**INTRODUCTION**

**INTRODUCTION:**

Agriculture plays a vital role in India’s economy and all around the world. 54.6% of the India’s population is engaged in agriculture and allied activities (census 2011) and it contributes 17% to the country’s Gross Value Added (current price 2015-16, 2011-12 series). Nowadays, it’s more necessary than ever to increase the crop yields food grain production as the matter of fact the world population is increasing day-by-day. Agricultural system/methodologies, technology and tools directly affect the production of crops and its quality. With the modernization of the world, agriculture is also affected and reports say that technology can make revolution in agriculture pattern and output. Good technology and tools helps us to make better analysis leading to better prediction and to take right action before occurrence of any hazard, by the use of the method of data mining.

Before technology advancement, farmers were depended upon their own experience, natural recourses and prediction without formulation; which sometimes failed resultantly due to lesser experience and lesser information. But with the revolution of technologies and tools agriculture graph is going up with lesser man-power and resources (basically land, machine and tools) in comparison with earlier days, and machines have somewhat replaced the human begin and animals for the purpose of irrigation and fertilization. And now farmers can predict monsoon, weather changing etc. before hazard by the use of satellite weather forecasting, periodical surveys conducted by the government, NGOs and other information technologies.

But, the current survey methods are not sufficient which is evident by visiting agricultural farms, collecting samples, take measurements from all around the country (India) by few numbers of govt. and NGO’s surveyors. Whereby, it is a very lengthy, time taking and long duration periodical process. And through this method we can’t collect daily samples and with the limited resources (man-power, tools etc.).

The revolution of Internet of Things (IoT) and existing Internet, embedded devices, sensors and smart phones could solve the problem of collecting daily samples and measurements of agriculture fields all over the country (India) even all over the world without man-power interference (only required during setup/installation time and it can be done by farmers after short duration training), each and every minute, hour, day, week, month and/or year (depends upon requirements). Collected data; such as information like geo-locations using GPS; crops status by image capturing using mobile camera and temperature, weather, humidity, soil properties (water, chemical properties etc.) using

sensors etc.; could exchange data and information among each other (if needed). And could send data to the local mobile device (which would be connected with Internet) via network-connectivity; may be with or without Internet connection. After that, the local mobile device could send the data / information to the Distributed Clouds for Analysis and Data-mining for forecasting and hazard prevention. Obviously it will be produce massive data by each smart-device which diversely forms the distributed Big-Data. Because of 'ALWAYS ON-LINE TO TALK' for every THING each THING will generate huge data, or big data! It is impossible to exercise the traditional centralized Data-Centre, it should be distributed Big Data, which requires distributed clouds. These distributed clouds also need to be connected by wireless for lots of scenarios.

The proposed Smart Agriculture Model would solve the problem of human limitation and would produce better result of regression analysis (big-data-mining) due to large number of data in comparison to existing conventional survey technique because IoT based agricultural field survey is regular (minutely, hourly or daily) and could be install by trained farmers in larger area in comparison to irregular (half yearly and yearly), human based field survey of limited/small area which is done by well experienced few government or NGO’s surveyors. This project main focused on IoT and Big-Data-Mining analysis Technique.

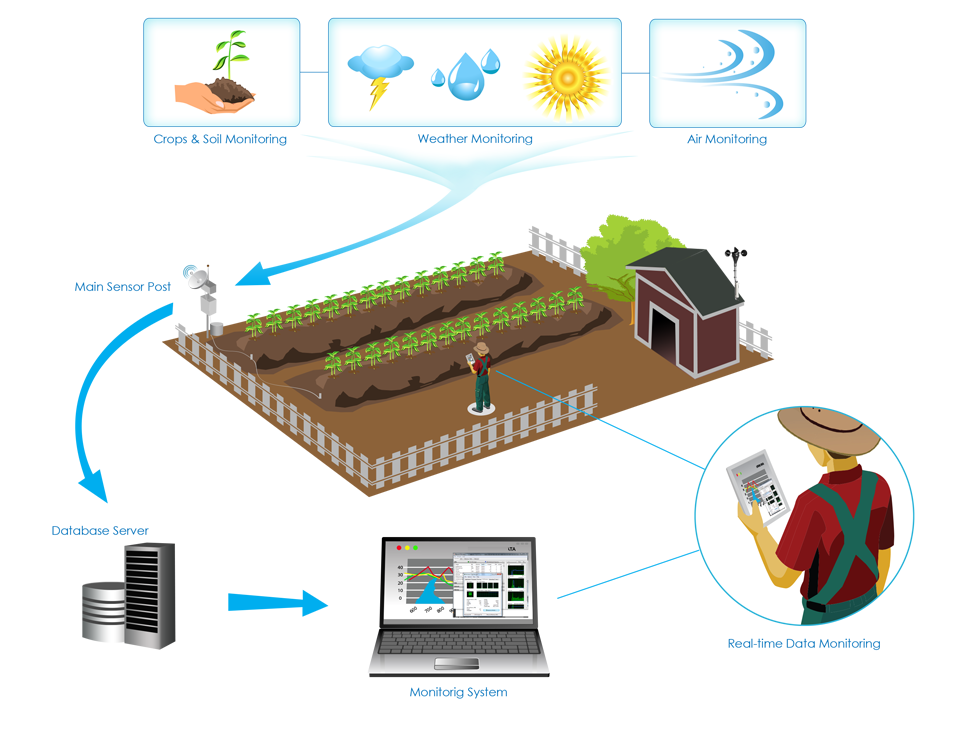
Internet Of Things (IoT) based agriculture model is better in comparison to conventional agriculture model due to its regular (huge data produce in short period i.e. per minute or per hour or per day) monitoring of agriculture field status like climate, environment, soil properties, crops status, temperature, pressure, humidity and GPS location etc. via IoT devices in comparison to conventional human based irregular (less data in long period i.e. quarterly or half yearly or yearly) agricultural survey of agriculture fields. These data and information are very important for making better quality/production of crops by analysis, prediction and learning using data-mining to make right decision and take action for hazard prevention. More data, information and knowledge give better forecasting and results.

There are several IoT based Agriculture Model have been proposed and implemented in several countries specially technology developed countries like US, Japan and China, where electronics, server, computer and IoT devices with software are available in very low cost due to production of these devices in their own countries. They are using Onsite IoT based Agriculture Model which require large number of IoT devices to be installed in agriculture fields with network connectivity basically Internet, regular huge electricity/power consumption and larger cloud servers for “big data mining” because this system produce huge amount of data per day.

But the scenario is different for India to implement IoT in agriculture field because electronics, server, computer and IoT devices with software cost is very high since make

in India production of these products with export quality is very lesser or almost nil. Therefore we have to import from technology developed countries which results into increase in cost of goods 3 to 1000 time[[1]](#footnote-1) due to currency exchange rate, transport cost etc. Electricity is not available in every rural areas and agriculture fields of India which is a bitter truth. Internet is very costly in compare to the income of maximum agricultural farmers of India. Maximum farmers are illiterate, poor and not technology friendly; they couldn’t operate smart phone, laptop and IoT till date. These things are the barriers to implement IoT in our Indian agriculture system. In the proposed model we are going to make intermediate solution to deal with the about problems.





**OBJECTIVE**

**OBJECTIVE**

* In this thesis project we are going to propose a Smart Agriculture Model (specially for India) with the use of Internet of Things (IoT)
* for collecting Agricultural data
  + Such as field geo-location, crops images, temperature, weather, humidity and soil properties etc. from fields using IoT Sensors with network connectivity.
* the collected data would be send to the distributed clouds
  + for analysis Data-mining, prediction and hazard prevention using regression analysis technique.
* The proposed Model would be solve the problems of human limitation and produce better result of regression analysis (big-data-mining) due to large number of data in comparison to existing conventional survey technique
  + because IoT based agricultural field survey would be regular,
  + can be installed by trained farmers in larger area in comparison.
  + irregular, human based field survey of limited/small area which is done by well experienced govt. or NGOs surveyors

**PROBLEM**

**STATEMENT**

**PROBLEM STATEMENT**

The conventional agriculture system (specially of India) is not sufficient to produce and supply the adequate product as per the demand of the increasing population also the produce is less than in comparison to the input and effort of farmers, due to lack of technology in our agricultural system/methodology since many things are manually done by farmers which consumes more time and uncertainty. For example, the existing agriculture survey system of India is based on irregular and long interval (half yearly and yearly) human based field survey in limited/small areas which is done by well experienced few government or NGO’s surveyors. Which is indeed a very time taking process and it is not possible to survey all the agricultural fields and collecting samples by the use of this system due to human limitation and can’t collect regular and short interval (minutely, hourly or daily) data as it is very necessary to generate and predict most probable result with the help of Big-Data-mining analysis technique (s) to prevent hazards by taking alternatives

**PROPOSED**

**TECHNIQUE**

**PROPOSED TECHNIQUE**

In the proposed model we used IoT (agricultural) embedded Drone[[2]](#footnote-2)with Solar Panels and wireless internet connectivity (as shown in Fig. 2) which will fly over agriculture fields and collect data. It will be controlled and operated by government offices and NGO type organizations of Blocks. This model replaces onsite IoT model to remote IoT model withlittle variation of periodical data collection (i.e. less data produce periodically) but economically this model will be proven as a blessing especially for India

* 1. PROPOSED MODEL STEPS

1. ***Fly Drone****:* IoT (agricultural) embedded Drone with Solar Panels and wireless internet connectivity fly over agriculture fields. It will be controlled and operated by Block’s govt. offices/head quarters and some NGO type organizations.
2. ***Collect agriculture fields’ data using IoT (agricultural) sensors****.* Data may be climate, environment, soil properties, crops status, temperature, pressure, humidity and GPS location etc. Preprocessing, measurement and interpretation maybe done here complete or partially.
3. ***Save data****:* Save data on Drone Memory (retrieve saved data after returning Drone to controller) and send to operator via wireless network from site.
4. ***Send to Cloud Server****:* Operator will be send retrieved data and information to the Central or Cloud server.
5. ***Analyze, Predict and Learn****:* In the server analysis, prediction and learning using data-mining processes have been done on the collected data and information.
6. ***Act and Optimize****:*On the basis of analysis, prediction and learning government, NGO, researchers and farmers can make right decision and take action for hazard prevention and produce better quality crops. Contact to agricultural fields owner/farmers if it is required.

In existing IoT models step (a) doesn’t involved and this makes the proposed “Smart Remote Internet of Things based Agriculture Model for INDIA” different from existing IoT models.

Note, this model is developed especially for India and Indian states but it will also beneficial for other countries and states as well; which are not technically developed plus technically developed countries and states as an alternative model to reduce IoT implementation cost on their agricultural fields with little variation of periodical data collection

**REQUIREMENT ANALYSIS**

**Hardware Specifications:**

1. Rectifier

2. Regulator

3. LCD Display

4. Water Pump

5. Sensors:

A>Temperature Sensor

B>Soil Moisture Sensor

C>Humidity Sensor

6. Raspberry Pi

**Software Specifications:**

Operating System: Windows/Linux

Programming Language: Java and Python

Database: Oracle

Driver: JDBC

**DATA DICTIONARY**

**Admin Registration**

Column name Type Constraints

UserID int Primary key

Name varchar2(40) Not null

Address varchar2(120) Not null

Mobile int Not null,Unique

Sex varchar(10) Not null

UID(Aadhar Card) int Not null,Unique

Age int Not null

**Crop Details**

Column Name Type Constraints

Record\_Id int Primary key

Soil type varchar2(30) Not Null

Crop type varchar2(40) Not Null

Temperature int Not Null

Soil Humidity int Not Null

**DATA FLOW**

**DIAGRAM**

**LEVEL 0**

USER CONFIGURE THE CROP DETAILS

ADMIN REGISTER IN APPLICATION

USER CAN CONTROL IRRIGATION SYSTEM

USER CAN VIEW LIVE STATUS OF IRRIGATION SYSTEM

Level-1DUJGBHHKJFIJ

**POWER SUPPLY**

**REGULATOR**

**RECTIFIER**

**DISPLAY**

**APPLICATION CONTROL**

**UNIT**

**RASPBERRY**

**PI**

**WATER PUMP**

**SOIL MOISTURE SENSOR**

**ENTITY RELATIONSHIP DIAGRAM**

**E-R Diagram:**

HAS

Crop Details

Admin Registration

**TOOLS DESCRIPTION**

**JAVA SWING**

**Swing** is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

Swing is currently in the process of being replaced by JavaFX.

Architecture

Swing is a platform-independent, Model-view-contrellor GUI framework for Java, which follows a single threaded programming model. Additionally, this framework provides a layer of abstraction between the code structure and graphic presentation of a Swing-based GUI.

#### Extensible

Swing is a highly modular-based architecture, which allows for the "plugging" of various custom implementations of specified framework interfaces: Users can provide their own custom implementation(s) of these components to override the default implementations using Java's inheritance mechanism.

Swing is a component-based framework, whose components are all ultimately derived from the javax.swing.jcomponent class. Swing objects asynchronously fire events, have bound properties, and respond to a documented set of methods specific to the component. Swing components are java beans components, compliant with the Java Beans Component Architecture specifications.

#### Customizable

Given the programmatic rendering model of the Swing framework, fine control over the details of rendering of a component is possible. As a general pattern, the visual representation of a Swing component is a composition of a standard set of elements, such as a border, inset, decorations, and other properties. Typically, users will programmatically customize a standard Swing component (such as a JTable) by assigning specific borders, colors, backgrounds, opacities, etc. The core component will then use these properties to render itself. However, it is also completely possible to create unique GUI controls with highly customized visual representation.

#### Configurable

Swing's heavy reliance on runtime mechanisms and indirect composition patterns allows it to respond at run time to fundamental changes in its settings. For example, a Swing-based application is capable of hot swapping its user-interface during runtime. Furthermore, users can provide their own look and feel implementation, which allows for uniform changes in the look and feel of existing Swing applications without any programmatic change to the application code.

**Lightweight UI**

Swing's high level of flexibility is reflected in its inherent ability to override the native host operating system (OS)'s GUI controls for displaying itself. Swing "paints" its controls using the Java 2D APIs, rather than calling a native user interface toolkit. Thus, a Swing component does not have a corresponding native OS GUI component, and is free to render itself in any way that is possible with the underlying graphics GUIs.

However, at its core, every Swing component relies on an AWT container, since (Swing's) JComponent extends (AWT's) Container. This allows Swing to plug into the host OS's GUI management framework, including the crucial device/screen mappings and user interactions, such as key presses or mouse movements. Swing simply "transposes" its own (OS-agnostic) semantics over the underlying (OS-specific) components. So, for example, every Swing component paints its rendition on the graphic device in response to a call to component.paint(), which is defined in (AWT) Container. But unlike AWT components, which delegated the painting to their OS-native "heavyweight" widget, Swing components are responsible for their own rendering.

#### Loosely coupled and MVC

The Swing library makes heavy use of the Model/View/Controller software design pattern which conceptually decouples the data being viewed from the user interface controls through which it is viewed. Because of this, most Swing components have associated models (which are specified in terms of Java interfaces), and the programmers can use various default implementations or provide their own. The framework provides default implementations of model interfaces for all of its concrete components. The typical use of the Swing framework does not require the creation of custom models, as the framework provides a set of default implementations that are transparently, by default, associated with the corresponding JComponent child class in the Swing library. In general, only complex components, such as tables, trees and sometimes lists, may require the custom model implementations around the application-specific data structures. To get a good sense of the potential that the Swing architecture makes possible, consider the hypothetical situation where custom models for tables and lists are wrappers over DAO and/or EJB services.

Typically, Swing component model objects are responsible for providing a concise interface defining events fired, and accessible properties for the (conceptual) data model for use by the associated JComponent. Given that the overall MVC pattern is a loosely coupled collaborative object relationship pattern, the model provides the programmatic means for attaching event listeners to the data model object. Typically, these events are model centric (ex: a "row inserted" event in a table model) and are mapped by the JComponent specialization into a meaningful event for the GUI component.

For example, the JTable has a model called Table model that describes an interface for how a table would access tabular data. A default implementation of this operates on a two-dimensional array.

The view component of a Swing JComponent is the object used to graphically represent the conceptual GUI control. A distinction of Swing, as a GUI framework, is in its reliance on programmatically rendered GUI controls (as opposed to the use of the native host OS's GUI controls). Prior to Java 6 Update 10, this distinction was a source of complications when mixing AWT controls, which use native controls, with Swing controls in a GUI Finally, in terms of visual composition and management, Swing favors relative layouts (which specify the positional relationships between components) as opposed to absolute layouts (which specify the exact location and size of components). This bias towards "fluid"' visual ordering is due to its origins in the applet operating environment that framed the design and development of the original Java GUI toolkit. (Conceptually, this view of the layout management is quite similar to that which informs the rendering of HTML content in browsers, and addresses the same set of concerns that motivated the former)

**MYSQL**

MySql includes full featured and government certified implementation of the ANSI/ISO standard SQL language. The MySql server can be an active part of the applications enforcing data integrity and business rules through declarative means (that is without programming) as well as through the industry's most complex and robust set of programming features, such as stored procedures and triggers. MySql also includes industrial strength database facilities enabling enterprise- wide access to data, no matter where it is stored or on what kind of platforms.

Last but not least, MySql? maximum the performance and reliability of on-line transaction processing applications through a range of architectural features such as row-level locking, on-line backup and recovery and support for a wide variety of platforms.

All businesses large and small must manage the company data in a fashion that business to run smoothly some businesses might use file cabinet to manage their data but most choose to use computerized database management systems that efficiently store, retrieve and manage large amount of data.

MySql is a multi-user database management system - a software product that specializes in managing a single, shared set of information among concurrent users.

MySql is a comprehensive operating environment that provides a set of functional programs that can be used as tools to build structures and perform tasks. Because applications developed on Professional MySql are completely portable to other versions of the program. We can create a complex application in a single user environment and then move it to multi-user platform.

In MySql, all data is stored and displayed in tables. A table consists of columns and of rows. A view, referred to as virtual table is a derived table that cam be created for purposes of display while the table while the tables they are derived from are called base tables. A view can be a combination of two base tables or a subset of one base table.

Because MySql is a relational system, we connect the data stored in various table to increase its usefulness and to avoid duplication. Selection is the process of producing a new table consisting of a set of rows from another table that match certain specified criteria. Projection is the process of creating a table from set of columns from another's table that match certain criteria. A join produces a new table that is the union of all rows in two tables less any duplicate rows.

At the core the MySql is SQL, which stands for structured query language. SQL is the language that is used to communicate with MySql. It consists of a set of common English words, such as 'Select and 'Create' which can be arrange din a highly structured commands and statements in order to access and manipulate data stored in a relational database.

The standard set of SQL commands fall into fan categories. Data Definition Language (DOL) commands are for creating and altering the structure of the database. Data manipulation language (DML) commands are for adding and modifying data. Data control language (DCL) commands are for controlling access to the database. Finally, query commands are used to extract information from the database.

**MySql Environment:-** MySql is a modular system that consists of the MySql database and several functional programs. These components can be viewed as tool; each has a special purpose and can be accessed and used independently.

MySqls tool do four major kinds of work:

* Database management
* Data base access and manipulation
* Programming
* Connectivity
* **Database Management Tools :-** This category includes the core programs of the MySql database management system: the MySql database with its associate tables and views, which are stored in MySqls data dictionary and a group of house keeping utilities listed in table.
* The data dictionary is MySql control documentation system. It stores information related to every facet of the database system. User names user access rights table names, table attribute names, table storage information and auditing data for disaster recovery are all stored in the data dictionary. A data dictionary can be either active or passive. MySql data dictionary is active, meaning that the system automatically updates the dictionary in 'real time' that is, as changes are made. The MySql database programs then reference the data dictionary as required.
* **Data Access And Manipulation tools :-** All of MySql data access and manipulation tools have one thing in common; they are firmly based on ANSI standard SQL. This programs on the gateway to MySql, the tool that will be used to be access and manipulate data, as well as to design or to use applications, each provides a separate point of entry and a unique approach to the MySql system. SQL \* Plus allows direst access to database with SQL commands. SQL \* forms offer a user friendly way to and to use forms. SQL \* reports writer can be used to create formatted output. SQL \* menu provides a way to integrate application using menus.
* **SQL\*Plus :-** SQL \* Plus is the main direct access interface to. the MySql relational DBMS. For DBA's, high level system developer or others who want to go straight to the heart if the MySql DBMS, SQL \* Plus is the right path. the program provides a full implementation of ANSI standard SQL, plus an assortment of extension that can be used when strict standardization is not required. With SQL \* Plus, we can Query our database and perform data definition, data manipulation, and data control operation. Extensive online help is also available.

**SOL \* Forms :-** SQL \* Forms provides a convenient method for non experts to query a data base and update, delete or add information. Its form driven, query by example approach is boom for users not familiar with the SQL language.

**Programming Tools :-** One of the most important categories of the tools available from MySql is its series pro plus programming interface these pre-compilers software tools provide convenient and is easy to use method of incorporating MySql SQL statements in high level programming language. Using these products we can run a supported high level procedure language merged with SQL data language statements through pre- compiler, which converts the SQL statements into that languages native code so that they can be competed normally. Currently, users of MySql can interface with Cobol, C, Fortran, PL/I, Ada. And Pascal, with C' being the most popular among this.

**Connectivity Tools :-** Connectivity tools make MySql available to networks and to other database managers. SQL .Star is a group of product including SQL .Net and SQL .Connect that allows the user to use data stored on remote machine as' if they are available locally.

**a) Security and control :-** MySql has several features that ensures integrity of the database. If an interruption occurs in processing, a roll back can reset the database to a point before the disaster. If a restore is necessary, MySql has a roll forward command for recreating the database to its most recent safe point.

MySql provides users with several function for securing data. Grant and Revoke command limit access to information down to the row and column levels. Views are a valuable feature for limiting access to the primary tables in the database.

**b) Competitiveness :-** MySql consists of routines that quickly and automatically calculate the best path to the data, and sophisticated fast indexing routines. MySql unique clustering techniques for storing data on the disk are another performance gain.

Additional function help control complex database installations. The active data dictionaries, which automatically update and log modifications to the database, provide documentation. A variety of auditing commands and journaling ease errors detection and tracking tasks. Database table, partition, and space altering " on the fly eliminates cumbersome data offloading and loading from the modification process.

**c) Applications Development :-** SQL \* Forms is an excellent user friendly tool for quickly creating forms. MySql provides unique control device called triggers to influence users action on field before, during and after data input. These triggers can execute SQL commands, native SQL \* Forms. Or External procedural language subroutines from within a form.

With SQL \* Menu we can link all forms, programs, queries in an easily maintained, secure menu structure.

**Creating database**

* Collection of Tables and Database Objects.
* There are many types of s/w to work on Database.

**E.g. 🡪**MySql, MS Sql server, Ms Access, My Sql, excel, FoxPro, Informix,

* There are two tools.

1. Sql Plus (Structure query langue plus, to write codes of Sql.)
2. Pl Sql (Procedure level Sql)

* Sql🡪 Structure query language, which is pre defined commands.

It has three categories:-

1. DDL 🡪 Data definition Language.

(Create, alter etc)

1. DML 🡪 Data manipulation Language.

Select, Update, Insert, and Delete.

1. DCL🡪 Data Control language.

Commit, Rollback.

Orcale is RDBMS DATA base and it is Server based products so it can handle many Clients at a time,on internet it is used in wilde form.

**IT has three Tools.**

1. SQL Plus (interface)
2. SQL (one command at a time on SQL Plus)
3. PL SQL (Procedure level Language)

**Sql**

It stands for structure query language,it has three parts.

1> DDL Data definition language.

Like Create Table,Alter Table, Drop table etc.

2> DML Data Manipulation Language

like insert into,delete from,update,select.

2>DCL Data Control Language.

Like commit,rollback.

MySql gives a powerful tool is called Constraint which is used to make restriction on field and used to make relation between Tables.

Like

\* Primary key

\* References key

\* NOT NULL

\* Composite key

\* Check

\* Default

# JDBC - Java Database Connectivity

## JDBC?

Java Database Connectivity (**JDBC**) is a programming framework for Java developers writing programs that access information stored in databases, spreadsheets, and flat files. JDBC is commonly used to connect a user program to a "behind the scenes" database, regardless of what database management software is used to control the database. In this way, JDBC is cross-platform . This article will provide an introduction and sample code that demonstrates database access from Java programs that use the classes of the JDBC API, which is available for free download from Sun's site

A database that another program links to is called a data source. Many data sources, including products produced by Microsoft and MySQL, already use a standard called Open Database Connectivity (ODBC). Many legacy C and Perl programs use ODBC to connect to data sources. ODBC consolidated much of the commonality between database management systems. JDBC builds on this feature, and increases the level of abstraction. JDBC-ODBC bridges have been created to allow Java programs to connect to ODBC-enabled database software

This article assumes that readers already have a data source established and are moderately familiar with the Structured Query Language (SQL), the command language for adding records, retrieving records, and other basic database manipulations. See Hoffman's tutorial on SQL if you are a beginner or need some refreshing

## Using a JDBC driver

Regardless of data source location, platform, or driver (MySQL, Microsoft, etc.), JDBC makes connecting to a data source less difficult by providing a collection of classes that abstract details of the database interaction. Software engineering with JDBC is also conducive to module reuse. Programs can easily be ported to a different infrastructure for which you have data stored (whatever platform you choose to use in the future) with only a driver substitution.

As long as you stick with the more popular database platforms (MySQL, Informix, Microsoft, MySQL, etc.), there is almost certainly a JDBC driver written to let your programs connect and manipulate data. You can download a specific JDBC driver from the manufacturer of your database management system (DBMS) or from a third party (in the case of less popular open source products) . The JDBC driver for your database will come with specific instructions to make the class files of the driver available to the Java Virtual Machine, which your program is going to run. JDBC drivers use Java's built-in DriverManager to open and access a database from within your Java program.

To begin connecting to a data source, you first need to instantiate an object of your JDBC driver. This essentially requires only one line of code, a command to the DriverManager, telling the Java Virtual Machine to load the bytecode of your driver into memory, where its methods will be available to your program. The String parameter below is the fully qualified class name of the driver you are using for your platform combination:

Class.forName("org.gjt.mm.mysql.Driver").newInstance();

## Connecting to your database

To actually manipulate your database, you need to get an object of the Connection class from your driver. At the very least, your driver will need a URL for the database and parameters for access control, which usually involves standard password authentication for a database account.

As you may already be aware, the Uniform Resource Locator (URL) standard is good for much more than telling your browser where to find a web page:

http://www.vusports.com/index.html

The URL for our example driver and database looks like this:

jdbc:mysql://db\_server:3306/contacts/

Even though these two URLs look different, they are actually the same in form: the protocol for connection, machine host name and optional port number, and the relative path of the resource. Your JDBC driver will come with instructions detailing how to form the URL for your database. It will look similar to our example.

You will want to control access to your data, unless security is not an issue. The standard least common denominator for authentication to a database is a pair of strings, an account and a password. The account name and password you give the driver should have meaning within your DBMS, where permissions should have been established to govern access privileges.

Our example JDBC driver uses an object of the Properties class to pass information through the DriverManager, which yields a Connection object:

Properties props = new Properties();

props.setProperty("user", "contacts");

props.setProperty("password", "blackbook");

Connection con = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/contacts/", props);

Now that we have a Connection object, we can easily pass commands through it to the database, taking advantage of the abstraction layers provided by JDBC.

## Structuring statements

Databases are composed of tables, which in turn are composed of rows. Each database table has a set of rows that define what data types are in each record. Records are also stored as rows of the database table with one row per record. We use the data source connection created in the last section to execute a command to the database.

We write commands to be executed by the DBMS on a database using SQL. The syntax of a SQL statement, or query, usually consists of an action keyword, a target table name, and some parameters. For example:

INSERT INTO songs VALUES (

"Jesus Jones", "Right Here, Right Now");

INSERT INTO songs VALUES (

"Def Leppard", "Hysteria");

These SQL queries each added a row of data to table "songs" in the database. Naturally, the order of the values being inserted into the table must match the order of the corresponding columns of the table, and the data types of the new values must match the data types of the corresponding columns. For more information about the supported data types in your DBMS, consult your reference material.

To execute an SQL statement using a Connection object, you first need to create a Statement object, which will execute the query contained in a String.

Statement stmt = con.createStatement();

String query = ... // define query

stmt.executeQuery(query);

### Example: Parsing a text file into a database table

In the course of modernizing a record keeping system, you encounter a flat file of data that was created long before the rise of the modern relational database. Rather than type all the data from the flat file into the DBMS, you may want to create a program that reads in the text file, inserting each row into a database table, which has been created to model the original flat file structure.

In this case, we examine a very simple text file. There are only a few rows and columns, but the principle here can be applied and scaled to larger problems. There are only a few steps:

* Open a connection to the database.
* Loop until the end of the file:
  + Read a line of text from the flat file.
  + Parse the line of text into the columns of the table.
  + Execute a SQL statement to insert the record.

Here is the code of the example program:

import java.io.\*;

import java.sql.\*;

import java.util.\*;

public class TextToDatabaseTable {

private static final String DB = "contacts",

TABLE\_NAME = "records",

HOST = "jdbc:mysql://db\_lhost:3306/",

ACCOUNT = "account",

PASSWORD = "nevermind",

DRIVER = "org.gjt.mm.mysql.Driver",

FILENAME = "records.txt";

public static void main (String[] args) {

try {

// connect to db

Properties props = new Properties();

props.setProperty("user", ACCOUNT);

props.setProperty("password", PASSWORD);

Class.forName(DRIVER).newInstance();

Connection con = DriverManager.getConnection(

HOST + DB, props);

Statement stmt = con.createStatement();

// open text file

BufferedReader in = new BufferedReader(

new FileReader(FILENAME));

// read and parse a line

String line = in.readLine();

while(line != null) {

StringTokenizer tk = new StringTokenizer(line);

String first = tk.nextToken(),

last = tk.nextToken(),

email = tk.nextToken(),

phone = tk.nextToken();

// execute SQL insert statement

String query = "INSERT INTO " + TABLE\_NAME;

query += " VALUES(" + quote(first) + ", ";

query += quote(last) + ", ";

query += quote(email) + ", ";

query += quote(phone) + ");";

stmt.executeQuery(query);

// prepare to process next line

line = in.readLine();

}

in.close();

}

catch( Exception e) {

e.printStackTrace();

}

}

// protect data with quotes

private static String quote(String include) {

return("\"" + include + "\"");

}

}

## Conclusions

In this article, you saw a quick introduction to manipulating databases with JDBC. More advanced features of JDBC require a greater knowledge of databases. See the references for more articles about JDBC and its applications. As a Java programmer, JDBC is a good tool to have in your arsenal.

I encourage you to copy the code in this article to your own computer. With this article and documentation for another JDBC driver, you are on your way to creating data source-driven Java programs. Experiment with this code, and adapt it to connect to data sources available to you.

**DSN LESS CONECTION**

**1> Oracle**

Class.forName("com.org.driver.OracleDriver");  
> > String connString = "jdbc:thin://localhost”;   
> > Connection conn = DriverManager.getConnection(connString, "Scott",  
> > "password");

**2> MS ACCESS**

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  
> > String connString = "jdbc:odbc:Driver={Microsoft Access Driver  
> > (\*.mdb)};DBQ=c:/test.mdb;DriverID=22;READONLY=true}";  
> > Connection conn = DriverManager.getConnection(connString, "Admin",  
> > "password");

**3> My Sql**

Class.forName("com.mm.driver.MySqlDriver");  
> > String connString = "jdbc:mm://localhost”;   
> > Connection conn = DriverManager.getConnection(connString, "Scott",  
> > "password");

**CODING**

**User login and Registration**

import javax.swing.\*;

import javax.swing.ImageIcon;

import java.awt.event.\*;

import java.awt.Toolkit;

public class AutoIrrigation extends JFrame implements ActionListener

{

JLabel l1,l2,l3,l4,l5,l6,l7,l8,l9,l10;

JTextField t1,t2,t3,t4,t5,t6,t7,t8;

JPasswordField p1;

JButton b1,b2,b3,b4;

JRadioButton r1,r2;

void display()

{

Toolkit kit=Toolkit.getDefaultToolkit();

int width=(int)kit.getScreenSize().getWidth();

int height=(int)kit.getScreenSize().getHeight();

ImageIcon img=new ImageIcon("fb\_post\_kk.jpg");

JLabel jl=new JLabel(img);

l1=new JLabel("User Registration");

l2=new JLabel("name");

l3=new JLabel("Age");

l4=new JLabel("Address");

l5=new JLabel("Gender");

l6=new JLabel("Mobile no");

l7=new JLabel("Aadhar no");

l8=new JLabel("User id");

l9=new JLabel("Password");

l10=new JLabel("User Login");

t1=new JTextField("");

t2=new JTextField("");

t3=new JTextField("");

t4=new JTextField("");

t5=new JTextField("");

t6=new JTextField("");

t7=new JTextField("");

t8=new JTextField("");

r1=new JRadioButton("male");

r2=new JRadioButton("female");

ButtonGroup bg=new ButtonGroup();

bg.add(r1);

bg.add(r2);

b1=new JButton("Register");

b2=new JButton("reset");

b3=new JButton("Login");

b4=new JButton("Cancel");

jl.setBounds(0,0,width,height);

l1.setBounds(50,50,140,50);

l2.setBounds(100,100,120,50);

t1.setBounds(210,100,120,50);

l3.setBounds(100,170,120,50);

t2.setBounds(210,170,120,50);

l4.setBounds(100,240,120,50);

t3.setBounds(210,240,120,50);

l5.setBounds(100,310,120,50);

r1.setBounds(210,310,110,50);

r2.setBounds(320,310,120,50);

l6.setBounds(100,380,120,50);

t4.setBounds(210,380,120,50);

l7.setBounds(100,450,120,50);

t5.setBounds(210,450,120,50);

l8.setBounds(690,100,100,50);

t7.setBounds(760,100,120,40);

l9.setBounds(680,160,120,50);

t8.setBounds(760,160,120,40);

l10.setBounds(600,50,120,50);

b1.setBounds(130,580,100,50);

b2.setBounds(240,580,100,50);

b3.setBounds(690,220,100,50);

b4.setBounds(800,220,100,50);

add(l1);add(l2);add(l3);add(l4);add(l5);add(l6);add(l7);add(t1);add(t2);add(t3);add(t4);add(t5);add(t6);add(b1);add(b2);add(r1);add(r2);add(l8);add(l9);

add(t7);add(t8);add(b3);add(b4);add(l10);add(jl);

setSize(width,height);

setLayout(null);

setVisible(true);

b1.addActionListener(this);

b2.addActionListener(this);

b3.addActionListener(this);

b4.addActionListener(this);

}

public void actionPerformed(ActionEvent ae)

{

if(ae.getSource()==b3)

{

FarmerDetails s=new FarmerDetails();

s.see();

}

}

public static void main(String[] s)

{

AutoIrrigation g=new AutoIrrigation();

g.display();

}

}

**Temperature and humidity**

import java.io.DataInputStream;

import java.sql.\*;

import java.io.InputStream;

import java.sql.Connection;

import javax.swing.JFrame;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

class ReadTH extends JFrame

{

String[] getDTH()

{

// 0-TEMP, 1- HUMIDITY, 2- DATE, 3- Time

String HT[]={"-55","-55","0000-00-00","00:00:00"};

int n;

int n1;

try{

Runtime run=Runtime.getRuntime();

run.exec("gpio -g mode 21 out");

Process pr=run.exec("python3 /home/pi/Desktop/mypi/jtmp.py");

InputStream in=pr.getInputStream();

DataInputStream din=new DataInputStream(in);

String line=din.readLine();

if(line!=null)

{

String[] two=line.split(",");

if(two!=null && two.length==4)

{

HT[0]=two[0];

n1=(int)(Double.parseDouble(HT[0]));

HT[1]=two[1];

n=(int)(Double.parseDouble(HT[1]));

if(n>39 && n1>30)

{

run.exec("gpio -g write 21 0");

System.out.println("MOTOR ON");

}

else

{ System.out.println("MOTOR OFF");

run.exec("gpio -g write 21 1");

}

HT[2]=two[2];

HT[3]=two[3];

}

}

}catch(Exception ee)

{

}

return(HT);

}

public static void main(String fg[]) throws Exception

{

System.out.println("Ranchi");

final ReadTH v1=new ReadTH();

final Connection con=Conn1.getConnection();

Runnable run=new Runnable()

{

public void run()

{

while(true)

{

String HT[]=v1.getDTH();

System.out.println("TEMPRAT="+HT[1]+"\t"+"HUMIDITY="+HT[0]+"\t"+"DATE="+HT[2]+"\t"+"TIME="+HT[3]);

try{

PreparedStatement ed=con.prepareStatement("insert into weatherinfo values(?,?,?,?)");

ed.setString(1,"RANCHI”);

ed.setFloat(2,Float.parseFloat(HT[1]));

ed.setFloat(3,Float.parseFloat(HT[0]));

ed.setString (4,HT[2]+" "+HT[3]);

ed.executeUpdate();

ed.close();

Thread.sleep(5000);

}

catch(Exception eer)

{

}

}

} // run()

}; // run object

Thread th=new Thread(run); th.start();

}

}

**Soil Moisture**

import java.io.DataInputStream;

import java.io.InputStream;

public class Moisture {

String[] getSoil()

{

// 0-TEMP, 1- HUMIDITY, 2- DATE, 3- Time

String HT[]={"NONE","0000-00-00","00:00:00"};

int n;

try{

Runtime run=Runtime.getRuntime();

Process pr=run.exec("python3 /home/pi/Desktop/mypi/moisture.py"); // 14 PIN

InputStream in=pr.getInputStream();

DataInputStream din=new DataInputStream(in);

String line=din.readLine();

if(line!=null)

{

String[] two=line.split(",");

if(two!=null && two.length==3)

{

HT[0]=two[0];

HT[1]=two[1];

HT[2]=two[2];

}

}

}catch(Exception ee)

{

}

return(HT);

}

public static void main(String fg[]) throws Exception

{

System.out.println("Ranchi");

final Moisture v1=new Moisture();

Runnable run=new Runnable()

{

public void run()

{

while(true)

{

String M[]=v1.getSoil();

System.out.println(M[0]+" , "+M[1]+" , "+M[2]);

try{Thread.sleep(5000);}catch(Exception ee){}

}

}};

Thread th=new Thread(run);

th.start();

}

}

**Farmer Details**

import javax.swing.\*;

import java.awt.Toolkit;

public class FarmerDetails extends JFrame

{

JLabel l1,l2,l3,l4,l5;

JTextField j1,j2,j3,j4;

void see()

{

try

{

Toolkit kit=Toolkit.getDefaultToolkit();

int height=(int)kit.getScreenSize().getHeight();

int weight=(int)kit.getScreenSize().getWidth();

l1=new JLabel("Registered User");

l2=new JLabel("Crop Details");

l3=new JLabel("Temperature");

l4=new JLabel("Soil Humidity");

j1=new JTextField("");

j2=new JTextField("");

j3=new JTextField("");

l1.setBounds(50,30,140,100);

l2.setBounds(110,110,100,50);j1.setBounds(230,110,80,50);

l3.setBounds(110,150,100,100);j2.setBounds(230,180,80,50);

l4.setBounds(110,230,100,100);j3.setBounds(230,250,80,50);

add(l1);

add(l2);

add(l3);

add(l4);

add(j1);

add(j2);

add(j3);

setSize(weight,height);

setLayout(null);

setVisible(true);

}

catch(Exception ee)

{}

}

public static void main(String [] args)

{

FarmerDetails s=new FarmerDetails();

s.see(); }}

**Farming Data**

import javax.swing.JFrame;

import javax.swing.table.DefaultTableModel;

import javax.swing.JTable;

import javax.swing.JScrollPane;

import java.sql.\*;

import java.sql.DriverManager;

import java.sql.Connection;

import java.awt.Toolkit;

public class ShowData extends JFrame

{

DefaultTableModel grid=null;

JTable tab=null;

JScrollPane pan=null;

String header[]={"sno","soil\_type","crop\_type","temperature","humidity"};

void display()

{

Toolkit kit=Toolkit.getDefaultToolkit();

int width=(int)kit.getScreenSize().getWidth();

int height=(int)kit.getScreenSize().getHeight();

grid=new DefaultTableModel(header,0);

tab=new JTable(grid);

pan=new JScrollPane(tab);

add(pan);

showrecs();

setSize(width,height);

setVisible(true);

}

void showrecs()

{

try

{

Connection con=Conn.getConnection();

System.out.println(con);

PreparedStatement ps=con.prepareStatement("select \* from crop\_details");

ResultSet rs=ps.executeQuery();

while(rs.next())

{

String arr[]={"","","","",""};

arr[0]=""+rs.getInt(1);

arr[1]=rs.getString(2);

arr[2]=rs.getString(3);

arr[3]=rs.getString(4);

arr[4]=rs.getString(5);

grid.addRow(arr); }}

catch(Exception ee)

{System.out.println(ee);

}

}

public static void main(String[] args)

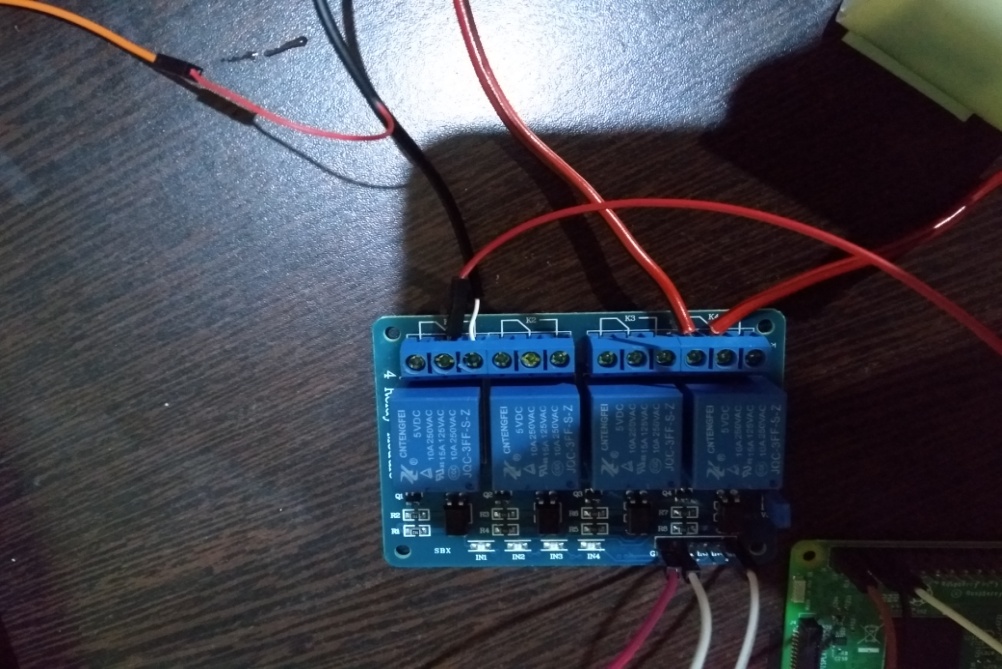
{

ShowData rod=new ShowData();

rod.display(); }

}

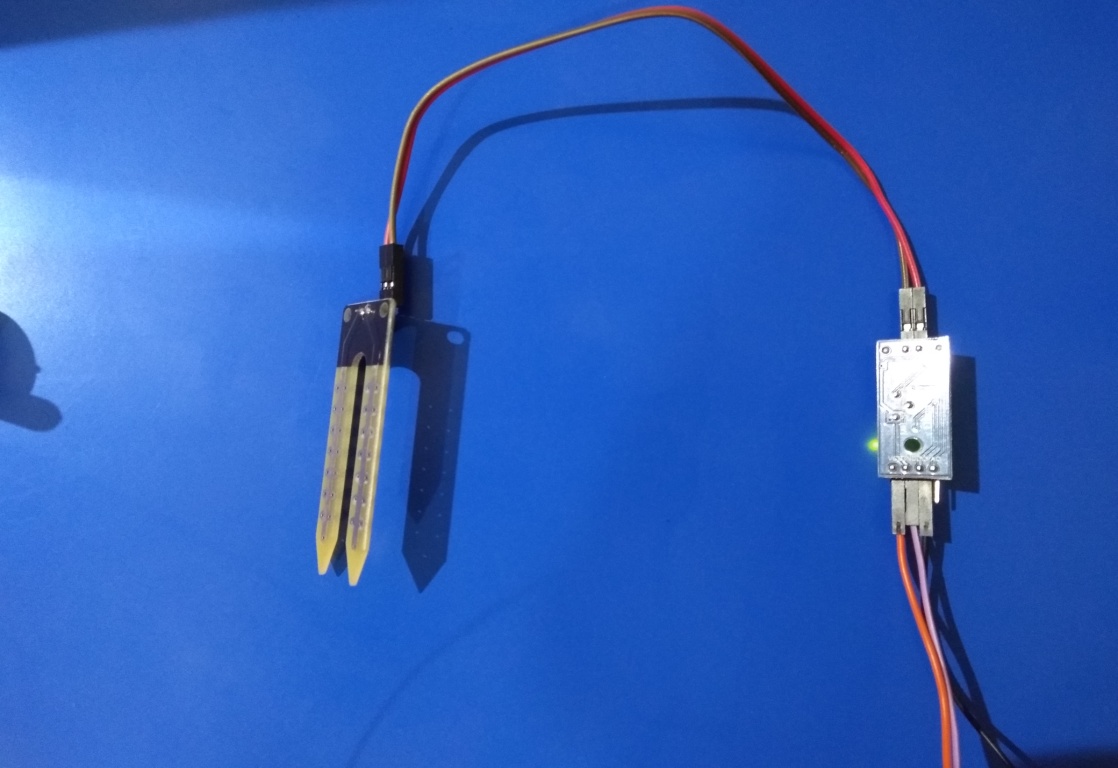
**SNAPSHOTS**



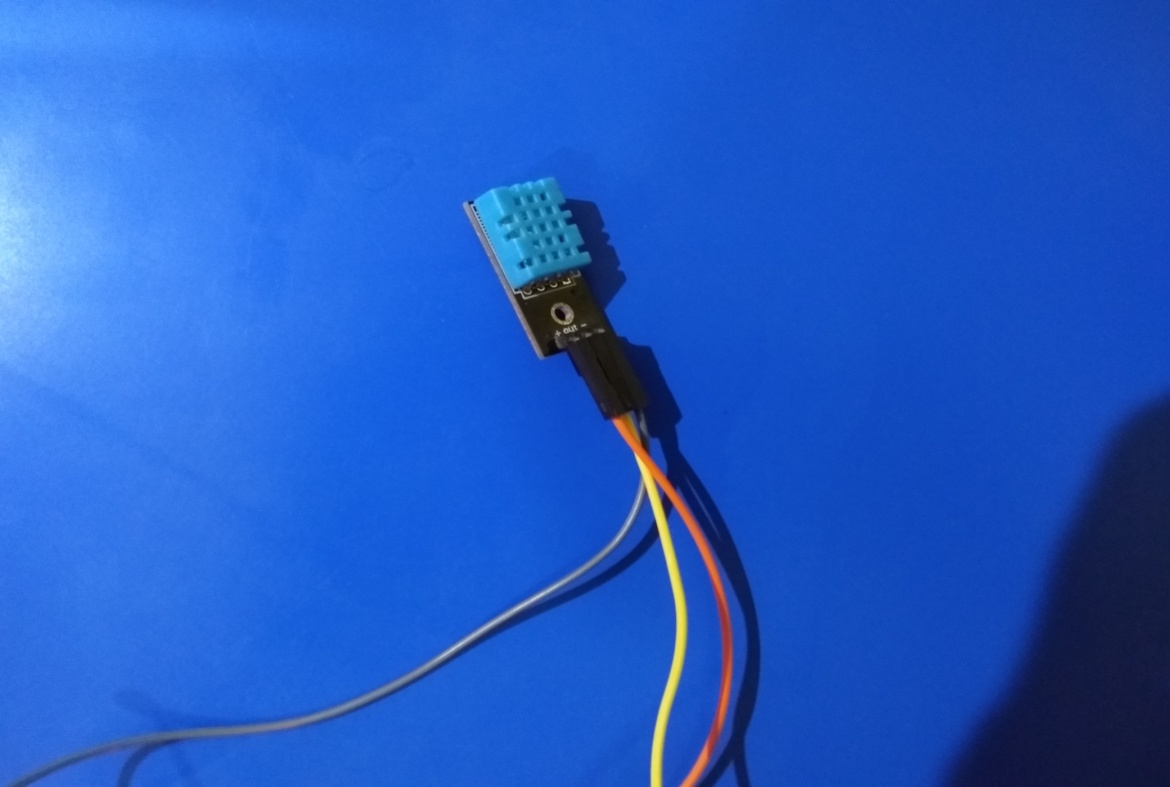
**RELAY**

****

**RASPBERRY PI**

****

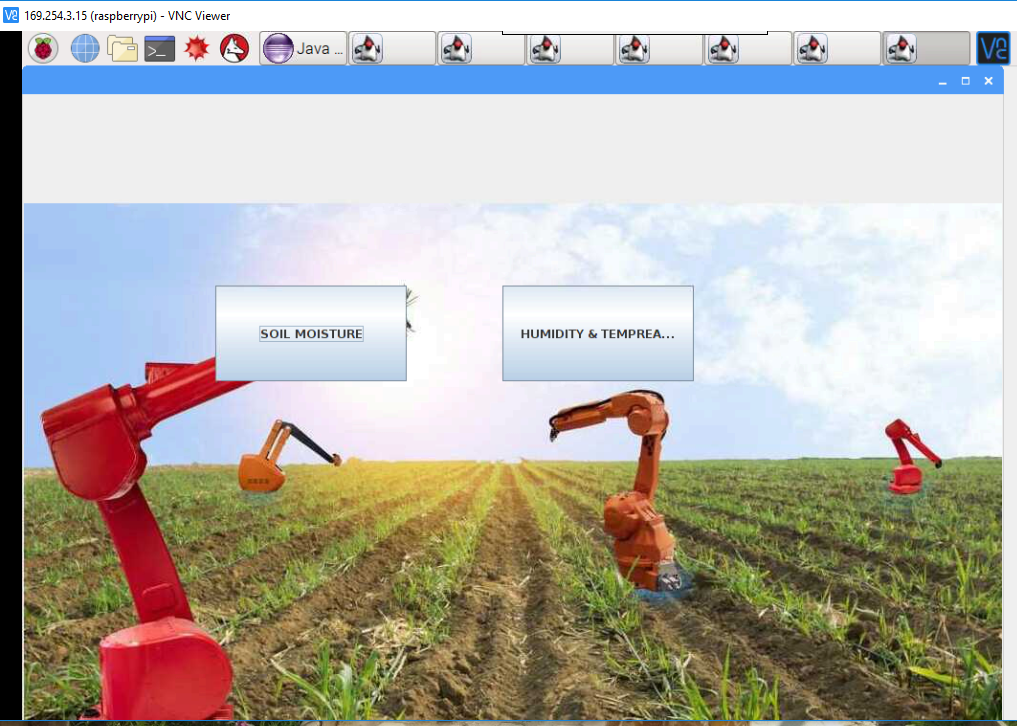
**SOIL MOISTURE SENSOR**



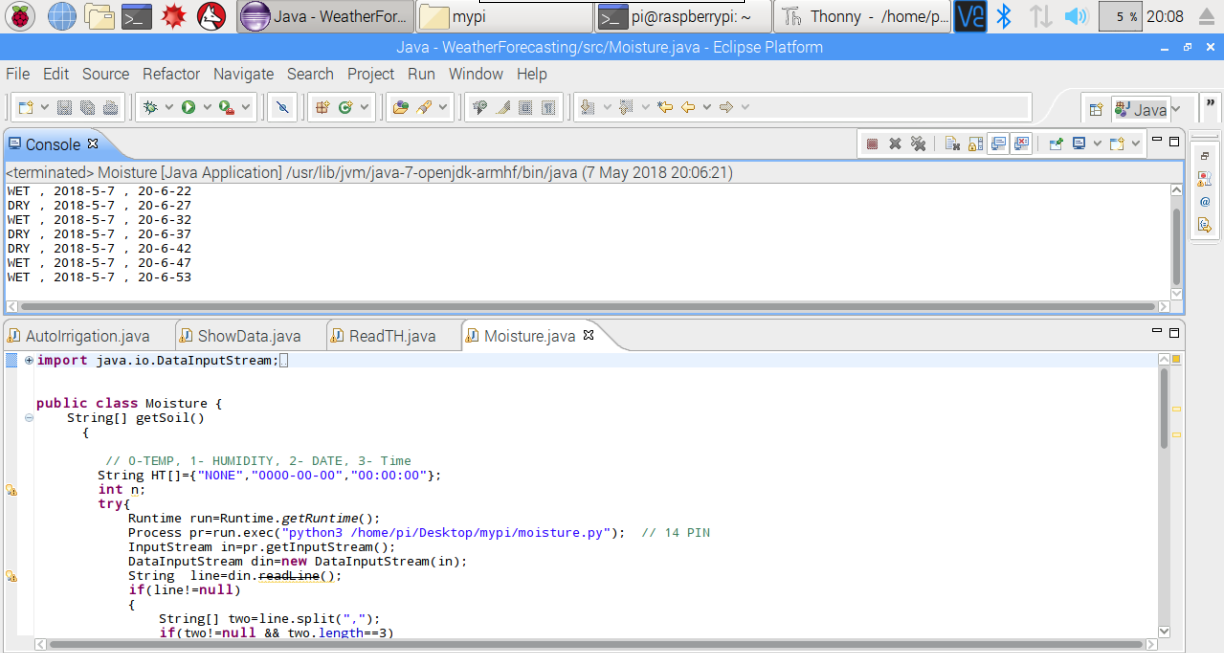
**TEMPERATURE AND HUMIDITY SENSOR**

****

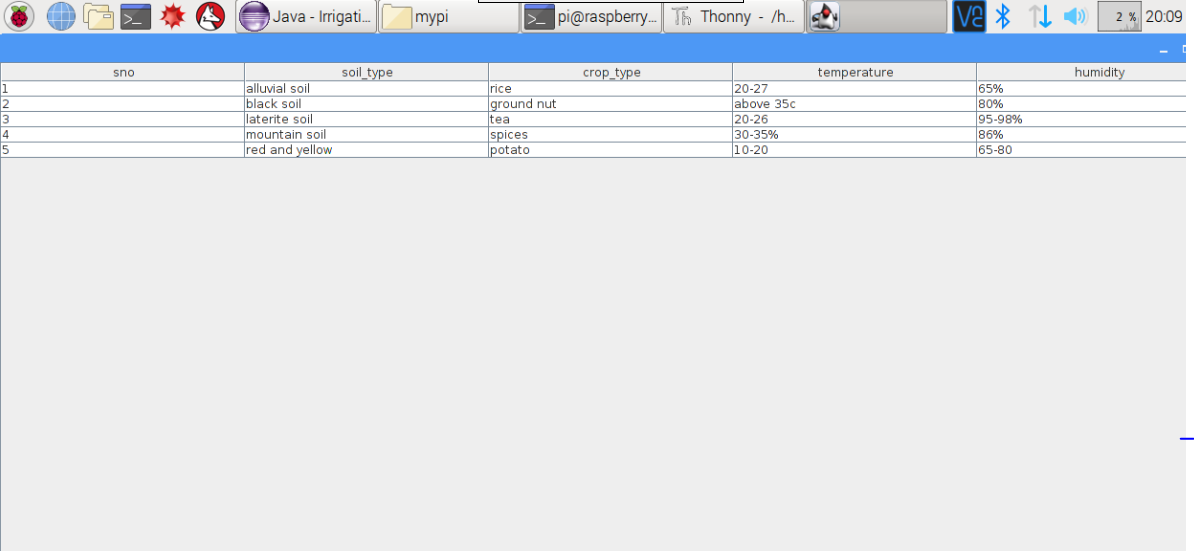
**SAMERSIBLE PUMP**

****

**USER REGISTRATION AND LOGIN**

****

**MOISTURE SENSOR READING**

****

**CROP DETAILS**

**TESTING**

INTRODUCTION TO TESTING:-

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

TESTING IN STRATEGIES

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

**Unit Testing:**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

**Each module can be tested using the following two Strategies:**

**Black Box Testing:**

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structure or external database access
* Performance errors
* Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

**White Box testing :**

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

* Guarantee that all independent paths have been Executed.
* Execute all logical decisions on their true and false Sides.
* Execute all loops at their boundaries and within their operational bounds
* Execute internal data structures to ensure their validity.
* **Integrating Testing :**

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

**System Testing :**

Involves in-house testing of the entire system before delivery to the user. It's aim is to satisfy the user the system meets all requirements of the client's specifications.

**Acceptance Testing :**

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

#### Test Approach :

**Testing can be done in two ways:**

Bottom up approach

Top down approach

**Bottom up Approach:** Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded with in the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

**Top down approach:**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

**Validation:**

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed

**Testing**

Systems are not designed as entire system are not they tested as single system. The analyst must perform both unit and system testing.

**Unit Testing**

In unit testing the analyst test the programs making up a system. For this reason, unit testing is someone called program testing. Unit testing gives stress on the modules independently of one another, to find error. This helps the tester in detecting error in coding and logic that are contained within that module alone. The error resulting from the interaction between modules is initially avoided.

**System Testing**

The important and essential part of the system development phase after designing and developing the software is system testing. We cannot say that every program or system design is perfect and because of lack of communication between the user and the designer, some error is there in the software development. The number and nature of error in a newly designed system depends on some usual factor like communication between the user and the designer; the programmer’s ability to generate a code that reflects exactly the systems specification and the time frame for the design.

Testing is an important function to the success of the system. System testing makes a logical assumption that if all the part of the system. System testing makes a logical assumption that if all the part of the system are correct, the goal will be successfully activated. Another reason for system testing is its utility as a user-oriented vehicle before implementation.

**System testing consists of the following Five steps**:

* Program Testing
* String Testing
* System Testing
* System Documentation
* User acceptance Testing

There are other Five tests which fall under special category. They are describing below:

1. **Peak Load Test** : It determines whether the system will handle the volume of activities that occur when the system is at the peak of its processing demands. For example, test the system by activating all terminals at the same time.
2. **Storage Testing** : It determines the capacity of the system to store transaction data on a disk or in other file. For example, verify documentation statement that the system will store 10,000 record of 400 bytes length on a single flexible disk.
3. **Performance Time Testing** : It determines the length of time system used by the system to process truncation data. This test is conducted prior to implementation to determine how long it takes to get a response to an inquiry, makes a backup copy of a file, or send a transmission and get a response.
4. **Recovery Testing** : This testing determines the ability of user to recover data or restart system after failure. For example, load backup copy of data and resume processing without or integrity loss.
5. **Procedure Testing** : It determines the clarity of documentation on operation and use of system by having users do exactly what manuals request. For example powering down system at the end of week or responding to paper-out light or printer.

**Software Engineering Paradigm applied:-**

We know that there are various approaches of Software engineering to analyze and design a system. There is growing recognition the software like all complex system evolves over a period of time. Business and product requirements often change as development proceeds making a straight path to an end product unrealistic. As broadly classified there are two Software Engineering paradigms.

1. Classical Software engineering paradigm
2. Evolutionary Software process model

The example of classical software engineering paradigm is

* 1. Waterfall Model / Linear sequential model / Classical life cycle model
  2. Prototyping Model
  3. Rapid Action Development (RAD) Model

As stated earlier, evolutionary models are interactive. They are characterized in a manner that enables software engineers to develop increasingly more complete version of Software. Thus the development of my project is an approach to design and develop it as an evolutionary Software process models.

**CONCLUSION**

CONCLUSION :

**The proposed irrigation system for agricultural purpose can measure the Soil moisture, temperature of the field and transmits the real time data to the user through the Wi-Fi and IoT server, if there is any deviation from the span of reference value, then the user can send the command through the IoT server to maintain the set point value of field parameter for a proper irrigation and proposed IoT based irrigation system is better than the recently proposed other irrigation systems.**

**ADVANTAGES**

**AND DISADVANTAGES**

**Advantages:**

1. It is a valuable tool for preserving water.

2. It is fully automated system for irrigation system.

3. There is less usage of water.

4. We can also control the amount of water we want deliver to the plants when it is needed based on types of plants by keeping an eye on soil moisture and temperature.

5. It also requires smaller water sources.

6. Reduce soil erosion and nutrient leaching.

7. No man power required.

8. Increase in productivity.

**Disadvantages:**

1. Lack of accuracy in sandy soil due to their large particles.

2. Need for each soil type to be calibrated.

**FUTURE SCOPE**

**FUTURE SCOPE**

IoT based smart farming system can prove to be very helpful for farmers since over as well as less irrigation is not good for farming. Threshold values for climatic conditions like humidity, temperature, moisture can be fixed based on the environmental conditions of that particular region. This system generates irrigation schedule based on the sensed real time data from field and data from the weather repository. This system can recommend farmer whether or not, is there a need for irrigation.

One of the limitations of this system is that continuous internet connectivity is required at user end which might prove to be costly for farmer. This can be overcome by extending the system to send suggestion via SMS to the farmer directly on his mobile using GSM module instead of mobile app. Weather data from the meteorological department can be used along with the sensed data to predict more information about the future which can help farmer plan accordingly and improve his livelihood

**BIBLIOGRAPHY**

**Bibliography:**

1. Programming in java

By: E Balagurusamy

2. The Complete Reference

By: Herbert Schildt

3. Java 7 Programming Black Book

By: Kongent Learning

4. en.m.wikipedia.org

1. [↑](#footnote-ref-1)
2. An unmanned aerial vehicle (UAV), commonly known as a drone [3], is an aircraft without a human pilot aboard. UAVs are a component of an unmanned aircraft system (UAS); which include a UAV, a ground-based controller, and a system of communications between the two. The flight of UAVs may operate with various degrees of autonomy: either under remote control by a human operator or autonomously by onboard computers. [↑](#footnote-ref-2)