Project Report of Major Project

on

IOT in Crop Production

Submitted to

I.K. GUJRAL PUNJAB TECHNICAL UNIVERSITY, JALANDHAR

In partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in

Computer Science Engineering

Submitted By

Gaurav Gunjan

PTU Roll No- 1326462

November 2016

Under the guidance of

Academic Tutor

Er. Harkomal Preet Kaur

Assistant Professor

CSE Department

To

Department of Computer Science and Engineering

Swami Vivekanand Institute of Engineering & Technology, Banur, Punjab

Declaration

I hereby declare that this project report entitled

IOT in Crop Production

is written by me and is my own effort and that no part has been plagiarized without citations.

STUDENT:____ DATE:__________

(GAURAV GUNJAN)

SUPERVISOR:____________________________________ DATE:__________

(Er. HARKOMAL PREET KAUR)

Acknowledgment

This project is a hard piece of work. This project is been developed as a partial vision and another way towards the Make in India initiative by Govt. of India. This project involved a number of people sharing their views and ideas and mentoring my way through. I would like to acknowledge and thank them all for their precious time and ideas. First of all, Honourable Prime Minister of India, Narendra Modi for giving us opportunities and platform under the Make in India initiative and promoting Internet of Things. I would like to thank Mr. Rakesh Ranjan(Program Director, IBM Research Lab, California, USA), Ms. Kawaljeet Kaur(HOD, CSE Department, SVIET, Banur, Punjab), Er. Vikas Zandu(Asst. Professor, SVIET, Banur, Punjab), Er. Harkomal (Asst. Professor, SVIET, Banur, Punjab), Ms. Geetika Luthra(B.Tech Student, SPMC, Delhi University, New Delhi), Mr. Anurag Kumar(Student, B.Tech CSE, SVIET, Banur, Punjab), Mr. Pankaj(Student, B.Tech CSE, SVIET, Banur, Punjab). I would also like to acknowledge my SVIET ACM Student Chapter under which I partially contributed in creating my first IOT Project. Last but not the least, My Parents for their extraordinary support and love.

Abstract

1. Objectives

The Internet of Things (IoT) is transforming the agriculture industry and enabling farmers to contend with the enormous challenges and predicament situations they face. Internet of Things in Farming can really boost farmers status and our country's economy. We will try to solve the basic problem of utilizing complete area of the farming land taking consideration the weather, resources and the land to cultivate using hardware inputs and some manual inputs through application on Mixed Integer Linear Programming (MILP) algorithm.

2. Beneficiaries (For whom)

Our vision is to provide maximum output a farmer can get from his field so as to live a good life.

3. Value of results (Use)

1. Project pushes forward the possibilities in agriculture. Maximum production means maximum profit to farmers and so as to the government.
2. It will provide moral boost to the farmers. Countries agriculture sector can once again somehow hold its back. Possibility of 5% - 15% rise in production.

4. Background

India is a developing nation and an agriculture major country. Our honorable PM Mr. Narendra Modi talks about Smart Cities. We can put smartness in every object possible. We have heard a number of incidents where farmers are not able to maximize their crop production. In this project we are trying to take into consideration wide number of factors which involves in crop production. Internet of Things in Farming can really boost farmers status and our country's economy. We will try to solve the basic problem of utilizing complete area of the farming land taking consideration the weather, resources and the land to cultivate by Mixed Integer Linear Programming (MILP).

Table of Contents

Chapter 1: Introduction

India is a developing nation and an agriculture major country. Our honorable PM Mr. Narendra Modi talks about Smart Cities. We can put smartness in every object possible. We have heard a number of incidents where farmers are not able to maximize their crop production. In this project we are trying to take into consideration wide number of factors which involves in crop production. Internet of Things in Farming can really boost farmers status and our countries economy. We will try to solve the basic problem of utilizing complete area of the farming land taking consideration the weather, resources and the land to cultivate by Mixed Integer Linear Programming (MILP).

Internet of Things has become an important area of consideration. A very interesting concept of connecting everything around each one of us and access from round the globe. Our vision is to provide maximum output a farmer can get from his field so as to live a good life. I don't think that our current approach is friendly enough to be used by a farmer but surely will be one day. Maximizing the output from a field is not just affected by one or two possible factors. It depends on a number of factors which we will be treating as variables in our program. This project can be subdivided into 3 phases i.e., Algorithm to solve the problem, hardware to gather the physical information of the field and the surroundings and third is integrating the hardware with a user accessible interface on the cloud. Arduino will be the board we are focusing on to be used for the project with integration of a number of sensors physically present like humidity, temperature, soil moisture, pH sensor etc and some of them would be virtual sensors to be used. IBM Bluemix will be used for hosting the application on the cloud and node-RED for prototyping and visualizing the working of complete application. Mixed Integer Linear Programming or MILP is the algorithm which is to be used to generate the output. Inputs in the algorithm will be variables like Soil Moisture, Soil Acidity, Atmospheric Temperature, Atmospheric Moisture, Field Size, Number of Crops, Type of Soil, pH of Soil, Rainfall pattern in the Area and many more. We would be considering all the factors in the algorithm. The program will give the outputs like division of field for the crops and which plants to be planted. Output's provided will really help the farmers optimize their productivity. As we have become the second largest country operating Smartphone's, the outputs will be provided in a very interactive Android application with a beautiful user interface. This will really bring a new dimension to the agriculture sector of India.

Project pushes forward the possibilities in agriculture. Moral boost to the farmers. Countries agriculture sector can once again somehow hold its back. Maximum production means maximum profit to farmers and so as to the government. Possibility of 5% - 15% rise in production. Updates to the application will be provided on a regular basis consisting of bug fixes and more. As an addition to this project we will be creating an extension to it or a sub-project which will have an API for unifying the selling price of the seeds and the cost price of the crops for the farmers.

Field of project will be Internet of Things, Cloud Computing, Mobile Computing, Arduino Programming, Linear Programming, OOPs and RDBMS. Keywords—*Make in India*; *Arduino*;

Cloud Computing; *Mixed Integer Linear Programming; Android, IBM Bluemix, Internet of Things, dashDB;*

1.1 Statement of Problem

The Internet of Things (IoT) is transforming the agriculture industry and enabling farmers to contend with the enormous challenges they face. The industry must overcome increasing water shortages, limited availability of lands, difficult to manage costs, while meeting the increasing consumption needs of a global population that is expected to grow by 70% by 2050. (Reference: Food and Agriculture Organization of the United Nations)

Our scope of problem is how to best utilize complete area of the farming land to increase productivity. I don't think that our current approach is friendly enough to be used by a farmer but surely will be one day. Maximizing the output from a field is not just affected by one or two possible factors.

Chapter 2: Research

2.1. Present methods of tackling the problem

There are number of companies working towards providing best possible ways to ameliorate agricultural problems like farmx which provides information about water stress in crops and many others like farm mobile and cropx.

2.2. Proposed Solution

Farmers will get suggested crops for their owned agriculture land, crops to plant for best profit to the farmers and way to plant the crops in their shape of land. This all will be provided in an easy to use android application. In our solution proposed we will be crowd sourcing a number of components for calculation like selling price of seeds and cost price of crops.

2.3. Novelty of Approach

Our solution focuses on providing solution to the farmers not just giving them numbers and analytics. Our prototype promises answers to a number of basic questions in farmers mind.

Chapter 3: Tools and Technologies Used

Being an Internet of Things project, a wide number of tools and technologies come into the scenario. There is both the hardware part of the project and the software part of the project. This project has interfacing on the Cloud, so concepts and application of Cloud Computing also comes into role. I will be discussing about all the tools and technologies in the following sections of this chapter.

3.1 Hardware Requirements

Sr. No.	Tool/ Technology	Use	Cost
1.	Laptop equiped with minimum Intel i5 processor, 8GB's of RAM, USB port and Internet Access.	Needed to write and transmit code into the Arduino D1 ESP8266 microprocessor, create the Android applicaion and for creating the Cloud application.	Pre-available.
2.	Arduino WEMOS D1	An Arduino UNO Compatible wifi board based on ESP8266EX.	~Rs. 800
3.	Soil Moisture Sensor	Arduino sensor used to detect moisture content of the soil.	~Rs. 150
4.	DHT-11 Sensor	Digital Humidity and Temperature Sensor for Arduino.	~Rs. 150
5.	Methane/ Butane Sensor	Methane or Butane detection sensor for Arduino.	~Rs. 200
6.	Bread Board	Board used to create connections for prototyping.	~Rs. 250
7.	Jump Wires	Wires used to create connections.	~Rs. 150
8.	Adapter	5V Power Adapter for WEMOS D1 Board.	~Rs. 350
9.	Android Device	Used to Run and Test Application	Pre-Available

3.2 Software Requirements

Sr. No.	Tool/ Technology	Use	Cost
1.	Arduino IDE	Used to program, code and debug Arduino Board.	Open-Source
2.	IBM Bluemix PaaS	Cloud Service from IBM which will work as an interface to Arduino and Android application. It will house the node-RED application.	Free
3.	node-RED service	This service enables to connect the hardware and applications and all databases. Coding can be done using JavaScript.	Free

4.	Virtual Sensor Application on Bluemix	This application enables a user to create virtual sensors. Sensors which are not phisically available but are virtually generated and could be used for experimental or prototyping uses.	Free
5.	dashDB DBMS Service	Database service offered by IBM.	Free
6.	Android Boilerplate on Bluemix	Prototype android application containing connectivity's with the Bluemix Cloud.	Free

The above mentioned technologies and tools are the ones used in creation and development of this project. Further in this report, detailed information about the technologies and tools are illustrated. Also the system how these technologies have been used are described in further chapters. Next chapter i.e., Chapter 4 contains the detailed Feasibility Study.

Chapter 4: Feasibility Study

An Internet of Things project contains a wide components in the scenario which can lead to successful completion or can create a hitch in the project. When a project contains both the hardware and the software part then feasibility study is a must. This project has interfacing on the Cloud, so concepts and application of Cloud Computing also comes into role. I will be discussing about the various feasibility studies in detail in this chapter.

The below feasibility study states and justifies if the project is a go/no-go. Various feasibility studies have been performed for the same.

Technical feasibility Study: Arduino Uno Prototyping Board, Soil Moisture Sensor, DHT-11 Humidity & Temperature Sensor are easily available on online stores. pH Sensor Kit is too much costly for the prototyping so we will be using Virtual Sensor. Development of the application is doable on Android and the prototyping on node-RED.

Economic feasibility Study: This following study evaluates the money aspects of the project by performing price profit analysis:

- Cost of doing full system study = Estimated Price of Hardware + Purchasing Space on Google Play App Store.
- Estimated price of hardware = Rs. 3000 (Max).
- Estimated price of purchase of Account on Google Play App Store = $25 ≈ Rs. 1700.
- Total Estimated cost of the project = Rs. 5000 (Min).

Operational feasibility Study: It appears that the system may get an integration issue of a wide number of variables into the algorithm. Prototyping on node-RED would be performed for in-depth check of the operational possibilities for long run and product deployment.

Schedule feasibility Study: This project can take a minimum of 4 - 5 months for completion and release. This time includes gathering of the resources, performing electrical assembly and deployment of application on cloud.

This was all about the feasibility study of the project. Moving ahead we will talk about the methodology of the project in Chapter 5.

Chapter 5: Methodology

A well laid out and logical methodology provides a great backbone for the entire project, and allows us to build an extremely strong results section. The writing for the method has to be clear and direct, concise and straight to the point. The major point is not to stray off into irrelevance, and this process is helped by making a few basic assumptions. For example, in a psychology paper, there is no need to describe a Skinner box, as that is well known to psychologists. However, you would need to explain exactly how the box was used, to allow exact replication.

The above explains a lot about how methodology must be written and with that approach I present the following data illustrations. Following are the phases followed for the completion of the project with the approximated time period required for completion of each step.

S. No.	Project Steps	Time Period (in days)
Phase I		
1.	Feasibility Study	1
2.	Gathering Resources	5-7
Phase II		
3.	Circuit Diagram	1-2
4.	Assembly of Electrical Components	1-2
5.	Coding of Arduino	2
Phase III		
6.	Node-RED Prototyping	5-7
7.	Integrating Arduino Setup	1-2
8.	Integration of Weather API	1-2
Phase IV		
9.	Algorithm generation	2-3
10.	User Interface	4-5
11.	Coding	Upto 10
Phase V		
12.	Deployment & Debugging	2-5
13.	Testing	2-4
14.	Finalizing	5

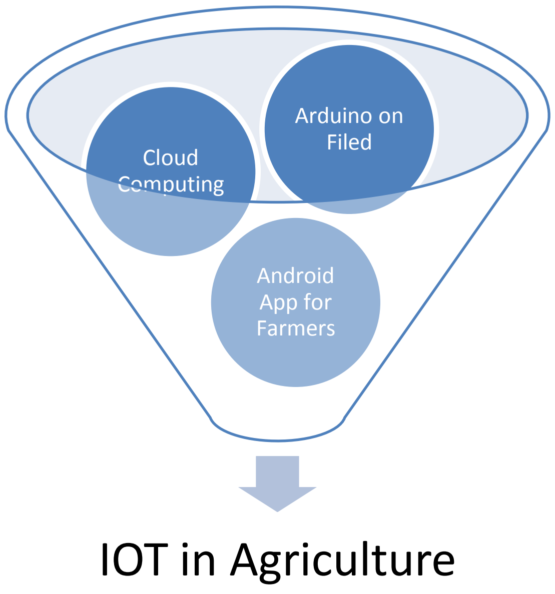
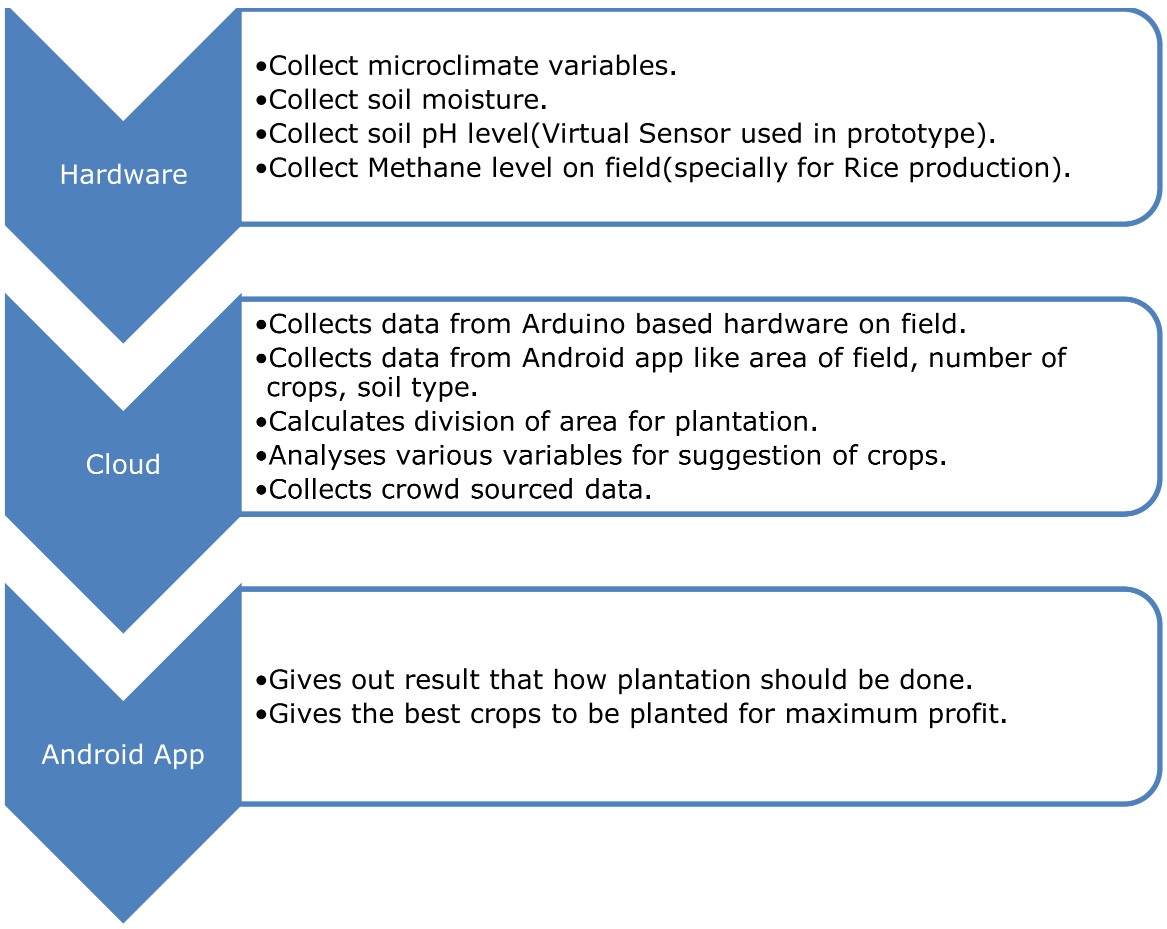


Fig. 1: Visual representation of the inside of the system of this Internet of Things Project.



Hardware

- Collect microclimate variables.
- Collect soil moisture.
- Collect soil pH level(Virtual Sensor used in prototype).
- Collect Methane level on field(specially for Rice production).

Cloud

- Collects data from Arduino based hardware on field.
- Collects data from Android app like area of field, number of crops, soil type.
- Calculates division of area for plantation.
- Analyses various variables for suggestion of crops.
- Collects crowd sourced data.

Android App

- Gives out result that how plantation should be done.
- Gives the best crops to be planted for maximum profit.

Chapter 6: Languages, Platforms & Hardware

We have discussed about the various tools, technologies be it software or hardware of the project. Also we have discussed the methodology and feasibility of this project. Now in this chapter we will be getting a insight of what all these technologies are and how they work. These information and knowledge has been scrapped from various sources from the internet and paperback books. These all have been acknowledged in the bibliography. Below are the various languages, platforms and hardware's in detail.

6.1 Arduino WEMOS D1

Arduino is an open-source electronic prototyping platform enabling users to create interactive electronic objects. WEMOS D1 is an Arduino UNO Compatible wifi board based on ESP8266EX.

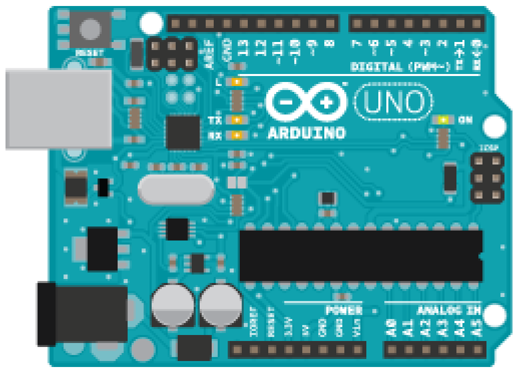


Fig.2: Image of Arduino Uno Board

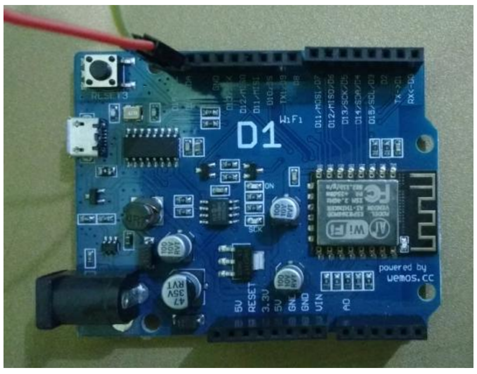


Fig.3: Image of WEMOS D1 used in the project.

Features:

- 11 digital input/output pins, all pins have interrupt/pwm/I2C/one-wire supported(except for **D0**)
- 1 analog input(3.2V max input)
- Micro USB connection
- Power jack, 9-24V power input.
- Compatible with Arduino.

Technical specs:

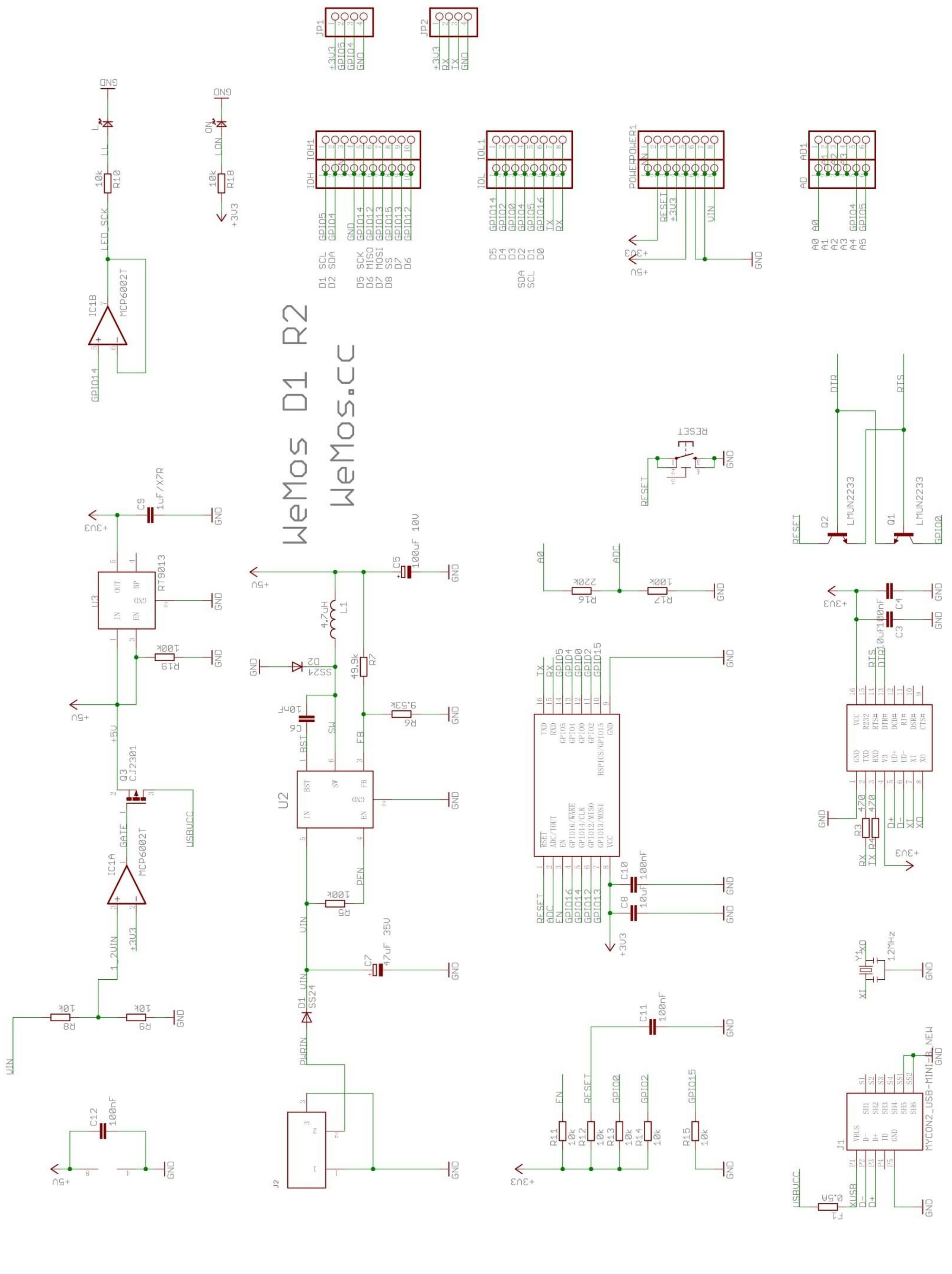
Microcontroller	**ESP-8266EX**
Operating Voltage	3.3V
Digital I/O Pins	11
Analog Input Pins	1(Max input: 3.2V)
Clock Speed	80MHz/160MHz

Flash	4M bytes
Length	68.6mm
Width	53.4mm
Weight	25g

Pin Description:

Pin	Function	ESP-8266 Pin
TX	TXD	TXD
RX	RXD	RXD
A0	Analog input, max 3.3V input	A0
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
5V	5V	-
3V3	3.3V	3.3V
RST	Reset	RST

Image in the next page provides the detailed circuit diagram of the components of the Arduino WEMOS D1 Microcontroller Board.


WeMos D1 R2
WeMos.CC
U2
U3 RT9013
IC1A MCP6002T
IC1B MCP6002T
Q3 CJ2301
Q1 LMUN2233
Q2 LMUN2233
L1 4.7uH
D1 SS24
D2 SS24
C5 100uF 10V
C7 47uF 35V
C9 1uF/X7R
C12 100nF
C11 100nF
C10 100nF
C8 10uF
C3 10uF
C4 100nF
C6 10nF
R5 100K
R6 9.53K
R7 49.9K
R8 10K
R9 10K
R10 10k
R16 220K
R17 100K
R18 10k
R19 100K
R11 10k
R12 10k
R13 10k
R14 10k
R15 10k
R3 470
R4 470
F1 0.5A
Y1 12MHz
J1 MYCON2_USB-MINI_B_NEW
J2
JP1
JP2
IOH
IOH1
IOL
IOL1
POWER
POWER1
AD
AD1
RESET
VIN
+5V
+3V3
GND
USBVCC
PWRIN
GPIO14
LED_SCK
DTR
RTS
A0
ADC
TX
RX

6.2 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 weighting 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.

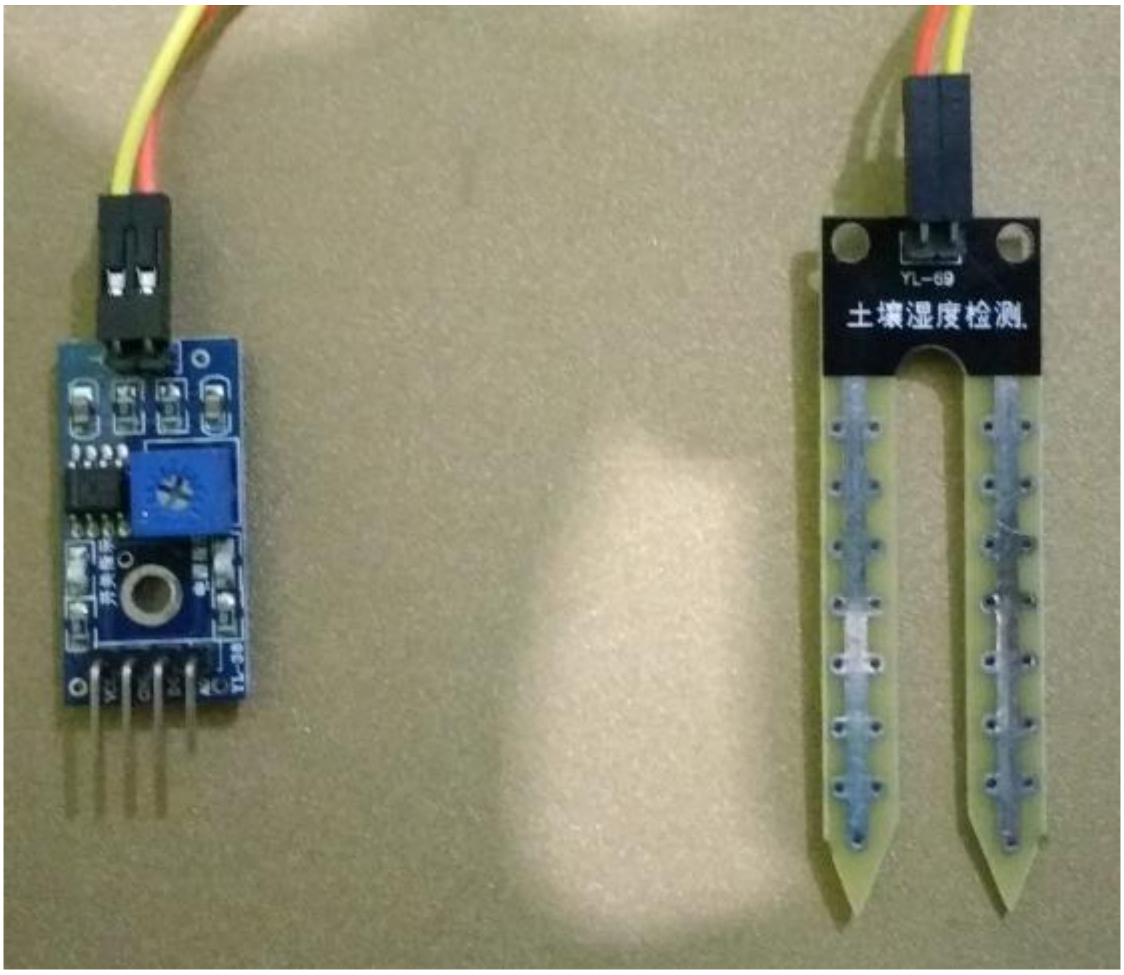


Fig. 4: Above image is of the Soil Moisture Sensor used in this project.

6.3 DHT-11 Sensor

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output.
By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a

resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high-performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness.

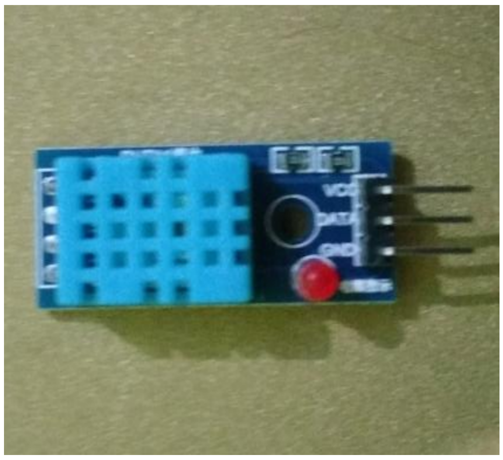


Fig.5: Image of the DHT-11 Sensor

Technical Specifications

Item	**Measurement Range**	**Humidity Accuracy**	**Temperature Accuracy**	**Resolution**	**Package**
DHT-11	20-90%RH 0-50^{o} C	±5%RH	±2^{o} C	1	4 Pin Single Row

6.4 Methane/Butane Sensor

MQ-2 gas sensor has high sensitivity to LPG, Propane and Hydrogen, also could be used to Methane and other combustible steam, it is low cost and suitable for various applications.

Using high-quality dual-panel design, with a power indicator and TTL signal output instructions; With a the DO switch signal (TTL) output, and AO analog signal output; TTL output valid signal is low level; (When the low level output signal lights, it can be connected directly to the microcontroller or relay module). TTL output signal can be connected directly to a microcontroller IO port or connect to the relay module, potentiometer is used to adjust the output level transition threshold.

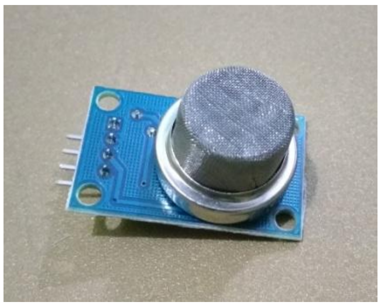


Fig.6: Image of the Methane/Butane Sensor

Features

- Wide detecting scope
- Stable and long life
- Fast response and High sensitivity
- Analog and Digital Outputs
- Trigger Level configuration Potentiometer

Specifications

- Model: FC-22-A
- Operating voltage: DC 5V
- Analog Output (AO): 0~5V analog output voltage
- Digital Outout (DO): 0V or 5V output
- Configuration: Through Potentiometer (adjusts the output level transition)
- Preheat Duration: 20s

Pinout

1. VCC (+5V)
2. GND (0V)
3. DO (Digital Out)
4. AO (Analog Out)

6.5 Arduino IDE and Language

Fig.7: Official Arduino Logo

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages *Processing* and *Wiring*. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and provides simple one-click mechanism to compile and load programs to an Arduino board. A program written with the IDE for Arduino is called a "sketch".

The Arduino IDE supports the languages C and C++ using special rules to organize code. The Arduino IDE supplies a software library called Wiring from the Wiring project, which provides many common input and output procedures. A typical Arduino C/C++ sketch consist of two functions that are compiled and linked with a program stub *main()* into an executable cyclic executive program:

- *setup()*: a function that runs once at the start of a program and that can initialize settings.
- *loop()*: a function called repeatedly until the board powers off.

After compiling and linking with the GNU toolchain, also included with the IDE distribution, the Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal coding that is loaded into the Arduino board by a loader program in the board's firmware.

Fig.8: Arduino Open-Source Community Logo

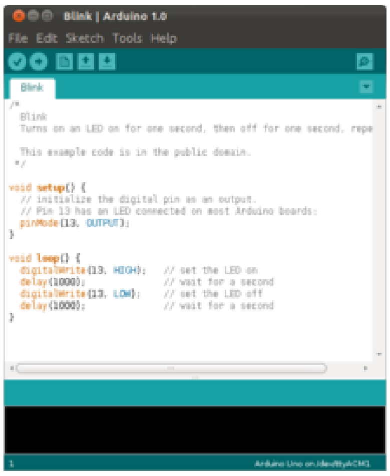
Arduino Board Development and the Arduino Integrated Development Environment are Open-Source Projects developed and maintained by Arduino Open-Source Community on Github.

Sample program

The bare minimum code to start a sketch program consists of two functions setup() and loop().

```
void setup() {
  // put your setup code here, to run once at startup
}

void loop() {
  // put your main code here, to run repeatedly
}
```



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

Fig.9: Screenshot of the Arduino IDE with a Sample Code of Blink Operation.

6.6 JavaScript

JavaScript (JS) is a programming language mostly used client-side to dynamically script webpages, but often also server-side. JavaScript is a programming language that adds interactivity to your website (for example: games, responses when buttons are pressed or data entered in forms, dynamic styling, animation). This article helps you get started with this exciting language and gives you an idea of what is possible.

Fig.10: Casual Logo of Javascript.

JavaScript ("JS" for short) is a full-fledged dynamic programming language that, when applied to an HTML document, can provide dynamic interactivity on websites. It was invented by Brendan Eich, co-founder of the Mozilla project, the Mozilla Foundation, and the Mozilla Corporation.

JavaScript is incredibly versatile. You can start small, with carousels, image galleries, fluctuating layouts, and responses to button clicks. With more experience you'll be able to create games, animated 2D and 3D graphics, comprehensive database-driven apps, and much more!

JavaScript itself is fairly compact yet very flexible. Developers have written a large variety of tools complimenting the core JavaScript language, unlocking a vast amount of extra functionality with minimum effort. These include:

- Application Programming Interfaces (APIs) built into web browsers, providing functionality like dynamically creating HTML and setting CSS styles, collecting and manipulating a video stream from the user's webcam, or generating 3D graphics and audio samples.
- Third-party APIs to allow developers to incorporate functionality in their sites from other content providers, such as Twitter or Facebook.
- Third-party frameworks and libraries you can apply to your HTML to allow you to rapidly build up sites and applications.

Sample Code

```
//Basic Code of If Else in JavaScript.

var iceCream = 'chocolate';
if (iceCream === 'chocolate') {
  alert('Yay, I love chocolate ice cream!');
} else {
  alert('Awwww, but chocolate is my favorite...');
}
```

6.7 JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

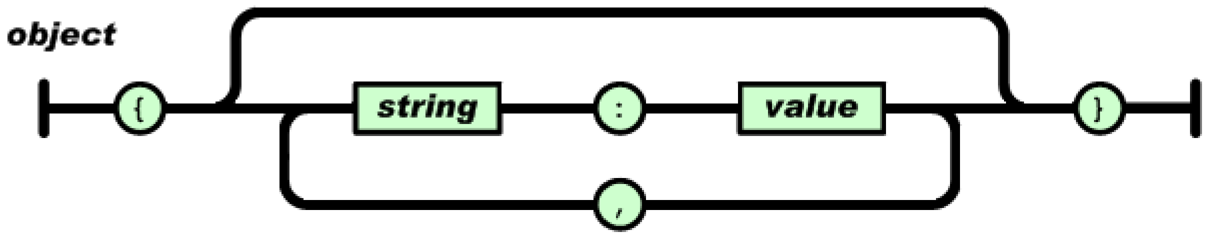
JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

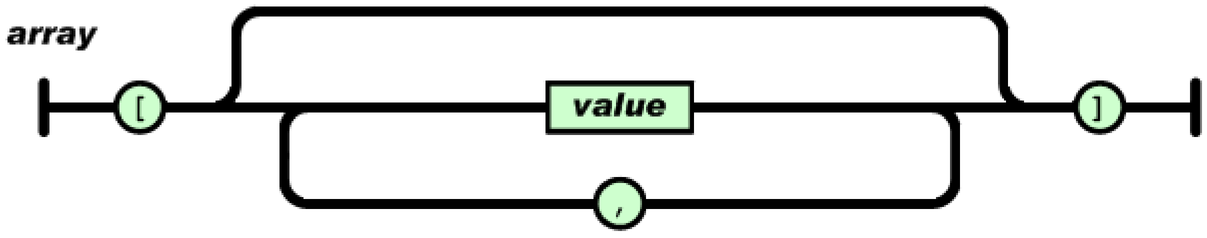
These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

In JSON, they take on these forms:

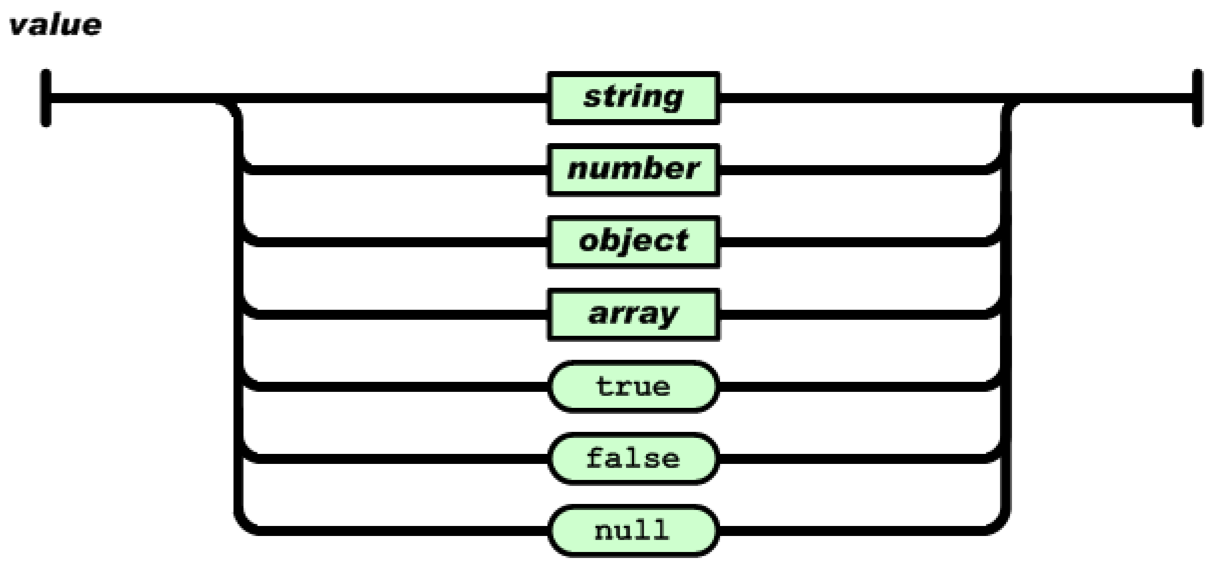
An *object* is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).



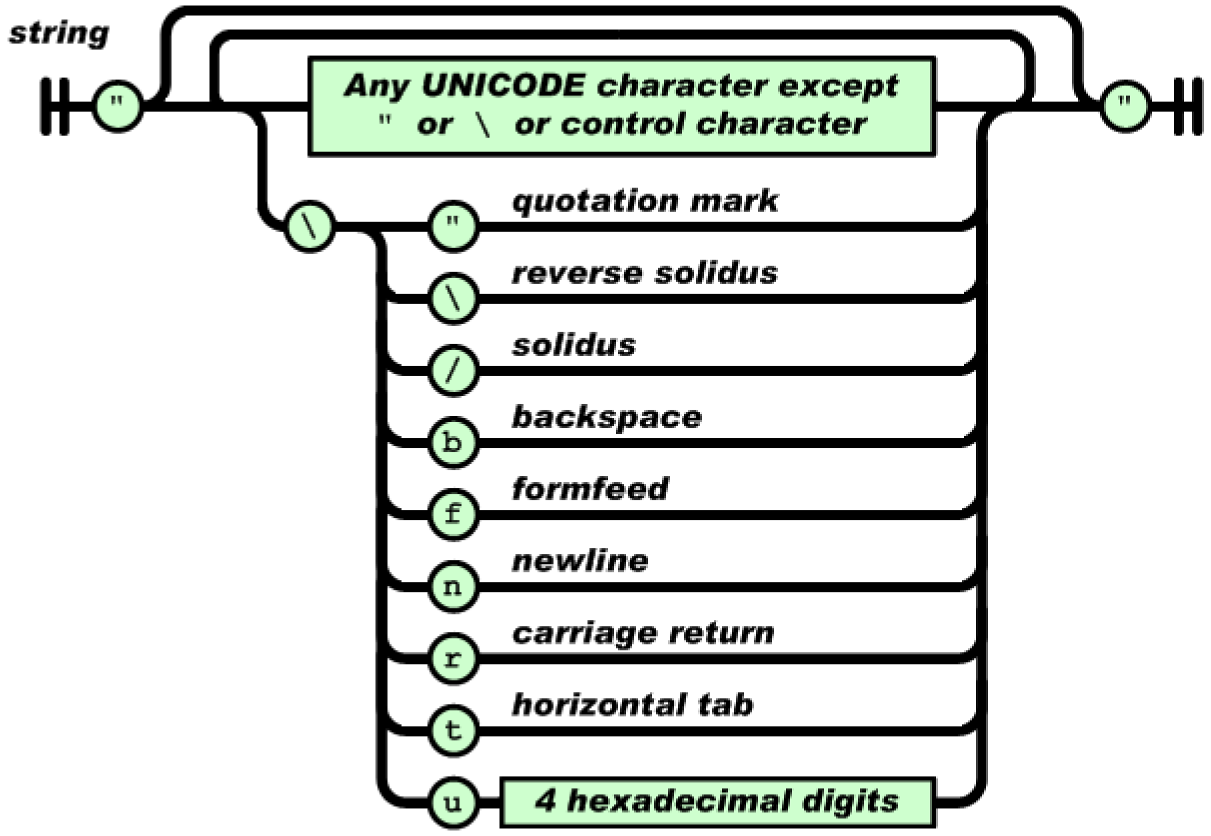
An *array* is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by , (comma).



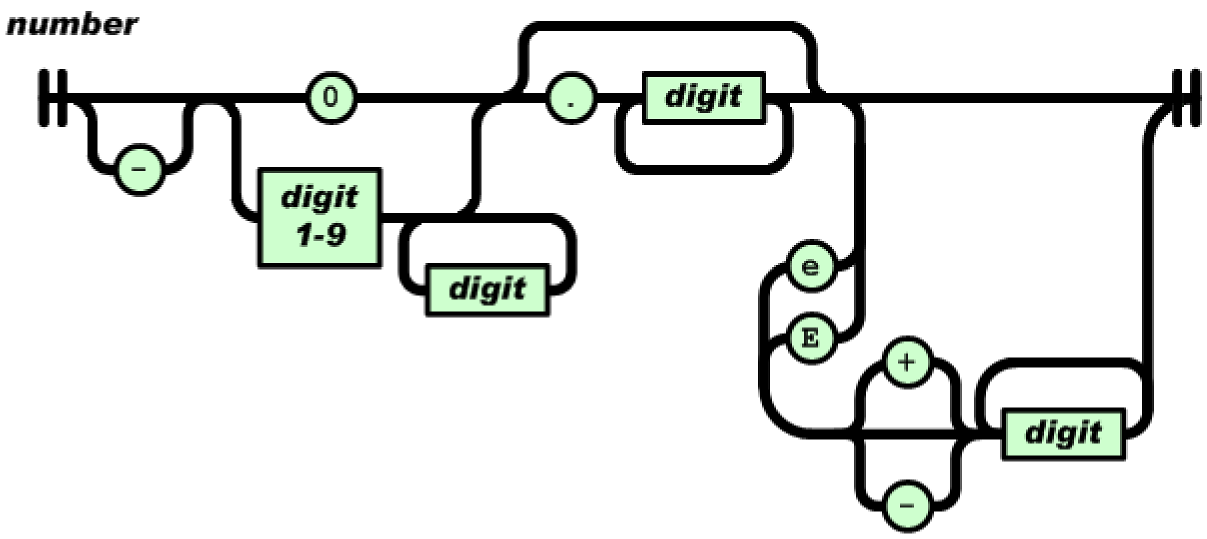
A *value* can be a *string* in double quotes, or a *number*, or true or false or null, or an *object* or an *array*. These structures can be nested.



A *string* is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. A string is very much like a C or Java string.



A *number* is very much like a C or Java number, except that the octal and hexadecimal formats are not used.



Whitespace can be inserted between any pair of tokens. Excepting a few encoding details, that completely describes the language.

JSON Example

```
{"employees":[
               {"firstName":"John",          "lastName":"Doe"},
               {"firstName":"Anna",          "lastName":"Smith"},
               {"firstName":"Peter",         "lastName":"Jones"}
]}
```

Why JSON?

For AJAX applications, JSON is faster and easier than XML:

Using XML

- Fetch an XML document
- Use the XML DOM to loop through the document
- Extract values and store in variables

Using JSON

- Fetch a JSON string
- JSON.Parse the JSON string

6.8 Node.js

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Fig.11: node.JS Official Logo

Fig.12: node.JS Official Logo Reverse Coloured

As an asynchronous event driven JavaScript runtime, Node is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection the callback is fired, but if there is no work to be done Node is sleeping.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

This is in contrast to today's more common concurrency model where OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node are free from worries of dead-locking the process, since there are no locks. Almost no function in Node directly performs I/O, so the process never blocks. Because nothing blocks, scalable systems are very reasonable to develop in Node.

6.9 IBM Bluemix Cloud PaaS

IBM Bluemix is a cloud platform as a service (PaaS) developed by IBM. It supports several programming languages and services as well as integrated DevOps to build, run, deploy and manage applications on the cloud. Bluemix is based on Cloud Foundry open technology and runs on SoftLayer infrastructure. Bluemix supports several programming languages including Java, Node.js, Go, PHP, Swift, Python, Ruby Sinatra, Ruby on Rails and can be extended to support other languages such as Scala through the use of buildpacks.

Fig.13: IBM Bluemix Logo

What is Cloud Foundry?

Cloud Foundry is an open source platform as a service (PaaS) that lets you quickly create and deploy applications on the cloud. Because of its open source roots, Cloud Foundry is not vendor specific and does not lock you into proprietary software or cloud infrastructure. Cloud Foundry abstracts the underlying infrastructure needed to run a cloud, letting you focus on the business of building cloud applications. The beauty of Cloud Foundry is that it provides choice. Developers and organizations can choose:

- **Development Frameworks**: Cloud Foundry supports Java™ code, Spring, Ruby, Node.js, and custom frameworks.
- **Application Services**: Cloud Foundry offers support for MySQL, MongoDB, PostgreSQL, Redis, RabbitMQ, and custom services.
- **Clouds**: Developers and organizations can choose to run Cloud Foundry in Public, Private, VMWare and OpenStack-based clouds.

Cloud Foundry's ability to provide choice comes through buildpacks, a convenient way to package frameworks and runtimes. Buildpacks can be community based, custom built, or built from scratch. In other words, if you cannot find a framework or service buildpack that suits your needs, you could modify an existing buildpack or create your own. By using buildpacks, companies are able to provide enterprise-level services like the Bluemix cloud offering.

What is Bluemix?

Bluemix is an implementation of IBM's Open Cloud Architecture, based on Cloud Foundry, that enables you to rapidly create, deploy, and manage your cloud applications. Because Bluemix is based on Cloud Foundry, you can tap into a growing ecosystem of runtime frameworks and services. In addition to providing additional frameworks and services, Bluemix provides a dashboard for you to create, view, and manage your applications and services as well as monitor your application's resource usage. The Bluemix dashboard also provides the ability to manage organizations, spaces, and user access.

Bluemix provides access to a wide variety of services that can be incorporated into an application. Some of these services are delivered through Cloud Foundry. Others are delivered from IBM and third party vendors. New and enhanced services are added to the catalog often. To see the current list of runtimes and services, and their status go to the Bluemix catalog.

Some of the commonly used runtimes are:

- Node.js
- PHP
- Python
- Ruby

Some of the Bluemix services available from the expanding catalog include:

Service name	**Description**
BigInsights for Hadoop	Powered by InfoSphere BigInsights, which is based on open source Hadoop, this service provides the open source capabilities of HBase, Hive, MapReduce, Pig and others, including your own open source packages.
Business Rules	Enables developers to spend less time recoding and testing when the business policy changes. This service minimizes your code changes by keeping business logic separate from application logic.
Cloudant NoSQL DB	Provides access to a fully managed NoSQL JSON data layer that's always on. This service is compatible with CouchDB, and accessible through a simple to use HTTP interface for mobile and web application models.
Data Cache	Improve the performance and user experience of web applications by retrieving information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases.
DevOps Auto-Scaling	Enables you to automatically increase or decrease the compute capacity of your application. The number of application instances are adjusted dynamically based on the Auto-Scaling policy you define.
DevOps Delivery Pipeline	Automate builds and deployments, test execution, configure build scripts, and automate execution of unit tests. Automatically build and deploy your application to IBM's cloud platform, Bluemix.
Embeddable Reporting	Use a simple cloud editor then embed reports and dashboards in your web or mobile app using a wide variety of languages such as Node.js or Java using a RESTful API.
Geospatial Analytics	Leverage real-time geospatial analytics to track when devices enter or leave defined regions.
Internet of Things	Lets your apps communicate with and consume data collected by your connected devices, sensors, and gateways.
Mobile Push Notifications	Push information to all application users or to a specific set of users and devices. You can even let users subscribe to specific tags or topics for notification.

Service name	Description
MongoDB	A popular NoSQL database
MQ Light	Develop responsive, scalable applications with a fully-managed messaging provider in the cloud. Quickly integrate with application frameworks through easy-to-use APIs.
Redis	A popular distributed dictionary server used by many distributed applications
Secure Gateway	Brings Hybrid Integration capability to your Bluemix environment. It provides secure connectivity from Bluemix to other applications and data sources running on-premise or in other clouds. A remote client is provided to enable secure connectivity.
Sendgrid	Sendgrid's cloud-based email infrastructure relieves businesses of the cost and complexity of maintaining email systems.
Session Cache	Improve application resiliency by storing session state information across many HTTP requests. Enable persistent HTTP sessions for your application and seamless session recovery in event of an application failure.
Single Sign-On	Implement user authentication for your web and mobile apps quickly, using simple policy-based configurations.
SQL Database	Add an on-demand relational database to your application. Powered by DB2, it provides a managed database service to handle web and transactional workloads.
Watson Alchemy API	Leverage natural language processing and computer vision in your apps to deeply understand the world's conversations, documents and photos.
Watson Language Translation	Converts text input in one language into a destination language for the end user. Translation is available among English, Brazilian Portuguese, Spanish, French, and Arabic.
Watson Personality Insights	Derives insights from transactional and social media data to identify psychological traits which determine purchase decisions, intent and behavioral traits; utilized to improve conversion rates.

For developers, Bluemix further optimizes the time you spend creating cloud application. You no longer have to be concerned about installing software or having to deal with virtual machine images or hardware. With a few clicks or keystrokes, you can provision instances of your applications with the necessary services to support them. This streamlining translates countless hours of setting up, configuring, and troubleshooting into time spent rapidly innovating and reacting to never-ending requirement changes.

For organizations, Bluemix provides a cloud platform that requires very little in-house technical know-how as well as cost savings. Bluemix provides the rapid development environment organizations need to react to users' demands for new features. The Bluemix platform and the cloud provide the elasticity and capacity flexibility organizations require when their applications explode in popularity. For users, Bluemix represents the key that enables organizations to quickly deliver the features they demand.

6.10 Node-RED Application

A visual tool for wiring the Internet of Things. Node-RED is a tool for wiring together hardware devices, APIs and online services in new and interesting ways.

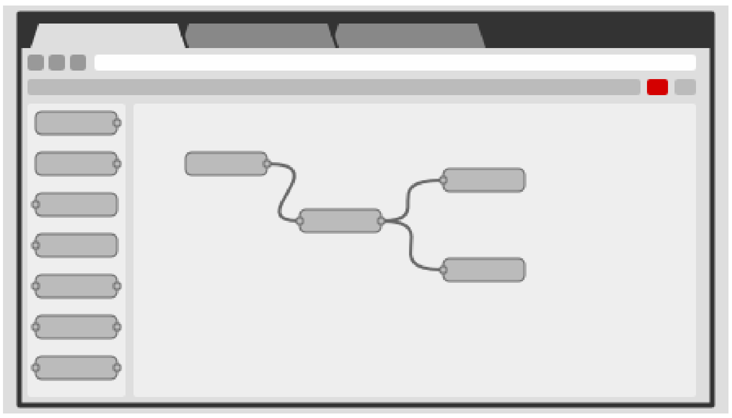


Fig.14: Browser-based flow editing

Node-RED provides a browser-based flow editor that makes it easy to wire together flows using the wide range nodes in the palette. Flows can be then deployed to the runtime in a single-click. JavaScript functions can be created within the editor using a rich text editor. A built-in library allows you to save useful functions, templates or flows for re-use.

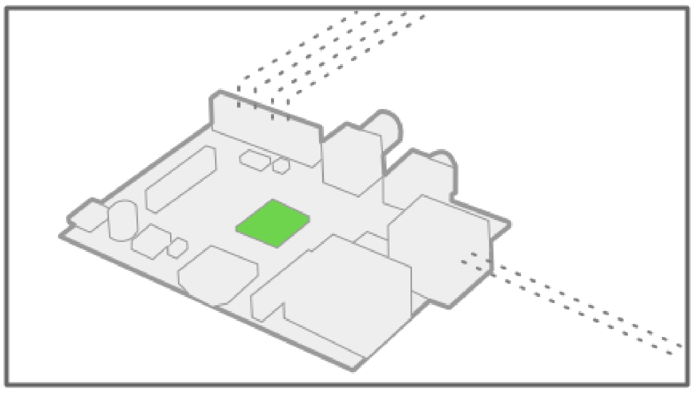


Fig.15: Built on Node.js

The light-weight runtime is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.

With over 225,000 modules in Node's package repository, it is easy to extend the range of palette nodes to add new capabilities.

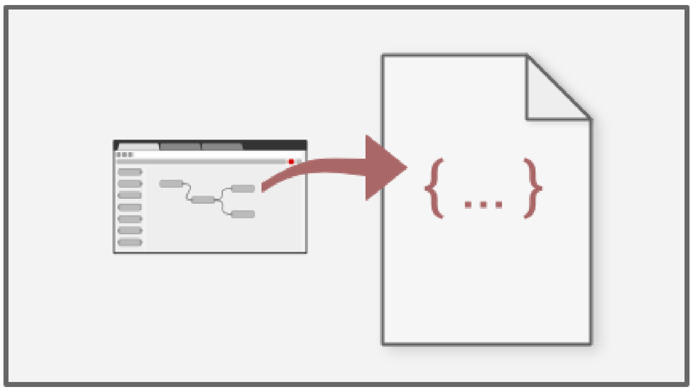


Fig.16: Social Development

The flows created in Node-RED are stored using JSON which can be easily imported and exported for sharing with others. An online flow library allows you to share your best flows with the world.

6.11 dashDB Database

dashDB gives you on demand data warehousing at the speed, scale and agility you need so you can get straight to building solutions. How? dashDB integrates key data warehousing and analytics technologies on a robust, secure cloud infrastructure that includes:

- In-database Netezza analytics, providing the immediate answers you need from your data.
- Massive scale and performance achieved through the MPP architecture, bringing greater storage and speed with each server instance you add to your network cluster.
- In-memory BLU processing technology that delivers an extreme performance boost so answers are available in seconds —without the “waiting-for-results” coffee break.
- Integration with the IBM Bluemix open cloud platform, with access to many and diverse cloud services to help you build your new solution, application or architecture.
- Direct integration with IBM Cloudant so you can easily bring JSON and NoSQL data into your data warehousing and analytics strategy.
- SoftLayer bare metal cloud infrastructure, a high performance enterprise cloud that addresses essential security needs at the infrastructure, database and operational levels—all fully managed by IBM.
- RStudio integration for predictive modeling, Watson Analytics compatibility for cognitive intelligence, and a comprehensive partner ecosystem that adds more and more specialized analytics capabilities.

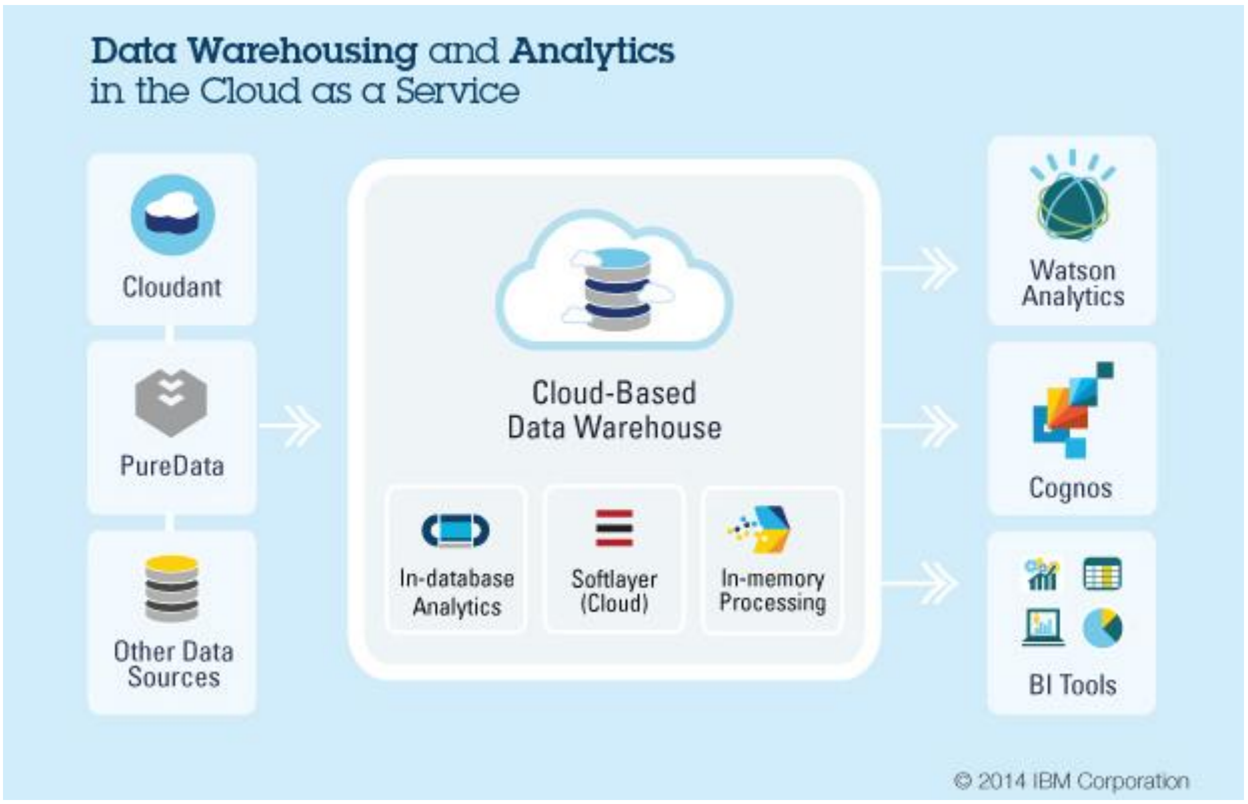


Fig.17: Simple data warehousing for every use case.

Easily ingest data from a myriad of sources — both structured and unstructured. It's easy to get your data into the cloud. dashDB can handle Cloudant-based JSON data stores, data on PureData for Analytics appliances, DB2, Excel and Oracle SQL data and many other types of data stores. Write and execute your own analytic queries, or leverage other analytic and BI capabilities that are available either on premises or in the cloud, such as Watson Analytics, Cognos or other BI tools.

6.12 MQTT Protocol

What is MQTT?

MQTT stands for MQ Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimise network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging "machine-to-machine" (M2M) or "Internet of Things" world of connected devices, and for mobile applications where bandwidth and battery power are at a premium.

Who invented MQTT?

MQTT was invented by Dr Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom (now Eurotech), in 1999.

Where is MQTT in use?

MQTT has been widely implemented across a variety of industries since 1999. A few of the more interesting examples are listed on the Projects page.

Is MQTT a standard?

As of March 2013, MQTT is in the process of undergoing standardisation at OASIS.

The protocol specification has been openly published with a royalty-free license for many years, and companies such as Eurotech (formerly known as Arcom) have implemented the protocol in their products.

In November 2011 IBM and Eurotech announced their joint participation in the Eclipse M2M Industry Working Group and donation of MQTT code to the proposed Eclipse Paho project.

How does MQTT relate to SCADA protocol and MQIsdp?

The “SCADA protocol” and the “MQ Integrator SCADA Device Protocol” (MQIsdp) are both old names for what is now known as the MQ Telemetry Transport (MQTT). The protocol has also been known as “WebSphere MQTT” (WMQTT), though that name is also no longer used.

What is WebSphere MQ Telemetry?

This is a product from IBM which implements the MQTT protocol in a very scalable manner and which interoperates directly with the WebSphere MQ family of products.

There are other implementations of MQTT listed on the Software page.

Are there standard ports for MQTT to use?

Yes. TCP/IP port 1883 is reserved with IANA for use with MQTT. TCP/IP port 8883 is also registered, for using MQTT over SSL.

Does MQTT support security?

You can pass a user name and password with an MQTT packet in V3.1 of the protocol. Encryption across the network can be handled with SSL, independently of the MQTT protocol itself (it is worth noting that SSL is not the lightest of protocols, and does add significant network overhead). Additional security can be added by an application encrypting data that it sends and receives, but this is not something built-in to the protocol, in order to keep it simple and lightweight.

6.13 Android Studio and Android OS

Android Studio is the official integrated development environment (IDE) for Android platform development.

Fig.18: Android Studio Logo

It was announced on May 16, 2013 at the Google I/O conference. Android Studio is freely available under the Apache License 2.0.

Android Studio was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

Based on JetBrains' IntelliJ IDEA software, Android Studio is designed specifically for Android development. It is available for download on Windows, Mac OS X and Linux, and replaced Eclipse Android Development Tools (ADT) as Google's primary IDE for native Android application development.

Android Studio – Android studio is and IDE developed by google itself for helping android developers in producing quality android applications. We used this environment for the same purpose as it is very helpful in development of those specific applications.

Download link: https://developer.android.com/studio/index.html

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Instant Run to push changes to your running app without building a new APK
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

Project Structure:

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules

- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under **Gradle Scripts** and each app module contains the following folders:

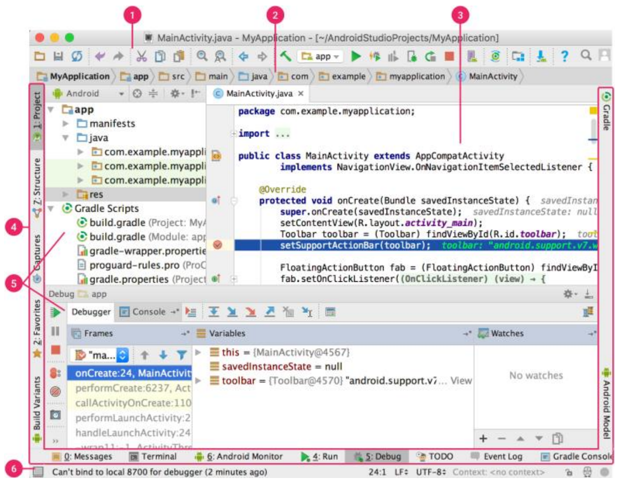
- **manifests**: Contains the AndroidManifest.xml file.
- **java**: Contains the Java source code files, including JUnit test code.
- **res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project**from the **Project** dropdown .

The User Interface:

The Android Studio main window is made up of several logical areas identified in figure below.

1. The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.
2. The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.
3. The **editor window** is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
4. The **tool window bar** runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.
5. The **tool windows** give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
6. The **status bar** displays the status of your project and the IDE itself, as well as any warnings or messages.



```
package com.example.myapplication;

import ...

public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {  savedInstan
        super.onCreate(savedInstanceState);  savedInstanceState: null
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  tool
        setSupportActionBar(toolbar);  toolbar: "android.support.v7.w

        FloatingActionButton fab = (FloatingActionButton) findViewByI
        fab.setOnClickListener((OnClickListener) (view) → {
```

```
onCreate:24, MainActivit
performCreate:6237, Act
callActivityOnCreate:110
performLaunchActivity:2
handleLaunchActivity:24

this = {MainActivity@4567}
savedInstanceState = null
toolbar = {Toolbar@4570} "android.support.v7... View
```

```
Can't bind to local 8700 for debugger (2 minutes ago)    24:1  LF÷  UTF-8÷  Context: <no context>
```

Fig.19: Snapshot of Android Studio

Chapter 7: Phases of Project in Detail

We will be discussing about the various phases involved in this project in detail. Some of the phases are still in work but we will be dealing with each of them individually explaining all. All the sub topics are explained with the help of various illustrations like tables, graphs, charts, images and programs. Let's begin with our first Phase i.e., Assembly of Electrical Components.

7.1 Assembly of Electrical Components

We have discussed previously about the various electronic components used in this project. So now it's time to put them together so that it works as we intend it to do. Building an IOT device is just a state of art. In this chapter I will show you, how these components have been put up to work and produce informative results.

Components

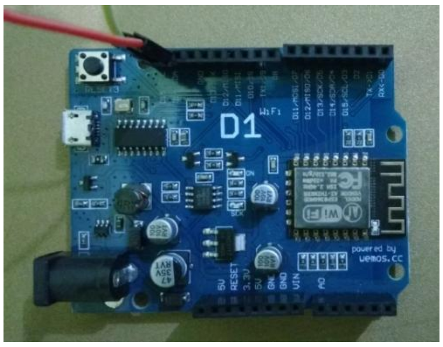


Fig.20: WEMOS D1 Board

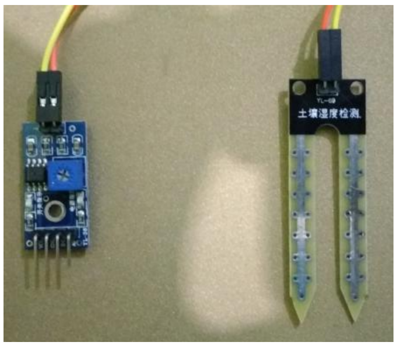


Fig.21: Soil Moisture Sensor

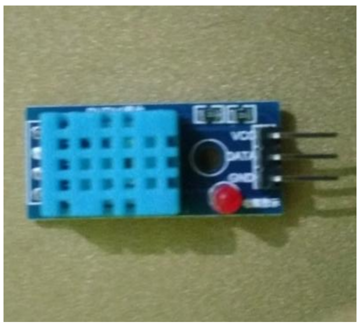


Fig.22: DHT-11 Sensor

These three components are put together so that the above mentioned sensors gather important information from the field on which the crop production has to be maximized. The circuit diagram below explains how these components have been assembled.

Circuit Diagram/ Fritz

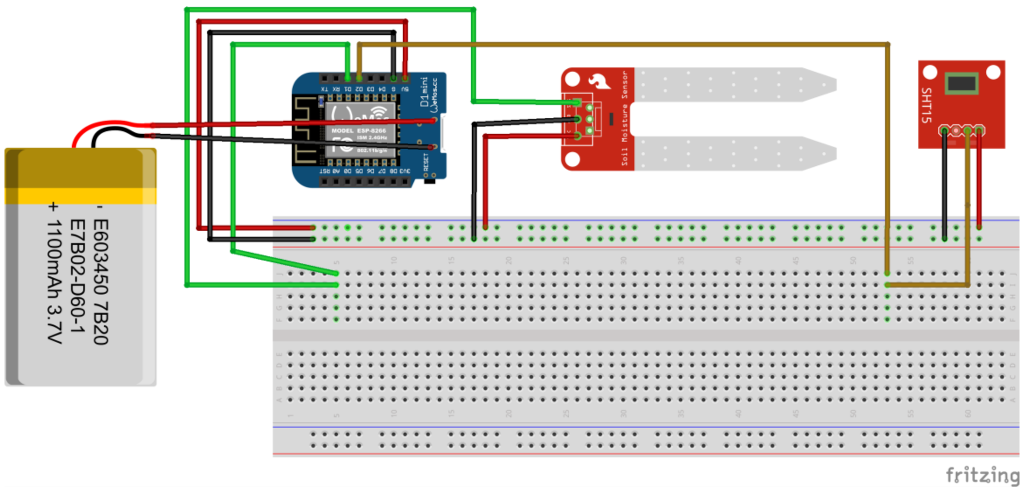


Fig.23: Fritz of the Project.

Wire Color	**Red**	**Black**	**Green**	**Olive**
Description	VCC	GND	Data from Soil Moisture Sensor	Data from DHT-11 Sensor

Assembly

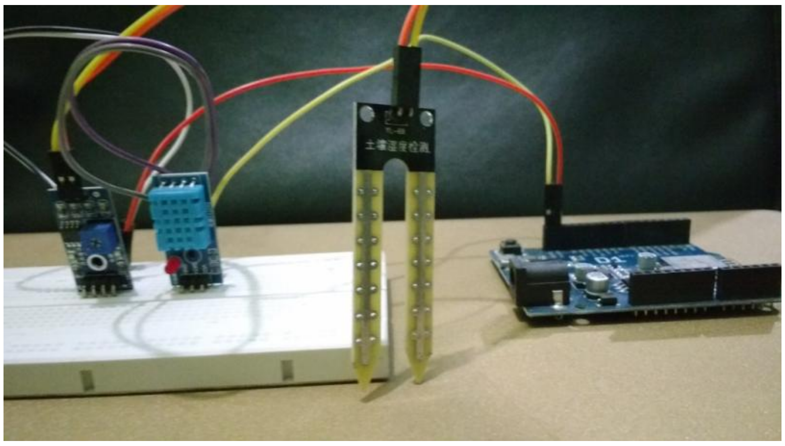


Fig.24: Front View of the Assembled Components

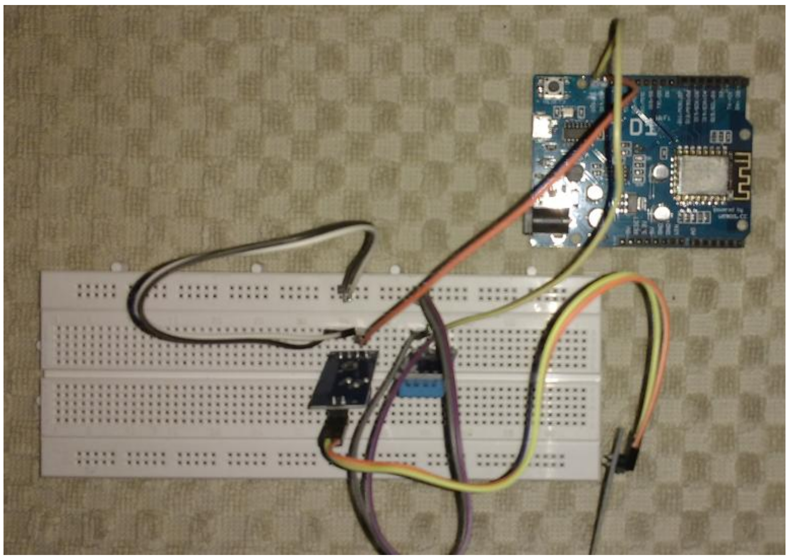


Fig.25: Top View of the Assembled Components

7.2 Coding of Arduino

```
#include <SPI.h> // needed in Arduino 0019 or later

#include <Ethernet.h>

#include <Twitter.h>

#include "DHT.h"
```

```
//DHT22 humidity and temperature sensor liberaries and initialisation definations.

#define DHTPIN 8     // what pin we're connected to

#define DHTTYPE DHT22   // DHT 22  (AM2302)

#define fan 13


DHT dht(DHTPIN, DHTTYPE);


//trigger event static variables

int maxHum = 75;

int maxTemp = 45;

int trig_h = 10;

int trig_t = 2;

int msg_count = 5;

//static utility variables

long picker;

float prev_h=100.0,prev_t=100.0;

//sentences set for temp increase

char incTmsg[][50]={

  "we going to be hot.",

  "be carefull of getting skin burn.",

  "tumhe garmi lagegi.",

  "you will need a hanky as you are going to sweat.",

  "its hot outside!"};

//sentences set for temp decrease

char decTmsg[][50]={

  "Finally it's getting cold.",

  "Man, it's freezing out here.",
  "I can't feel my leaves in this cold",

  "Abe AC kisne on ki!",

  "I can feel the freezing soil at my roots.",};

//sentences set for hum increase

char incHmsg[][50]={

  "It's too sweaty.",

  "Man, it's getting humid.",

  "Humidity MODE ON!",

  "I don't like being sweaty.",

  "it's to moisture outside."};

//sentences set for hum decrease

char decHmsg[][50]={

  "Finally, Humidity is gone.",

  "Humidity MODE OFF",

  "It's a good breeze outside.",

  "I miss being all sweaty.",

  "thank god, it's not humid anymore."};

// The includion of EthernetDNS is not needed in Arduino IDE 1.0
or later.

// Please uncomment below in Arduino IDE 0022 or earlier.

//#include <EthernetDNS.h>

// Ethernet Shield Settings

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED  };

// If you don't specify the IP address, DHCP is used(only in
Arduino 1.0 or later).3694679955-
1X9YzgVBgsTG1vQA3g6YJksypVDtXkgDbTIo20J //3694679955-
4IT3HibxylI75f1JufKxSRJFzsy98YnM5CfKoQh

byte ip[] = { 172, 16, 200, 124 };

byte gateway[] = { 172, 16, 200, 1 }; //Manual setup only
byte subnet[] = { 255, 255, 255, 0 };//Manual setup only

byte mydns[] = { 172, 16, 100, 2 };

// Your Token to Tweet (get it from http://arduino-tweet.appspot.com/)

Twitter twitter("3694679955-4IT3HibxylI75f1JufKxSRJFzsy98YnM5CfKoQh");

void setup()

{ //physical pinMode configuration for directly stacking the sensor on the board.

  pinMode(9,OUTPUT);

  pinMode(8,INPUT_PULLUP);

  pinMode(7,OUTPUT);

  digitalWrite(9,HIGH);

  digitalWrite(7,LOW);

  pinMode(fan, OUTPUT);

  Serial.begin(9600);

  dht.begin();

 // or you can use DHCP for autoomatic IP address configuration.

 // Ethernet.begin(mac);

  //Ethernet.begin(mac,ip);

  Ethernet.begin(mac, ip, mydns, gateway);

  Serial.println(Ethernet.localIP());

// if analog input pin 0 is unconnected, random analog

  // noise will cause the call to randomSeed() to generate

  // different seed numbers each time the sketch runs.

  // randomSeed() will then shuffle the random function.

  randomSeed(analogRead(0));
```

```
}

void loop() // functions to execute repeatively here...

{  // Wait a few seconds between measurements.

  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!

  // Sensor readings may also be up to 2 seconds 'old' (its a very
slow sensor)

  float h = dht.readHumidity();

  // Read temperature as Celsius

  float t = dht.readTemperature();


  // Check if any reads failed and exit early (to try again).

  if (isnan(h) || isnan(t)) {

    Serial.println("Failed to read from DHT sensor!");

    return;

  }

  //logic for triggering the twitter post if the change in
humidity or temperature is considerable here...

  float change_h = h - prev_h;

  float change_t = t - prev_t;

  prev_h = h;

  prev_t = t;

  Serial.print("new prev_h value : ");

  Serial.print(prev_h);

  Serial.print("%\t");

  Serial.print("new prev_t value : ");

  Serial.print(prev_t);

  Serial.println();
```

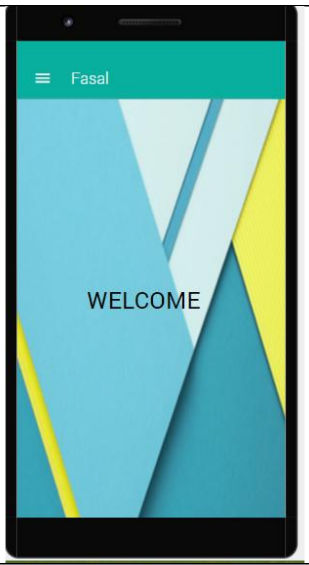
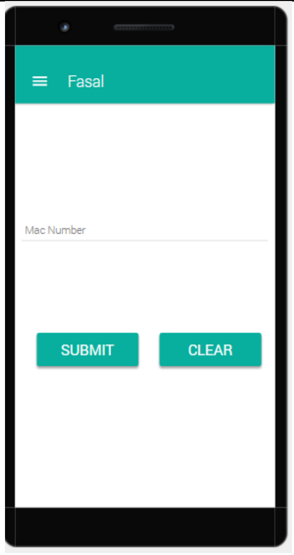
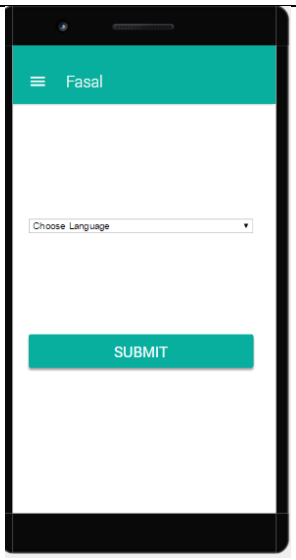
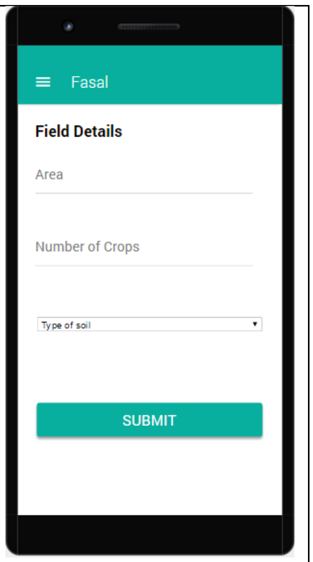
```
if(h > maxHum || t > maxTemp) {
      digitalWrite(fan, HIGH);
  } else {
     digitalWrite(fan, LOW);
  }
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C ");

  Serial.print("Changed Humidity: ");
  Serial.print(change_h);
  Serial.print(" %\t");
  Serial.print("Changed Temperature: ");
  Serial.print(change_t);
  Serial.println(" *C ");

      if(change_t >= (trig_t)){
    // statement for considerable increase in Temperature.
    picker = random(msg_count);
    doitbro(incTmsg[picker]);
    }
    else if(change_t < -(trig_t)){
    // statement for considerable decrease in Temperature.
    picker = random(msg_count);
    Serial.print(" Picker value : ");
    Serial.print(picker);
    Serial.println();
    doitbro(decTmsg[picker]);
    }
     else if(change_h >= (trig_h)){
    // statement for considerable increase in humidity.
    picker = random(msg_count);
    doitbro(incHmsg[picker]);
    }
    else if(change_h < -(trig_h)){
   // statement for considerable decrease in humidity.
    picker = random(msg_count);
    doitbro(decHmsg[picker]);
    }
    else {}
   // Wait an 2hour between measurements.
    delay(1000*60*24*2);
    //delay(1000*5);
}
void doitbro(char msg[]){
 delay(1000*5);
        // Twitter message posting implementation code here...
      Serial.println("connecting ...");
      if (twitter.post(msg)) {
        // Specify &Serial to output received response to Serial.
        // If no output is required, you can just omit the
argument, e.g.
```

```
      // int status = twitter.wait();
      int status = twitter.wait(&Serial);
      if (status == 200) {
        Serial.println("OK.");
      }
      else {
        Serial.print("failed : code ");
        Serial.println(status);
      }
    }
    else {
      Serial.println("connection failed.");
     doitbro(msg);
    }
}
```

7.3 Android User Interface

			 Fasal Field Details Area Number of Crops Type of soil SUBMIT
Fig.25: Welcome Screen	Fig.26: MAC Number	Fig.27: Choose Language	Fig.28: Acquire Data from User

Chapter 9: Results and Discussions

Till the date, of all progress we have achieved till date, we have came up with the following Results.

1. Educated farmers can easily use this application to maximize their profits.

2. Food producing organisations and companies can use to maximize their profit and fulfil the needs of population.

3. Crowd sourcing will not bias the prices of the crops and reduce corruption in marketing of agricultural goods.

Chapter 10: Project Log

This chapter contains the Project Log from the author. This contains information to backtrack and reinvent if any mishap happens or as if thought so to redevelop.

Project Log

```
16:33 27-08-2016

Creating a Test application which will be taking Android Phone
Sensor Data on the Bluemix Cloud App and then get response back to
Android Phone.

16:34 27-08-2016

Gathering Resources for the test application.

1. A Bluemix account.

2. Download and install the Cloud Foundry command line interface.
(https://console.ng.bluemix.net/docs/starters/install_cli.html)(ht
tps://github.com/cloudfoundry/cli/releases)

3. A mobile phone (an Android or iOS device)

16:37 27-08-2016

Waiting for download of CloudFoundary CLI Setup.

16:38 27-08-2016

Create an IoT app in Bluemix. Follow Steps on the Link Below.

(https://www.ibm.com/developerworks/library/iot-mobile-phone-iot-
device-bluemix-apps-trs/)

16:46 27-08-2016

Created a IOT Platform Starter Boilerplate Application named
"HotDevice" which will gather temperature sensor data from Android
device.

17:05 27-08-2016

Staging complete for AppName "HotDevice"
(http://hotdevice.mybluemix.net/)

17:06 27-08-2016
```

Waiting for download of android-studio-bundle-143.310....exe to finish.

17:17 27-08-2016

Launched the IoT Platform Dashboard and added my Android Device.

Device 24DA9B70EDBA

Your Device Credentials

Organization ID	4acgx4
Device Type	Android
Device ID	24DA9B70EDBA
Authentication Method	token
Authentication Token	cq91wv)AjvwYYnF4_y

17:20 27-08-2016

IoT Starter for Android app will be used to read and send sensor data on your mobile phone.(https://github.com/ibm-watson-iot/iot-starter-for-android)

Open this app in the android studio and develop according to your need.

Or Else

Download a Quick Starter from the link. (https://ibm.box.com/iotstarterapp)

17:27 27-08-2016

Installed the Android Starter App and Started Sensor Data.

17:28 27-08-2016

Started installation of Android Studio.

Bibliography

i. Guidance from Er. Harkomal Preet Kaur(Department of CSE, SVIET, Banur, Punjab)
ii. Github Repository: https://github.com/codegauravg/IOT-in-Crop-Production
iii. Android Studio: https://developer.android.com/studio/intro/index.html
iv. MQTT Protocol: http://mqtt.org/faq
v. dashDB: https://www-01.ibm.com/software/data/dashdb/technology.html
vi. fritzing: http://fritzing.org/learning/
vii. node.JS: https://nodejs.org/en/about/
viii. Arduino Wiki: https://en.wikipedia.org/wiki/Arduino
ix. Cloud Foundary: https://www.ibm.com/developerworks/cloud/library/cl-bluemixfoundry/cl-bluemixfoundry-pdf
x. Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/Glossary/JavaScript
xi. JSON: http://www.json.org/
xii. JSON Introduction: http://www.w3schools.com/js/js_json_intro.asp
xiii. Node-RED Documentation: https://nodered.org/docs/
xiv. Soil Moisture Sensor Wiki: https://en.wikipedia.org/wiki/Soil_moisture_sensor