

CHAPTER 1

INTRODUCTION

1.1 General

The idea of precision agriculture has been around since the advent of various wireless communication technologies like GPS, GNSS, Wi-Fi etc. In today's world, farmers can use these technologies along with sensors in order to control various parameters within their fields. WSN (Wireless Sensor Network) is best suited for this purpose, as it assists in computing, communication and control within devices. By implementing a smart decision making algorithm, various parameters various parameters inside a farm like moisture, temperature, pH and light intensity can be controlled. This will help the farmers increase and monitor the yield of their crops.

1.2 Objectives

The following are the objectives of this project:

- To create a wireless sensor network to monitor the various conditions inside a rice field. (Soil moisture, temperature, pH and light intensity).
- To control an irrigation mechanism using solenoid valve.
- To collect sensor values (using Atmega 328).
- To communicate these values to the border router (raspberry pi) using Zigbee transceivers.
- To classify these values (decision mechanism) at the server and send appropriate response to the nodes.
- To instruct the nodes to act according to the server's instructions .i.e. control the water flow.
- To host a website, to which all the collected data will be pushed.
- To show the various parameters of the field in real time.

1.3 Background and Literature Survey

A similar project was undertaken by students of World Academy of Science, Engineering and Technology to monitor plants inside a greenhouse. They used Star network topology to monitor the micro climate close to plants and control irrigation process. Sensing nodes were deployed inside the greenhouse among plants to measure climate parameters (Temperature and Humidity). Each sensing node transceiver module was configured to work as ZigBee end device (ZED). On the other hand, Control node was fixed near to the solenoid valve. It was also responsible to drive actuators, start network, and manage network data traffic.

They found that the system had the ability for real-time monitoring the temperature and humidity surround plants and controlling soil water contain to obtain maximum crop growth in the greenhouse. The system depended on high accuracy and short time response sensors to collect climate data to extend life time of the system. ZigBee protocol is emerging wireless technology. It supports low cost, low data rate, low power consumption, communication range about (40m), self-organize, and self-healing. So that, it is best choose to use with WSN. Their work can be found at the following link [5]. But unlike this, in our project, intends to measure parameters other than temperature and moisture, i.e. pH and light intensity. The amount of lime required will also be calculated based on the pH value. The main assumption related to node communication is, that the range of ZigBee is 60 meters instead of 40 meters so that more area can be covered.

Table 1.1 shows the parameters and conditions that govern the optimum yield of rice fields. Also describes the pH values and corresponding fertilizers required in order to maintain the optimum conditions.

Table 1.1 Literature Survey

SL.No	Name of article	Name of journal	Information included
1.	Fertility and organic matter in submerged rice soils	CURRENT SCIENCE, VOL. 88, NO. 5, 10 MARCH 2005	Soil temperature and pH
2.	Efficient Fertilizer Use — Soil pH Management	Efficient Fertilizer Use Manual(www.plantstress.com/articles/toxicity_m/soilph%20amend.pdf)	Liming conditions
3.	The role of nitrogen fertilizer in soil pH levels	AG Professional	Fertilizer and pH relationship
4.	Greenhouse micro climate monitoring based on WSN with smart irrigation technique	International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering Vol:7, No:12, 2013	<ul style="list-style-type: none"> - Sampling rate t0(20 mins) - Idea for developing wireless sensor based smart agriculture
5.	Cultivation of Rice: Suitable Conditions Required for the Cultivation of Rice (6 Conditions)	Your article library (http://www.yourarticlelibrary.com/cultivation/cultivation-of-rice-suitable-conditions-required-for-the-cultivation-of-rice-6-conditions/25491/)	Optimum soil moisture level
6.	Goodput Evaluation of AODV, OLSR and DYMO Protocols for Vehicular Networks Using CAVENET	Network-Based Information Systems (NBIS), 2011 14th International Conference on , vol., no., pp.118,125, 7-9 Sept. 2011	Information about AODV protocol

1.4 Need for Precision Agriculture

India being an agricultural country needs some innovation in the field of agriculture. This can be achieved through modern technologies which assist computing, communication and control within devices. WSN is well suited for this purpose. Using a smart decision making algorithm, we can control the various parameters inside a farm like moisture, temperature, pH and light intensity. This will help farmers to ensure the yield of their crop.

1.5 Organization of the Report

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the methodology, design approach, standards used and constraints of the methodology used in the project.
- Chapter 3 contains the implementation of precision agriculture with server node algorithm and end node algorithm. It also explains the communication between end node and server node.
- Chapter 4 gives the cost involved in the implementation of the project and analyses its economic feasibility.
- Chapter 5 compiles the results obtained after the project was implemented.
- Chapter 6 concludes the report with discussions about the results obtained and their future implications.

CHAPTER 2

PRECISION AGRICULTURE SYSTEM FOR RICE

This Chapter describes the methodology, design, standards, calibrations, sampling rate and constraints of Precision Agriculture system.

2.1 Methodology

The idea implemented in this project is server side decision making, which gives user flexibility while deploying the node, as the thresholds are decided and implemented at the server itself, so if the values change the end-nodes don't have to be re-programmed, plus the node can have low processing capability.

For demonstration of our project we grew rice in a small pot and attached the various sensors for measuring soil moisture, pH, temperature and light intensity, with Atmega 328 through the analog input pins as shown in Figure 2.1. The Atmega328 also controls the 12V solenoid valve through the relay. It is also connected to Xbee module through UART. Rx and Tx pins are used for connecting to the Xbee, as shown in Figure 2.2. The Xbee is configured as leaf nodes/routers. Different soil samples (without rice) of varying pH and soil moistures are used to compare with the values received from the rice pot and also to simulate a WSN network with three nodes.

The data is collected and sent to the coordinator node (Raspberry Pi) using Zigbee module. Raspberry pi is connected to the internet using a WiFi module, and hosts the website which displays these statistics and corresponding graphs.

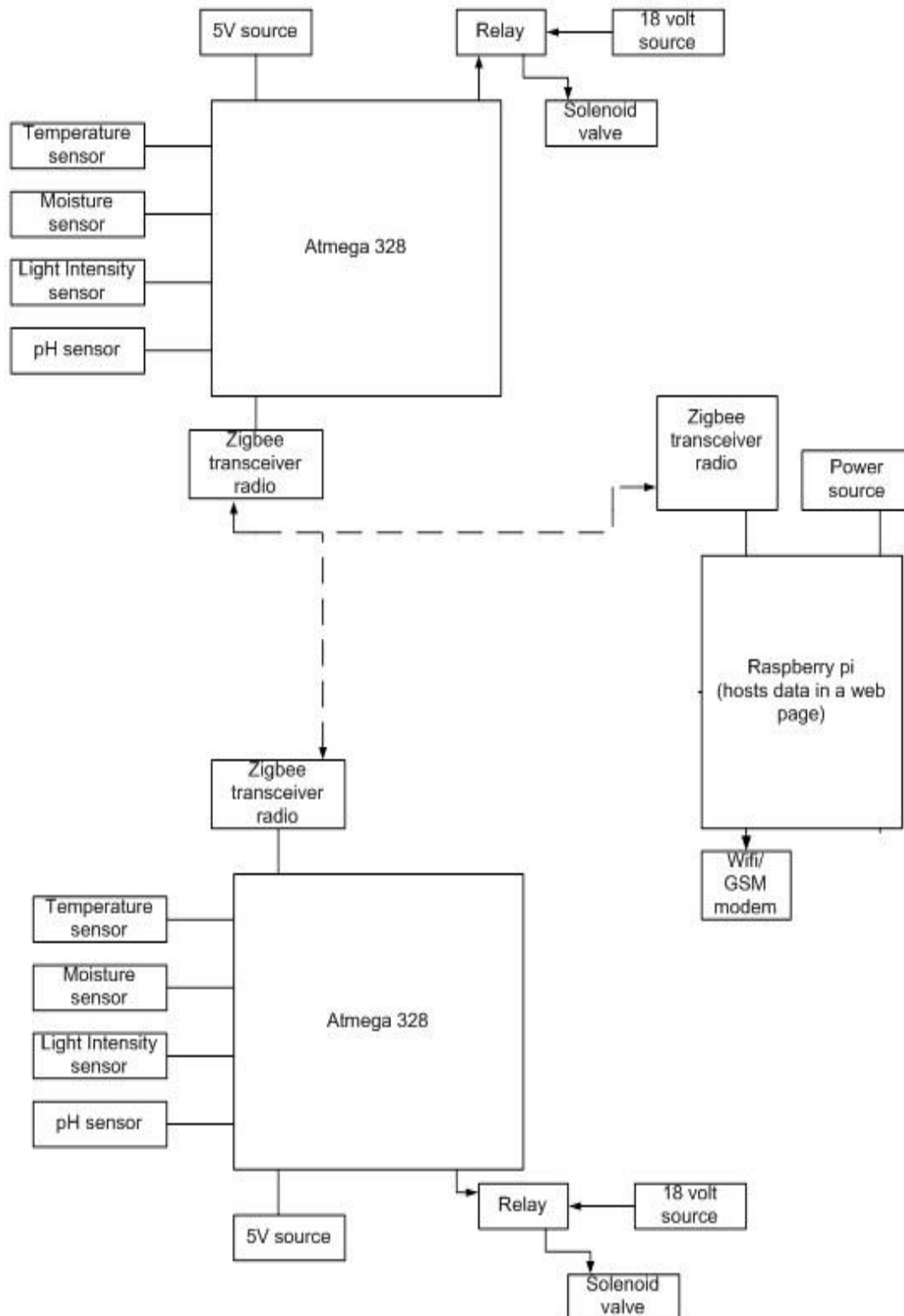


Figure 2.1 Block diagram of Precision Agriculture System for Rice Crop

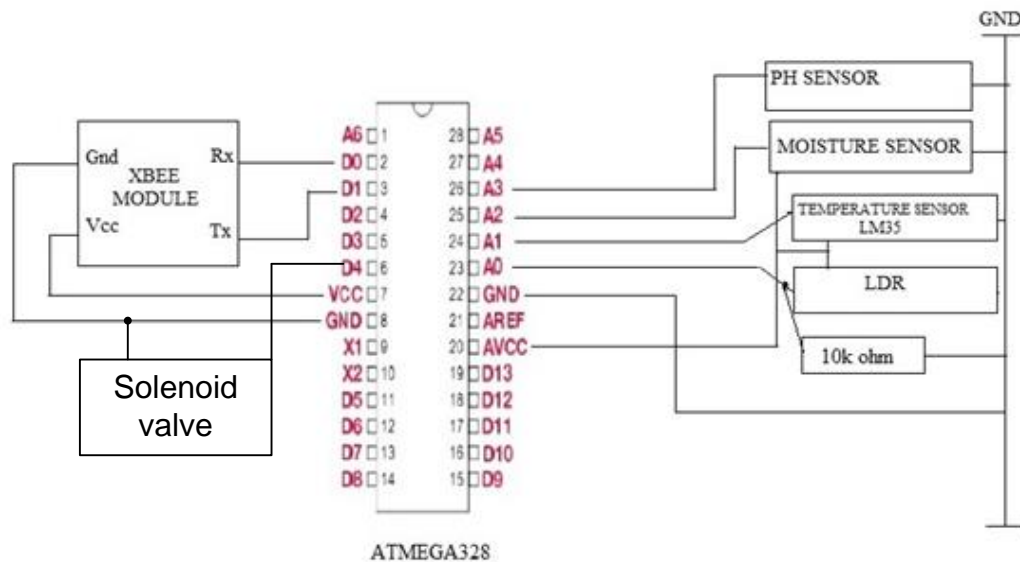


Figure 2.2 End-Node Sensor connections

2.2 Design Approach

The server side approach is implemented in the system, so as to reduce the required processing capability at the end-node. The added benefit to this approach is that when the threshold values change, the end-node don't need to be re-programmed, and only the server has to be programmed.

The Raspberry pi acts as the server and performs the following functions:

- It checks the individual headers of the packet and determines which category of data has been received.
- Then it stores valid data in corresponding variables and only when all the variables have been assigned, they are uploaded into Sqlite3 database for 24hrs, as shown in Figure 2.3.
- After 24hrs the table is checked for missing nodes i.e. Nodes that haven't sent data that day, and then these node numbers are stored into the MySQL database. During this process we also check the number of times that the nodes have sent the light intensity greater than threshold, then this value is compared with the

desired value and if it is insufficient then the time that the node received desired intensity of sunlight is stored in the MySQL database.

- After this process the data in the Sqlite3 table is deleted.
- Parallel to this process, if the data exceeds the set threshold, that set of data is uploaded into MySQL database, as shown in Figure 2.4.

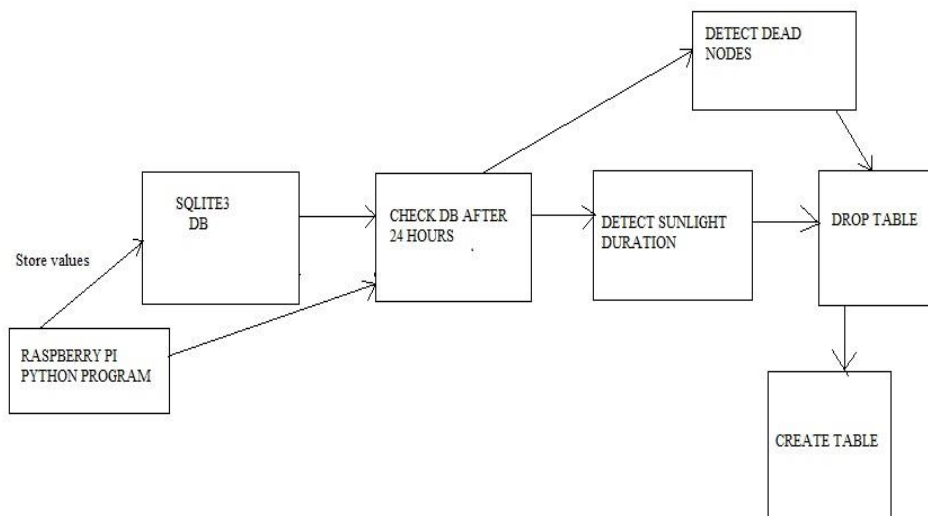


Figure 2.3 Node Status and Sunlight Duration

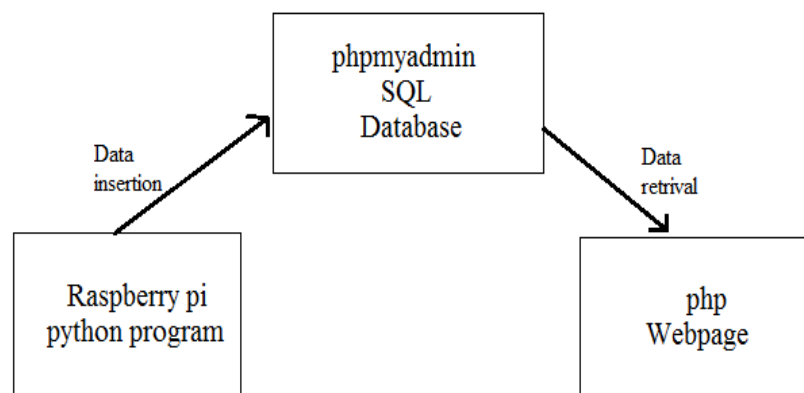


Figure 2.4 MySQL Database functionality diagram

2.3 Standards

This section describes the standards used for the WSN and the border router.

2.3.1 For WSN (local wireless Sensor network):

Physical, Mac and Datalink layer: 802.14.5 standard

The IEEE 802.15 TG4 was chartered to investigate a low data rate solution with multi-month to multi-year battery life and very low complexity. It is operating in an unlicensed, international frequency band. Potential applications are sensors, interactive toys, smart badges, remote controls, and home automation.

- Data rates of 250 kbps, 40 kbps, and 20 kbps.
- Two addressing modes; 16-bit short and 64-bit IEEE addressing.
- Support for critical latency devices, such as joysticks.
- CSMA-CA channel access.
- Automatic network establishment by the coordinator.
- Fully handshaked protocol for transfer reliability.
- Power management to ensure low power consumption.
- 16 channels in the 2.4GHz ISM band.

Network layer: The routing protocol used by the network layer is AODV.

[8]Ad hoc On Demand Distance Vector (AODV) is a reactive protocol. It is an improvement of DSDV, because it minimizes the number of broadcast packets [7] generated by creating route only when required. All nodes in the network maintain the route information in a table and participate in sharing their routing tables. [7] The source node starts the route discovery process, when it wants to transfer data to the destination node. During this process, source node disseminates Route Request [7](RREQ) packet to its neighbors. If the neighbor nodes which receive RREQ, lack the information pertaining to the request, they forward the packets to their neighbor nodes. The following process

goes on until RREQ reaches, the destination or the node who knows the path to the destination. Intermediate nodes set up a reverse path by recording the addresses of the neighbors in their tables when, they receive RREQ. [7] When the destination node or the node which has information about the path to the destination receives RREQ, it sends back Route Reply (RREP) packet to the source node. This RREP packet is sent through the reverse course. The source node assimilates the course to destined node and the discovered course information is placed in the routing table, when it receives the RREP packet.

Application layer: Zigbee

[10] It enables the services necessary for Zigbee device object and application objects to interface with the network layers for data managing services. This layer is responsible for matching two devices according to their services and needs. It provides two types of data services as key value pair and generic message services. Generic message is a developer defined structure, whereas the key value pair is used for getting attributes within the application objects. ZDO provides an interface between application objects and APS layer in Zigbee devices. It is responsible for detecting, initiating and binding other devices to the network.

2.3.2 For Border Router:

Physical layer: 802.11n standard

1. Uses multiple input / multiple output (MIMO) technology and a wider radio frequency channel.
2. Provides a mechanism called frame aggregation to decrease time between transmissions
3. Frequencies 5GHz or 2.4 GHz.
4. There is sufficient room at 2.4 GHz for three 20 MHz channels, but only one 40 MHz channel can be accommodated.
5. Data rate as high as 540 Mbps with typical rates between 100 and 200 Mbps

Application layer: TCP (To display the page), FTP (to retrieve data from the MySQL database of pHpMyadmin), UDP (to get the DNS) and DHCP (to assign IP to the Raspberry pi 2) protocol

Xbee S2: As they provide multi-hop facility so the range increases without increasing the power supplied to the transmitter, and this parameter is of high importance in WSN where power efficiency is the goal.

2.4 Calibrations and Sampling Rate

This section describes the calibrations and the sampling rate of various sensors used in Precision agriculture system.

2.4.1 Calibrations

1. Moisture: Insert the sensor 3 cm deep into the soil. Table 2.1 shows the calibrations of the moisture sensor.

Table 2.1 Moisture sensor calibration

Analog value	Condition
1023	totally dry
500	higher bound (dry if greater)
185	lower bound (excess water if greater)

2. Light: 120 and greater analog value, for 6 hours at least
3. pH: Table 2.2 shows the calibration of the pH sensor.

Table 2.2 pH sensor calibration

pH	Analog value
4	30> = >=24
4.5	23
5	21,22& 20
5.5	18>= >=14
6	13>= >=9
6.5	8>= >=5

7	4 & 3
7.5	2 & 1
8	0

4. Temperature (in Celsius): optimum temperature range for growing rice is 25 to 35 Celsius.

For converting analog temperature value: $(A0/1024.0)*500$, where A0 is the analog reading.

5. Liming conditions :

The range of soil pH for rice crop is 5.5 to 7.5 for optimum growth [12]. So we can select 6.5 to be the optimum pH for rice. To increase the pH of soil, lime is generally added, as given in Table 2.3.

Table 2.3 Liming conditions

Soil water pH	Amount of lime to add(Tons/Acre) to give pH 6.5
4.5 – 4.9	4.0
5.0 – 5.2	3.5
5.3 – 5.5	3.0
5.6 – 5.8	2.5
5.9 – 6.0	2.0
6.1	1.5
6.2 – 6.3	1.0

2.4.2 Sampling rate:

Sampling time t_0 has been fixed as 20 mins as a similar project was previously implemented successfully with the same sampling time [5]. Sampling time t_1 has been fixed to 24 hours to forcibly check the light received and the status of the nodes.

2.5 Constraints, Alternatives and Trade offs

1. **Constraint:** Xbee communication range is 100 m.

Alternative: We selected Xbee range as 60 m, assuming the soil properties to be constant within this range.

Trade off: Soil properties may vary spatially across a field within the communication range of Xbee.

2. **Constraint :** pH sensor gives value between 0-8 only. It is also very expensive.

Alternative : We selected optimum pH for rice to be 6.5(5.5 to 7.5). We attached pH sensor only to one node due to cost constraint.

Trade off : Any pH higher than 8 cannot be measured.

Node Deployment: As this element depends on the geographical conditions present in the real time environment, for our project we have assumed that the conditions are nearly constant/uniform for the range of 11309.7336 sq. Meters (approx. 2.7 acres) as the zigbee transceiver can easily communicate in range of the 60-100m. So, each node will cover the area of approx 2.7 acres.

2.6 TECHNICAL SPECIFICATIONS

Components:

- Moisture sensor: Soil Moisture Sensors measure soil moisture with patented TDT (Time Domain Transmission) technology. biSensors are self-calibrating for all soil types and conditions and are unaffected by salty soil conditions or soils with a high pH. biSensors can be connected to a two-wire path or conventional wire and provide continuous measurements and real-time feedback for the controller to make smart irrigation decisions specific to the landscape the biSensor is installed in.
- Temperature sensor LM35: The LM35 series are precision integrated-circuit Calibrated Directly in Celsius (Centigrade) temperature devices Factor proportional to the Centigrade temperature.
- Hydrofarm MGMLP1 Active Air 3-Way pH, Light and Moisture Meter:_ pH readings between pH 4 and 8.
- LDR: Two cadmium sulphide (cdS) photoconductive cells with spectral responses similar to that of the human eye. The cell resistance falls with increasing light intensity.

- Solenoid valve: 12V, 10W Used to control the water flow.
- Raspberry pi: will be used as border router.
- Wifi router: to provide internet access to the raspberry pi.
- Jumper wires: for connections.
- Xbee series 2: Zigbee transceivers that use 802.14.2 standard. Outdoor range 100m. .
- Atmega328: used to get sensor values and send them to the border router.

2.7 Summary

Thus the methodology, design approach, standards implemented, technical specifications and the constraints of the project were discussed in this chapter. The implementation of the precision agriculture system is presented in Chapter 3.

CHAPTER 3

IoT BASED PRECISION AGRICULTURE IMPLEMENTATION

This chapter describes the implementation of Precision Agriculture System. It describes server-node algorithm, end-node algorithm and the communication between server-node and end-node.

3.1 Server-node algorithm

1. The collected data (from the end node) will be compared to corresponding threshold values and if it exceeds/is less than it, then we will push the data to the table called 'Sensor1' of the MySQL database called 'Sensor_Statistics', as shown in Figure 3.2 and Figure 3.3.
2. To determine if the node is dead/alive we will store all the data regardless of its nature i.e. even if it does not violate the threshold, in Sqlite3 database and check will be conducted on daily (after time t1)bases to determine the status of the nodes. After this determination the table will be emptied to store next day's data, as shown in Figure 3.1.
3. The Raspberry pi will make certain decisions based on the sensor values and send the Atmega328 instructions to take appropriate action when the thresholds are violated, as shown in Figure 3.6.
 - a. If the water level is low, the raspberry pi will send the threshold value to the particular node and the solenoid valve will be activated and will remain active till the level becomes optimum. If it is excess the farmer will be alerted.
 - b. If the pH is not in the desired range the lime required to be added will be calculated.
 - c. If the light intensity is above threshold for less than 6 hours the farmer will be alerted. All the data is stored in the Sqlite3 database on daily basis and at the end of the day (t1 time duration), we will check the number of times the node sends values above threshold and multiply it by the sampling rate, to get the total duration.

- d. If the temperature is not in the optimum range the farmer will be alerted.
4. The data sent to the Raspberry pi will be pushed to the webpage hosted by the raspberry pi. This webpage can be accessed by a specific username and password.

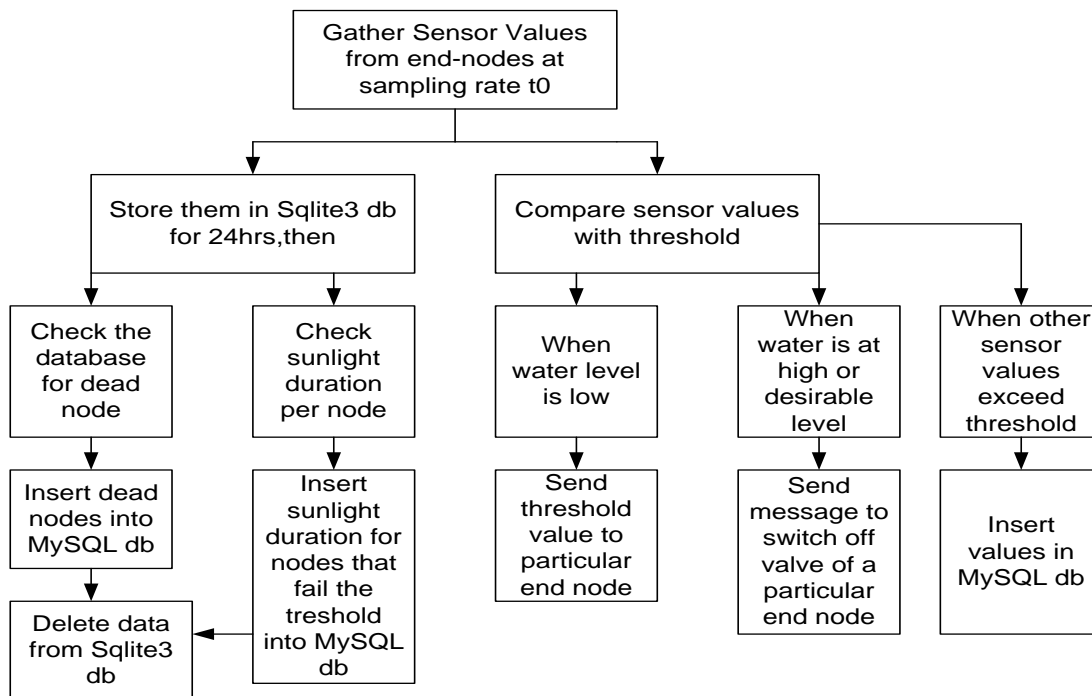


Figure 3.1 Server-node flow diagram

Showing rows 0 - 29 (~126 total) . Query took 0.0025 sec

```

SELECT *
FROM `Sensors1`
LIMIT 0, 30
  
```

	Id_no	Node_no	time_stamp	Crop	Temp	light_Intensity	moisture	pH	lime_req	nodes_down
<input type="checkbox"/>	1	2	2016-03-21 23:04:21	Rice	24	90	600	5.5	3	0
<input type="checkbox"/>	2	2	2016-03-21 23:04:21	Rice	24	90	600	5.5	3	0
<input type="checkbox"/>	3	1	2016-03-21 23:04:59	Rice	24	190	300	4.5	4	0
<input type="checkbox"/>	4	1	2016-03-22 00:32:48	Rice	20	130	200	8	R	0
<input type="checkbox"/>	5	1	2016-03-22 00:32:48	Rice	20	130	200	8	R	0
<input type="checkbox"/>	6	1	2016-03-22 00:32:48	Rice	20	130	Remove_water	8	R	0
<input type="checkbox"/>	7	1	2016-03-22 00:35:05	Rice	25	130	240	7	0	0

Figure 3.2 Database start page

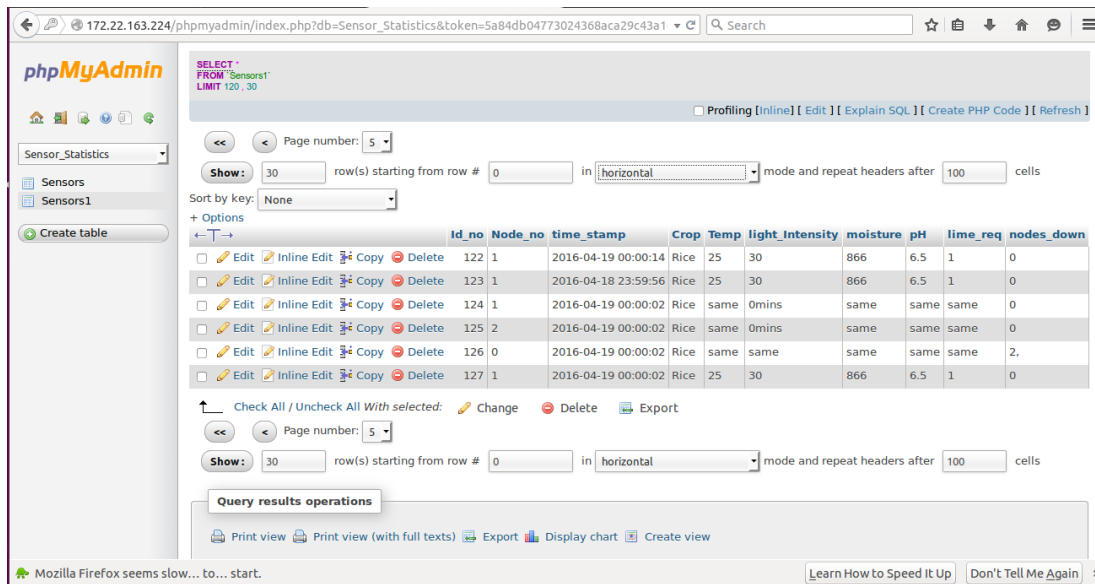


Figure 3.3 Database end page

3.2 End-node algorithm

1. The Atmega328 will collect the analog reading from the sensors at a sampling rate of t_0 and it will be in sleep mode at other time, as shown in Figure 3.4.
2. This data will be sent to the border router (raspberry pi) using Xbee S2 transceiver.
3. The server sends a message to switch on the solenoid valve when the moisture content is low and specifies the condition in the message so that the node powers the valve until it attains desirable moisture level. Otherwise it sends valve close message, as shown in Figure 3.5.

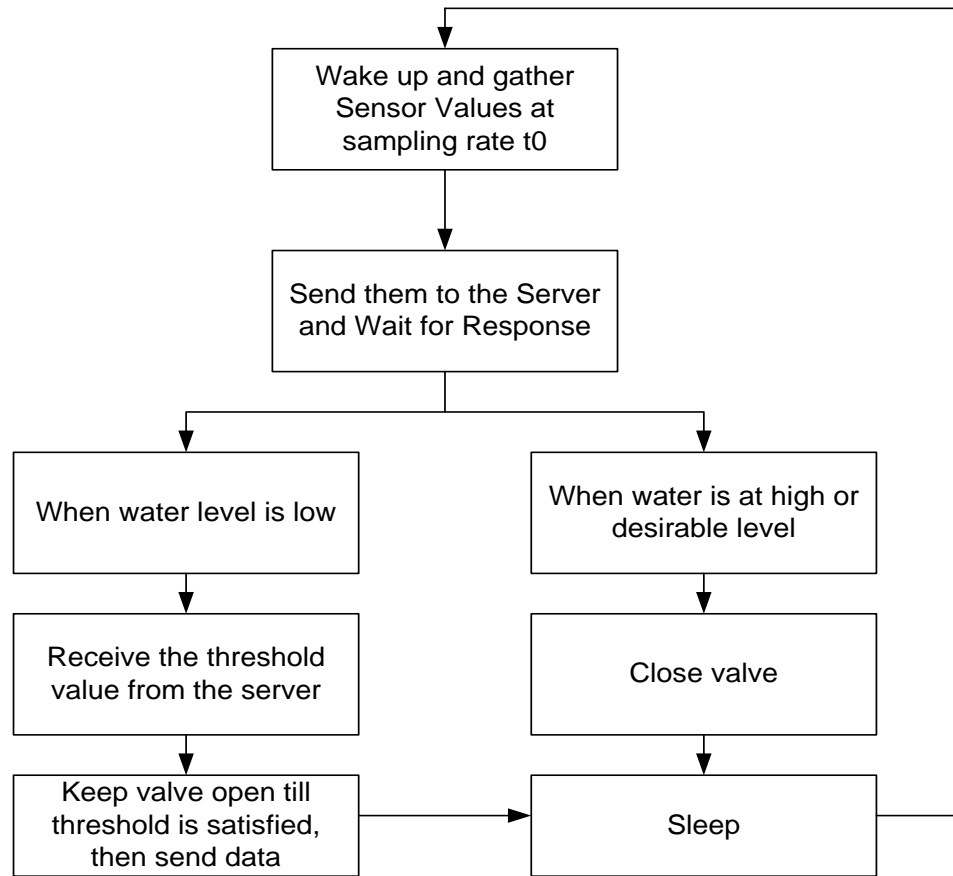


Figure 3.4 End-node flow diagram

3.3 Transaction between Server-node and end-node

The data is sent by the end nodes to the raspberry pi in the form of frame.

E.g.; N1; CRice; T24; L180; M220; P7;

Here the ‘;’ acts as the delimiter and separates the various sensor data.

- N1: denotes node number1.
- CRice: denotes crop type is Rice.
- T24: denotes temperature of the node is 24C.
- L180:denotes the light intensity of the node is 180
- M220:denotes the moisture of the sensor is 220(within the limits)
- P7: denotes the pH is 7.

The data sent by the Raspberry pi is of two types:

1. To switch on the solenoid valve:

E.G.; N1; W500;

Here the ';' acts as the delimiter

- N1: denotes node number1.
- W500: denotes 500 is the threshold for moisture and the valve has to be on till the moisture analog value becomes less than it.

2. To switch on the solenoid valve:

E.G.; N1; W0;

Here the ';' acts as the delimiter

- N1: denotes node number1.
- W0: denotes that analog value of moisture sensor is less than the threshold and the valve has to be switched off.

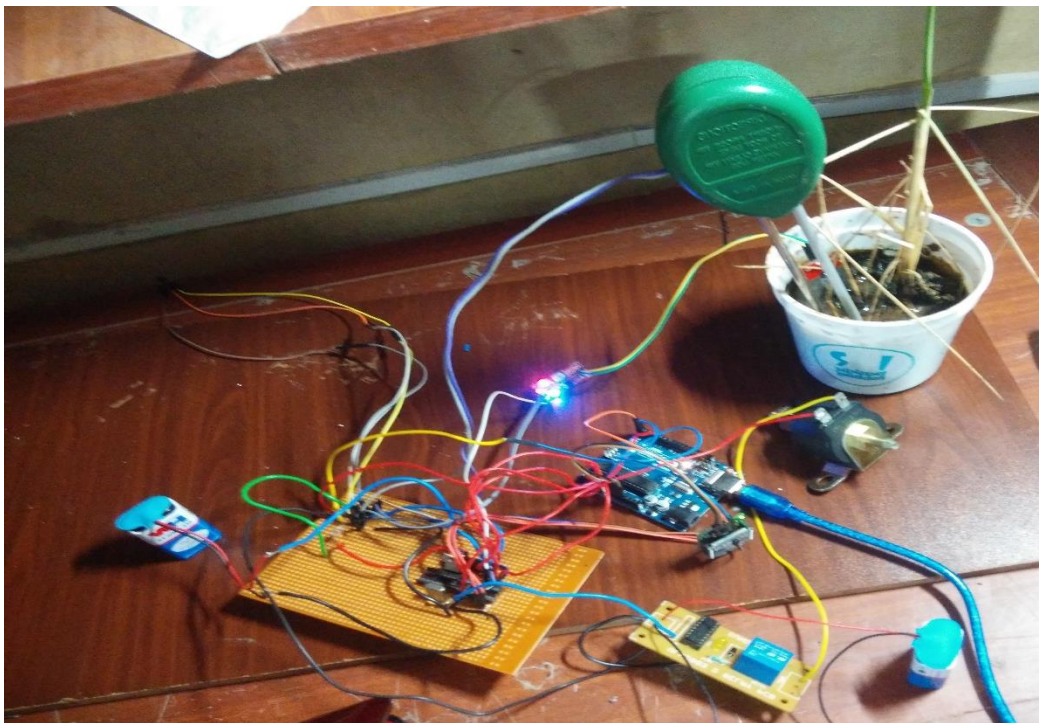


Figure 3.5 Collection of readings from sensors and sending them to coordinator

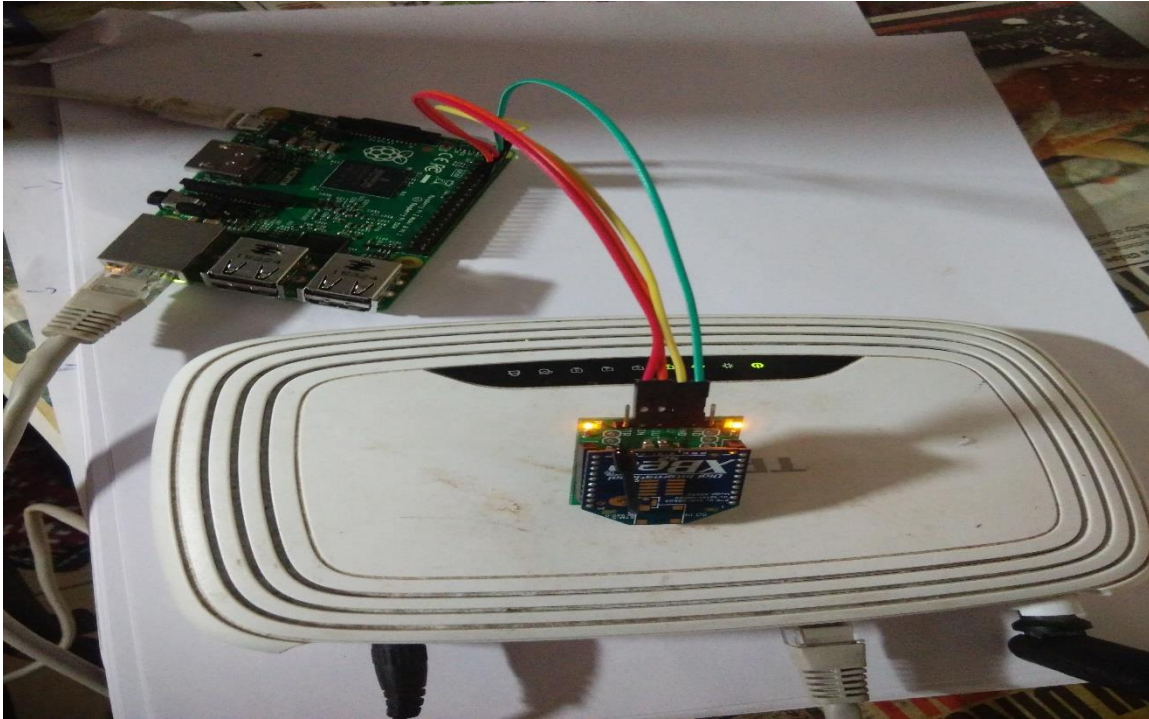


Figure 3.6 Raspberry Pi setup as coordinator to receive values from end nodes

3.4 Summary

In this chapter the implementation of Iot based precision agriculture has been described. The server node algorithm describes how the server node processes the data collected by the end nodes and the decisions it makes based on that data. End node algorithm describes how the data is collected and sent to the server by the end nodes, and how they implement the action specified by the server. The communication between the server and the end node is described in subsection 3.3. In the next chapter the cost analysis of various components of the project is given.

CHAPTER 4

COST ANALYSIS

4.1 List of components and their cost

The costs of the various components used in this project are given below in Table 4.1.

Table 4.1 List of components and their costs

COMPONENT	COST
Raspberry Pi	Rs 3000
Atmega328 (2 no.s)	Rs 360
Xbee modules (2 no.s)	Rs 2000
Xbee shields (2 no.s)	Rs 400
pH sensor (2 no.s)	Rs 1500
Moisture sensors (2 no.s)	Rs 290
LDR (2 no.s)	Rs 10
LM 35 (2 no.s)	Rs 70
Solenoid valve (2 no.s)	Rs 600
TOTAL	Rs 8230

4.2 Cost Effectiveness

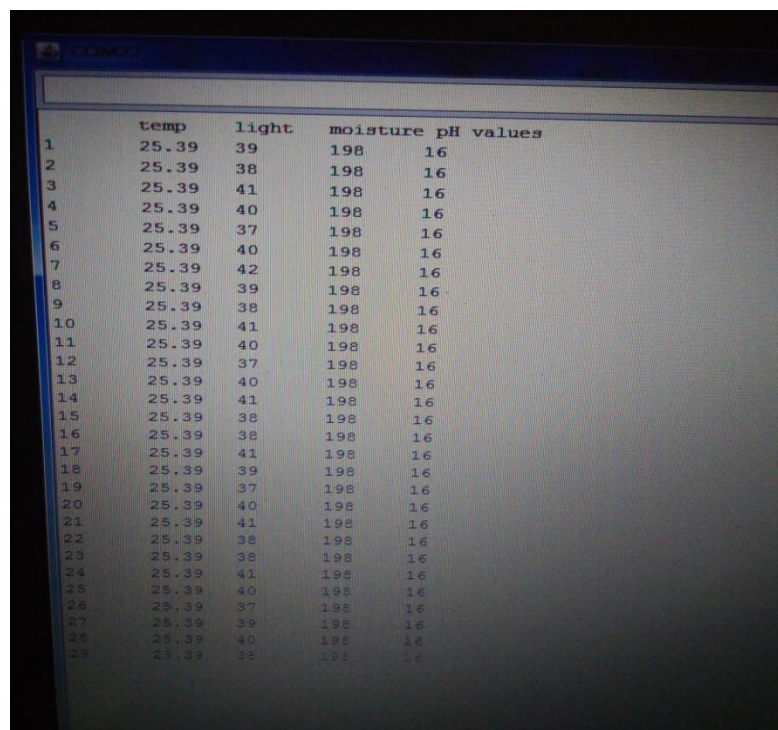
We have put one node per 2.7 acres (11309.7336 sq. Meters), which is the area covered by each Xbee module (60 m radius). We have assumed that the soil properties are constant within this range. Cost of each node is Rs 2615. Thus cost per acre is around Rs 970. It is economically viable for farmers with large areas of land.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Sensor Readings

The end nodes were able to transmit the values collected from the sensors to the Raspberry Pi, as shown in Figure 5.1.



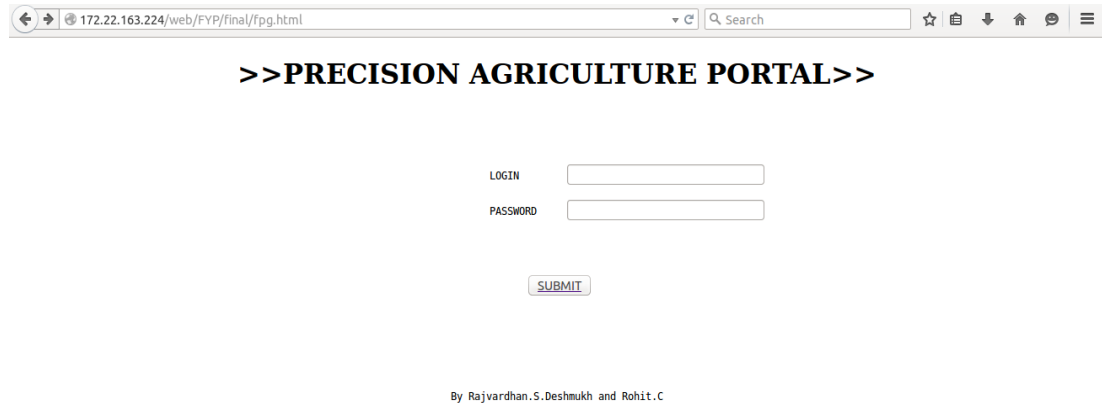
	temp	light	moisture	pH values
1	25.39	39	198	16
2	25.39	38	198	16
3	25.39	41	198	16
4	25.39	40	198	16
5	25.39	37	198	16
6	25.39	40	198	16
7	25.39	42	198	16
8	25.39	39	198	16
9	25.39	38	198	16
10	25.39	41	198	16
11	25.39	40	198	16
12	25.39	37	198	16
13	25.39	40	198	16
14	25.39	41	198	16
15	25.39	38	198	16
16	25.39	38	198	16
17	25.39	41	198	16
18	25.39	39	198	16
19	25.39	37	198	16
20	25.39	40	198	16
21	25.39	41	198	16
22	25.39	38	198	16
23	25.39	38	198	16
24	25.39	41	198	16
25	25.39	40	198	16
26	25.39	37	198	16
27	25.39	39	198	16
28	25.39	40	198	16
29	25.39	38	198	16

Figure 5.1 Values collected from sensors

5.2 Integration of Client Nodes with Webserver

The website consists of a login page, as shown in Figure 5.2, which further leads us to the option page, as shown in Figure 5.3. The values were simultaneously uploaded in the website hosted by the Pi, as shown in Figure 5.4. Individual node values received were presented in form of Graphs, as shown in Figure 5.5 and Figure 5.6. Along with displaying the values on the website, the Pi was also able to take some decisions based on

the values. For example : when the moisture level became too low, the solenoid valve was activated; and when the pH level exceeded or became less than the threshold, the amount of lime to be added to correct it was calculated and displayed on the website.



>>PRECISION AGRICULTURE PORTAL>>

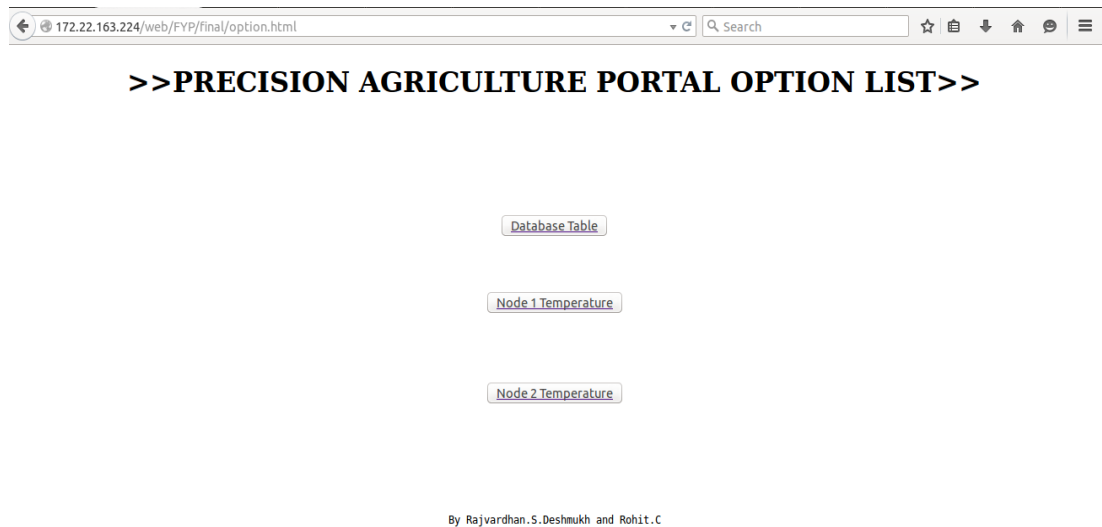
LOGIN

PASSWORD

[SUBMIT](#)

By Rajvardhan.S.Deshmukh and Rohit.C

Figure 5.2 Website login page



>>PRECISION AGRICULTURE PORTAL OPTION LIST>>

[Database Table](#)

[Node 1 Temperature](#)

[Node 2 Temperature](#)

By Rajvardhan.S.Deshmukh and Rohit.C

Figure 5.3 Website option page

Precision Agriculture										
Id	Node_no.	Time_Stamp	Crop	Temp(C)	Light_Intensity	Moisture	pH	Lime Required	Nodes down	
138	2	2016-04-21 11:41:31	Rice	24	120	210	8	R	0	
137	2	2016-04-24 11:41:07	Rice	same	0mins	same	same	same	same	0
136	1	2016-04-24 11:41:07	Rice	same	0mins	same	same	same	same	0
135	2	2016-04-24 11:41:07	Rice	24	120	210	8	R	0	
134	2	2016-04-21 17:15:20	Rice	24	120	210	8	R	0	
133	1	2016-04-21 16:22:28	Rice	24	121	Remove_water	7	0	0	0
132	2	2016-04-21 15:34:18	Rice	32	119	700	7	0	0	
131	0	2016-04-21 15:34:17	Rice	same	same	same	same	same	same	2,
130	2	2016-04-21 15:34:17	Rice	same	0mins	same	same	same	same	0
128	1	2016-04-19 05:19:30	Rice	32	119	700	7	0	0	
127	1	2016-04-19 00:00:02	Rice	25	30	866	6.5	1	0	
126	0	2016-04-19 00:00:02	Rice	same	same	same	same	same	same	2,
125	2	2016-04-19 00:00:02	Rice	same	0mins	same	same	same	same	0

Precision Agriculture										
Id	Node_no.	Time_Stamp	Crop	Temp(C)	Light_Intensity	Moisture	pH	Lime Required	Nodes down	
18	2	2016-04-18 23:59:24	Rice	20	80	600	4.5	4	0	
17	2	2016-04-18 23:59:24	Rice	20	80	600	4.5	4	0	
16	2	2016-04-18 23:59:24	Rice	20	80	600	4.5	4	0	
15	1	2016-04-18 23:58:18	Rice	25	130	600	6.5	1	0	
14	2	2016-03-22 02:08:42	Rice	20	80	600	4.5	4	0	
13	2	2016-03-22 02:08:42	Rice	20	80	600	4.5	4	0	
12	2	2016-03-22 02:08:42	Rice	20	80	600	4.5	4	0	
11	3	2016-03-22 00:57:17	Rice	25	130	240	7	0	0	
9	2	2016-03-22 00:50:59	Rice	25	130	240	7	0	0	
8	2	2016-03-22 00:38:13	Rice	25	130	240	7	0	0	
7	1	2016-03-22 00:35:05	Rice	25	130	240	7	0	0	
6	1	2016-03-22 00:32:48	Rice	20	130	Remove_water	8	R	0	
4	1	2016-03-22 00:32:48	Rice	20	130	200	8	R	0	
3	1	2016-03-21 23:04:59	Rice	24	190	300	4.5	4	0	
2	2	2016-03-21 23:04:21	Rice	24	90	600	5.5	3	0	
1	2	2016-03-21 23:04:21	Rice	24	90	600	5.5	3	0	

Figure 5.4 Website updating values received from sensors

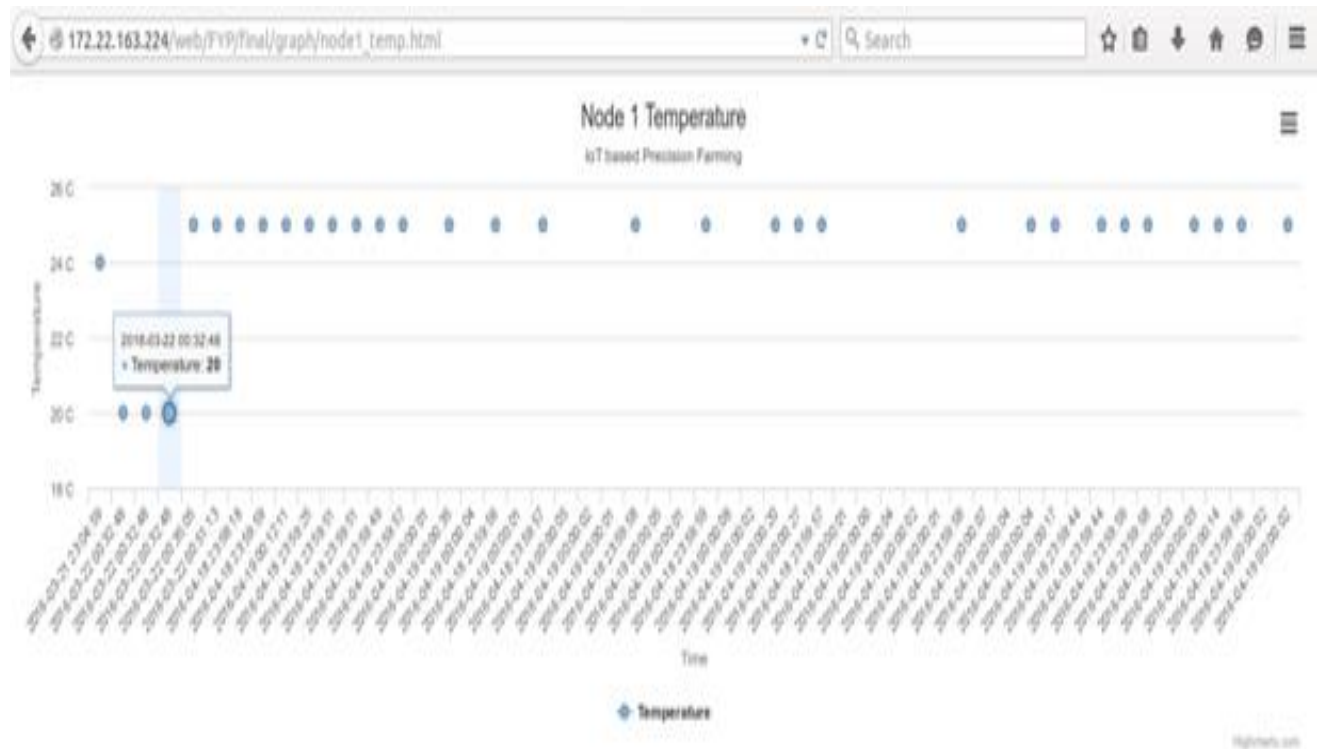


Figure 5.5 Node 1 temperature graph

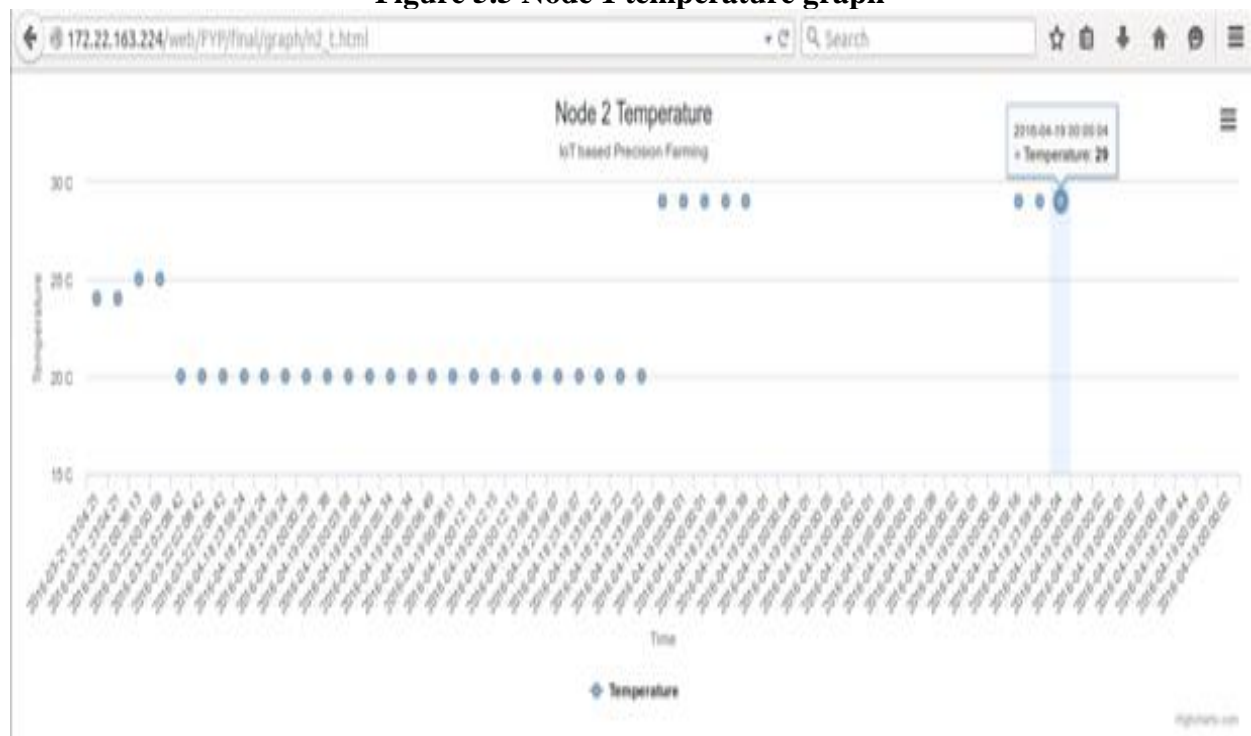


Figure 5.6 Node 2 temperature graph

CHAPTER 6

CONCLUSION AND FUTURE WORK

This chapter concludes the thesis on Precision agriculture system for rice crop and suggest additional functionalities for future implantation.

6.1 Conclusion

In this project, a method to measure and control the various parameters within a rice field using WSN was demonstrated. Parameters over vast areas (one node covering 2.7 acres) owing to the large range of Xbee module (60 m) were measured. If the soil properties could be considered to be more or less constant over this area range, then the project would also be cost effective (in terms of cost per unit area). However, soil properties may vary, and more nodes per acre maybe required to accurately monitor the field. The parameter of the field like soil moisture, temperature, pH and the light intensity were uploaded into a website hosted locally by a Raspberry Pi. The site could be accessed by a host specific username and password.

6.2 Future work

With the help of weather forecast/prediction we can control the water flow so as to get optimum results without spending excess resource (power). The circuit, when manufactured at a large scale will be more cost effective and durable.

APPENDIX

Atmega328 coding (node side)(for N1)

```
#include <SoftwareSerial.h>
#include <avr/sleep.h>
#include <avr/power.h>
#include <avr/wdt.h>

#define valve_PIN (4)
#define temp (analogRead(A1)*500)/1023
#define light analogRead(A0)//node level
#define water analogRead(A2)//node level
#define pH analogRead(A3)//computation at pi

volatile int f_wdt=1;

// Watchdog Interrupt Service. This is executed when watchdog timed out.

ISR(WDT_vect)
{
  if(f_wdt == 0)
  {
    f_wdt=1;
  }
  else
  {
    Serial.println("WDT Overrun!!!");
  }
}
// Enters the arduino into sleep mode.

void enterSleep(void)
{
  set_sleep_mode(SLEEP_MODE_PWR_DOWN); /* EDIT: could also use
SLEEP_MODE_PWR_SAVE for higher power consumption. */
  sleep_enable();

  /* Now enter sleep mode. */
  sleep_mode();

  /* The program will continue from here after the WDT timeout*/
  sleep_disable(); /* First thing to do is disable sleep. */

  /* Re-enable the peripherals. */
```

```

    power_all_enable();
}

// Setup for the serial comms and the Watch dog timeout.
void frame()
{
    Serial.print("N1;");
    Serial.print("T");
    Serial.print(temp);
    Serial.print(";L");
    Serial.print(light);
    Serial.print(";M");
    Serial.print(water);
    Serial.print(";P");
    Serial.print(pH);
    Serial.println(";");
    delay(1000);
}

void setup()
{
    Serial.begin(9600);
    pinMode(valve_PIN,OUTPUT);
    Serial.println("setup");
    delay(1000);
    digitalWrite(valve_PIN,LOW);
    /*** Setup the WDT ***/

    /* Clear the reset flag. */
    MCUSR &= ~(1<<WDRF);

    /* In order to change WDE or the prescaler, we need to
     * set WDCE (This will allow updates for 4 clock cycles).
     */
    WDTCR |= (1<<WDCE) | (1<<WDE);

    /* set new watchdog timeout prescaler value */
    WDTCR = 1<<WDP0 | 1<<WDP3; /* 8.0 seconds */

    /* Enable the WD interrupt (note no reset). */
    WDTCR |= _BV(WDIE);
}

//int i,j,k;//ifor 20mins, j for 6 hrs
int i=0;

```

```

int r;
char c[9];
char wlevel[3];
void loop()
{
  /* Don't forget to clear the flag. */
  f_wdt = 0;
  /* Re-enter sleep mode. */
  enterSleep();
  //Serial.print(i);
  //Serial.println(". 8 secs");
  //delay(1000);
  //sleep_watchdog_8sec
  if(i==1)//150
  {
    frame();
    //delay(1000);
    i=0;
    if((c[0]=Serial.read())=='N'){
      for(r=1;r<8;r++){
        c[r]=Serial.read();
      }
      if(c[1]=='N'&&c[2]=='1')
      {
        if(c[4]=='W' && c[5]!='0')
        {
          wlevel[0]=c[5];
          wlevel[1]=c[6];
          wlevel[2]=c[7];
          while(water>atoi(wlevel))
          {
            Serial.print(atoi(wlevel));
            digitalWrite(valve_PIN,HIGH);
          }
          digitalWrite(valve_PIN,LOW);
        }
      }
    }
    i++;
  }
}

```

Atmega328 coding (node side)(for N2)

```

#include <SoftwareSerial.h>
#include <avr/sleep.h>
#include <avr/power.h>

```

```

#include <avr/wdt.h>

#define valve_PIN (4)
#define temp (analogRead(A1)*500)/1023
#define light analogRead(A0)//node level
#define water analogRead(A2)//node level
#define pH analogRead(A3)//computation at pi

volatile int f_wdt=1;

// Watchdog Interrupt Service. This is executed when watchdog timed out.

ISR(WDT_vect)
{
  if(f_wdt == 0)
  {
    f_wdt=1;
  }
  else
  {
    Serial.println("WDT Overrun!!!");
  }
}
// Enters the arduino into sleep mode.

void enterSleep(void)
{
  set_sleep_mode(SLEEP_MODE_PWR_DOWN); /* EDIT: could also use
SLEEP_MODE_PWR_SAVE for higher power consumption. */
  sleep_enable();

  /* Now enter sleep mode. */
  sleep_mode();

  /* The program will continue from here after the WDT timeout*/
  sleep_disable(); /* First thing to do is disable sleep. */

  /* Re-enable the peripherals. */
  power_all_enable();
}

// Setup for the serial comms and the Watch dog timeout.
void frame()
{
  Serial.print("N2;");
  Serial.print("T");

```

```

    Serial.print(temp);
    Serial.print(";L");
    Serial.print(light);
    Serial.print(";M");
    Serial.print(water);
    Serial.print(";P");
    Serial.print(pH);
    Serial.println(";");
    delay(1000);
}

void setup()
{
    Serial.begin(9600);
    pinMode(valve_PIN,OUTPUT);
    Serial.println("setup");
    delay(1000);
    digitalWrite(valve_PIN,LOW);
    /*** Setup the WDT ***/

    /* Clear the reset flag. */
    MCUSR &= ~(1<<WDRF);

    /* In order to change WDE or the prescaler, we need to
     * set WDCE (This will allow updates for 4 clock cycles).
     */
    WDTCR |= (1<<WDCE) | (1<<WDE);

    /* set new watchdog timeout prescaler value */
    WDTCR = 1<<WDP0 | 1<<WDP3; /* 8.0 seconds */

    /* Enable the WD interrupt (note no reset). */
    WDTCR |= _BV(WDIE);
}

//int i,j,k;//ifor 20mins, j for 6 hrs
int i=0;
int r;
char c[9];
char wlevel[3];
void loop()
{
    /* Don't forget to clear the flag. */
    f_wdt = 0;
    /* Re-enter sleep mode. */

```



```

    enterSleep();
//Serial.print(i);
//Serial.println(". 8 secs");
//delay(1000);
//sleep_watchdog_8sec
if(i==1)//150
{
    frame();
    //delay(1000);
    i=0;
    if((c[0]=Serial.read())=='N'){
        for(r=1;r<8;r++){
            c[r]=Serial.read();
        }
        if(c[1]=='N'&&c[2]=='2')
        {
            if(c[4]=='W' && c[5]!='0')
            {
                wlevel[0]=c[5];
                wlevel[1]=c[6];
                wlevel[2]=c[7];
                while(water>atoi(wlevel))
                {
                    Serial.print(atoi(wlevel));
                    digitalWrite(valve_PIN,HIGH);
                }
                digitalWrite(valve_PIN,LOW);
            }
        }
    }
    i++;
}

```

Raspberry Pi coding(server side)(python)

```

import MySQLdb
import serial
import sqlite3
import datetime
import time

d=datetime.datetime.date(datetime.datetime.now())# in main
se=2#se-1 no.of nodes
db = MySQLdb.connect(host="localhost", user="root", passwd="raspberr",
db="Sensor_Statistics") #change the database

```

```
cur = db.cursor()
```

```
ser = serial.Serial('/dev/ttyAMA0', 9600)
```

```
string = 'Hello from Raspberry Pi'
```

```
sp = ';'

```

```
print 'Sending "%s"' % string
```

```
ser.write('%s\n' % string)
```

```
o=0
```

```
f=0
```

```
n1=0
```

```
c1=0
```

```
t1=0
```

```
l1=0
```

```
m1=0
```

```
p1=0
```

```
w1=0
```

```
s=0
```

```
n=[]
```

```
c=[]
```

```
t=[]
```

```
l=[]
```

```
m=[]
```

```
p=[]
```

```
w=[]
```

```
a=[0]
```

```
def light():
```

```
    global n,t,l,m,p,w,d,se
```

```
    conn = sqlite3.connect('test.db')
```

```
    #print "Opened database successfully";
```

```
    i=1#node no. counter
```

```
    j=0#
```

```
    n_d=[]
```

```
    global s
```

```
    cursor = conn.execute("SELECT Id,Node_no,Temp,light_Intensity,moisture,pH,lime_req  
from Dlogs")
```

```
    conn.commit()
```

```
    e=datetime.datetime.date(datetime.datetime.now())
```

```
    if d!=e:
```

```
        while i<=se:
```

```
            for row in cursor:
```

```
                if i==int(row[1]):
```

```
                    print "node no. = ", i, "\n"
```

```
                    j=j+1
```

```
                    print "j:",j
```

```

        if int(row[3])>=120:
            s=s+1
            print 'inner',i, ' '
    if j==0:
        coma=str(i)+','
        n_d.append(coma)
        n_d+=[]
    j=0
    if s<=18:
        sun= j*20
        sui=str(sun)+'mins'
        print 'hrs',sui
        cur.execute("INSERT INTO Sensors1(Node_no,light_Intensity) VALUES
('%s','%s')" % (i,sui))
        db.commit()
        s=0
        cursor = conn.execute("SELECT
Id,Node_no,Temp,light_Intensity,moisture,pH,lime_req from Dlogs")
        conn.commit()
        i=i+1#while se ;
    if n_d:
        n_d="".join(n_d)
        print 'nodes:',n_d,'down '
        n_down= "INSERT INTO Sensors1(nodes_down) VALUES('%s')" % (n_d)
        cur.execute(n_down)
        db.commit()
        del n_d

# conn.execute("PRAGMA busy_timeout =15000")
# conn.commit()
conn.close()
conn = sqlite3.connect('test.db')
conn.execute("DROP TABLE Dlogs")
conn.commit()
conn.close()
time.sleep(1/1000)
conn = sqlite3.connect('test.db')
conn.execute("""CREATE TABLE Dlogs
(Id          INTEGER PRIMARY KEY AUTOINCREMENT,
Node_no     VARCHAR(10) NOT NULL,
Temp        VARCHAR(10) NOT NULL,
light_Intensity VARCHAR(10) NOT NULL,
moisture     VARCHAR(10) NOT NULL,
pH           VARCHAR(10) NOT NULL,
lime_req     VARCHAR(10) NOT NULL);""")

```

```

    conn.commit()
# conn.close()
    d=e#ife d ;
    elif e==d:
    # conn = sqlite3.connect('test.db')
    conn.execute("INSERT INTO Dlogs
(Node_no,Temp,light_Intensity,moisture,pH,lime_req)
VALUES('%s','%s','%s','%s','%s','%s')" % (n,t,l,m,p,w))
    conn.commit()
# conn.close()
return

def sq():
    global n,c,t,l,m,p,w
    sql= "INSERT INTO Sensors1(Node_no,Crop,Temp,light_Intensity, moisture,
pH, lime_req) VALUES ('%s','%s','%s','%s','%s','%s','%s')" % (n,c,t,l,m,p,w)
    cur.execute(sql)
    db.commit()
    return

def phr():
    global p,w
    if int(p)>=23 and int(p)<=30:
        p=4.5
    elif int(p)>18 and int(p)<23:
        p=5
    elif int(p)<=18 and int(p)>=14:
        p=5.5
    elif int(p)>=9 and int(p)<=13:
        p=6
    elif int(p)<=8 and int(p)>=5:
        p=6.5
    elif int(p)==4 or int(p)==3:
        p=7
    elif int(p)==2 or int(p)==1:
        p=7.5
    elif int(p)==0:
        p=8
    else: p=0

    if float(p)==4.5:
        w=4
    elif float(p)==5:
        w=3.5
    elif float(p)==5.5:
        w=3

```

```

elif float(p)==6:
    w=2
elif float(p)==6.5:
    w=1
elif float(p)==7 or float(p)==7.5:
    w=0
else:
    w='R'

if float(p)!=6.5 and int(p)!=7 and int(p)!=7.5:
    sq()
return

def rice():
    global t,l,m
    phr()
    light()
    if int(t)< 24 or int(t)> 36:
        sq()

    if int(m)>500:
        pac=';N'+n+';W500;'
        ser.write(pac)
        sq()
    else:
        ser.write(';W0')
    if int(m)<210:
        m="Remove_water"
        sq()
    return

def crop():
    global c
    if c=='Rice':
        rice()
    return

while True:
    incoming = ser.read().strip()
    print '%s' % incoming
    if incoming!=sp:
        a+=[]
        if a[0]==0:
            a[0]=incoming

```

```

else:
    a.append(incoming)
print 'a= %s' %a
o+=1
else:
    print '%s' % a
    if a[0]=='N':
        if n1>0:
            print 'yo'
            del n
        n=[]
        n1=0
        for i in range(1,o):
            n.append(a[i])
            n+=[ ]
        del a
        a=[0]
        o=0
        n1+=1
        n="".join(n)
        print 'Node_no:%s ' % n
        #sql= "INSERT INTO Sensors1(Node_no) VALUES ('%s')" % n
    elif a[0]=='C':
        if c1>0:
            del c
            c=[]
            c1=0
            for i in range(1,o):
                c.append(a[i])
                c+=[ ]
            del a
            a=[0]
            o=0
            c1+=1
            c="".join(c)
            print 'crop:%s ' % c#sql= "INSERT INTO Sensors1(Temp) VALUES ('%s')" %
t
    elif a[0]=='T':
        if t1>0:
            del t
            t=[]
            t1=0
            for i in range(1,o):
                t.append(a[i])
                t+=[ ]
            del a

```

```

        a=[0]
        o=0
        t1+=1
        t="".join(t)
        print 'Temperature:%s ' % t#sql= "INSERT INTO Sensors(Temp) VALUES
('%s') " % t
    elif a[0]=='L':
        if l1>0:
            del l
            l=[]
            l1=0
            for i in range(1,o):
                l.append(a[i])
                l+=[]
            del a
            a=[0]
            o=0
            l1+=1
            l="".join(l)
            print 'Light_intensity:%s' % l#sql= "INSERT INTO Sensors(light_Intensity)
VALUES ('%s') " % l
        elif a[0]=='M':
            if m1>0:
                del m
                m=[]
                m1=0
                for i in range(1,o):
                    m.append(a[i])
                    m+=[]
                del a
                a=[0]
                o=0
                m1+=1
                m="".join(m)
                print 'Moisture:%s' % m#sql= "INSERT INTO Sensors(moisture) VALUES
('%s') " % m
            elif a[0]=='P':
                if p1!=0:
                    del p
                    p=[]
                    p1=0
                    for i in range(1,o):
                        p.append(a[i])
                    p1+=1
                    del a
                    a=[0]

```

```

o=0
p="".join(p)
print 'pH:%s' %p# sql= "INSERT INTO Sensors(pH) VALUES ('%s')" % p
else:
    print 'error:%s' % a
    del a
    a=[0]
    o=0
f=n1+c1+t1+l1+m1+p1
if f==6:
    crop()
    del n,c,t,l,m,p,w
    n=[]
    c=[]
    t=[]
    l=[]
    m=[]
    p=[]
    w=[]
    o=0
    f=0
    n1=0
    c1=0
    t1=0
    l1=0
    m1=0
    p1=0

```

```
#
```

Raspberry Pi coding(server side)(Login page)

```

<html>
<center><H1>>>PRECISION AGRICULTURE PORTAL>><H1></center>
<form>
<pre>

<center>
                                LOGIN      <input type="text" name="lname"><BR>
                                PASSWORD    <input type="text" name="lname"><BR>
</center>

<center>
    <a href="option.html"><input type="button" value="SUBMIT" ></a>
</center>

```



```
<center>
By Rajvardhan.S.Deshmukh and Rohit.C
</center>
```

```
</pre>
</form>
```

```
</html>
```

Raspberry Pi code (server side)(option page)

```
<html>
<center><H1>>>PRECISION AGRICULTURE PORTAL OPTION LIST>><H1></center>
<form>
<pre>
```

```
<center>
```

```
<a href="farm.php"><input type="button" value="Database Table" ></a>
```

```
<a href="graph/node1_temp.html"><input type="button" value="Node 1
Temperature" ></a>
```

```
<a href="graph/n2_t.html"><input type="button" value="Node 2
Temperature" ></a>
```

```
</center>
```

```
<center>
By Rajvardhan.S.Deshmukh and Rohit.C
</center>
```

```
</pre>
</form>
```

```
</html>
```

Raspberry Pi code (server side)(Dynamic Content page)

[illegible]

Raspberry Pi code (server side)(Database connection)

```
<?php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = 'raspberry';

$db = 'Sensor_Statistics';
$conn = mysql_connect($dbhost,$dbuser,$dbpass);
mysql_select_db($db);
?>
```

Raspberry Pi graph coding (node1_temp.js)(for drawing graph)

```
$(function() {  
    var x values = [];
```

```

var y_values = [];
var switch1 = true;
$.get('graph_temp.php', function(data) {

    data = data.split('/');
    for (var i in data)
    {
        if (switch1 == true)
        {
            var ts = data[i];
            x_values.push(ts);
            switch1 = false;
        }
        else
        {
            y_values.push(parseFloat(data[i]));
            switch1 = true;
        }
    }
    x_values.pop();

    $('#chart').highcharts({
        chart : {
            type : 'spline'
        },
        title : {
            text : 'Node 1 Temperature'
        },
        subtitle : {
            text : 'IoT based Precision Farming'
        },
        xAxis : {
            title : {
                text : 'Time'
            },
            categories : x_values
        },
        yAxis : {
            title : {
                text : 'Temperature'
            },
            labels : {
                formatter : function() {
                    return this.value + ' C'
                }
            }
        },
        tooltip : {
            crosshairs : true,
            shared : true,
            valueSuffix : ''
        },
        plotOptions : {
            spline : {
                marker : {

```

```

        radius : 4,
        lineColor : '#666666',
        lineWidth : 1
    }
},
series : [{
    name : 'Temperature',
    data : y_values
}]
});
});
});

```

Raspberry Pi graph program (graph_temp.php)(for connecting to database)

```

<?php
$con = mysql_connect("localhost","root","raspberrypi");
if (!$con) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("Sensor_Statistics", $con);
$result = mysql_query("SELECT * FROM `Sensors1` WHERE Node_no=1 ") or
die ("Connection error");
while($row = mysql_fetch_array($result)) {
    echo $row['time_stamp'] . "/" . $row['Temp'] . "/" ;
}
mysql_close($con);
?>

```

Raspberry Pi graph program (node1_temp.html)(for providing framework of the website)

```

<html>
<head>
<title>Node 1 Temperature</title>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"
type="text/javascript"></script>
<script src="http://code.highcharts.com/highcharts.js"></script>
<script src="http://code.highcharts.com/modules/exporting.js"></script>
<script type="text/javascript" src="node1_temp.js" ></script>
</head>
<body>
<div id="chart" style="height: 400px; margin: 0 auto"></div>
</body>
</html>

```

REFERENCES

- [1] http://www.onlysupport.co.in/index.php?main_page=product_info&cPath=13_49&products_id=404(soil moisture sensor)

- [2] <http://www.ti.com/lit/ds/symlink/lm35.pdf>(lm 35)

- [3] <https://www.hydrofarm.com/p/MGMLP1>(active air three way meter)

- [4] http://www.biltek.tubitak.gov.tr/gelisim/elektronik/dosyalar/40/LDR_NSL19_M51.pdf(light detecting resistor)

- [5] Mahmoud Shaker, Ala'a Imran, "Greenhouse Micro Climate Monitoring Based On WSN with Smart Irrigation Technique" International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering Vol:7, No:12, 2013.

- [6] "Climate and soil for rice cultivation" , agricultureandupdates.blogspot.in, October 12,2011

- [7] Spaho, E.; Barolli, L.; Mino, G.; Xhafa, F.; Kolici, V., "Goodput Evaluation of AODV, OLSR and DYMO Protocols for Vehicular Networks Using CAVENET," Network-Based Information Systems (NBIS), 2011 14th International Conference on , vol., no., pp.118,125, 7-9 Sept. 2011

- [8] Rajvardhan Somraj Deshmukh,Tushar Singh Chouhan,P.Vetrivelan, "VANETS Model: Vehicle-to-Vehicle,Infrastructure-to-Infrastructure," *International Journal of Current Engineering and Technology* , vol. 5, no. June 2015, p. 2347 – 5161, 2015.

- [9] <https://www.elprocus.com/what-is-zigbee-technology-architecture-and-its-applications/>

- [10] K. L. Sahrawat," Fertility and organic matter in submerged rice soils" CURRENT SCIENCE, VOL. 88, NO. 5, 10 MARCH 2005

- [11] Smriti Chand" Cultivation of Rice: Suitable Conditions Required for the Cultivation of Rice (6 Conditions)",yourarticlelibrary.com

- [12] Dr. Cliff Snyder, "Soil pH management",Efficient Fertilizer Use Manual(www.plantstress.com/articles/toxicity_m/soilph%20amend.pdf)

LIST OF PUBLICATIONS

International Journals

- 1) Rajvardhan Somraj Deshmukh, Tushar Singh Chouhan, P. Vetrivelan, "VANETS Model: Vehicle-to-Vehicle, Infrastructure-to-Infrastructure," *International Journal of Current Engineering and Technology*, vol. 5, no. June 2015, p. 2347 – 5161, 2015.
- 2) Rajvardhan Somraj Deshmukh, Rohith.C, Dr. Vetrivelan P, "IoT BASED PRECISION AGRICULTURE SYSTEM FOR RICE CROP", ARPN Journal (April 2016 Communicated).