Moisture Value	Motor Status	Time			
192	OFF	April 27, 2018, 5:12 p.m.			
8	OFF	April 27, 2018, 2:13 p.m.			
300	OFF	April 27, 2018, 2:13 p.m.			
200	OFF	April 27, 2018, 2:13 p.m.			
2	OFF	April 27, 2018, 2:11 p.m.			
174	OFF	April 27, 2018, 2:10 p.m.			
574	OFF	April 27, 2018, 2:09 p.m.			
474	OFF	April 27, 2018, 2:08 p.m.			
374	OFF	April 27, 2018, 2:08 p.m.			
74	OFF	April 27, 2018, 2:07 p.m.			

PRACTICAL IMPLEMENTATION

This technology can be practically implemented using numerous sensors and creating a WSN (Wireless Sensor Network) with nodes which collect data from sensors and send it to the IoT gateway which is basically used to handle data from various sensors which may operate on different protocols and data formats. Also, to serve more number of users, they will be provided their specific user IDs. Also methods will be employed to reduce the power consumptions.



RESULTS

This technology is recommended for efficient automated irrigation systems and it may provide a valuable tool for conserving water planning and irrigation scheduling which is

extendable to other similar agricultural crops. Maximum absorption of the water by the plant is ensured by spreading the water uniformly using a motor. So there is minimal wastage of water. This system also allows controlling the amount of water delivered to the plants when it is needed based on types of plants by monitoring soil moisture and temperature. This project can be used in large agricultural area where human effort needs to be minimized. Many aspects of the system can be customized and fine tuned through software for a plant requirement

CONCLUSION

A Smart Irrigation and Monitoring System have been proposed so as to reduce wastage of water and to automate the irrigation structure of large areas of crops. The system mainly monitors the behavior of soil moisture and see how it contributes to evaluate the needs of water in a plant. The system uses IoT. It compares actual values obtained from sensors with a threshold value that has been fed to the system for analysis. The farmer he also can choose to turn on the water pump with a button click i.e. option of manual control is made available. Moreover, the system has a web app and is helpful if ever the farmer wants to see the statistical sensor data and assess the change in sensor readings throughout a time period.

In the present era, the farmers use irrigation technique through the manual control, in which the farmers irrigate the land at regular intervals. This process seems to consume more water and results in water wastage. Moreover in dry areas irrigation

becomes more difficult. Hence we require an automatic system that will precisely monitor and control the water requirements in the field. Installing Smart irrigation system saves time and ensures judicious usage of water.

FUTURE ASPECTS

The functionality of the system can be made dependent on one more aspect i.e. weather forecast. This will further help in making judicious use of water by saving it in situations when there is probability of rainfall. For this the current device can be installed with a GPS module or the coordinates(latitude and longitude) can be given as input by the user according to the location of the place where device is installed.

This location can be further used to acquire the weather forecast of that particular area and accordingly inform the user so that the irrigation is done keeping in mind, the possibility of rainfall.

In addition to this, another protocol namely mqtt can be used instead of http . MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It is being widely used due to its various advantages over http.

- It offers very less response time i.e,its is comparatively faster.
- It requires less bandwidth since the data transmission is done in the form of byte array.
- It reduces the power consumption i.e. it is beneficial for low lower battery operations.

Furthermore a solenoid valve can be used for varying the volume of water flow according to the requirement of the type of crop.

BIBLIOGRAPHY

- <https://www.ibm.com/developerworks>
- <http://www.instructables.com>
- <https://electronicsforu.com/nodemcu-development-board-iot>
- <http://www.circuitstoday.com/arduino-soil-moisture-sensor>

