

farmhouse manually. So towards this, GSM been used towards reporting the status about irrigation for farmer's mobile handset.

ICT Based Agricultural Monitoring

Sensor based paddy growth monitoring system [11] been developed by researchers Kait towards improving the rice productivity or yield. This system been considered to be cost effective as well as durable at outdoor operations. The architecture of the system is shown in Figure 1 where sensor nodes like Temperature, humidity, light, water level etc. are deployed in the field to gather the appropriate data and accordingly transmitted to the Base station using multi hop routing. The data are processed locally at the BS which are then sent to the remote server for further processing and analysis. After data analysis, message sent to farmers notifying the field conditions and providing suggestions.. Knowledge base can be created based on the data collected using Expert systems for further analysis. This system is not scalable and cannot be reused on other applications as it is not based on the concept of Machine to Machine Communications (M2M) which cover a large geographical area with many sensors deployed in the field.

Research [12] been carried out by employing Bluetooth Wireless Transmitter that sense soil moisture, temperature etc and accordingly send the data to the Base station (BS) which makes the decision towards irrigation decision based on field and time. The irrigation control unit which is responsible towards irrigating the field pertaining to operating the sprinkler would receive the control signal from the BS. This is based on water Requirement of the fields. Some researchers are also working towards Variable rate Sensor based Irrigation System.

Researchers Wall and King [13] developed an automated field specific irrigation system with soil moisture sensor and sprinkler valve controller. These systems do not take into consideration monitoring the water pollution in lakes or rivers and also do not consider M2M Communication concept.

Research been carried out in developing an intensive sensor based irrigation monitoring system which is scalable and self-organizing [14] The system architecture of their proposed sensor-based irrigation monitoring system is divided into two layers: bottom layer and upper layer as shown in Figures 2 and 3. Hierarchical sensor network are placed in bottom layer where nodes are placed in widely separated clusters. These nodes send the data to Base station (BS) which are connected Wireless LAN that holds the data logger software.[14]. Upper layer consist of five modules which are “acquisition module, network management module, alarm/network status display module and business module”. Real and non-real time data are collected using the data gathering modules from sensor network which are stored in database for decision/alert notifications. Alert notifications or displaying information to the end users are carried out by the

Alarm/Network status display module. The Alarm/Network status display module acts as an access point between end users and other modules/networks.

The condition of networks such as localizations, collision and network configuration are carried out by network management module. This system here developed have introduced the concept of M2M communication where water and energy conserved by using intelligent sensor network and efficient routing protocol Research also been done in Crop field assessment towards irrigations, applying fertilizers and pesticides. So accordingly a sensor network based field management system been developed where solar powered moisture nodes and low resolution camera deployed. The information from the two sources are captured for comparative assessment and analysis. Crop height, coverage and greenery information are sent through the sensor networks to the Base Station (BS) by

means of Self-contained, self-powered low resolution camera. In addition crop images are sent from these camera nodes by allocating the time. Lastly the cattle position and behavior can be observed by means of camera nodes.

Research [15] also been carried out by developing an automated irrigation system (A2S) which is based on sensor network. Wireless sensors are being employed for monitoring and controlling the agricultural fields. The management sub system controls the sensor network and accordingly provides service to farmer's by means of PDA. In this system, long distance communicated provided by means of WLAN between sensor network and server. Management subsystems consist of database, application and web server. Data from sensors are received by the application server which are stored in the database server. Web server is accessed by farmer's using PDA. In addition to the above mentioned research employing wireless sensors in agricultural monitoring, mobile phones also have been adopted in many rural areas by the farmers

[16]. In one of the research, information about the seeds is delivered to the farmer's in two ways. First is the periodic broadcast of seed information from web-portal through SMS on mobile phones of farmers depending upon the season. Second, farmer can query the system by sending names of the crops in SMS and our system will automatically reply with the available seed information of the crop. So this way farmer will be able to get information on their mobile phones at a reasonable cost as compared to expenses incurred in travelling to agricultural offices. Moreover, wastage in time is also an issue that will be solved through this system Low cost technological solutions are provided by sensor networks for PA towards in field crop management. These crop conditions/growth are monitored for a longer period of time, remotely make decision and accordingly evaluate the potential of new crops. With the data collected by sensors, database or knowledge base created. GUIs are integrated with these sensor networks in these monitoring systems.

Machine Learning in Agricultural Monitoring

The machine learning algorithm has various uses in the field of agriculture monitoring which are being discussed here.

In one of the **research [17]**, Machine learning been applied towards Grape cultivation. In here farmers are unable to identify the disease manually on the grapes. The disease on grapes is identifiable only after the infection which takes lot of time and also has adverse effect on the vineyards. So accordingly a monitoring system developed for grape cultivation where temperature, relative humidity and leaf wetness sensors are deployed in the vineyard. The data collected at regular intervals are sent using Zigbee module to the server. The server here employs the hidden markov model algorithm towards training the data sets pertaining to Temperature, relative humidity and leaf wetness for analysing the data towards predicting the chance of disease on grapes before getting infected. This information is sent as alert message via sms to the farmer and expert. The system employs machine learning in early and accurate detection of disease in grapes and suggests pesticide to protect the crop from disease and reduce manual disease detection efforts. Also this system can be helpful for farmer's towards giving information on schedule of fertilizer's, pesticide spraying, irrigation etc which would help in improving the quality and quantity of grapes.

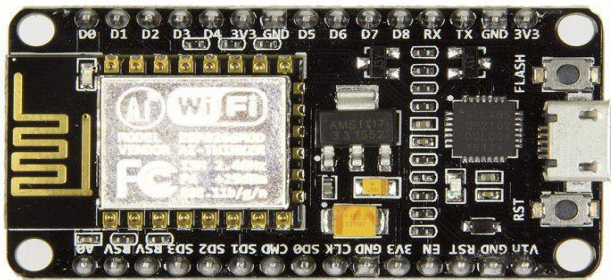
Extreme Machine Learning [8] been employed which is a simple, reliable and efficient single hidden layer feed forward neural networks. This methodology is based on weather factors and time series of soil moisture. Data sets are obtained from Dookie Applied Orchard in Victoria, Australia. Results have shown that future trend of soil moisture can be predicted accurately and accordingly decision support can be developed for irrigation scheduling. This method provides better accuracy compared to Support Vector machines which is the traditional and conventional soil moisture forecasting In another

Research [19], Random Forest Machine algorithm been developed for detection and classification of different. Grape diseases like Anthracnose, Powdery Mildew and Downy Mildew from the images collected under uncontrolled environment with random background. In here, the performance of Random forest been compared with Probabilistic Neural Network, Back propagation Neural Network and Support Vector Machine. Also performance of different texture features been studied. The proposed system achieves best classification accuracy of 86% using Random Forest and GLCM features for background separation and disease classification.

Crop Selection Method research [20] been developed towards selecting the sequence of crops planted over a season. Selection of Crops resolved based on prediction yield rate which is dependent on parameters like weather, soil type, water density, crop type etc. Crop, Sowing time, plantation days and predicted yield rate are given as input for the season in this method and accordingly sequence of crops given as output. The performance and accuracy of CSM is dependent on predicted value of influenced parameters.

COMPONENTS AND TECHNOLOGY USED

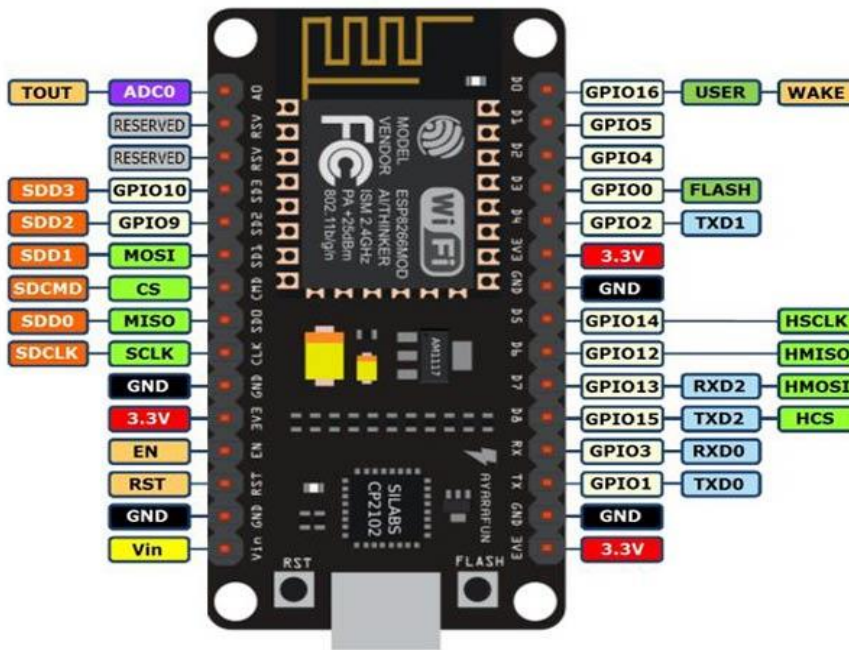
- **NodeMCU (ESP8266 Microcontroller)**



NodeMCU is an open source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits. Both the firmware and prototyping board designs are open source. The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.

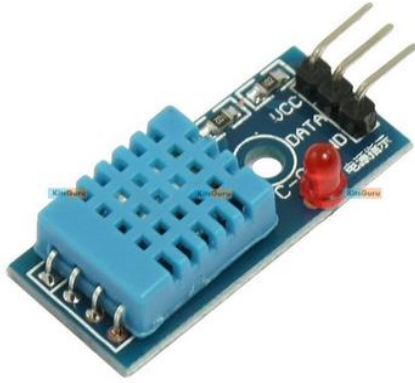
The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially was based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications.

- NODEMCU PIN CONFIGURATIONS



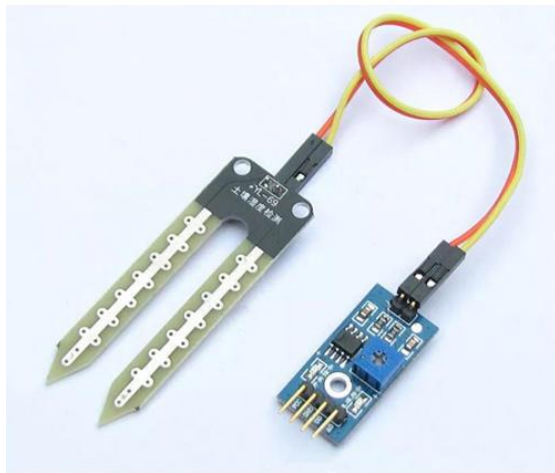
Pin Category	Name	Description
Power	Micro-USB, 3.3V, GND, Vin	Micro-USB: NodeMCU can be powered through the USB port 3.3V: Regulated 3.3V can be supplied to this pin to power the board GND: Ground pins Vin: External Power Supply
Control Pins	EN, RST	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.

- **DHT11 Sensor**



The digital temperature and humidity sensor DHT11 is a composite sensor that contains a calibrated digital signal output of temperature and humidity. The technology of a dedicated digital modules collection and the temperature and humidity sensing technology are applied to ensure that the product has high reliability and excellent long-term stability. The sensor includes a resistive sense of wet component and an NTC temperature measurement device, and is connected with a high-performance 8-bit microcontroller.

- **Soil moisture sensor**

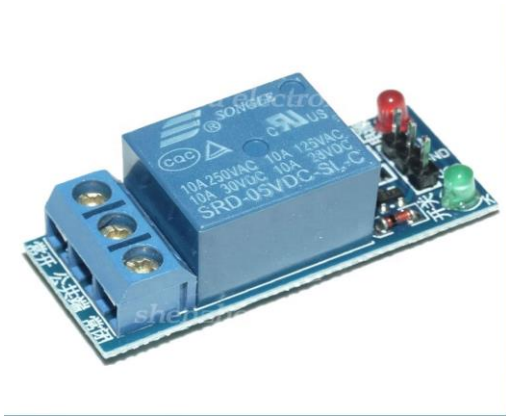


Soil moisture sensors measure the volumetric water content in soil. Since the direct gravimetric measurement of free soil moisture requires removing, drying, and weighing of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content.

The relation between the measured property and soil moisture must be calibrated and may vary depending on environmental factors such as soil type, temperature, or electric conductivity. Reflected microwave radiation is affected by the soil moisture and is used for remote sensing in hydrology and agriculture. Portable probe instruments can be used by farmers or gardeners.

Soil moisture sensors typically refer to sensors that estimate volumetric water content. Another class of sensors measure another property of moisture in soils called water potential; these sensors are usually referred to as soil water potential sensors and include tensiometers and gypsum blocks.

- **1- channel Relay Module**



A relay is an electrically operated device. It has a control system and (also called input circuit or input contactor) and controlled system (also called output circuit or output contactor). It is frequently used in automatic control circuit. To put it simply, it is an automatic switch to controlling a high-current circuit with a low-current signal.

The advantages of a relay lie in its lower inertia of the moving, stability, long-term reliability and small volume. It is widely adopted in devices of power protection, automation technology, sport, remote control, reconnaissance and communication, as well as in devices of electro mechanics and power electronics. Generally speaking, a relay contains an induction part which can reflect input variable like current, voltage, power, resistance, frequency, temperature, pressure, speed and light etc. It also contains an actuator module (output) which can energize or de-energize the connection of controlled circuit. There is an intermediary part between input part and output part that is used to coupling and isolate input current, as well as actuate the output. When the rated value of input (voltage, current and temperature etc.) is above the critical value, the controlled output circuit of relay will be energized or de-energized.

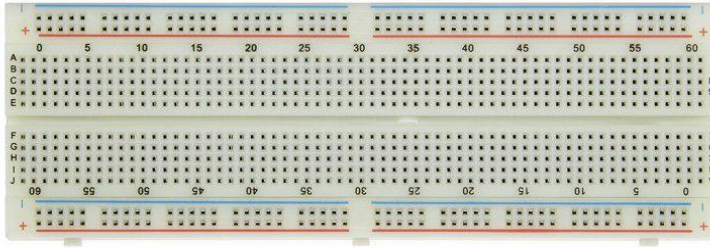
- **Motor Pump (mini DC 12V)**



A DC motor is any of a class of rotary electrical motors that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor.

DC motors were the first form of motor widely used, as they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight brushed motor used for portable power tools and appliances. Larger DC motors are currently used in propulsion of electric vehicles, elevator and hoists, and in drives for steel rolling mills. The advent of power electronics has made replacement of DC motors with AC motors possible in many applications.

- **Breadboard**



A **breadboard** is a construction base for prototyping of electronics. Originally the word referred to a literal bread board, a polished piece of wood used for slicing bread. In the 1970s the **solderless breadboard** (a.k.a. **plugboard**, a terminal array board) became available and nowadays the term "breadboard" is commonly used to refer to these.

Because the solderless breadboard does not require soldering, it is reusable. This makes it easy to use for creating temporary prototypes and experimenting with circuit design. For this reason, solderless breadboards are also popular with students and in technological education. Older breadboard types did not have this property. A stripboard (Veroboard) and similar prototyping printed circuit boards, which are used to build semi-permanent soldered prototypes or one-offs, cannot easily be reused. A variety of electronic systems may be prototyped by using breadboards, from small analog and digital circuits to complete central processing units (CPUs).

- **Connecting wires**



A jump wire (also known as jumper wire, or jumper) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment.

Jumper wires typically come in three versions: male-to-male, male-to-female and female-to-female. The difference between each is in the end point of the wire. Male ends have a pin protruding and can plug into things, while female ends do not and are used to plug things into. Male-to-male jumper wires are the most common and what you likely will use most often. When connecting two ports on a breadboard, a male-to-male wire is what you'll need.

- **Google Firebase Cloud**



Firebase is a Backend-as-a-Service — BaaS — that started as a YC11 startup and grew up into a next-generation app-development platform on Google Cloud Platform.

Realtime Database

Real-time data is the way of the future. Nothing compares to it. Most databases require you to make HTTP calls to get and sync your data. Most databases give you data only when you ask for it. When you connect your app to Firebase, you're not connecting through normal HTTP. You're connecting through a WebSocket. WebSockets are much, much faster than HTTP. You don't have to make individual WebSocket calls, because one socket connection is plenty. All of your data syncs automatically through that single WebSocket as fast as your client's network can carry it. Firebase sends you new data as soon as it's updated. When your client saves a change to the data, all connected clients receive the updated data almost instantly.

File Storage

Firebase Storage provides a simple way to save binary files — most often images, but it could be anything — to Google Cloud Storage directly from the client. Firebase Storage has its own system of security rules to protect your Cloud bucket from the masses, while granting detailed write privileges to your authenticated clients.

Hosting

Firebase includes an easy-to-use hosting service for all of your static files. It serves them from a global CDN with HTTP/2.

- **ARDUINO IDE**



The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

METHODOLOGY

The system designed consists briefly of two major components:

1. Artificial Intelligence to predict the amount of rainfall and the soil moisture content.
2. Implementation of system using Internet of Things.

- **Central Controller**

A central system is utilized to control the nodes, and to decide the time for which the pump affixed to any particular node is switched on. This implementation utilizes a Cloud API controller [4]. The controller utilizes a Python3 and JAVA script to execute all its functions. Support scripts are imported to the main script, and the control logic calls methods from the fortification scripts as and when required. The functions include communication with the nodes, data preprocessing and deriving inferences utilizing PLSR algorithms. Training is performed on a high-end computer, and the trained models are then deployed in the engenderment environment.

- **Distributed Nodes**

Each node is comprised of an ESP8266 [8] predicated microcontroller board (NodeMCU) and an array of analog sensors. These sensors are acclimated to fetch information about weather conditions, which is converted to digital 10-bit integers utilizing the Analog to Digital converters present on each NodeMCU. An algorithm converts this data to a 32-bit floating point value required by the inference model, on the NodeMCU itself. Each node contains adscititious support hardware to provide required electrical power to the NodeMCU and the sensors and to avail facile debugging in the event of failure of the Node. The puissance circuit consists of a 9V battery input given to a 3.3V regulator. The 3.3V output is then used to power the NodeMCU and the sensors [13]. A mundane Vcc line and GND line are habituated to supply power to the aforementioned components. Adscititiously, the 9V supply is directly provided to an L293D motor drive IC, which potencies the water pump predicated on the inputs it receives from the NodeMCU. The implementation may withal include one or more solar panel(s) to power the NodeMCU and the sensors, utilizing battery power only when the pump is required. The pre-processed data, along with some adscititious bytes used to identify each individual node, is sent to the central controller via the HTTP GET request.

- **THE NETWORK**

The nodes are connected to the central controller through a shared WiFi access point, over the IPv4 protocol [9]. A client-server architecture is utilized for communication between the nodes and the central controller. The nodes do not communicate with each other. This implementation utilizes a Cloud server on the central controller. The server elongates a URL in the form of a webhook. The nodes connect to the server via this webhook to send data. The HTTP GET request is utilized to transmit the sensor data after post-calculation to the server [14]. The default replication of the server conveys two messages to the node that sent the request: 1. if the pump annexed to the particular node has to be switched on or not, and 2. the duration for which the pump has to be switched on. The duration is calculated utilizing the output of the regression algorithm. Each node sends a request to the server at an interval of 30 minutes. The network is thus essentially a wireless Local Area Network. Since each node works independent of other nodes, incidental failure of a node does not result in the collapse of the entire system. Supplementally, isolation of the failed node becomes facile. The IPv4 Address of the central controller is hard-coded into each

NodeMCU. The central controller is assigned a static IPv4 address utilizing the access point settings. The port utilized for communication is set to 8080 for this implementation.

- **Rainfall Prediction using Artificial Intelligence**

The rainfall presage involved a two-phase solution:

- Prediction of Probability of Rainfall in the next 30 minutes
- Estimation of the Amplitude of Rainfall

The first phase is to avail the network to realize whether there is a chance of rainfall to occur in the next 30 minutes or not and the contrivance keeps checking the status at customary intervals of time when it is switched on periodically. This is achieved by soothsaying the probability of rainfall to occur in the next 30 minutes. The system is made power efficient by not perpetually keeping it on for checking the status of rainfall rather checks at customary time intervals itself. The amplitude of rainfall depends on multiple parameters including mean of temperature, pressure, maximum and minimum sultriness as well as the mean dew associated with the air [15]. The dataset used is accountable for local areas and regions since it contained parameters which can be generalized to presage the rainfall and had desultory values of all the parameters. The data used is taken from the rainfall data available on the Regime of India Portal for local regions. The initial deployment of the model took account of the data provided for the state of Punjab primarily for wheat crop. The data was divided into three sets, namely, training, validation and testing, with a percentage of 70, 15 and 15 respectively. The values of the dataset were facilely accountable with Arbitrary Forest Relegation and Regression. The rainfall estimation involves many parameters highly dynamic and prone to transmutations in authentic time and ergo it made an artificial neural network [16] and support vector machines inefficient comparatively [17]. The conception is to design a deployable model and ergo desultory forest regression proved to be more adaptive to dynamic genuine time data and gives proximately precise results for weather presage needed to determine the whether it would rain or not [5]. Once it has been determined that the rainfall would occur or not the next challenge is to presage the amplitude of rainfall which would occur. This majorly deals with the historic precipitation data available for different regions and the model has to acclimate to a plethora of such historic data spread over a wide duration. For this, we took the historic data available for rainfall for different states provided by the Regime of India for research purposes. The rainfall data involved non-linear patterns and has varied intensities associated with it. The system is designed to acclimate to any state and withal updates the dataset with the incipient rainfall values which is being recorded with the avail of the Raspberry Pi. The overall implementation of the system makes it highly dynamic and more precise as well as adaptable to any region with onset of time of deployment. The rainfall estimation in India involved monthly and yearly patterns which were studied through utilization of Autoregressive Integrated Moving Average Time Series popularly kenned as the ARIMA Model utilized for rainfall estimation from historic data [10]. The results were found to be as expected with variations in intensity of rainfall and non-linear patterns visually examined uniquely for each state.

The dataset used is described as follows:

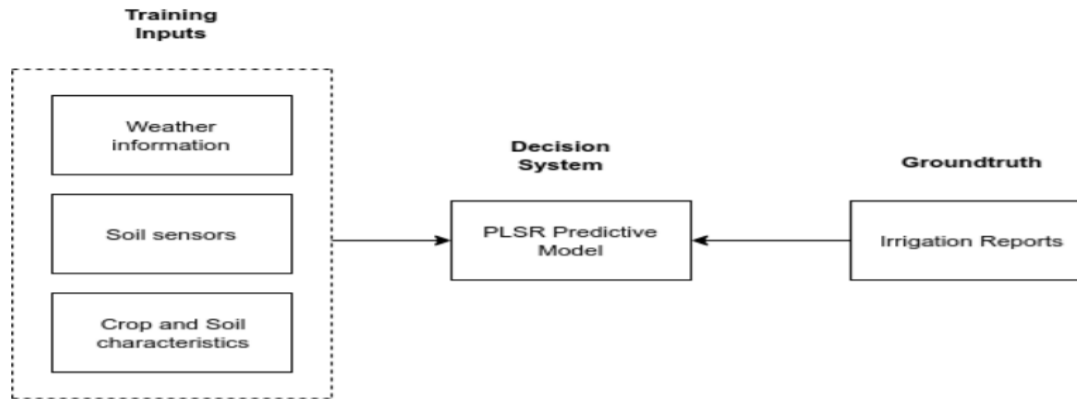
Serial	Description	Observation
1	Mean Temperature (monthly)	Celsius
2	Mean Dew (monthly)	Range of Temperature
3	Mean Pressure (monthly)	Millibar
4	Minimum Humidity	Gram per cubic metre
5	Maximum Humidity	Gram per cubic metre
6	Maximum Temperature	Celsius
7	Minimum Temperature	Celsius
8	Maximum Dewpoint	Celsius
9	Minimum Dewpoint	Celsius
10	Maximum Pressure	Millibar
11	Minimum Pressure	Millibar

The details of the rainfall patterns were determined through the time series [18] and the probability of the rainfall was determined already combining both of which we get the desired output.

- **Estimation of Soil Moisture Content required**

The decision system takes into account the water level content required by the crop to calculate its irrigation requisites. In traditional applications, these decisions are taken by the farmer predicated on their experience or an expert agricultural expert. Information from sundry sources is accumulated; these include weather statistics, crop and soil properties, moisture content data accumulated from soil sensors deployed in the fields, and moisture influencing factors such as evapotranspiration [11]. An agricultural expert analyses this data to decide an estimate of the quantity of water that the crop would require on a particular day. Predicated on this conception, our autonomous irrigation system is developed. We accumulate historical data which consists of decisions taken by an agricultural expert for the water content required by a crop predicated on available information and current crop condition. The accumulated information is analysed utilizing machine learning to presage the water requisites for the crop in authentic-time. Decision history by an agricultural expert is utilized to evaluate our system's performance. The decisions taken by the machine learning model are evaluated against those taken by the agricultural expert. The machine learning system is to be trained with historical data and decision reports of the agricultural expert, taking

into account the needed water level requisite of the particular crop and sundry authentic-time factors mentioned anteriorly. The aim of the machine learning model is to be as proximate as possible to the decisions taken by an agronomist, which are utilized as ground truth for evaluation. We tested sundry machine learning models to achieve the best performance with the minimum indispensable computation required. Figure illustrates a schematic representation of the machine learning system.



The machine learning predicated decision system, when trained with precise data, can provide a precise estimate of the irrigation requisites of the crop in consideration, given the required authentic-time information including the crop and soil condition. Table 2 shows the set of possible factors to be considered as input to the decision model. Evapotranspiration is derived from the words “evaporation” and “transpiration”. It refers to the process of moisture eluding from the soil and crop to the atmosphere. Scrutinizing the evapotranspiration rates of a crop is essential to estimating its irrigation requisites [19].

The water to be irrigated to the crop is the magnitude of water level (moisture content) required to compensate for the evapotranspiration moisture loss from the field. Hence, ascertaining the evapotranspiration of a crop field is a major deciding factor in estimating the irrigation requisites of the crop. The FAO Penman-Monteith formula can be used to calculate the reference crop evapotranspiration (ET₀) on a daily basis, using information from weather stations or alternatively weather sensors [20]. The crop evapotranspiration can be calculated under standard conditions using the single crop coefficient formula given below:

$$ET_c = K_c \cdot ET_0$$

Here K_c denotes the crop coefficient. It depends on various factors such as the crop type, crop evaporation, and crop growth stage and weather information.

We’ve utilized the Partial Least Square Regression (PLSR) [21] algorithm to design the machine learning decision model. It is a statistical algorithm that identifies fundamental cognations between input and output variables. Predictor (input) variables, denoted by X , are defined as the visually examined variables that are quantified for providing input to the decision model. Replication variables, denoted by Y , are the outputs that must be estimated provided the input.

We utilize the PLSR technique among other regression algorithms since it efficiently tackles cases when the number of inputs is much higher than the number of output variables, the outputs are strepitous and there subsists a high probability of having multicollinearity among the input variables. The multicollinearity quandary occurs when input variables are highly correlated, due to redundancy between meteorological factors and soil sensors data. We ascertain that all of these factors appear in our irrigation decision system.

The training set $\{X, Y\}$ of S samples is used to train the PLSR model. It comprises of the predictor input matrix $X = [x_1, \dots, x_i, \dots, x_S]^T$ and the response output matrix $Y = [y_1, \dots, y_i, \dots, y_S]^T$. Here x_i is a vector of N elements which contain all the soil sensor and weather data measured at a particular day i . Since in this model, the output is only the time of irrigation recommended for that day, y_i is a scalar, containing the variable to be estimated at a given day i .

The trained PLSR model takes the predictor matrix as input which contains the soil and weather information for a particular day, and predicts the soil moisture percentage required for the crop. This soil moisture percentage helps us to calculate the minutes of irrigation required as a function of the area of the crop field and the power of the electric motor being used.

PARTIAL LEAST SQUARE REGRESSION (PLSR) ALGORITHM (AI ALGORITHM USED IN OUR PROJECT):-

The goal of pls regression is to predict Y from X and to describe their common structure. When Y is a vector and X is full rank, this goal could be accomplished using ordinary multiple regression. When the number of predictors is large compared to the number of observations, X is likely to be singular and the regression approach is no longer feasible (i.e., because of multicollinearity). Several approaches have been developed to cope with this problem. One approach is to eliminate some predictors (e.g., using stepwise methods) another one, called principal component regression, is to perform a principal component analysis (pca) of the X matrix and then use the principal components of X as regressors on Y . The orthogonality of the principal components eliminates the multicollinearity problem. But, the problem of choosing an optimum subset of predictors remains. A possible strategy is to keep only a few of the first components. But they are chosen to explain X rather than Y , and so, nothing guarantees that the principal components, which “explain” X , are relevant for Y . By contrast, pls regression finds components from X that are also relevant for Y . Specifically, pls regression searches for a set of components (called latent vectors) that performs a simultaneous decomposition of X and Y with the constraint that these components explain as much as possible of the covariance between X and Y . This step generalizes pca. It is followed by a regression step where the decomposition of X is used to predict Y .

Simultaneous decomposition of predictors and dependent variables

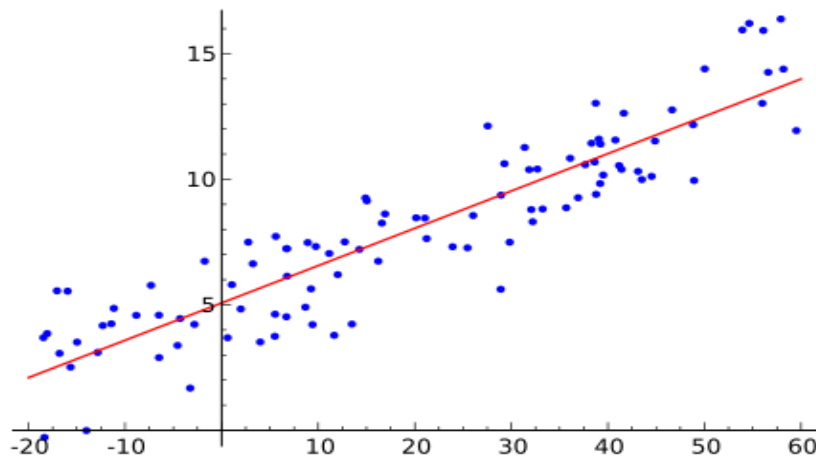
Pls regression decomposes both X and Y as a product of a common set of orthogonal factors and a set of specific loadings. So, the independent variables are decomposed as $X = TPT^T$ with $TT^T = I$ with I being the identity matrix (some variations of the technique do not require T to have unit norms). By analogy with pca T is called the score matrix, and P the loading matrix (in pls regression the loadings are not orthogonal). Likewise, Y is estimated as $Yb = TBCT$ where B is a diagonal matrix with the “regression weights” as diagonal elements (see below for more details on these weights). The columns of T are the latent vectors. When their number is equal to the rank of X , they perform an exact decomposition of X . Note, however, that they only estimate Y . (i.e., in general Yb is not equal to Y).

Pls regression and covariance

The latent vectors could be chosen in a lot of different ways. In fact in the previous formulation, any set of orthogonal vectors spanning the column space of X could be used to play the role of T . In order to specify T , additional conditions are required. For pls regression this amounts to finding two sets of weights w and c in order to create (respectively) a linear combination of the columns of X and Y such that their covariance is maximum.

Specifically, the goal is to obtain a first pair of vectors $t = Xw$ and $u = Yc$ with the constraints that $w^T w = 1$, $t^T t = 1$ and $t^T u$ to be maximal. When the first latent vector is found, it is subtracted from both X and Y and the procedure is re-iterated until X becomes a null matrix (see the algorithm section for more).

A pls regression algorithm



The properties of pls regression can be analyzed from a sketch of the original algorithm. The first step is to create two matrices: $E = X$ and $F = Y$. These matrices are then column centered and normalized (i.e., transformed into Zscores). The sum of squares of these matrices are denoted SSX and SSY . Before starting the iteration process, the vector u is initialized with random 2 values. (in what follows the symbol \propto means “to normalize the result of the operation”).

Step 1. $w \propto E^T u$ (estimate X weights).

Step 2. $t \propto Ew$ (estimate X factor scores).

Step 3. $c \propto F^T t$ (estimate Y weights).

Step 4. $u = Fc$ (estimate Y scores).

If t has not converged, then go to Step 1, if t has converged, then compute the value of b which is used to predict Y from t as $b = t^T u$, and compute the factor loadings for X as $p = E^T t$. Now subtract (i.e., partial out) the effect of t from both E and F as follows $E = E - tp^T$ and $F = F - btc^T$. The vectors t , u , w , c , and p are then stored in the corresponding matrices, and the scalar b is stored as a diagonal element of B . The sum of squares of X (respectively Y) explained by the latent vector is computed as $p^T p$ (respectively b^2), and the proportion of variance explained is obtained by dividing the explained sum of squares by the corresponding total sum of squares (i.e., SSX and SSY). If E is a null matrix, then the whole set of latent vectors has been found, otherwise the procedure can be reiterated from Step 1 on.

FORMULA USED

With one independent variable, we may write the regression equation as:

$$Y = a + bX + e$$

Where Y is an observed score on the dependent variable, a is the intercept, b is the slope, X is the observed score on the independent variable, and e is an error or residual.

We can extend this to any number of independent variables:

$$Y = a + b_1 X_1 + b_2 X_2 + \dots + b_k X_k + e_{(3.1)}$$

Note that we have k independent variables and a slope for each. We still have one error and one intercept. Again we want to choose the estimates of a and b so as to minimize the sum of squared errors of prediction. The prediction equation is:

$$Y' = a + b_1 X_1 + b_2 X_2 + \dots + b_k X_k \quad (3.2)$$

Finding the values of b (the slopes) is tricky for $k > 2$ independent variables, and you really need matrix algebra to see the computations. It's simpler for $k=2$ IVs, which we will discuss here. But the basic ideas are the same no matter how many independent variables you have. If you understand the meaning of the slopes with two independent variables, you will likely be good no matter how many you have.

For the one variable case, the calculation of b and a was:

$$b = \frac{\sum xy}{\sum x^2}$$

$$a = \bar{Y} - b\bar{X}$$

For the two variable case:

$$b_1 = \frac{(\sum x_2^2)(\sum x_1 y) - (\sum x_1 x_2)(\sum x_2 y)}{(\sum x_1^2)(\sum x_2^2) - (\sum x_1 x_2)^2}$$

and

$$b_2 = \frac{(\sum x_1^2)(\sum x_2 y) - (\sum x_1 x_2)(\sum x_1 y)}{(\sum x_1^2)(\sum x_2^2) - (\sum x_1 x_2)^2}$$

At this point, you should notice that all the terms from the one variable case appear in the two variable case. In the two variable case, the other X variable also appears in the equation. For example, X_2 appears in the equation for b_1 . Note that terms corresponding to the variance of both X variables occur in the slopes. Also note that a term corresponding to the covariance of X_1 and X_2 (sum of deviation cross-products) also appears in the formula for the slope.

The equation for a with two independent variables is:

$$a = \bar{Y} - b_1 \bar{X}_1 - b_2 \bar{X}_2$$

This equation is a straight-forward generalization of the case for one independent variable.

EQUATION USED TO CALCULATE SOIL MOISTURE REQUIRED BY THE CROP AT A GIVEN INSTANCE:

The FAO Penman-Monteith method to estimate ETo

$$ET_o = \frac{0.408\Delta (R_n - G) + \gamma \frac{900}{T + 273} u_2 (e_s - e_a)}{\Delta + \gamma(1 + 0.34 u_2)}$$

The FAO Penman-Monteith method to estimate ETo can be derived [Eq. 1]:

Where,

ET_o = reference evapotranspiration, mm day⁻¹;

R_n = net radiation at the crop surface, MJ m⁻² d⁻¹;

G = soil heat flux density, MJ m⁻² d⁻¹;

T = mean daily air temperature at 2 m height, °C;

u_2 = wind speed at 2 m height, m s⁻¹;

e_s = saturation vapor pressure, kPa;

e_a = actual vapor pressure, kPa;

$e_s - e_a$ = saturation vapor pressure deficit, kPa;

Δ = slope of the vapor pressure curve, kPa °C⁻¹;

γ = psychrometric constant, kPa °C⁻¹.

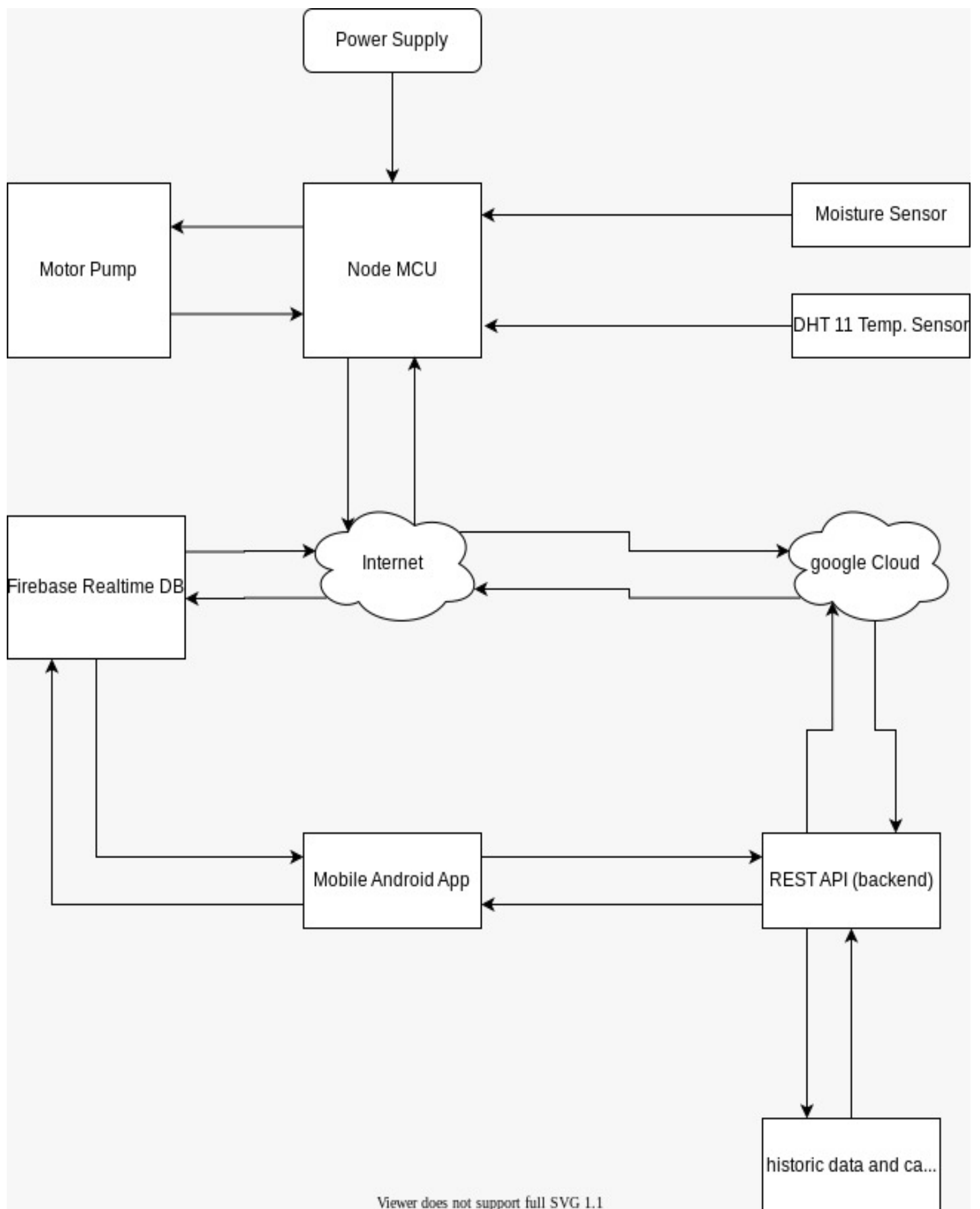
Tabulated K_c values

The below given Table lists typical values for K_{c ini}, K_{c mid} and K_{c end} for various agricultural crops. The coefficients presented are organized by group type (i.e., small vegetables, legumes, cereals, etc.) to assist in locating the crop in the table and to aid in comparing crops within the same group. There is usually close similarity in the coefficients among the members of the same crop group, as the plant height, leaf area, ground coverage and water management are normally similar.

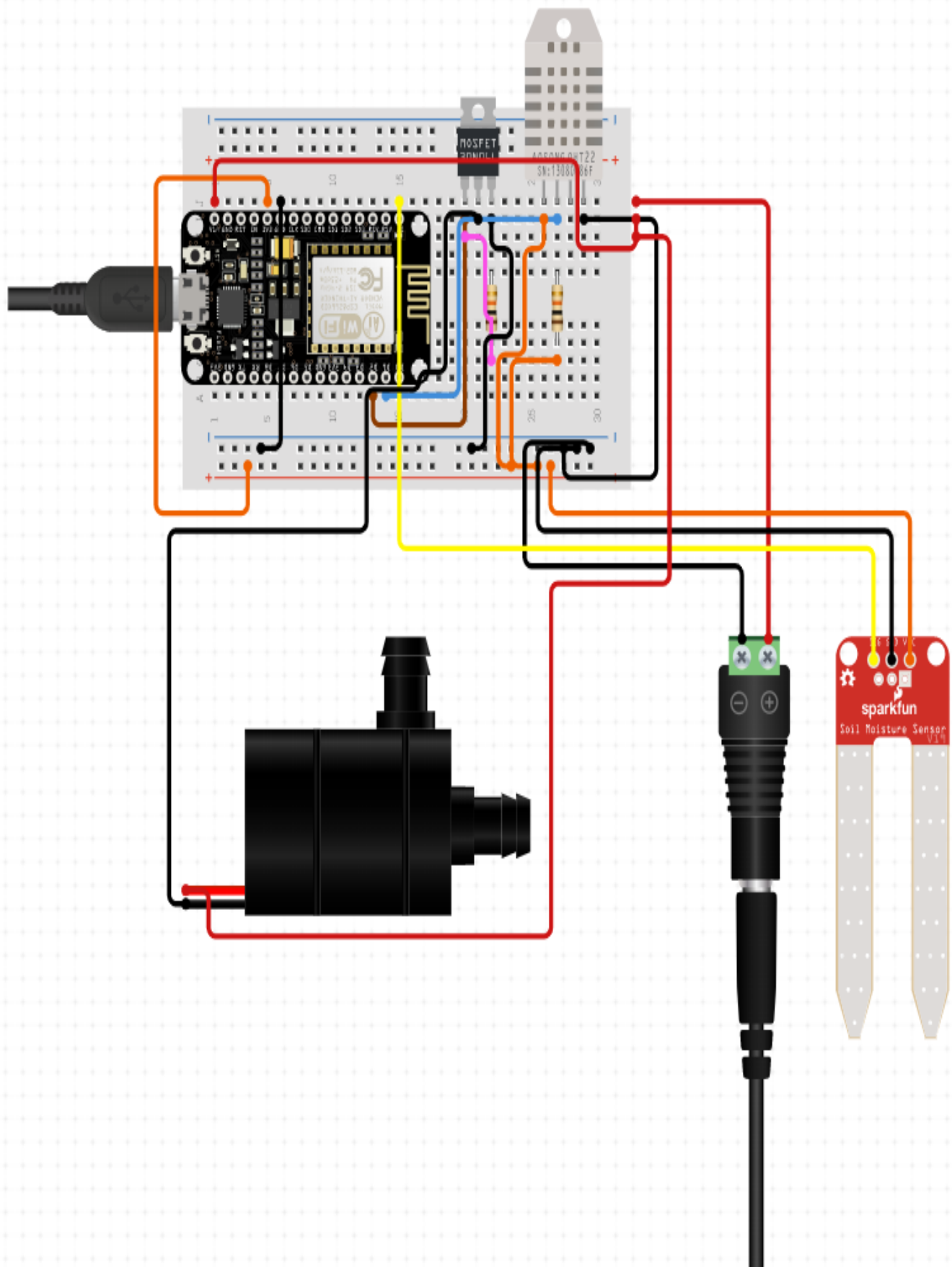
The coefficients in Table integrate the effects of both transpiration and evaporation over time. The effects of the integration over time represent an average wetting frequency for a 'standard' crop under typical growing conditions in an irrigated setting. The values for K_c during the initial and crop development stages are subject to the effects of large variations in wetting frequencies and therefore refinements to the value used for K_{c ini} should always be made. For frequent wettings such as with high frequency sprinkler irrigation or rainfall, the values for K_{c ini} may increase substantially.

Crop	$K_{c\ ini}^1$	$K_{c\ mid}$	$K_{c\ end}$	Maximum Crop Height (h) (m)
a. Small Vegetables	0.7	1.05	0.95	
Broccoli		1.05	0.95	0.3
Brussel Sprouts		1.05	0.95	0.4
Cabbage		1.05	0.95	0.4
Carrots		1.05	0.95	0.3
Cauliflower		1.05	0.95	0.4
Celery		1.05	1.00	0.6
Garlic		1.00	0.70	0.3
Lettuce		1.00	0.95	0.3
Onions				
- dry		1.05	0.75	0.4
- green		1.00	1.00	0.3
- seed		1.05	0.80	0.5
Spinach		1.00	0.95	0.3
Radish		0.90	0.85	0.3
b. Vegetables - Solanum Family (<i>Solanaceae</i>)	0.6	1.15	0.80	
Egg Plant		1.05	0.90	0.8
Sweet Peppers (bell)		1.05 ²	0.90	0.7
Tomato		1.15 ²	0.70-0.90	0.6
c. Vegetables - Cucumber Family (<i>Cucurbitaceae</i>)	0.5	1.00	0.80	
Cantaloupe	0.5	0.85	0.60	0.3
Cucumber				
- Fresh Market	0.6	1.00 ²	0.75	0.3
- Machine harvest	0.5	1.00	0.90	0.3
Pumpkin, Winter Squash		1.00	0.80	0.4
Squash, Zucchini		0.95	0.75	0.3
Sweet Melons		1.05	0.75	0.4
Watermelon	0.4	1.00	0.75	0.4


FLOWCHART OF OUR PROJECT MODEL METHODOLOGY:



ARCHITECTURE AND DESIGN OF OUR MODEL



BACKEND CLOUD DATABASE:

 **Firebase**

Project Overview

Develop

Quality

Spark

Authentication

Database

Storage

Hosting

Functions

ML Kit

Crashlytics

Performance

Test Lab

App Distribution

Extensions

Free \$0/month

Upgrade

SmartGardener

smartgardener-833c2

CurrentET0: 1.978

CurrentWeather: "broken clouds" X

DirectNSI

DisplayHumidity: 38

DisplayPrecip: 3.2

DisplayPressure: 1005

DisplayTempMax: 32.1499

DisplayTempMin: 32.1499

DisplayWindSpeed: 2.4197

ET0

ET0F

GSoilFlux

KCET0: 2.274

Kc: 1.15

PrecipProbability: 0

PressureCalc

RelativeHumidity


SolarRad

SumX1: -101.896

SumX1X2: -0.03089

SumX1Y: -892.306

SumX1square: 3306.00

 **Firebase**

Project Overview

Develop

Quality

Spark

Authentication

Database

Storage

Hosting

Functions

ML Kit

Crashlytics

Performance

Test Lab

App Distribution

Extensions

Free \$0/month

Upgrade

SmartGardener

SolarRad

SumX1: -101.896

SumX1X2: -0.03089

SumX1Y: -892.306

SumX1square: 3306.00

SumX2: 0.47641

SumX2Y: 3.12597

SumX2square: 0.00715

SumY: 271.14

WindSpeed

-M7HrcGH0uRO705H5uci: 7.6733

-M7HtvZuGftjtseRma9h: 7.6733

-M7HwvNWqr8nOloliYG7: 7.6733

-M7IRDyZSTOBcuev8i4y: 4.168

-M7IRvpYWp3HOn-fD_3I: 4.168

-M7ISUbQjDsiGf14CIAo: 4.168

-M7ITZhDnJSkNeje3dTd: 4.168

-M7IUjSyyM5s-Wrf7amw: 4.168

-M7IV9IqOqGyilERuiT: 4.168

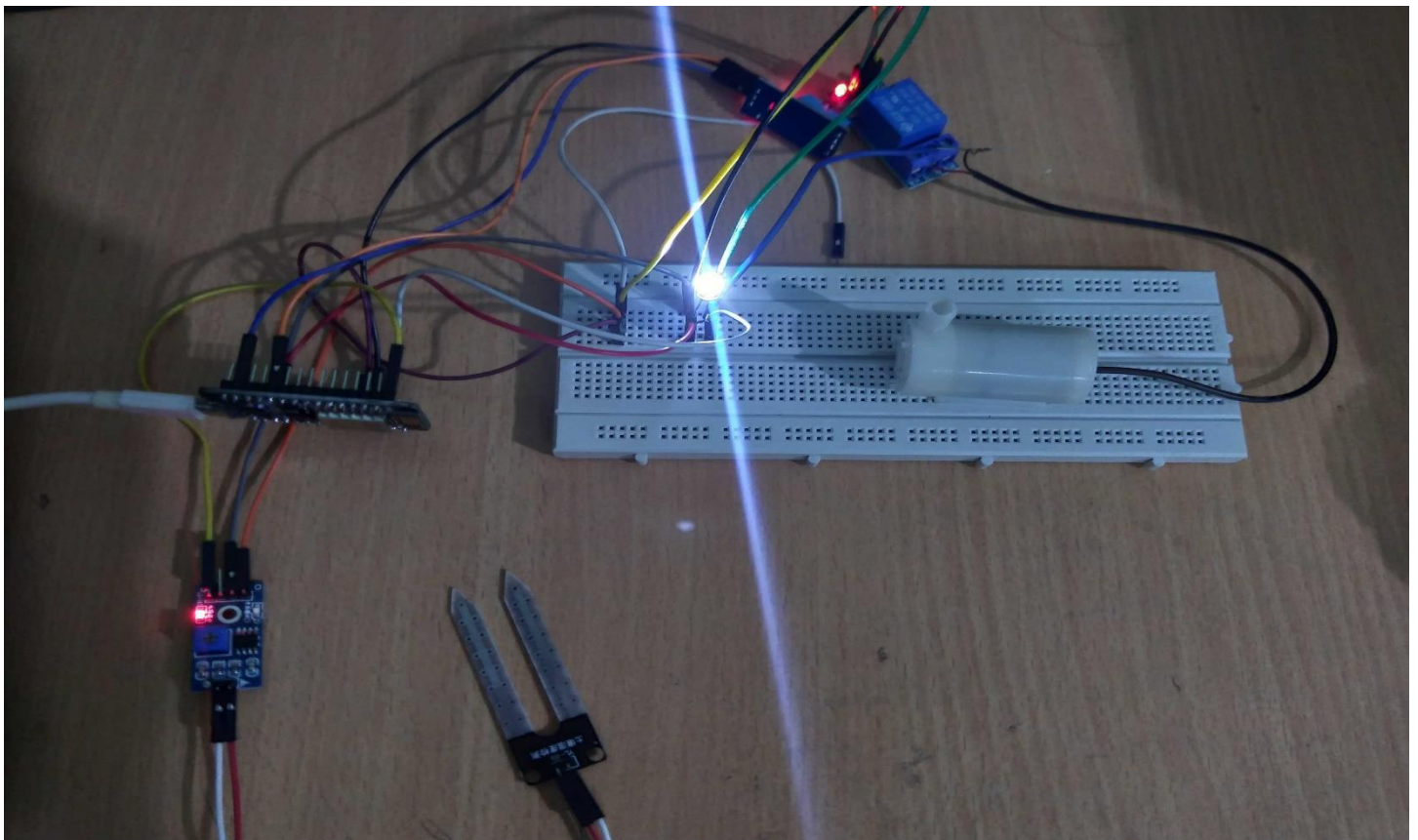
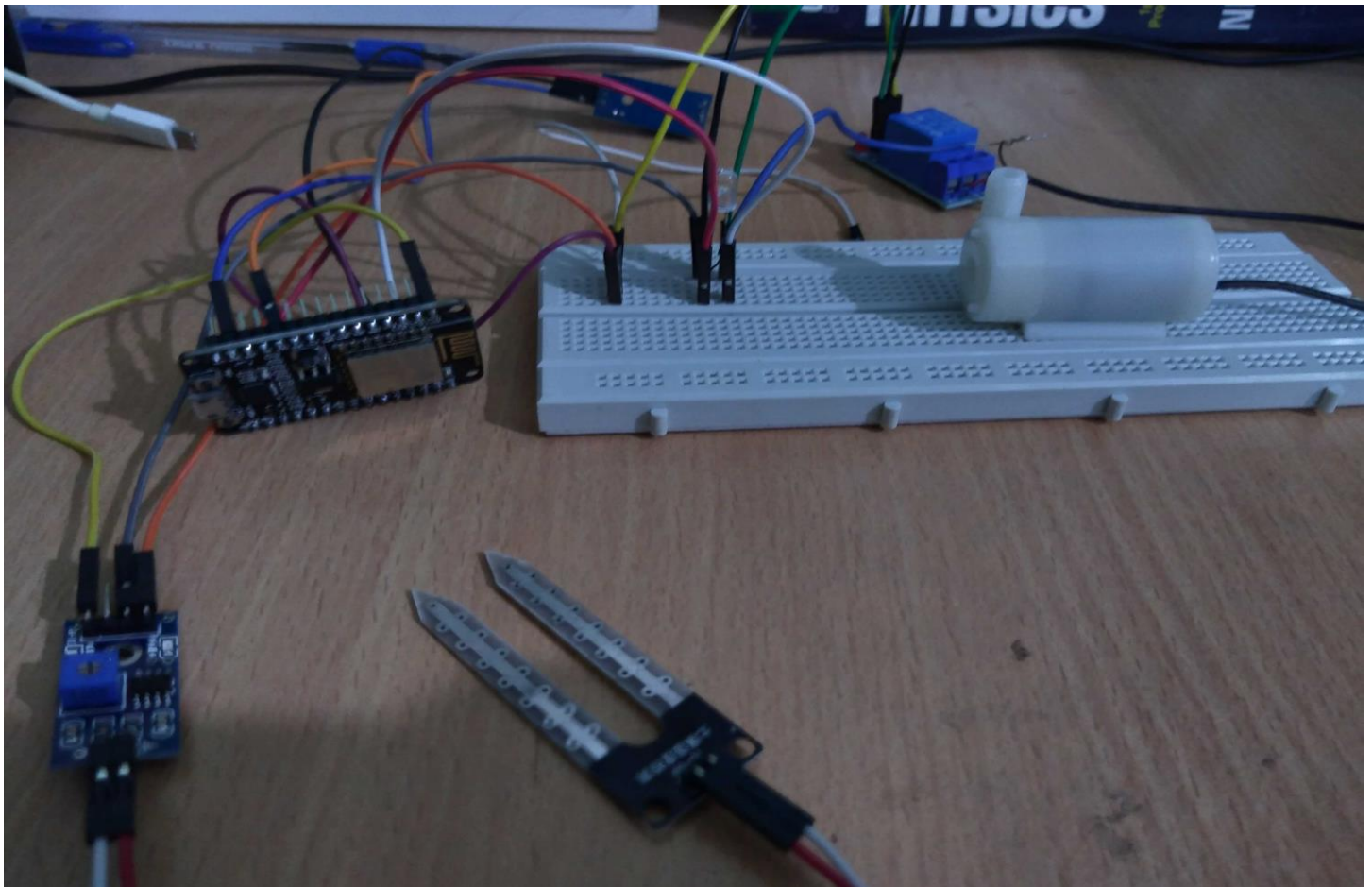
-M7IVn7a4_GAZYgn8uOj: 4.168

-M7IWGa3r5fL_c9OMoNi: 4.168

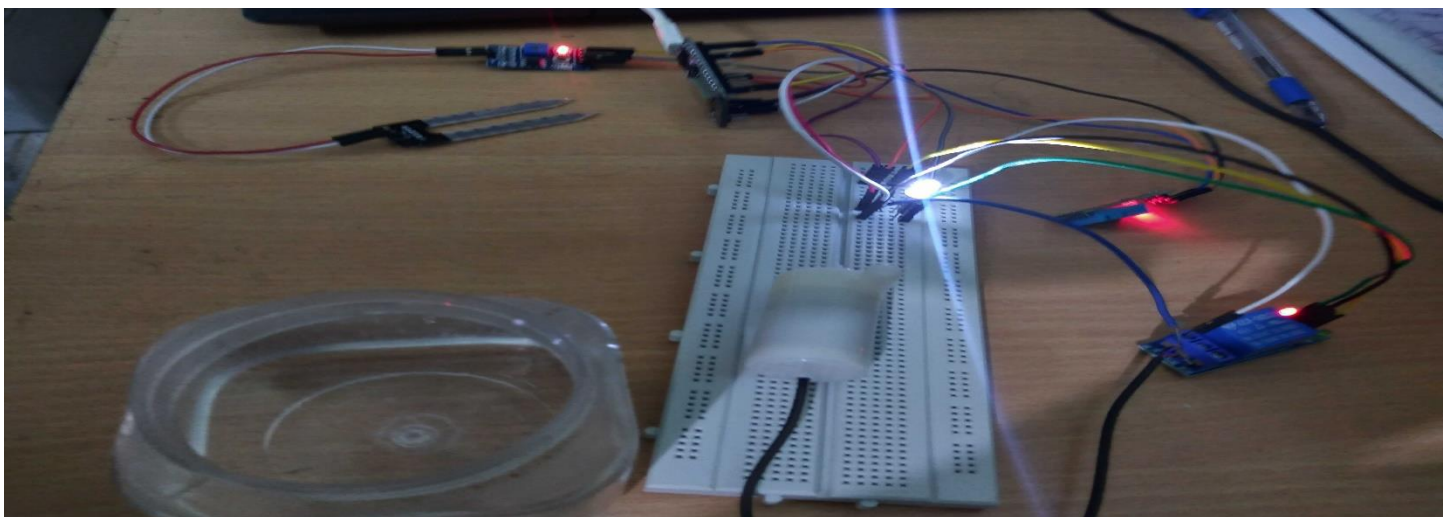
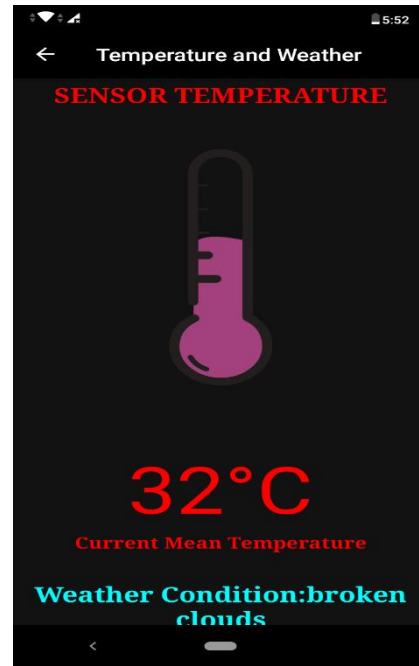
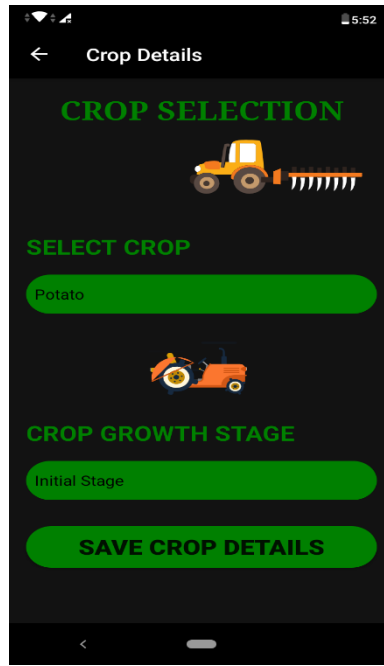
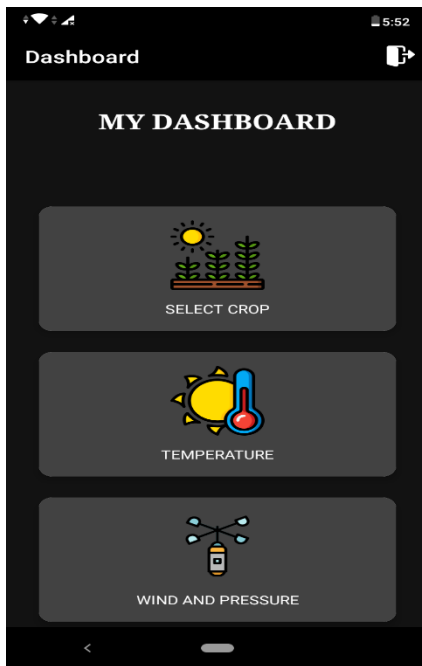
-M7IXBkDLiBB8IrnUdUn: 4.168

-M7IZ0yk85_ydkS1VL4h: 4.168

HARDWARE DEVICE MODEL:-



ANDROID APPLICATION CONNECTED TO IOT DEVICE VIA CLOUD:



SOURCE CODE OF OUR MODEL:-

```
#include <ArduinoJson.h>

#include <ESP8266WiFi.h>

#include <WiFiClientSecure.h>

#include <ESP8266WebServer.h>

#include <ESP8266HTTPClient.h>

#include <FirebaseArduino.h>                                // firebase library

#define FIREBASE_HOST "smartgardener-833c2.firebaseio.com"      // the project
name address from firebase id

#define FIREBASE_AUTH "f1BiVsro8Ft3oDXKfu5XGXyssYNVd8Q2BauvthRy" // the
secret key generated from firebase

//SENSOR AREA

#include "DHT.h"      // including the library of DHT11 temperature and humidity sensor

#define DHTTYPE DHT11 // DHT 11

#define dht_dpin 0

DHT dht(dht_dpin, DHTTYPE);

//Variable

float sensorMoisture;

float h;

float t;

float moisture_percentage;

int sensor_analog;

float ET0;

float KCET0;

float Kc;
```

```

float PrecipProb;

float G;

float SumY;

float tempMin;

float tempMax;

float Pres;

float rh;

float solarRad;

float windSpd;

float dni;

/* Set these to your wifi credentials. */

const char *ssid = "Bsharma"; //ENTER YOUR WIFI SETTINGS

const char *password = "9993936448";


const char *host = "irrisat-cloud.appspot.com";

const int httpsPort = 443; //HTTPS= 443 and HTTP = 80


//SHA1 finger print of certificate use web browser to view and copy

const char fingerprint[] PROGMEM = "e0 5c 17 d7 4a 85 68 8d d4 69 ba d3 d5 da 5c 89 cf 2d b0
29";

//=====

//          Power on setup

//=====


void setup() {

  delay(1000);

  digitalWrite(4,HIGH);

```

```

pinMode(4, OUTPUT);

Serial.begin(115200);

WiFi.mode(WIFI_OFF);    //Prevents reconnection issue (taking too long to connect)

delay(1000);

WiFi.mode(WIFI_STA);    //Only Station No AP, This line hides the viewing of ESP as wifi
                        hotspot

WiFi.begin(ssid, password); //Connect to your WiFi router

Serial.println("");


Serial.print("Connecting");

// Wait for connection
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}

//If connection successful show IP address in serial monitor

Serial.println("");

Serial.print("Connected to ");

Serial.println(ssid);

Serial.print("IP address: ");

Serial.println(WiFi.localIP()); //IP address assigned to your ESP

Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

}

//=====

//          Main Program Loop

//=====

void loop() {

```

```
//=====IRRISTAT API=====//

HttpClient httpAgro; //Declare an object of class HttpClient

httpAgro.begin("http://api.weatherbit.io/v2.0/forecast/agweather?lat=23.2756&lon=77.4560&key=10fe4d3c09b64cb18e44c87dd9003578");//("http://api.weatherbit.io/v2.0/current?lat=23.2756&lon=77.4560&key=10fe4d3c09b64cb18e44c87dd9003578"); //Specify request destination
int httpAPICode = httpAgro.GET(); //Send the request

if (httpAPICode > 0) { //Check the returning code

String payload = httpAgro.getString(); //Get the request response payload

int ind = payload.indexOf('}');

////////////////JSON PARSINNG////////////////////////////////

Serial.println("\n");

Serial.println(payload.substring(0,ind+1)); //Print the response payload

String final =payload.substring(0,ind+1)+"}";

DynamicJsonBuffer jsonBuffer(907);

char json[final.length() + 1];

final.toCharArray(json, final.length() + 1);

JsonObject& root = jsonBuffer.parseObject(json);

// Test if parsing succeeds.

if (!root.success()) {

    Serial.println("parseObject() failed");

    return;

}

ET0 = root["data"][0]["evapotranspiration"];

Kc = Firebase.getFloat("/Kc");

KCET0 = ET0*Kc;

Firebase.setFloat("/KCET0",KCET0);
```



```

    PrecipProb = root["data"][0]["precip"];
    G = root["data"][0]["soilt_0_10cm"];
    Firebase.pushFloat("/ET0",ET0);
    Firebase.pushFloat("/GSoilFlux",G);
    Firebase.setFloat("/CurrentET0",ET0);
    SumY = Firebase.getFloat("/SumY");
    SumY = SumY +ET0;
    Firebase.setFloat("/SumY",SumY);
    Firebase.setFloat("/PrecipProbability",PrecipProb);
}

httpAgro.end(); //Close connection

//=====OpenWeatherAPI=====//

HTTPClient http; //Declare an object of class HTTPClient

http.begin("http://api.openweathermap.org/data/2.5/weather?lat=23.2756&lon=77.4560&appid=8f9bc27b855db221dae3a2aa4880ce73");//Specify request destination

int httpCode = http.GET(); //Send the request

if (httpCode > 0) { //Check the returning code

    String openW = http.getString(); //Get the request response payload
    Serial.println(openW);

    //////////////////////////////////////////////////JSON PARSINNG////////////////////////////////////

    DynamicJsonBuffer jsonBufferO(782);

```

```
char jsonO[openW.length() + 1];
openW.toCharArray(jsonO, openW.length() + 1);

JsonObject& rootO = jsonBufferO.parseObject(jsonO);

// Test if parsing succeeds.
if (!rootO.success()) {
    Serial.println("parseObject() failed");
    return;
}

const char* desc = rootO["weather"][0]["description"];
Firebase.setString("/CurrentWeather",desc);

tempMin = (rootO["main"]["temp_min"]);
tempMax = (rootO["main"]["temp_max"]);
tempMin = tempMin - 273;
tempMax = tempMax - 273;

Firebase.pushFloat("/dataTempMin",tempMin);
Firebase.pushFloat("/dataTempMax",tempMax);
Firebase.setFloat("/DisplayTempMin",tempMin);
Firebase.setFloat("/DisplayTempMax",tempMax);

float disPres = rootO["main"]["pressure"];
float disHum = rootO["main"]["humidity"];
Firebase.setFloat("/DisplayPressure",disPres);
Firebase.setFloat("/DisplayHumidity",disHum);
```

```

}

http.end(); //Close connection

//=====OPENWEATHERCLOSE_CONNECTION=====//

//=====WEATHERBIT API=====//

HTTPClient httpWeatherBit; //Declare an object of class HTTPClient

httpWeatherBit.begin("http://api.weatherbit.io/v2.0/current?lat=23.2756&lon=77.4560&key=10fe4d3c09b64cb18e44c87dd9003578"); //Specify request destination

int httpBitCode = httpWeatherBit.GET(); //Send the request

if (httpBitCode > 0) { //Check the returning code

String WeatherBit = httpWeatherBit.getString(); //Get the request response payload

Serial.println(WeatherBit);

////////////////JSON PARSING////////////////////////////////

DynamicJsonBuffer jsonBufferW(782);

char jsonW[WeatherBit.length() + 1];

WeatherBit.toCharArray(jsonW, WeatherBit.length() + 1);

JsonObject& rootW = jsonBufferW.parseObject(jsonW);

// Test if parsing succeeds.

if (!rootW.success()) {

    Serial.println("parseObject() failed");

    return;

}

Pres = rootW["data"][0]["pres"];

rh = rootW["data"][0]["rh"];

```

```

solarRad = rootW["data"][0]["solar_rad"];
windSpd = rootW["data"][0]["wind_spd"];
dni = rootW["data"][0]["dni"];
Firebase.pushFloat("/PressureCalc",Pres);
Firebase.pushFloat("/RelativeHumidity",rh);
Firebase.pushFloat("/SolarRad",solarRad);
Firebase.pushFloat("/WindSpeed",windSpd);
Firebase.setFloat("/DisplayWindSpeed",windSpd);
Firebase.pushFloat("/DirectNSI",dni);
}

httpWeatherBit.end(); //Close connection

//=====PLSR ALGO=====//

float tempMean = (tempMin+tempMax)/2;
float p = (17.27*tempMean)/(tempMean+273.3);
float del = (4098*( 0.6108 * pow(2.71828,p)))/pow((tempMean + 237.3),2);
float gama = 0.000665*Pres;
float Rn = solarRad;
float u2 = windSpd;
float p1 = (17.27*tempMax)/(tempMax+273.3);
float p2 = (17.27*tempMin)/(tempMin+273.3);
float eTmax = 0.6108*(pow(2.71828,p1));
float eTmin = 0.6108*(pow(2.71828,p2));
float es = (eTmax + eTmin)/2;
float ea = (eTmin)*(rh/100);
float X1 = (del*(Rn-G))/(del + (gama*(1+ 0.34*u2)));
float X2 = (gama*u2*(es-ea))/((tempMean+273)*(del + (gama*(1+ 0.34*u2))));

```

```
Firestore.pushFloat("/X1",X1);
Firestore.pushFloat("/X2",X2);
float Y = ET0;
float X1Y = X1*Y;
float X2square = X2*X2;
float X1X2 = X1*X2;
float X2Y = X2*Y;
float X1square = X1*X1;
float SumX1 = Firestore.getFloat("/SumX1");
SumX1 = SumX1+X1;
float SumX2 = Firestore.getFloat("/SumX2");
SumX2 = SumX2+X2;
float SumX1Y = Firestore.getFloat("/SumX1Y");
SumX1Y=SumX1Y+X1Y;
float SumX2square = Firestore.getFloat("/SumX2square");
SumX2square=SumX2square+X2square;
float SumX1X2 = Firestore.getFloat("/SumX1X2");
SumX1X2=SumX1X2+X1X2;
float SumX2Y = Firestore.getFloat("/SumX2Y");
SumX2Y=SumX2Y+X2Y;
float SumX1square = Firestore.getFloat("/SumX1square");
SumX1square=SumX1square+X1square;

Firestore.setFloat("/SumX1",SumX1);
Firestore.setFloat("/SumX2",SumX2);
Firestore.setFloat("/SumX1Y",SumX1Y);
Firestore.setFloat("/SumX2square",SumX2square);
```

```

Firebase.setFloat("/SumX1X2",SumX1X2);
Firebase.setFloat("/SumX2Y",SumX2Y);
Firebase.setFloat("/SumX1square",SumX1square);

int c = Firebase.getInt("/count");
float b1 = ((SumX2square*SumX1Y)-(SumX1X2*SumX2Y))/((SumX1square*SumX2square)-
pow(SumX1X2,2));
float b2 = ((SumX1square*SumX2Y)-(SumX1X2*SumX1Y))/((SumX1square*SumX2square)-
pow(SumX1X2,2));
float b0 = (SumY/c) - (b1*(SumX1/c)) - (b2*(SumX2/c));

float YFinal = b0 +(b1*X1)+(b2*X2);
float ET0F = (((0.408*X1)+(900*X2))*2)/10;

Firebase.pushFloat("/ET0F",ET0F);
Firebase.pushFloat("/b1",b1);
Firebase.pushFloat("/b2",b2);
Firebase.pushFloat("/b0",b0);

sensor_analog = analogRead(A0);
moisture_percentage = ( 100 - ( ((sensor_analog-300)/724.00) * 100 ) );
Serial.print("Moisture Percentage = ");
Serial.print(moisture_percentage);
Serial.print("%\n\n");

h = dht.readHumidity();
t = dht.readTemperature();
sensorMoisture= ((float)(moisture_percentage)/10);
delay(5000);

```



```

if (sensorMoisture<6 && sensorMoisture<KCET0 && PrecipProb<KCET0){
  Serial.println("\MotorON");
  digitalWrite(4,LOW);
  delay(2000);
}
while(sensorMoisture<7 && sensorMoisture<KCET0 && PrecipProb<KCET0){
  sensor_analog = analogRead(A0);
  moisture_percentage = ( 100 - ( ((sensor_analog-300)/724.00) * 100 ) );
  sensorMoisture= ((float)(moisture_percentage)/10);
  Serial.println("Updated Moisture:");
  Serial.println(sensorMoisture);
  delay(5000);
}
digitalWrite(4,HIGH);
Serial.println("\MOTOR OFF");
Firebase.setInt("/count",c+1);
delay(1800000);
}

```

COMPARISON BETWEEN TWO BEST ALGORITHMS FOR REQUIRED SOIL MOISTURE DETECTION

1) Artificial neural network model (ANN)

An ANN consists of a large number of highly interconnected processing elements (nodes or units), which simulate basic functions of biological neurons. Fig. 4 illustrates a simple oneneuron model within an artificial network, where an input vector is passed through the neuron for providing an output value. A multiple-layer system consists of input, hidden and output layers. The theory and mathematical basis of ANNs have been extensively described in Haykin (1999) and Bishop (1995). In this study, a multiple-layer feed-forward backpropagation network with three layers was used: input, hidden and output layer. This type of network generally provides better performances in comparison to other types (Hornik et al., 1989). Tan– sigmoid transfer functions (non-linear) and linear transfer functions were selected for the

hidden and output layers, respectively. Using a tan– sigmoid function in the hidden layer allows to approximate only non-linear relations present between input and output layers (Haykin, 1999). The number of neurons in the hidden layer can be defined using a formula recommended by Fletcher and Goss (1993) or using a trial- and-error approach as suggested by Chang et al. (2004). The number of neurons in the hidden layer is of great importance, as too many neurons may cause over-fitting problems (Huang & Foo, 2002). For improving network generalization, Demuth and Beale (2004) recommended to use a network that is just large enough to provide an adequate fit between predictor and response variables. To avoid over-fitting problems and to provide an effective means to stop the time demanding training phase, the so called “early stopping” approach has been used (Demuth & Beale, 2004). The network input layer used in this study relates to reflectance, while the network output layer relates to soil salinity. Data preprocessing techniques (centering or standardizing to a mean of zero and standard deviation of one) and principal component analysis (PCA) were applied to the inputs to normalize the reflectance data and to reduce the data dimension. The numbers of neurons used for training the networks were varied systematically between five and 13 to allow subsequent selection of the most appropriate network size based on the performance on the test data set (Adeloye & De Munare, 2005). For ANN modeling, the computer software MatLab and the Neural Network Toolbox were used (Demuth & Beale, 2004). In each scale of the study, each data set was divided into three subsets, one for training (half of input data), one for validation (a quarter) and one for testing (a quarter of input data). The Levenberg–Marquardt algorithm (Demuth & Beale, 2004), which provides a fast optimization, was used for network training. The performance of a trained network was assessed by comparing the mean squared error (MSE) and root mean squared error (RMSE) calculated from training, validation and testing data subsets. Only a training data subset is used for updating the network weights and biases. During training, error with respect to validation data subset is monitored. When the validation error increases for a specified number of iterations, the training is stopped. Error with respect to testing data subset is not monitored during training, but is quantified to assess the final performance of a trained ANN model.

2) Quantification of model performance and estimation errors between PLSR and ANN Algorithms

To quantify performance of salinity–reflectance models based on PLSR and ANN methods, various parameters between estimated values (Y') and independent reference measurements (Y) were calculated (Table 2): root mean square error (RMSE), relative RMSE (RMSE%), mean and relative mean absolute errors (MAE and MAE%). Both MAE and RMSE indicate absolute estimation errors, but RMSE is more sensitive to outliers and thus, RMSE% and MAE% are also calculated. To analyse the goodness of estimated values versus the observed values, three other model performance parameters were calculated (Miehle et al., 2006; Nash & Sutcliffe, 1970; Williams, 2001): coefficient of determination (R^2), model efficiency (ME) and ratio of prediction to deviation (RPD). Detailed descriptions and definitions of these model performance parameters are given in Taylor (1997) and Cacuci (2003). The equations for quantification of the model performance parameters are given in Table 2. The R^2 and RPD indicate strength of statistical correlation between measured values and predicted values both for PLSR and ANN predictive models. A predictive model is accurate if R^2 and RPD values are higher than 0.91 and 2.5, respectively (Williams, 2001). R^2 between 0.82–0.9 and RPD higher than 2 indicate a good prediction, whereas the R^2 of an approximate prediction is considered to lie between 0.66–0.81 with a RPD of higher than 1.5. R^2 between 0.5 and 0.65 indicates a poor relationship. MAE and ME indices were also used to examine the performance of the PLSR and ANN predictive models. MAE and ME values indicate degree of agreement between measured and predicted values; hence, they provide a measure of prediction model efficiency (Nash & Sutcliffe, 1970). Larger values MAE% and RMSE%, thus mean high prediction errors while ME values close to 1 indicate small errors in prediction values. To examine the performance of PLSR-calibration models RMSE and RPD values is analysed. Generalization of the

ANN-training, validation and test models was analysed using the R^2 and RMSE values calculated from each data sets (training, validation and test).

RESULTS AND DISCUSSION

The system designed is an astute irrigation solution predicated on artificial astuteness which makes utilization of the soil moisture content and the moisture requisites of the crop to make the entire process of irrigation automatic. Its core benefit is its efficiency and economic feasibility. The main conception about the rainfall estimation proposed is to get the virtually correct estimation of the rainfall of a concrete local region as well as the annual rainfall to get data for future estimation of the rainfall in states. The variation in the non-linear patterns of rainfall for India over the historic data is scrutinized and patterns visually examined in the historic period from 1871 to 2017 are as follows:

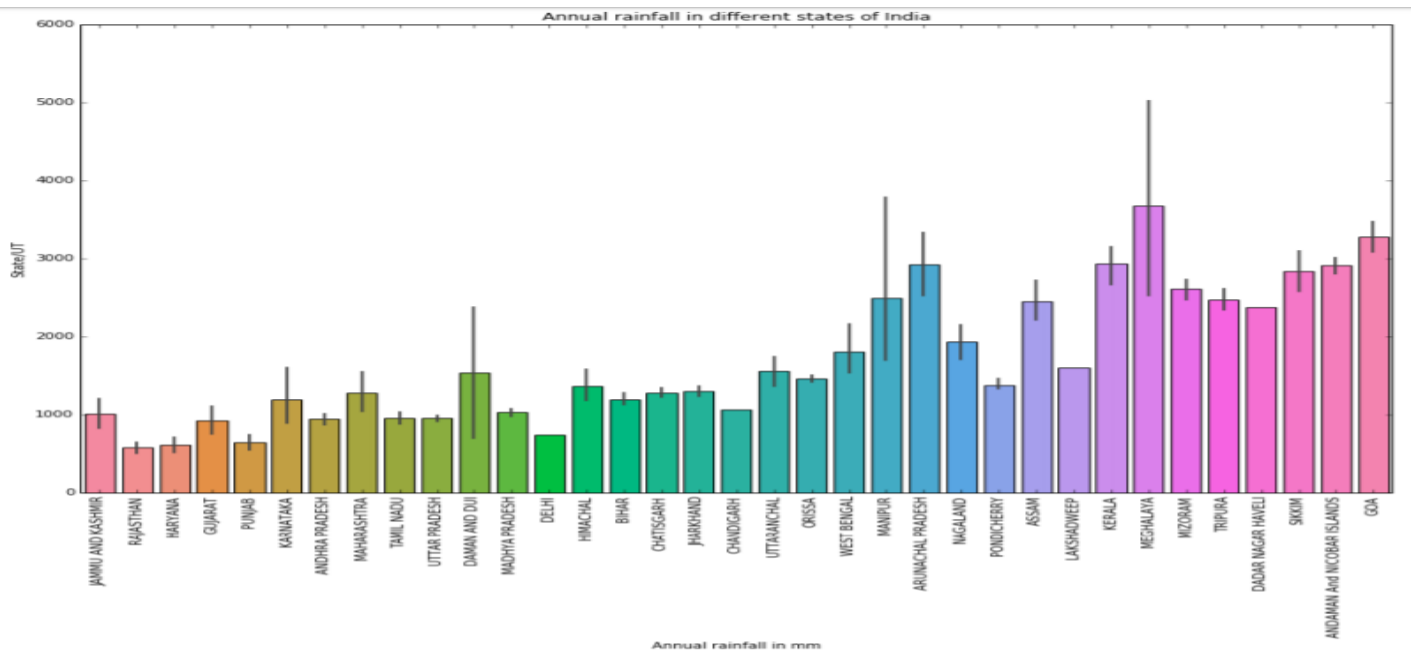
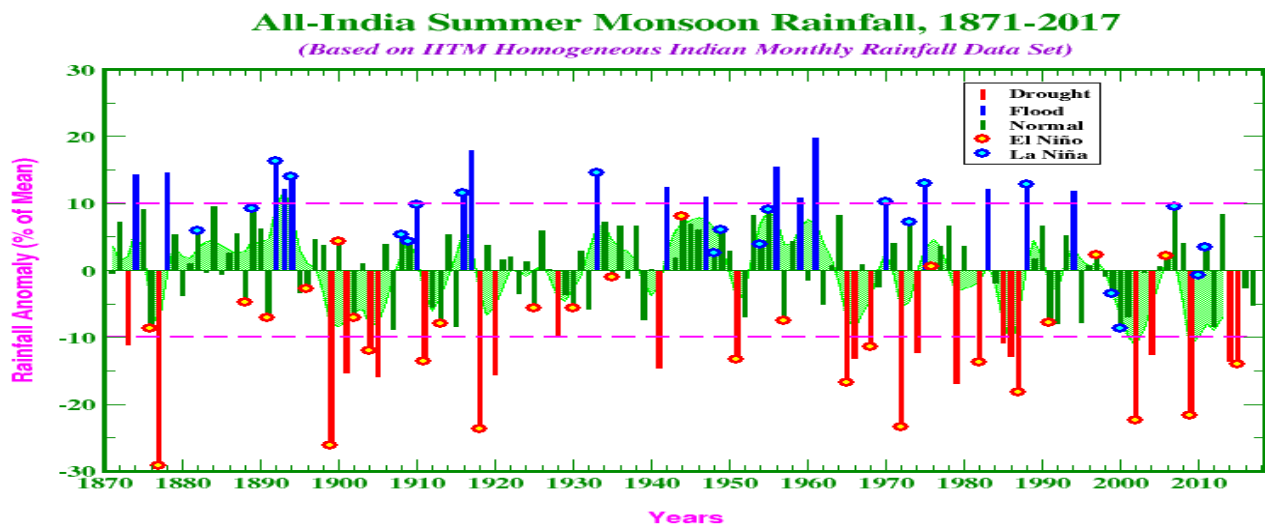


Figure 4: Annual Rainfall Prediction (in mm) for Indian States

The annual rainfall presaged for the state of Punjab utilizing Time Series ARIMA Model is 545 mm for the year 2015 which is proximately to the pristine value of 549.5 mm with a precision of 83%. This is further verified with the monthly rainfall presage in the state. The system prognosticates the probability of rain through this and

thus if the probability of rain emerges to be higher than 0.5 then the motor is not switched. If the probability of rain is lesser than 0.5 then the soil moisture requisite is calculated provided the soil moisture sensor data thereby switching on the motor for only the required duration automatically.

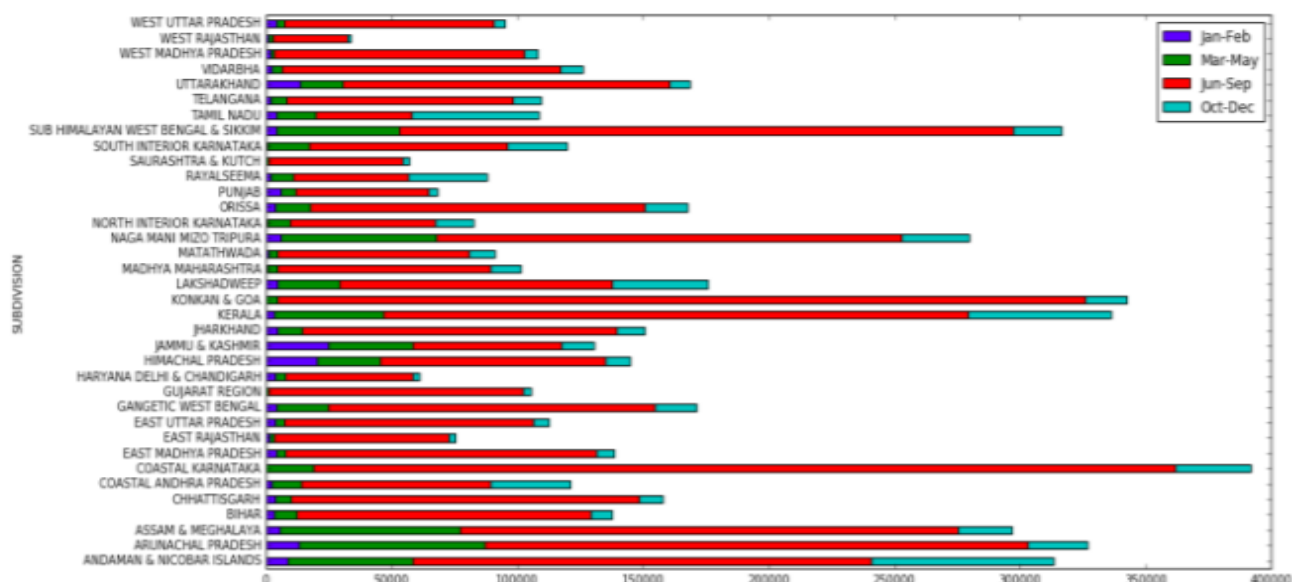


Figure 5: Monthly Rainfall Prediction (in mm) for Indian States

CONCLUSION

The autonomous irrigation system we developed uses Artificial Intelligence learning and predictive algorithms to integrate perspicacity to subsisting concept of automatic irrigation systems. The methods discussed in this paper can avail increase irrigation efficiency while decrementing effort required and avails water conservation compared to current irrigation methods. The system currently depends on weather station information for its calculation. This dependency can be superseded with on-premise sensors for deployment in rural areas widely found in the Indian subcontinent and arid regions where water is available in constrained quantity.

REFERENCES

- [1] Edordu C. and Sacks L., "Self Organizing Wireless Sensor Networks as a Land Management Tool in Developing Countries: A Preliminary Survey," In Proceedings of the 2006 London Communications Symposium, September 2006, Communications Engineering Doctorate Centre, London, UK.
- [2] Kim Y., Evans R.G. and Iversen W.M., "Remote Sensing and Control of an Irrigation System Using a Distributed Wireless Sensor Network," Instrumentation and Measurement, IEEE Transactions on, vol.57, no.7, pp.1379-1387, July 2008. <http://dx.doi.org/10.1109/TIM.2008.917198>
- [3] Joaquín G, Juan F , Alejandra N.G, and Miguel Ángel, "Automated Irrigation System Using a Wireless Sensor Network and GPRS Module", IEEE Transactions On Instrumentation and Measurement, Vol.63, no.1, pp.166-176, 2013
- [4] Karandeep K, "Machine Learning : Applications in Indian Agriculture", International Journal of Advanced Research in Computer and Communication Engineering, Vol.5, no.4, pp.342- 344, 2016.

- [5] Washington O and Joseph O, "Machine Learning Classification Technique for Famine Prediction". Proceedings of the World Congress on Engineering 2011 Vol II WCE 2011, July 6 - 8, London, U.K, 2011.
- [6] Snehal S, Sandeep V.R, "Agricultural Crop Yield Prediction Using Artificial Neural Network Approach". International Journal of Innovative Research in Electrical, Electronic, Instrumentation and Control Engineering, Vol. 2, Issue 1, January 2014.
- [7] Kumar R, Singh M .P, Prabhat K, and Singh J.P. "Crop Selection Method to Maximize Crop Yield Rate Using Machine Learning Technique." Proceedings of International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM) ,2015.
- [8] Rumpf, T., A.-K. Mahlein, U. Steiner, E.-C. Oerke, H.-W. Dehne, and L. Plümer. "Early Detection and Classification of Plant Diseases with Support Vector Machines Based on Hyperspectral Reflectance." Computers and Electronics in Agriculture, Vol. 74, no.1, pp.91-99, 2010.
- [9] Carles Antón-Haro, Thierry Lestable, Yonghua Lin, Navid Nikaein, Thomas Watteyne, Jesus AlonsoZarate, "Machine-to-machine: An emerging communication paradigm. Trans. on Emerging Telecommunications Technologies", Volume 24, Issue 4, pages 353–354, June 2013. DOI: <http://dx.doi.org/10.1002/ett.2668>
- [10] Koushik Anand, Jayakumar C, Mohana Muthu and Sridhar A, "Automatic Drip Irrigation using Fuzzy Logic and Mobile Technology", Proceedings of Technological Innovation in ICT for Agriculture and Rural Development, 2015.
- [11] Kait L.K., Kai C.Z., Khoshdelniat R., Lim S.M., and Tat E.H., "Paddy Growth Monitoring with Wireless Sensor Networks," Proceedings of Intelligent and Advanced Systems, Kuala Lumpur, Malaysia, pp.966- 970, 2007. <http://dx.doi.org/10.1109/ICIAS.2007.4658529>
- [12] Kim Y., Evans R.G. and Iversen W.M., "Remote Sensing and Control of an Irrigation System Using a Distributed Wireless Sensor Network," Instrumentation and Measurement, IEEE Transactions on, vol.57, no.7, pp.1379-1387, July 2008. <http://dx.doi.org/10.1109/TIM.2008.917198>
- [13] Wall R.W. and King B.A., "Incorporating plug and play technology into measurement and control systems for irrigation management", presented at the ASAE/CSAE Annu. Int. Meeting, Ottawa, ON, Canada, Aug. 2004
- [14] Yang W., Liusheng H., Junmin W. and Hongli X., "Wireless Sensor Networks for Intensive Irrigated Agriculture," Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE, pp.197-201, Las Vegas, Nevada, Jan. 2007. <http://dx.doi.org/10.1109/CCNC.2007.46>
- [15] Konstantinos K., Apostolos X., Panagiotis K. and George S., "Topology Optimization in Wireless Sensor Networks for Precision Agriculture Applications," Sensor Technologies and Applications, 2007. SensorComm 2007. International Conference on, pp.526-530, Valencia, Spain, 14-20 Oct. 2007. <http://dx.doi.org/10.1109/SENSORCOMM.2007.101>
- [16] Masuki, K. F. Ga, Kamugisha, Rb, Mowo, J. Gc, Tanui, Jc, Tukahirwa, Ja. Mogoi, Jc. and Adera E.O, "Role of mobile phones in improving communication and information delivery for agricultural development", ICT and Development - Research Voices from Africa. International Federation for Information Processing (IFIP), Technical Commission 9 – Relationship Between Computers and Society. Workshop at Makerere University, Uganda. 22-23 March 2010

- [17] Suyash S P and Sandeep A T, “Early Detection of Grapes Disease using Machine Learning and IoT”, Proceedings of Second International Conference on Cognitive Computing and Information Processing, Mysore, India, 2016
- [18] Yue L, Long M and Ooi S K, “Prediction of Soil Moisture based on Extreme Learning Machine for an Apple Orchard”, Proceedings of 3rd IEEE International Conference on Cloud Computing and Intelligent System”, Shenzhen, Hong Kong, 2014.
- [19] Biswas S, Saunshi A, Sarangi S and Pappula S, “Random Forest based Classification of Diseases in Grapes from Images Captured in Uncontrolled Environment”, Proceedings of IET International Conference on Signal Processing, 2016.
- [20] Rakesh K, Singh M.P, Prabhat K and Singh J.P, “Crop Selection Method to maximize Crop Yield Rate using Machine Learning Technique”, Proceedings of International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials, Chennai, India, 2015.



SMART IRRIGATION USING ARTIFICIAL INTELLIGENCE AND IOT

18BCE0107, 18BCE0154,18BCE0172|PROF. BHULAKSHMI BONTHU| SCOPE

Introduction

we provide an Automatic Irrigation System predicated on Artificial Perspicacity and Internet of Things, which can autonomously irrigate fields utilizing soil moisture data. The system is predicated on prognostication algorithms which make utilization of historic weather data to identify and presage rainfall patterns and climate changes.

Motivation

Farmers primarily depend on personal monitoring and their experience in irrigating the fields, and as a result, irrigation becomes largely inefficient and eccentric, therefore our device will help them in irrigation.

SCOPE of the Project

Agriculture is the primary source of livelihood for about 58 percent of India’s population. 70 % of rural India is dependent on agriculture for money. There is clearly a huge market for irrigational products.

To achieve this, we make utilization of Node MCUs, and soil moisture sensors, placed inside waterproof boxes and spread evenly throughout the area to be irrigated. All these nodes are connected to cloud via Wi-Fi connection

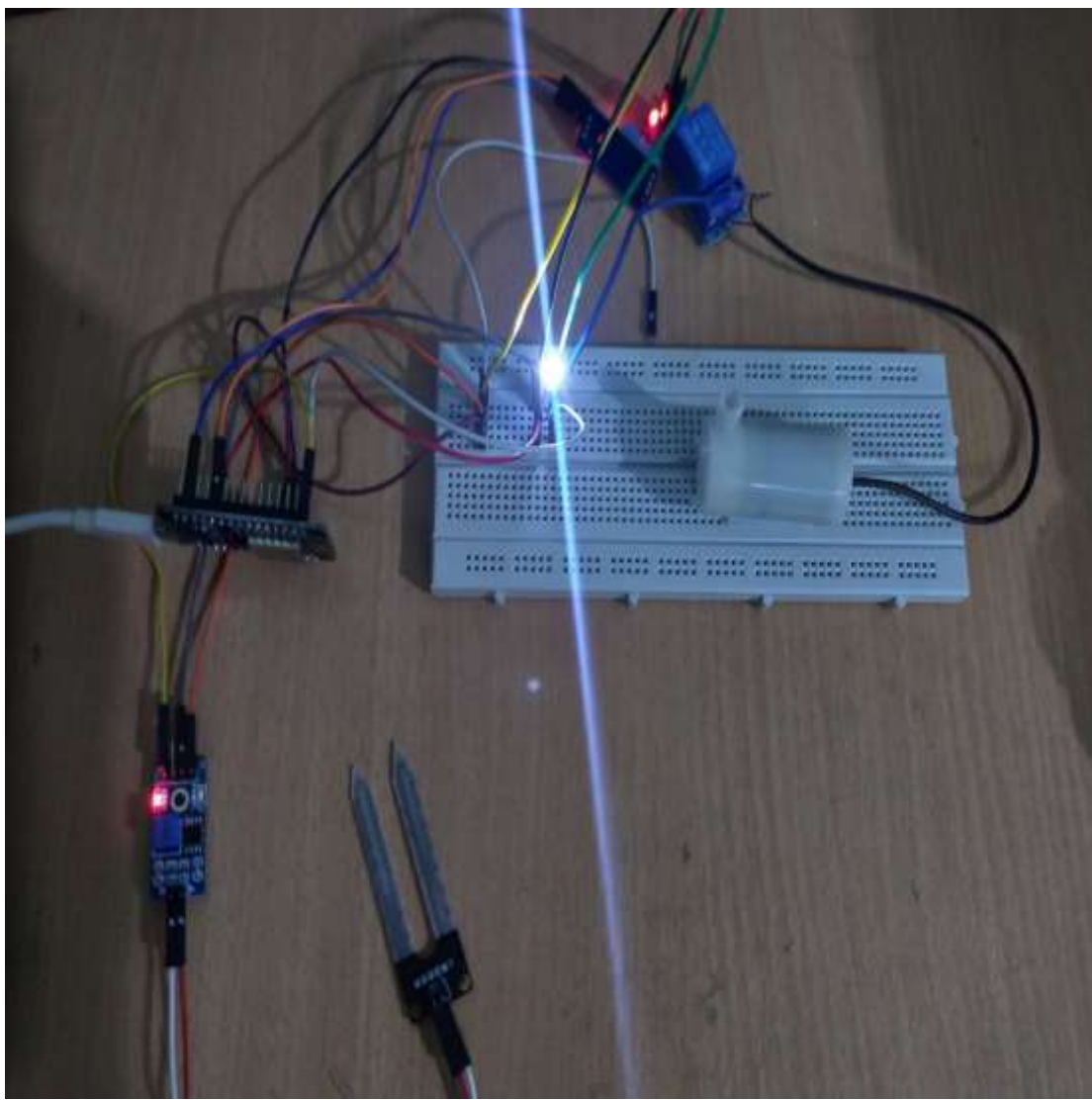
Methodology

The system designed consists briefly of two major components:

- Artificial Intelligence to predict the amount of rainfall and the soil moisture content.
- Implementation of system using Internet of Things.

Distributed Nodes

Each node is comprised of an ESP8266 [8] predicated microcontroller board (NodeMCU) and an array of analog sensors. These sensors are acclimated to fetch information about weather conditions, which is converted to digital 10-bit integers utilizing the Analog to Digital converters present on each NodeMCU.



The nodes are connected to the central controller through a shared Wi-Fi access point, over the IPv4 protocol. A client-server architecture is utilized for communication between the nodes and the central controller. The nodes do not communicate with each other. This implementation utilizes a Cloud server on the central controller. The server elongates a URL in the form of a web hook. The nodes connect to the server via this web hook to send data. The HTTP GET request is utilized to transmit the sensor data after post-calculation to the server. The default replication of the server conveys two messages to the node that sent the request: 1. if the pump annexed to the particular node has to be switched on or not, and 2. the duration for which the pump has to be switched on. The duration is calculated utilizing the output of the regression algorithm. Each node sends a request to the server at an interval of 30 minutes.

PARTIAL LEAST SQUARE REGRESSION (PLSR) ALGORITHM (AI ALGORITHM USED IN OUR PROJECT):-

The properties of pls regression can be analyzed from a sketch of the original algorithm. The first step is to create two matrices: $E = X$ and $F = Y$. These matrices are then column centered and normalized (i.e., transformed into Zscores). The sum of squares of these matrices are denoted SSX and SSY . Before starting the iteration process, the vector u is initialized with random 2 values. (in what follows the symbol \propto means “to normalize the result of the operation”).

Step 1. $w \propto ETu$ (estimate X weights).

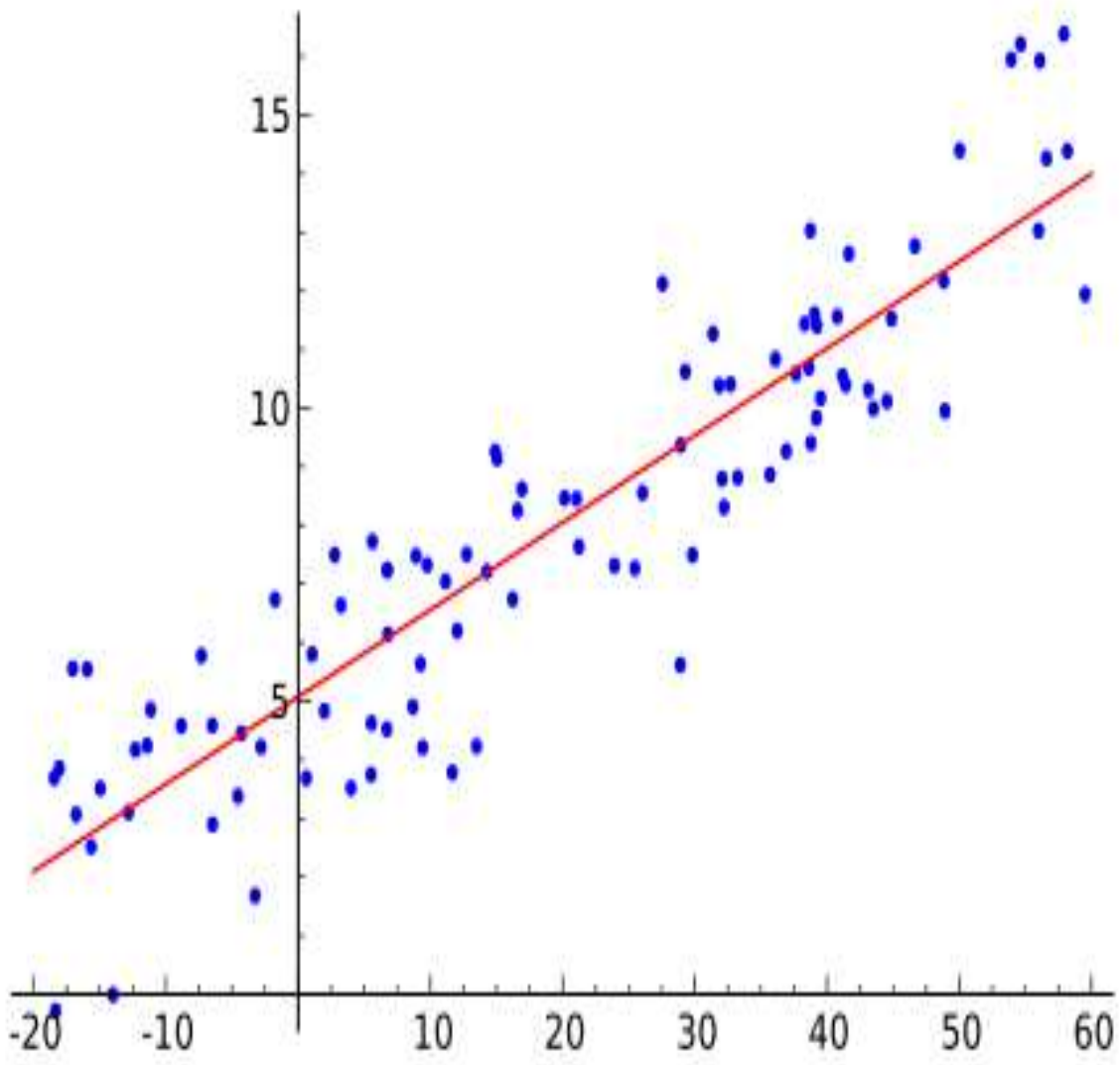
Step 2. $t \propto Ew$ (estimate X factor scores).

Step 3. $c \propto F Tt$ (estimate Y weights).

Step 4. $u = Fc$ (estimate Y scores).

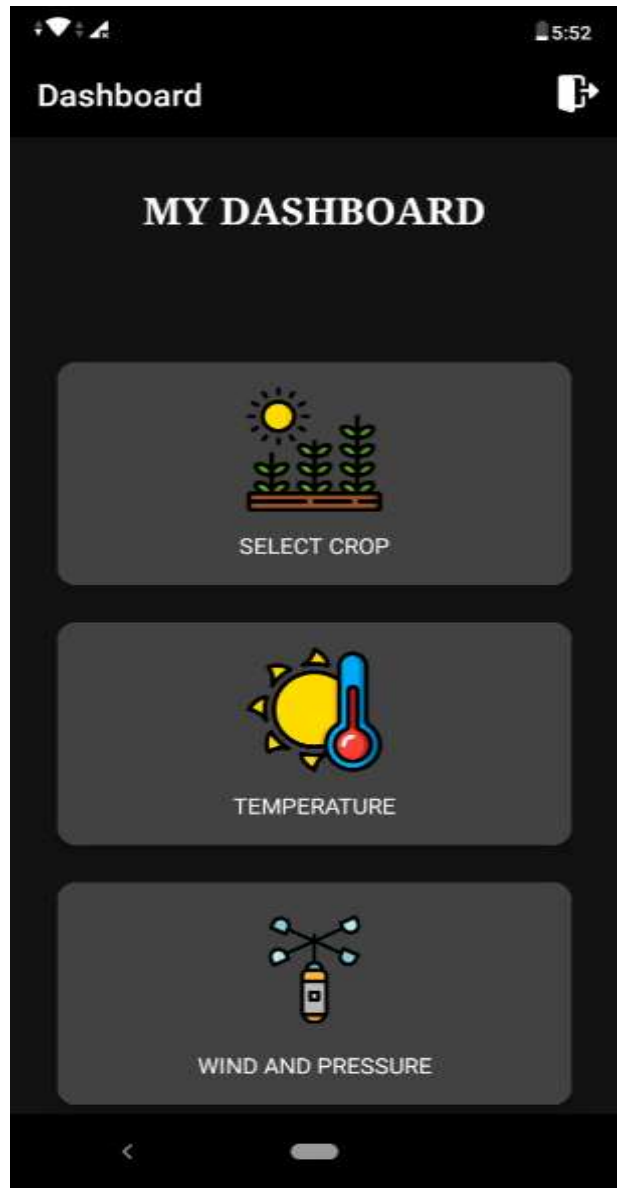
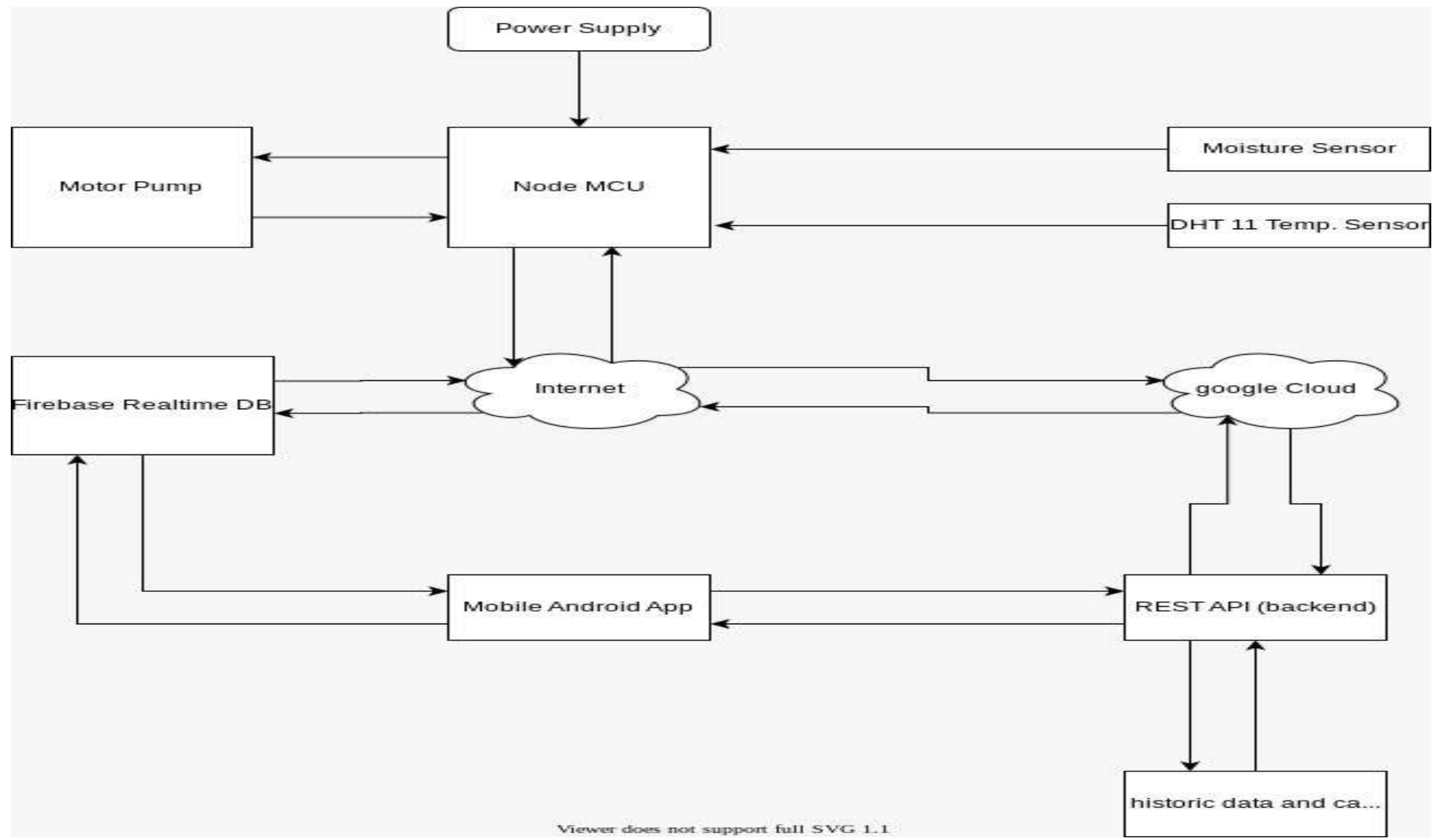
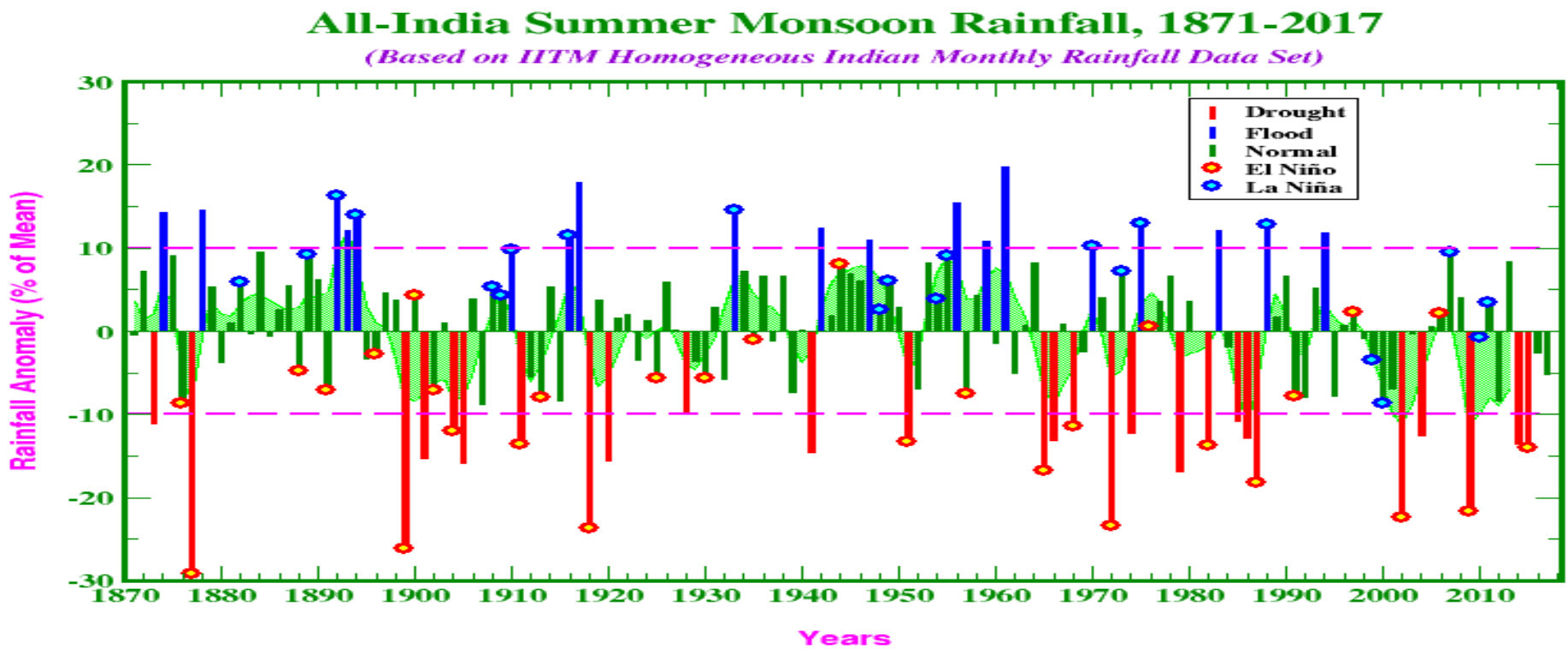
$$Y = a + bX + e$$
$$Y = a + b_1X_1 + b_2X_2 + \dots + b_kX_k + e$$

$$b_1 = \frac{(\sum x_2^2)(\sum x_1y) - (\sum x_1x_2)(\sum x_2y)}{(\sum x_1^2)(\sum x_2^2) - (\sum x_1x_2)^2}$$
$$b_2 = \frac{(\sum x_1^2)(\sum x_2y) - (\sum x_1x_2)(\sum x_1y)}{(\sum x_1^2)(\sum x_2^2) - (\sum x_1x_2)^2}$$



Results

The system designed is an astute irrigation solution predicated on artificial astuteness which makes utilization of the soil moisture content and the moisture requisites of the crop to make the entire process of irrigation automatic. Its core benefit is its efficiency and economic feasibility. The main conception abaft the rainfall estimation proposed is to get the virtually correct estimation of the rainfall of a concrete local region as well as the annual rainfall to get data for future estimation of the rainfall in states. The variation in the non-linear patterns of rainfall for India over the historic data is scrutinized and patterns visually examined in the historic period from 1871 to 2017 are as follows:



Farmers will be able to do the irrigation easily by following the below steps:

1. They just have to select the type of crop they are growing in their farm and the current growth stage. That's it rest our IOT device will handle.

RESULT ANALYSIS:

Our IOT device and app are connected to a single cloud server and can communicate through the same. IOT device finds the current environment data and send it to cloud and also calculate the required amount of soil moisture on the basis of current factors through PLSR algorithm and at the same time updates all the information on our Android App. This will conserve water and will increase the crop production of farmers and it is easy to use device.

Conclusion

The autonomous irrigation system we developed uses Artificial Intelligence learning and predictive algorithms to integrate perspicacity to subsisting concept of automatic irrigation systems. The methods discussed in this paper can avail increase irrigation efficiency while decrementing effort required and avails water conservation compared to current irrigation methods. The system currently depends on weather station information for its calculation. This dependency can be superseded with on-premise sensors for deployment in rural areas widely found in the Indian subcontinent and arid regions where water is available in constrained quantity.

References

[1] Kim Y., Evans R.G. and Iversen W.M., “Remote Sensing and Control of an Irrigation System Using a Distributed Wireless Sensor Network,” Instrumentation and Measurement, IEEE Transactions on, vol.57, no.7, pp.1379-1387, July 2008. <http://dx.doi.org/10.1109/TIM.2008.917198>

[2] Joaquín G, Juan F , Alejandra N.G, and Miguel Ángel, “Automated Irrigation System Using a Wireless Sensor Network and GPRS Module”, IEEE Transactions On Instrumentation and Measurement, Vol.63, no.1, pp.166-176, 2013

[3] Karandeep K, “Machine Learning : Applications in Indian Agriculture”, International Journal of Advanced Research in Computer and Communication Engineering, Vol.5, no.4, pp.342- 344, 2016.

MICROPROCESSOR AND INTERFACING – CSE2006 J - COMPONENT

SMART IRRIGATION USING ARTIFICIAL INTELLIGENCE AND IOT



ABSTRACT :

- Agriculture plays a consequential role in the economy and its contribution is predicated on quantifiable crop yield which is highly dependent upon irrigation.
- Through our project the farm can be modernized with electronic technology that continuously monitors the conditions of plants and soil, so that the plants could be provided watering as required.
- The water pump is automatically controlled based on the values of the various environmental factors like temperature, humidity, soil moisture and light intensity which we can measure through sensors like DHT-11 Temperature and Humidity sensor, Moisture sensor.
- Through our research work we will try to make the model intelligent by storing the previous scanned values in database and doing the pre-perception on the basis of historic values stored in learning phase of the working model. **We are using PLSR algorithm to train our model.**



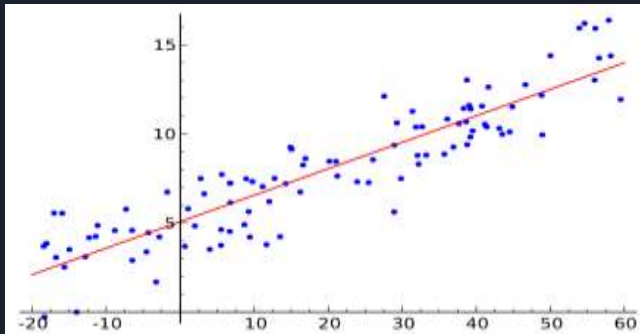
OBJECTIVE

- we have developed a perspicacious system which can study the patterns of rainfall in a region, and soothsay weather conditions in order to habituate to the geography, thus prognosticate the quantity of water needed for irrigation, to minimize wastage and increment the crop yield

ALGORITHM USED TO SOLVE PROBLEM STATEMENT

PARTIAL LEAST SQUARE REGRESSION (PLSR) ALGORITHM (AI ALGORITHM USED IN OUR PROJECT):-

- The goal of pls regression is to predict Y from X and to describe their common structure. When Y is a vector and X is full rank, this goal could be accomplished using ordinary multiple regression. When the number of predictors is large compared to the number of observations, X is likely to be singular and the regression approach is no longer feasible (i.e., because of multicollinearity).
- The trained PLSR model takes the predictor matrix as input which contains the soil and weather information for a particular day, and predicts the soil moisture percentage required for the crop. This soil moisture percentage helps us to calculate the minutes of irrigation required as a function of the area of the crop field and the power of the electric motor being used.



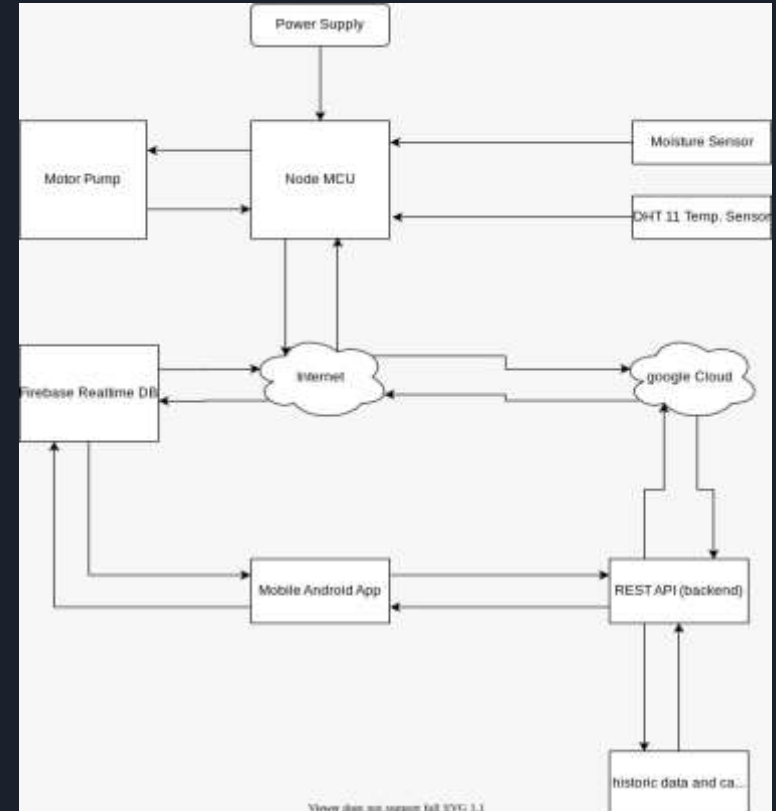
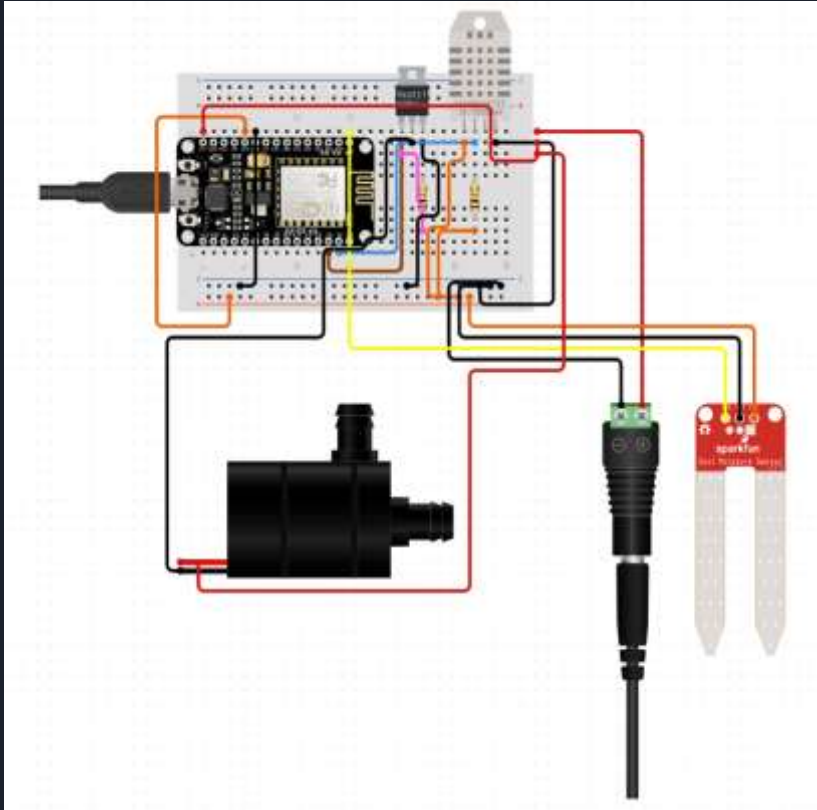
THE EQUATIONS TO CALCULATE THE SOIL MOISTURE RESULT INTO THE BELOW GIVEN MATHEMATICAL FORM:

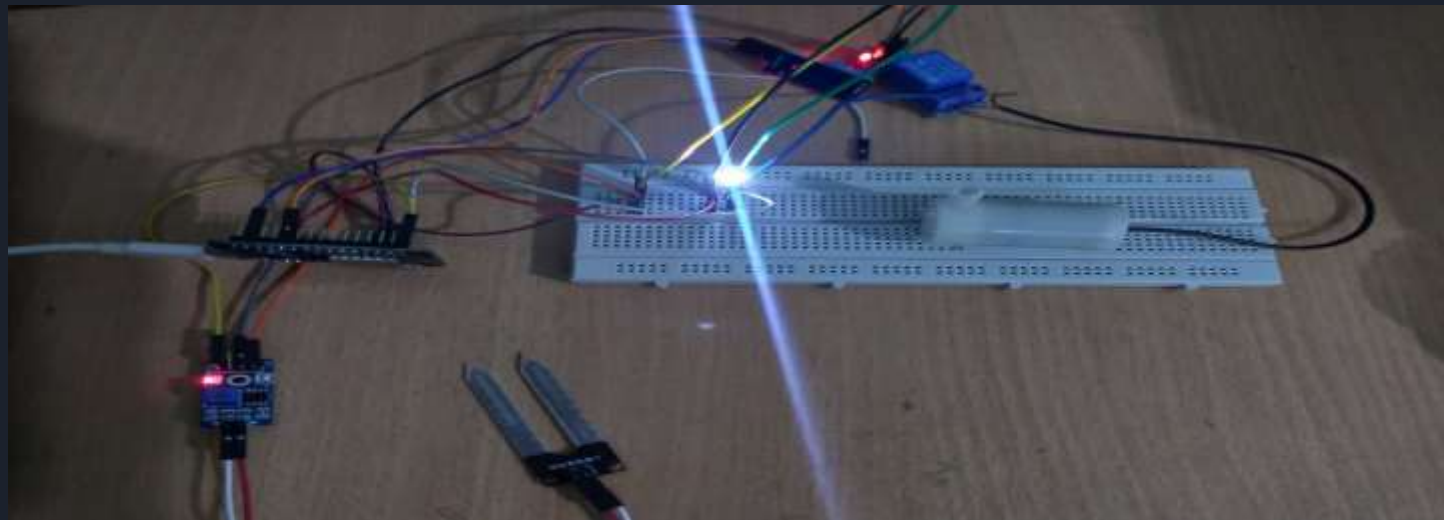
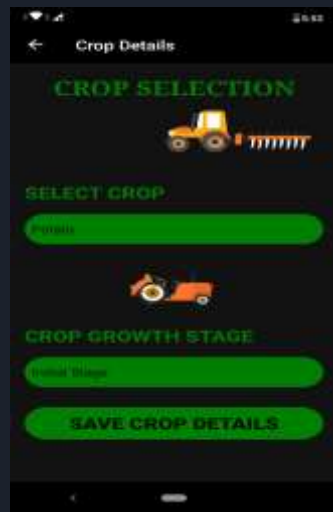
$$a = Y - b_1 X_1 - b_2 X_2$$

$$b_1 = \frac{(\sum x_2^2)(\sum x_1 y) - (\sum x_1 x_2)(\sum x_2 y)}{(\sum x_1^2)(\sum x_2^2) - (\sum x_1 x_2)^2}$$

$$b_2 = \frac{(\sum x_1^2)(\sum x_2 y) - (\sum x_1 x_2)(\sum x_1 y)}{(\sum x_1^2)(\sum x_2^2) - (\sum x_1 x_2)^2}$$

ARCHITECTURE (DESIGN, BLOCK DIAGRAM)





IOT MODULES USED

NodeMCU



DHT11 Sensor



Soil Moisture Sensor



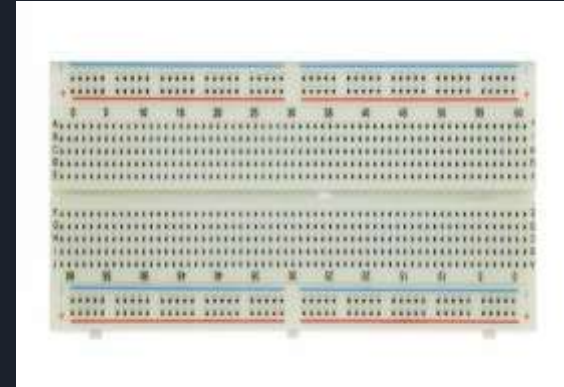
1- channel Relay Module



Motor Pump (mini DC 12V)



Breadboard



GROUP MEMBER'S CONTRIBUTION

1. AYUSH SHARMA (18BCE0172)

- IOT Device build up, Arduino coding and its integration with android app
- Coded PLSR algorithm in IOT device
- Connected IOT device to cloud using Arduino IDE coding and connected it to Android app too.

2. RITIK GUPTA (18BCE0154)

- Coded backend API to communicate with Historic data
- Coded API to fetch important factors for moisture content calculation.
- Worked on PLSR Algorithm in integrating it with backend API

3. NAMAN AGRAWAL (18BCE0107)

- Built Android App and connected it with IOT device
- Coded backend for cloud server (Server side code)
- Connected app and cloud together with IOT device. (Coded in Android Studio)



CONCLUSION

- The autonomous irrigation system we developed uses Artificial Intelligence learning and predictive algorithms to integrate perspicacity to subsisting concept of automatic irrigation systems. The methods discussed in this ppt can avail increase irrigation efficiency while decrementing effort required and avails water conservation compared to current irrigation methods.
- The system currently depends on sensors and weather station information for its calculation. This dependency can be superseded with on-premise sensors for deployment in rural areas widely found in the Indian subcontinent and arid regions where water is available in constrained quantity.

```
COM8
01:26:05.225 -> .....
01:26:14.241 -> Connected to Bohaima
01:26:14.241 -> IP address: 192.168.1.5
01:26:15.487 ->
01:26:15.487 ->
01:26:15.487 -> {"data":{"v_soilm_40_100cm":0.12,"dlwrf_avg":426,"skin_temp_min":22.9,"v_soilm_10_40cm":0.134,"soilm_40_100cm":72,"dlwrf_max":462,"dlwrf_net":-59.132,"soil":
01:26:20.207 -> ("coord":{"lon":77.46,"lat":23.28},"weather":{"id":803,"main":"clouds","description":"broken clouds","icon":"04n"},"base":{"station":"temp":299.15,
01:26:23.353 -> {"data":{"rh":69,"pod":"n","lon":77.46,"pres":949.474,"timezone":"Asia/Kolkata","ob_time":"2020-08-03 19:54","country_code":"IN","clouds":56,"ts":15913860
01:26:23.421 ->
01:26:33.010 -> Moisture Percentage = 0.00%
01:26:33.010 ->
01:26:38.042 -> MotorON
01:26:40.032 -> Updated Moisture:
01:26:40.032 -> 0.36
01:26:45.032 -> Updated Moisture:
01:26:45.032 -> 0.76
01:26:50.025 -> Updated Moisture:
01:26:50.025 -> 2.52
01:26:55.025 -> Updated Moisture:
01:26:55.025 -> 3.34
01:27:00.020 -> Updated Moisture:
01:27:00.020 -> 4.48
01:27:05.020 -> Updated Moisture:
01:27:05.020 -> 4.50
01:27:10.044 -> Updated Moisture:
01:27:10.044 -> 4.09
```

```
SmartGarden | Arduino 1.8.10
File Edit Sketch Tools Help

SmartGarden

//=====WEATHER API=====//

HTTPClient httpWeatherBit; //Declare an object of class HTTPClient

httpWeatherBit.begin("http://api.weatherbit.io/v1.0/current?lat=23.2756&lon=77.45604&key=10fe4d3c09b64cb1e44c67d49903570"); //Specify request destination
int httpBitCode = httpWeatherBit.GET(); //Send the request

if (httpBitCode > 0) { //Check the returning code

String WeatherBit = httpWeatherBit.getString(); //Get the request response payload
Serial.println(WeatherBit);

////////////////////////////////////JSON PARSEING////////////////////////////////////
DynamicJsonBuffer jsonBuffer(768);

char jsonW[WeatherBit.length() + 1];
WeatherBit.toCharArray(jsonW, WeatherBit.length() + 1);

JsonObject rootW = jsonBuffer.parseObject(jsonW);

// Test if parsing succeeds.
if (!rootW.success()) {

}

}

//=====WEATHER API=====//

Writing at 0x00000000... (92 B)
Writing at 0x00000000... (10 B)
Writing at 0x00000000... (14 B)
Writing at 0x00000000... (128 B)
Wrote 25012 bytes (27482 compressed) at 0x00000000 in 24.3 seconds (effective 124.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```




Firebase



Project Overview



Develop



Authentication



Database



Storage



Hosting



Functions



ML Kit

Quality



Crashlytics



Performance



Test Lab



App Distribution



Extensions

Spark

Free \$0/month

Upgrade

SmartGardener ▾



SolarRad

SumX1: -101.896!

SumX1X2: -0.03089!

SumX1Y: -892.306!

SumX1square: 3306.00!

SumX2: 0.47641!

SumX2Y: 3.12597!

SumX2square: 0.00715!

SumY: 271.14!



WindSpeed

-M7HrcGH0uRO705H5uci: 7.6733!

-M7HtvZuGftjtseRma9h: 7.6733!

-M7HwvNWqr8nOlokiYG7: 7.6733!

-M7IRDyZSTOBcuev8i4y: 4.168!

-M7IRvpYWp3HOn-fD_3l: 4.168!

-M7ISUbQjDsiGf14CIAo: 4.168!

-M7ITZhDnJSkNeje3dTd: 4.168!

-M7IUjSyyM5s-Wrf7amw: 4.168!

-M7IV9lqOqGyilERruiT: 4.168!

-M7IVn7a4_GAZYgn8uOj: 4.168!

-M7IWGa3r5fL_c9OMoNi: 4.168!

-M7IXBkDLiBB8lrmUdUn: 4.168!

-M7IZ0yk85_ydkS1VL4h: 4.168!



MAIN SOURCE CODE

```
void setup() {  
    delay(1000);  
    digitalWrite(4,HIGH);  
  
    pinMode(4, OUTPUT);  
  
    Serial.begin(115200);  
    WiFi.mode(WIFI_OFF);    //Prevents reconnection issue (taking too long to connect)  
    delay(1000);  
    WiFi.mode(WIFI_STA);    //Only Station No AP, This line hides the viewing of ESP as wifi hotspot  
  
    WiFi.begin(ssid, password);    //Connect to your WiFi router  
    Serial.println("");  
  
    Serial.print("Connecting");  
    // Wait for connection  
    while (WiFi.status() != WL_CONNECTED) {  
        delay(500);  
        Serial.print(".");  
    }  
    //If connection successful show IP address in serial monitor  
    Serial.println("");  
    Serial.print("Connected to ");  
    Serial.println(ssid);  
    Serial.print("IP address: ");  
    Serial.println(WiFi.localIP()); //IP address assigned to your ESP  
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);  
}
```



```
//          Main Program Loop
//=====================================================
void loop() {

    //=====IRRISTAT API=====//
    HTTPClient httpAgro; //Declare an object of class HTTPClient

    httpAgro.begin("http://api.weatherbit.io/v2.0/forecast/agweather?lat=23.2756&lon=77.4560&key=10fe4d3c09b64cb18e44c87dd9003578");//("http://api.weatherbit.io/v2.0/current?lat=23.2756&lon=77.4560&key=10fe4d3c09b64cb18e44c87dd9003578"); //Specify request destination
    int httpAPICode = httpAgro.GET();                                //Send the request

    if (httpAPICode > 0) { //Check the returning code

        String payload = httpAgro.getString(); //Get the request response payload

        int ind = payload.indexOf('');
        //////////////JSON PARSINNG////////////////////
        Serial.println("\n");
        Serial.println(payload.substring(0,ind+1));                //Print the response payload
        String final =payload.substring(0,ind+1)+"}"}";

        DynamicJsonBuffer jsonBuffer(907);

        char json[final.length() + 1];
        final.toCharArray(json, final.length() + 1);

        JsonObject& root = jsonBuffer.parseObject(json);

    }
};
```

```

// Test if parsing succeeds.
if (!root.success()) {
    Serial.println("parseObject() failed");
    return;
}
ET0 = root["data"][0]["evapotranspiration"];
Kc = Firebase.getFloat("/Kc");
KCET0 = ET0*Kc;
Firebase.setFloat("/KCET0",KCET0);
PrecipProb = root["data"][0]["precip"];
G = root["data"][0]["soilt_0_10cm"];
Firebase.pushFloat("/ET0",ET0);
Firebase.pushFloat("/GSoilFlux",G);
Firebase.setFloat("/CurrentET0",ET0);
SumY = Firebase.getFloat("/SumY");
SumY = SumY +ET0;
Firebase.setFloat("/SumY",SumY);
Firebase.setFloat("/PrecipProbability",PrecipProb);
// Print values.
//Serial.println(ET0);

}

```

```

httpAgro.end(); //Close connection

```

```

//=====PLSR ALGO=====//

```

```

float tempMean = (tempMin+tempMax)/2;
float p = (17.27*tempMean)/(tempMean+273.3);
float del = (4098*( 0.6108 * pow(2.71828,p)))/pow((tempMean +
237.3),2);
float gama = 0.000665*Pres;
float Rn = solarRad;
float u2 = windSpd;
float p1 = (17.27*tempMax)/(tempMax+273.3);
float p2 = (17.27*tempMin)/(tempMin+273.3);
float eTmax = 0.6108*(pow(2.71828,p1));
float eTmin = 0.6108*(pow(2.71828,p2));
float es = (eTmax + eTmin)/2;
float ea = (eTmin)*(rh/100);

float X1 = (del*(Rn-G))/(del + (gama*(1+ 0.34*u2)));
float X2 = (gama*u2*(es-ea))/(((tempMean+273)*(del + (gama*(1+
0.34*u2))));
Firebase.pushFloat("/X1",X1);
Firebase.pushFloat("/X2",X2);
float Y = ET0;

float X1Y = X1*Y;
float X2square = X2*X2;
float X1X2 = X1*X2;
float X2Y = X2*Y;
float X1square = X1*X1;

```

```
//=====PLSR ALGO=====//
float SumX1 = Firebase.getFloat("/SumX1");
SumX1 = SumX1+X1;
float SumX2 = Firebase.getFloat("/SumX2");
SumX2 = SumX2+X2;
float SumX1Y = Firebase.getFloat("/SumX1Y");
SumX1Y=SumX1Y+X1Y;
float SumX2square = Firebase.getFloat("/SumX2square");
SumX2square=SumX2square+X2square;
float SumX1X2 = Firebase.getFloat("/SumX1X2");
SumX1X2=SumX1X2+X1X2;
float SumX2Y = Firebase.getFloat("/SumX2Y");
SumX2Y=SumX2Y+X2Y;
float SumX1square = Firebase.getFloat("/SumX1square");
SumX1square=SumX1square+X1square;

Firebase.setFloat("/SumX1",SumX1);
Firebase.setFloat("/SumX2",SumX2);
Firebase.setFloat("/SumX1Y",SumX1Y);
Firebase.setFloat("/SumX2square",SumX2square);
Firebase.setFloat("/SumX1X2",SumX1X2);
Firebase.setFloat("/SumX2Y",SumX2Y);
Firebase.setFloat("/SumX1square",SumX1square);

int c = Firebase.getInt("/count");

float b1 = ((SumX2square*SumX1Y)-
(SumX1X2*SumX2Y))/((SumX1square*SumX2square)-
pow(SumX1X2,2));
```

```
float b2 = ((SumX1square*SumX2Y)-
(SumX1X2*SumX1Y))/((SumX1square*SumX2square)-
pow(SumX1X2,2));
float b0 = (SumY/c) - (b1*(SumX1/c)) - (b2*(SumX2/c));
float YFinal = b0 +(b1*X1)+(b2*X2);
float ET0F = (((0.408*X1)+(900*X2))*2)/10;
```


```
Firebase.pushFloat("/ET0F",ET0F);
Firebase.pushFloat("/b1",b1);
Firebase.pushFloat("/b2",b2);
Firebase.pushFloat("/b0",b0);
```

```
//=====SENSOR VALUES TO CLOUD=====//
```

```
sensor_analog = analogRead(A0);
moisture_percentage = ( 100 - ( ((sensor_analog-300)/724.00) * 100
) );
Serial.print("Moisture Percentage = ");
Serial.print(moisture_percentage);
Serial.print("%\n\n");
```

```
h = dht.readHumidity();
t = dht.readTemperature();
```

```
//Firebase.setFloat("/SensorTemperature",t);
//Firebase.setFloat("/SensorHumidity",h);
// Firebase.setFloat("/SensorMoisture",sensor_analog);
sensorMoisture= ((float)(moisture_percentage)/10);
delay(5000);
```



```
if (sensorMoisture<6 && sensorMoisture<KCET0 && PrecipProb<KCET0){
  Serial.println("\nMotorON");
  digitalWrite(4,LOW);
  delay(2000);
}
if (sensorMoisture<6 && sensorMoisture<KCET0 && PrecipProb<KCET0){
  Serial.println("\nMotorON");
  digitalWrite(4,LOW);
  delay(2000);
}
while(sensorMoisture<7 && sensorMoisture<KCET0 && PrecipProb<KCET0){
  sensor_analog = analogRead(A0);
  moisture_percentage = ( 100 - ( ((sensor_analog-300)/724.00) * 100 ) );
  sensorMoisture= ((float)(moisture_percentage)/10);
  Serial.println("Updated Moisture:");
  Serial.println(sensorMoisture);
  delay(5000);
}
digitalWrite(4,HIGH);
Serial.println("\nMOTOR OFF");

//=====//

Firebase.setInt("/count",c+1);

  delay(1800000);
}
//=====
```



THANK YOU