CROP MONITORING SYSTEM BASED ON WIRELESS SENSOR NETWORK USING ESP32 AND LINUX BASED MQTT SERVER

S.Jyothirmaye[1] , Amirineni Rama L. Padmaja[2]
Associate Professor[1] , Assistant Professor[2]
Geethanjali College of Engineering and Technology, Hyderabad, India

ABSTRACT

This paper presents one of the method for monitoring and control over the Green-House is done by measuring critical parameters that effect the plant growth such as temperature, humidity, luminosity, soil Moisture, pH levels. In this paper, we use ESP32 to collect the data provided by sensors and perform necessary actions depending upon the data collected. The main node is the raspberry pi which is connected to the internet and the data captured by the individual ESP32's are transmitted via the wireless sensor network. This network is formed using the ESP32's by making them act as a node in the network to transmit information to the main node.

Typically, the plants in the green house are grown in a particular fashion. The most used patterns include array pattern, block pattern, step pattern etc. We locate a particular plant by using any appropriate indoor positioning mechanism such as magnetic positioning, Wi-Fi positioning, Bluetooth signal positioning, beaconing etc. Since we are using ESP32 we can use the Wi-Fi signals or Bluetooth signals emitted to represent a particular block or array. If any further advances are made to automate a task that requires a plant position in the greenhouse, then this positioning system can be used. For example, if we want to capture the image data using a line follower robot then certainly this indoor positioning mechanism is helpful.

Keywords: Greenhouse, Internet of Things, ESP32, Raspberry Pi, Indoor positioning system, Wireless Sensor Network.

1. INTRODUCTION

The concept of Internet of Things (IoT) is based on computing the data that is being exchanged by everyday objects that are connected to internet and being able to connect themselves to other devices to transfer and receive data. Considerable number of innovations has been done during the latest decade in the field of agriculture. Be it home, industry, agriculture we see automation engaged in it. Greenhouse is the methodological approach in which farmers in the rural areas will be profited by programmed monitoring & control of greenhouse environment that can potentially replace the direct supervision of the human. This paper concentrates on the specific usage of the **ESP32**'s running on **Mongoose OS** and a Raspberry Pi hosting a **MQTT Server** and this Raspberry Pi is running on **Raspbian OS** which is a flavour of Linux designed by the Raspberry Pi foundation in association with Broadcomm. The ESP32 will collect the data from the sensors, processes it and performs necessary actions depending upon the situation it faces. These actions can be automated by the ESP32 itself or the user can choose to manually override the ESP32 and perform his/her own actions. The system will provide the optimal prompts to the user to perform but the final decision is left to the user.

2. DESIGN AND IMPLEMENTATION

2.1 BLOCK DIAGRAM OF A SINGLE NODE IN THE NETWORK

The below figure 2.1.1 is the block diagram represents a single node in the Wireless Sensor Network comprised of ESP32s. The rest of the nodes comprise the same structure which is placed in the other blocks or arrays in the green house.

- Respective Sensors mentioned in the above block diagram are connected to ESP32 and the micro controller unit continues to monitor the values for a particular interval of time. Here the interval chosen is of 15 seconds (because for accurate readings, averaging of the values from the sensors is done in the code in order to minimize the errors).
- The ESP32 that is being used here runs on Mongoose OS which is a lot efficient compared to the default Real Time OS that comes out of the box with the ESP32. The main advantage here is the power consumption and execution speed. Mongoose OS has a much optimised kernel compared to RTOS. So, the management utilities perform efficiently thus the ESP32 consumes very less power and performs calculations faster.

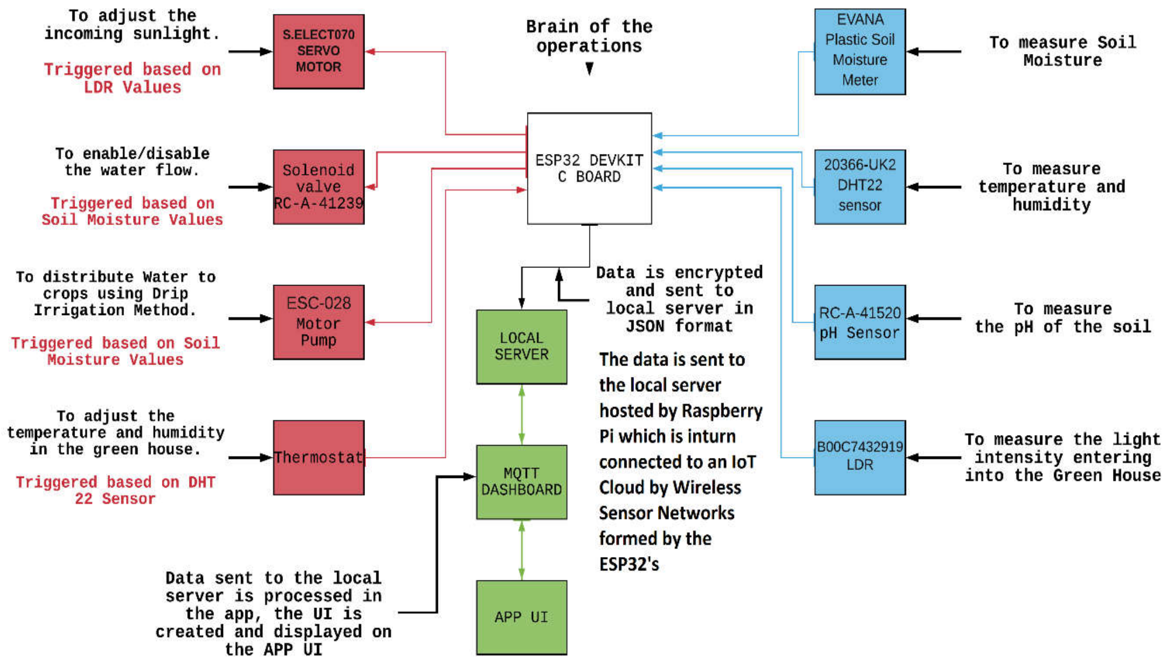

To adjust the incoming sunlight.
Triggered based on LDR Values
S.ELECT070 SERVO MOTOR
To enable/disable the water flow.
Triggered based on Soil Moisture Values
Solenoid valve RC-A-41239
To distribute Water to crops using Drip Irrigation Method.
Triggered based on Soil Moisture Values
ESC-028 Motor Pump
To adjust the temperature and humidity in the green house.
Triggered based on DHT 22 Sensor
Thermostat
Brain of the operations
ESP32 DEVKIT C BOARD
Data is encrypted and sent to local server in JSON format
LOCAL SERVER
MQTT DASHBOARD
APP UI
Data sent to the local server is processed in the app, the UI is created and displayed on the APP UI
The data is sent to the local server hosted by Raspberry Pi which is inturn connected to an IoT Cloud by Wireless Sensor Networks formed by the ESP32's
EVANA Plastic Soil Moisture Meter
To measure Soil Moisture
20366-UK2 DHT22 sensor
To measure temperature and humidity
RC-A-41520 pH Sensor
To measure the pH of the soil
B00C7432919 LDR
To measure the light intensity entering into the Green House

Figure 2.2.1 Block Diagram of Single Node in the Greenhouse

2.2 OVERVIEW OF THE WIRELESS SENSOR NETWORK

- Each block has an ESP32 node which has the configuration as shown in the Figure I above. For the sake of demonstration consider the above layout with 7 nodes. Each node is represented by an alphabet in the above diagram A, B, C, D, E and F respectively and every node is interconnected and it is possible to communicate with each node by making the transmitter as slave node and the receiver as the master node or vice versa depending upon our requirement and there is one master node that is the raspberry pi, it is connected to the internet and this information can be sent to any cloud or can be hosted locally. The general case here is for every 15 seconds the data is collected from sensors by individual node in its respective block.
  - Then all of the ESP32s will enter into transmission mode. Each node transmits in a round robin fashion to the master node by transmitting to the adjacent node by using an optimal path.
  - The optimal path that is the nearest path to the master node is determined by using the Dijkstra's shortest path algorithm.

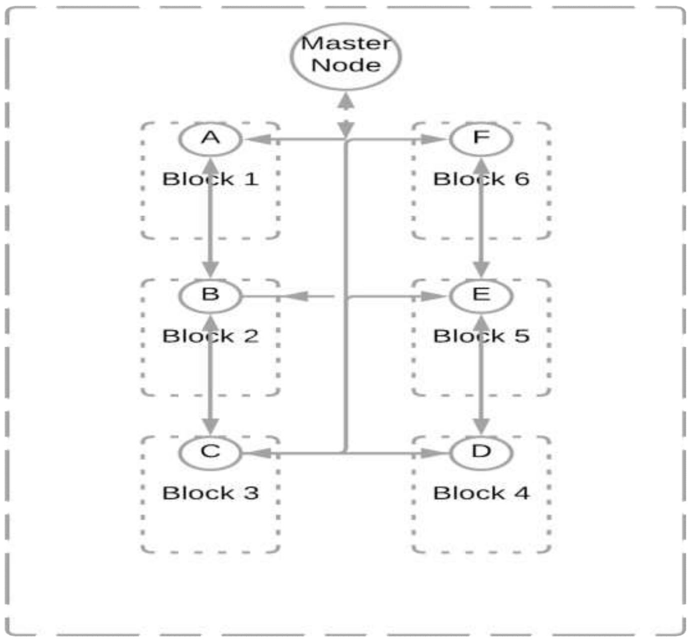


Figure 2.2.1 Overview of the WSN

2.3 MQTT PROTOCOL

The protocol used in-order to transmit and receive data is MQTT. MQTT stands for Message Queuing Telemetry Transport it is an open OASIS and ISO Standard (ISO/EIC 20922) lightweight, publish-subscribe network protocol that transports messages between devices. MQTT runs over TCP/IP. Nevertheless, any network protocol that provides ordered, lossless, bi-directional connections can support MQTT. MQTT is ideal for connections that require less bandwidth and a need for smaller code footprint. The library that has been used in implementing this paper is the mosquitto library. It is a very stable library provided by Eclipse. Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers. The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers. The Mosquitto project also provides a C library for implementing MQTT clients, and the very popular mosquitto_pub and mosquitto_pub command line MQTT clients. The eclipse software is installed in the Raspberry Pi and initiated to listen to on a particular port typically 1883 for incoming and outgoing connections.

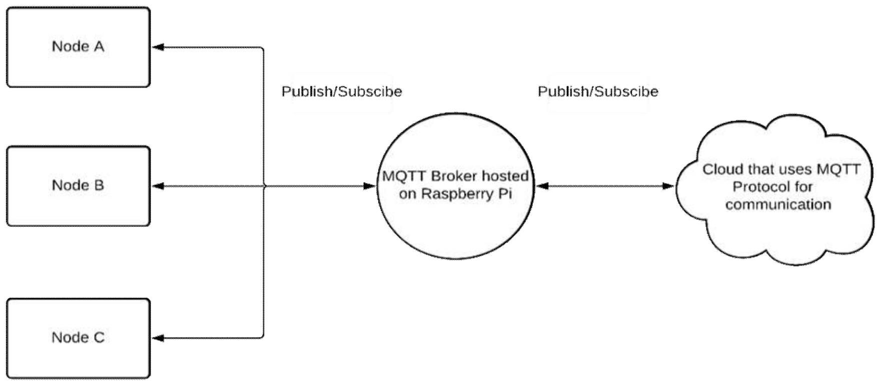

Node A
Node B
Node C
Publish/Subscibe
Publish/Subscibe
MQTT Broker hosted on Raspberry Pi
Cloud that uses MQTT Protocol for communication

Figure 2.3.1 Communication Layout using MQTT Protocol

3. OVERVIEW OF ESP32 AND RASPBERRY PI

3.1 . ESP32

- **Processors**:
  - **Main Processor**: Tensilica Xtensa 32-bit LX6 microprocessor
    - Cores: 1
    - Clock Frequency: 240MHz
    - Performance: 600DMIPS
  - **Ultra-Low Power Co-Processor**: allows ADC conversions, computation, and level thresholds while in deep sleep.
- **Wireless Connectivity:**
  - Wi-Fi: 802.11 b/g/n/e/i (802.11n @ 2.4 GHz up to 150 Mbit/s)
  - Bluetooth: v4.2 BR/EDR and Bluetooth Low Energy (BLE)
- **Memory:**
  - **Internal Memory:**
    - ROM: 448KiB (for booting and core functions)
    - SRAM: 520KiB (for data and instruction)
    - RTC fast SRAM: 8KiB (for data storage and main CPU during RTC Boot from the deep sleep mode)
    - eFuse: 1KiBit (Of which 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including Flash-Encryption and Chip-ID.)
- **Embedded Flash:**
  - 0 MiB (ESP32-D0WDQ6, ESP32-D0WD, and ESP32-S0WD chips)
  - 2 MiB (ESP32-D2WD chip)
  - 4 MiB (ESP32-PICO-D4 SiP module)
- **External flash & SRAM:** ESP32 supports up to four 16 MiB external QSPI flashes and SRAMs with hardware encryption based on AES to protect developers' programs and data. ESP32 can access the external QSPI flash and SRAM through high-speed caches.
  - Up to 16 MiB of external flash are memory-mapped onto the CPU code space, supporting 8-bit, 16-bit and 32-bit access. Code execution is supported.

- Up to 8 MiB of external flash/SRAM memory are mapped onto the CPU data space, supporting 8-bit, 16-bit and 32-bit access. Data-read is supported on the flash and SRAM. Data-write is supported on the SRAM.

- **Peripheral input/output:** Rich peripheral interface with DMA that includes capacitive touch, ADCs (analog-to-digital converter), DACs (digital-to-analog converter), I²C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), I²S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulse width modulation), and more.
- **Security:**
  - IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
  - Secure boot
  - Flash encryption
  - 1024-bit OTP, up to 768-bit for customers
  - Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)

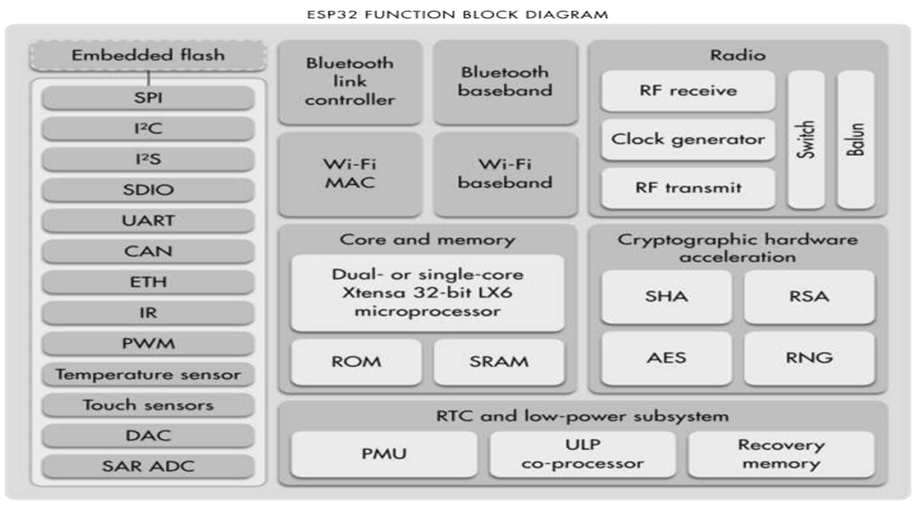


Figure 3.1.1 Functional Block Diagram of ESP32

3.2. RASPBERRY PI 4

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 2 × micro-HDMI ports (up to 4kp60 supported)
- 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A*)
- 5V DC via GPIO header (minimum 3A*)
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)
- Operating temperature: 0 – 50 degrees C ambient

Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom.The Raspberry Pi 4 uses a Broadcom BCM2711 SoC with a 1.5 GHz 64-bit quad-core ARM Cortex-A72 processor, with 1 MiB shared L2 cache.

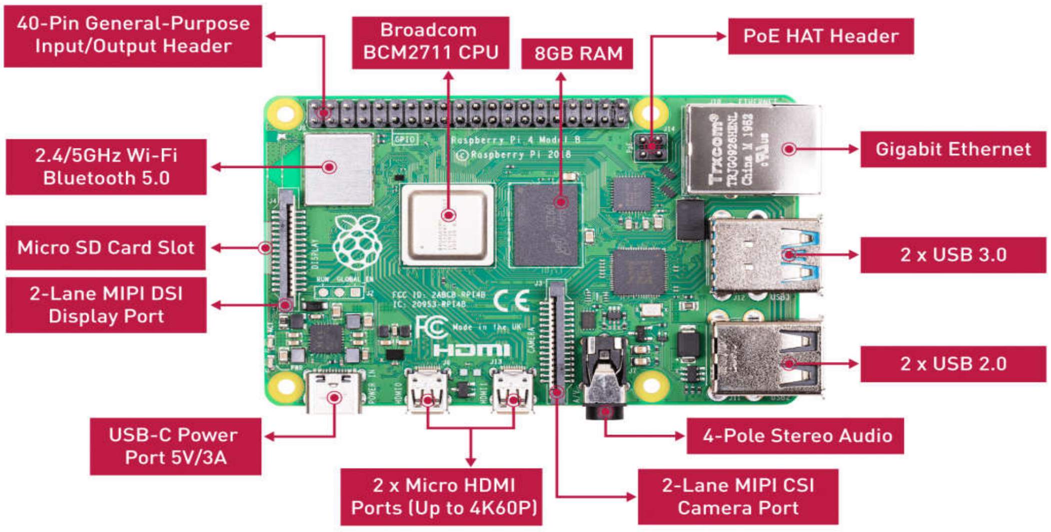
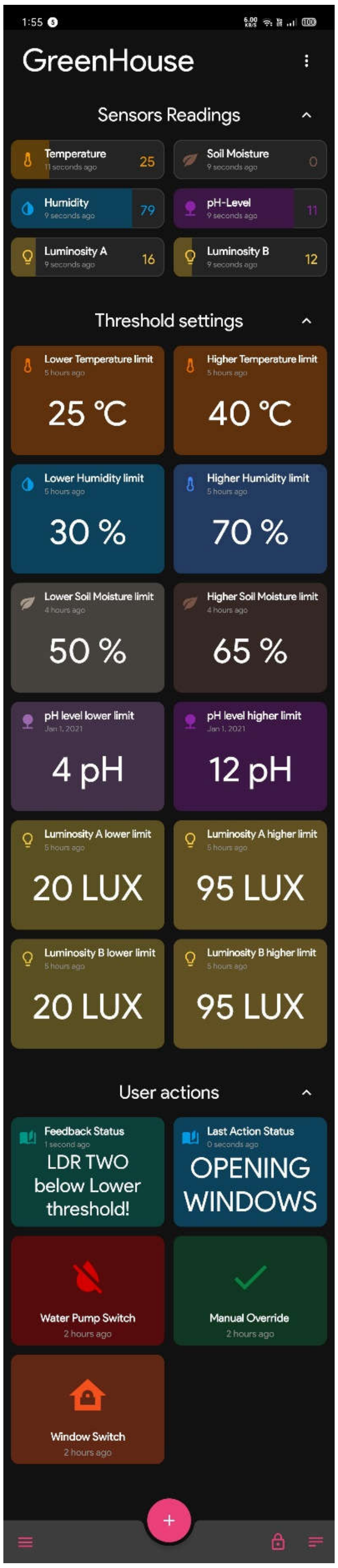

40-Pin General-Purpose Input/Output Header
Broadcom BCM2711 CPU
8GB RAM
PoE HAT Header
2.4/5GHz Wi-Fi Bluetooth 5.0
Gigabit Ethernet
Micro SD Card Slot
2 x USB 3.0
2-Lane MIPI DSI Display Port
2 x USB 2.0
USB-C Power Port 5V/3A
4-Pole Stereo Audio
2 x Micro HDMI Ports (Up to 4K60P)
2-Lane MIPI CSI Camera Port

Figure 3.2.1. Specification Diagram of Raspberry Pi 4

4 RESULT

4.1 APP LAYOUT OF SINGLE NODE

- The adjacent image is the App layout for a singular node in Green House.
- Each Node can be controlled independently and specific thresholds can be set for each parameter.
- The first segment of the App is the Sensors Readings. Here the data sensed from the sensors is sent via the MQTT broker.
- In the second segment of the App is the threshold settings. Here the user can set individual threshold limits for each parameter sensed by the sensor.
- The user can choose whether the system can perform actions automatically or just prompt the user with optimal action.
- In the third segment of the App is the User Actions. Here the user can enable Manual Override.
- This means the user can perform against the action suggested by the ESP32 or proceed with its decision.
- If the Manual Override options is turned off, ESP32 will perform the best action possible to neutralize the situation.
- Feedback Status displays the status of the control operations that are being done by ESP32.



1:55

GreenHouse

Sensors Readings

Temperature (11 seconds ago)	25	Soil Moisture (9 seconds ago)	0
Humidity (9 seconds ago)	79	pH-Level (9 seconds ago)	11
Luminosity A (9 seconds ago)	16	Luminosity B (9 seconds ago)	12

Threshold settings

Lower Temperature limit (5 hours ago)	25 °C	Higher Temperature limit (5 hours ago)	40 °C
Lower Humidity limit (5 hours ago)	30 %	Higher Humidity limit (5 hours ago)	70 %
Lower Soil Moisture limit (4 hours ago)	50 %	Higher Soil Moisture limit (4 hours ago)	65 %
pH level lower limit (Jan 1, 2021)	4 pH	pH level higher limit (Jan 1, 2021)	12 pH
Luminosity A lower limit (5 hours ago)	20 LUX	Luminosity A higher limit (5 hours ago)	95 LUX
Luminosity B lower limit (5 hours ago)	20 LUX	Luminosity B higher limit (5 hours ago)	95 LUX

User actions

Feedback Status (1 second ago): LDR TWO below Lower threshold!

Last Action Status (0 seconds ago): OPENING WINDOWS

Figure 4.1.1 UI of Singular Node in Green House

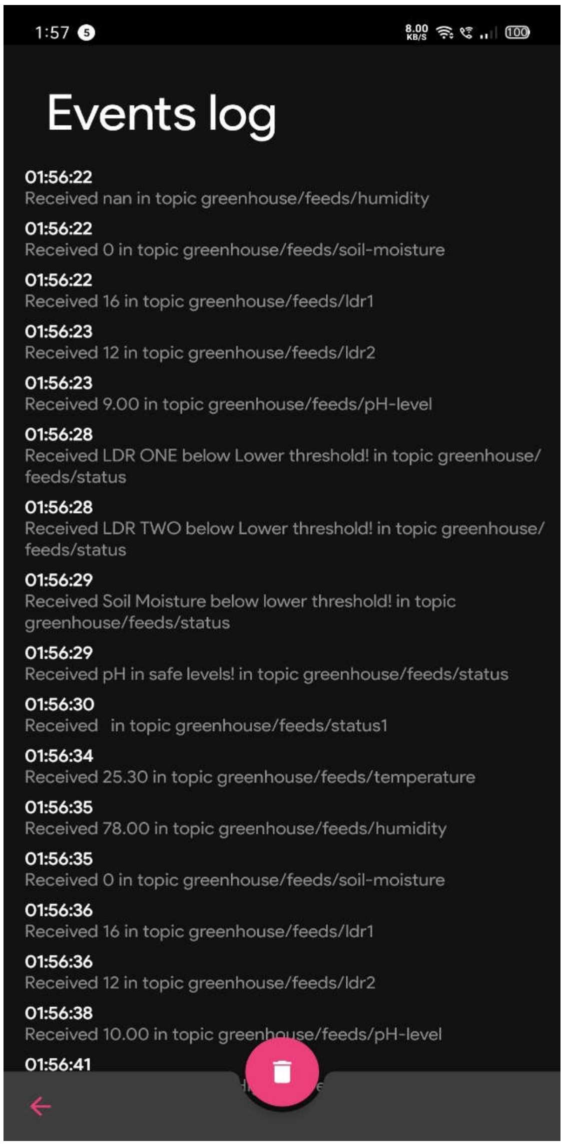
4.2 APP ACTION LOGGER

The adjacent image is the Events Log.

- As the name suggests it logs every action that is being performed by the system and what data it is receiving and transmitting to other nodes.
- The log is stored in the user’s phone and can be accessed anytime.
- This log hepls to troubleshoot if anything wrong happens.
- For Example look at the first log it says “Received nan(not a number) in topic greenhouse/feeds/humidity”
- By this we can know that the humidity and temperature sensor i.e DHT22 is malfunctioning.
- So, the user can quickly go to that block and verify what is wrong with that sensor.

4.3 APP BROKER

- The adjacent image is where we can edit our broker settings.
- The setup above is locally hosted on the Raspeberry Pi 4
- We have the flexibility to connect to any IoT cloud that support MQTT protocol such as adafruit.io, Google IoT core, AWS Cloud, IBM Cloud etc.



```
01:56:22
Received nan in topic greenhouse/feeds/humidity
01:56:22
Received 0 in topic greenhouse/feeds/soil-moisture
01:56:22
Received 16 in topic greenhouse/feeds/ldr1
01:56:23
Received 12 in topic greenhouse/feeds/ldr2
01:56:23
Received 9.00 in topic greenhouse/feeds/pH-level
01:56:28
Received LDR ONE below Lower threshold! in topic greenhouse/
feeds/status
01:56:28
Received LDR TWO below Lower threshold! in topic greenhouse/
feeds/status
01:56:29
Received Soil Moisture below lower threshold! in topic
greenhouse/feeds/status
01:56:29
Received pH in safe levels! in topic greenhouse/feeds/status
01:56:30
Received  in topic greenhouse/feeds/status1
01:56:34
Received 25.30 in topic greenhouse/feeds/temperature
01:56:35
Received 78.00 in topic greenhouse/feeds/humidity
01:56:35
Received 0 in topic greenhouse/feeds/soil-moisture
01:56:36
Received 16 in topic greenhouse/feeds/ldr1
01:56:36
Received 12 in topic greenhouse/feeds/ldr2
01:56:38
Received 10.00 in topic greenhouse/feeds/pH-level
01:56:41
```

Figure 4.3.1 App Events Log

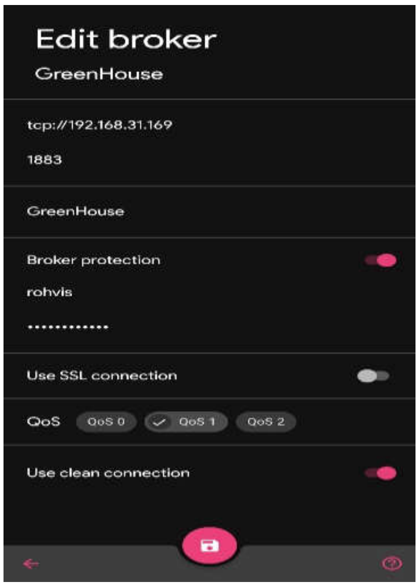

Edit broker
GreenHouse
tcp://192.168.31.169
1883
GreenHouse
Broker protection: on
rohvis
••••••••••••
Use SSL connection: off
QoS: QoS 0 [✓ QoS 1] QoS 2
Use clean connection: on

Figure 4.3.2 App Broker Settings

5 CONCLUSION

Thus, the paper offers an idea of uniting the modern technology into the agricultural field to turn the outdated methods of irrigation to modern methods, thus making fruitful, and economical harvesting. Some degree of automation is presented enabling the concept of supervising the field and the crop conditions within some long-distance ranges using mosquitto brokers. The benefits like water saving and labour-saving are originated using sensors that work automatically as they are programmed. This concept of modernization of agriculture is simple, reasonable and operable.

6 REFERENCES

[1] LIU Dan, Cao Xin, Huang Chongwei, JI Liang, “Intelligent agent greenhouse environment monitoring system based on IOT technology”,2015 International Conference on Intelligent Transportation, Big Data &Smart City.

[2] Joseph Haule, Kisangiri Michael, “Deployment of wireless sensor networks (WSN) in automated irrigation management and scheduling systems: a review”, Science, Computing and Telecommunications(PACT), 2014, Pan African Conference.

[3] http://esp32.net/

[4] https://en.wikipedia.org/wiki/Internet_of_Things.

[5] https://mongoose-os.com/docs/mongoose-os/userguide/intro.md

[6] T. Thaker, "ESP8266 based implementation of wireless sensor network with Linux based web-server," 2016 Symposium on Colossal Data Analysis and Networking (CDAN), Indore, India, 2016, pp. 1-5, doi: 10.1109/CD

S.Jyothirmaye, is an Associate Professor in ECE Department, Geethanajali College of Engineering and Technology, JNTUH University, India. She has completed her B.E (ECE) from Manipal Institute of Technology, MAHE University in the year 2002, M.Tech (Embedded Systems) from JNTUH University, India . He has about 14 years of teaching experience at various levels. Her research interest includes WSN, Deep Learning and IoT.

A R L Padmaja is an Assistant Professor in ECE Department, Geethanajali College of Engineering and Technology, JNTUH University, India. She completed her B. Tech in ECE, M. Tech in Embedded Systems. Her research interests include MEMS/NEMS based sensors, Analog and RF IC design, IC technology and SoC design.