STAND-ALONE DEVICE FOR IOT APPLICATIONS WITH A USE CASE IN PRECISION AGRICULTURE SCENARIO

Neha Kailash Nawandar*[1], Naveen Cheggoju[2], Vishal Ramesh Satpute[3]
[1]Post Doctoral Fellow, Department of Electrical Engineering, IIT Delhi
Hauz Khas, New Delhi, India 110016
Email: nnawandar@ee.iitd.ac.in

[2]Assistant Professor, Department of Electronics and Communication Engineering, School of Electronics, IIIT Una,
Una, Himachal Pradesh, India 177209
Email: naveen.c@iiitu.ac.in

[3]Associate Professor, Department of Electronics and Communication Engineering, VNIT Nagpur
South ambazari road, Nagpur, Maharashtra, India 440010
Email: vrsatpute@ece.vnit.ac.in

KEY WORDS: IoT, stand-alone device, smart agriculture.

ABSTRACT: The Internet has become a necessity, and with the Internet era came an astonishing domain, the Internet of Things (IoT), a digital world of connected and communicating devices that provide room for a plethora of smart applications. The IoT is an umbrella domain with enormously prolific applications like smart homes, smart waste management, smart agriculture, smart industry, retail, etc. IoT-enabled devices communicate with each other with minimal or no user involvement making it a primary element while developing a product nowadays. The work presented in this paper presents a similar stand-alone device and its use case in precision agriculture. The device works in three different modes, viz., admin mode, one-time setup mode, and continuous monitoring mode. Admin mode is a new provision for device-user interaction where the user provides details about the required application for which he intends to use the device. Users use this mode to select an application and enter application-specific data for setup and calibration. Then the device goes into a one-time setup mode, which is used for pre-hand application-specific estimation. The last mode is continuous monitoring, where the device captures data using sensors, sends this data to the server, and makes decisions based on this real-time data. We have created a smart home-garden environment to verify and test the device's capability. For this purpose, the device is interfaced with ambient temperature, humidity, and soil moisture sensors and actuators. The data gathered by the sensors is sent to the server using MQTT. MQTT is preferred over other protocols due to its lightweight and low power consumption features. The device sends the decision to the actuators, thus, achieving a continuous monitoring process. The paper provides sample results for this smart agriculture application, demonstrating the device's viability for an IoT- based smart agriculture application. The device prototype is developed and can be used in numerous IoT-based smart applications. The codes related to the device application in the presented smart agriculture scenario are available at https://github.com/nehakn/Smart-irrigation-system

1. INTRODUCTION

Due to the rapid growth of digitization, the Internet has become necessary, transforming the world into a digital space where humans require more precision than machines. Without human intervention, it would be much easier to live if we had systems that know everything there is to know about various things based on the data collected. We would know when an item needs to be repaired, replaced, etc. The Internet of Things (IoT) (Ashton 2009) is a system of interconnected computing devices, objects, things, animals, people, etc., that can share data over a network without human intervention. Every day, the expansion of the Internet creates new opportunities for establishing a connection between the real physical world and the virtual world. This collaboration makes computing and networking more efficient, adaptable, secure, and reliable for e-planes, vehicle collision avoidance, WSNs, device monitoring, e-health, and other applications. It enables the development of various applications, many of which have been successfully implemented. Examples are wearables, smart grids, smart infrastructure, connected vehicles, smart retail, intelligent supply chain, and industrial internet. These applications have been investigated, and some existing Internet of Things (IoT) products are listed in Table 1.

Table 1. Examples of some existing IoT products

Application	Product
Home safety	Birdi
Connected car	Cloud your car
Home ordering tool	Amazon dash button
Connected garden	Parrot pot
Thing tracker	Tile
Medical alert system	Lively watch, apple watch
Pet activity tracker	FitBark

These things communicate with each other and with the server using various protocols at different layers as given below,

- Infrastructure: 6LowPAN, IPv4/IPv6, RPL.
- Identification: EPC, uCode, IPv6, URIs.
- Communication: Wifi, Bluetooth, LPWAN.
- Discovery: Physical Web, mDNS, DNS-SD.
- Data protocols: MQTT, CoAP, AMQP, Websocket, Node.
- Device management: TR-069, OMA-DM.
- Semantic: JSON-LD, Web Thing Model.
- Multi-layer frameworks: Alljoyn, IoTivity, Weave, Homekit.

Now being called the language of the IoT, it seems evident that any standard aiming to bring a typical network service layer to the IoT architecture should be able to utilize MQ Telemetry Transport (MQTT) (Velez 2018). In this paper, the device-server communication is done using the MQTT protocol due to its simplicity and low latency for data transfer. The paper is organized as follows: Section 2 introduces the stand-alone device, explains its algorithm, and explores its capabilities, while its results for a home gardening application are shown in Section 3. Finally, the paper has been concluded in Section 4, and references have been mentioned at the end.

2. PROPOSED STAND-ALONE DEVICE

This section introduces the proposed stand-alone device and the algorithm. As seen in Figure 1, the device's capabilities have intelligence, using which it gets data from the sensors. It uses current and previous data history to predict and act accordingly for numerous applications. It also allows the tracking of sensor data using the MQTT protocol. The proposed device's system architecture is shown in Figure 2, and the device algorithm is given by Algorithm 1. Figure 2 shows that multiple sensors are connected to the device ESP8266, which operates as per proposed Algorithm 1, and the data is published to the broker, which can be remotely observed on mobile, dashboard, etc. The device runs an HTML webpage where the user can provide inputs per the application environment and has sensing and decision-making capabilities using sensors and actuators, respectively. The GPIO pins on the device connect multiple sensors and actuators as per requirement, where application-specific calibration can be done. The algorithm on the device runs multiple routines responsible for its efficient working, and its pseudo-code can be seen in Algorithm 1. Here, S: set of sensors for fetching physical data, δ: application-specific decision, α: actuators required.

The algorithm runs in a user-device communication mode for a pre-defined time interval. In this mode, the device takes inputs from the user, if any, and saves them temporarily onto the device for further usage. After a time-out, it disables this mode and starts working, usually where necessary computations are done in a one-time device setup routine. After successful calibration and setup, the device begins sensing the data using the attached sensors and continuously looks for actuation if needed. It also lets the user track and monitor the sensor data anytime and anywhere. This is achieved by uploading the data to an MQTT server, using which the user is aware of the present scenario and can manually control the actuators if needed.

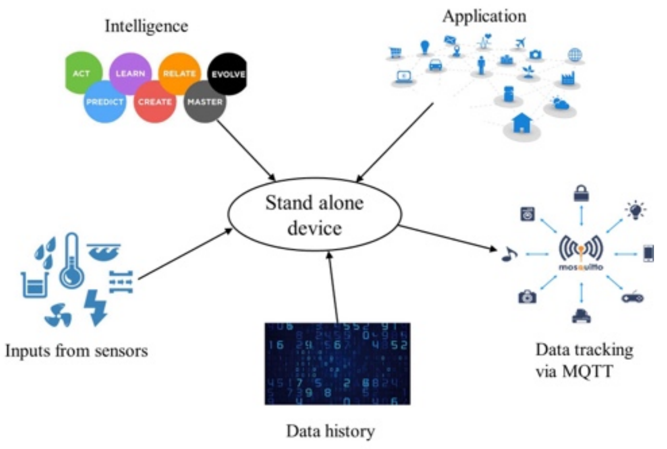


Figure 1. Proposed device capabilities

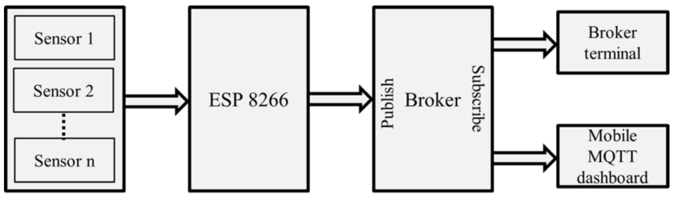


Figure 2. System architecture using the proposed stand-alone device

Algorithm 1. Proposed stand-alone device pseudo-code

```
Input: S
Output: δ, α
    1.  User-device communication mode:
        n_config → n/w configuration;
        n_info → n/w information;
        application:
        user input;
        save data on device;
    2.  if !Time-out then Goto 1;
    3.  else
    4.  Disable current mode;
    5.  One-time device setup;
    6.  Monitor S:
        T → ambient temperature;
        H → ambient humidity;
        SM → soil moisture
    7.  Connect to MQTT server:
        Mosquitto broker → 10.4.1.100;
        Create topics → /usp/temp, /usp/humid, /usp/sm;
        Publish data;
    8.  Save data offline;
    9.  Alert to relay → control α
    10. end
```

Table 2. Various sensor requirements in smart applications

Application	Sensors
Smart home (Sequeiros 2021)	Temperature, humidity, motion, light intensity, fire detection, CO_2, etc.
Smart e-health (Scarpato 2017)	Epidermal (Bio-patches, digital tattoos, body-worn tags), Soluble ingestible embedded in pills, etc.
Intelligent parking (Said 2021)	IR, proximity, RADAR, and Rear-mounted cameras.
Waste management (Anagnostopoulos 2017)	PID, trash SONAR, temperature, fill-level.
Smart farming (Zamora-Izquierdo 2019)	Temperature, humidity, soil moisture, soil nutrient, rain, soil salinity, etc.
Environmental monitoring (Kelly 2013)	CO and leakage detection, temperature and humidity, snow level, thermo-hygrometer.

As step 7 of Algorithm 1 shows, the device establishes a connection with Mosquitto, an open-source MQTT broker, for publishing the sensor data. MQTT is lightweight, energy efficient, and way faster than HTTP is preferred here to reduce the power requirement and latency. The data is sent to this Mosquitto broker, where the user continuously tracks it and generates statistics. If an unsuccessful connection occurs, it might create data loss, so to prevent it, a compressed backup is also stored offline on the device. This increases the device's validity, which implies that it can adapt to changes, thus making it efficient for future use without recalibration. The sensor data is used for actuation, so a decision δ based on the sensed data alerts the α, thus maintaining a continuous monitoring loop. The device connects to relays that can be used to connect to multiple application-specific actuators α. Table 2 mentions possible sensors for some intelligent applications. The proposed device can be used for these applications, and the corresponding sensors, when connected to the GPIO pins, complete the work.

As mentioned earlier, the device hosts an HTML page, and Figure 4 (a) shows three options: (i) network configuration, (ii) network information, and (iii) application that are available in user-device communication mode. The device scans for nearby networks and can be configured to connect to any of these using the network configuration button. It displays the connected network, its details, and nearby networks on selection. The final network information is then available on the network information page. The user inputs are obtained using the application button. An intelligent home gardening application environment is created to test the efficiency and work of the proposed device, and S, α are chosen accordingly. Section 3 introduces the application and presents the individual results obtained.

3. APPLICATION AND RESULTS

The proposed stand-alone device is tested for its efficacy using a smart home gardening application. Some of the sensors mentioned in the smart farming portion of Table 2 are used for getting data. It is used to:

- Get the ambient characteristics.
- Fetch soil moisture.
- Data publish to MQTT broker.
- Compute irrigation (IN) requirements.
- Give 'time' for drip.
- Motor control for IN.

The device is used for an intelligent farming application, and apart from its basic capabilities, those mentioned above are achieved using it. It can be used to track the growth and determine IN needs of crops planted in backyard gardens, indoor or organic farming, vertical gardens, fields, etc. To achieve this, temperature and humidity sensors for ambient data and soil moisture sensors for getting soil moisture levels are connected to the device. In this case, the user inputs are the crop, the type of soil to be planted in, and the date of crop plantation. These data are used to compute the IN requirements and an irrigation schedule. In this case, it is then connected to an actuator, a water pump to supply water when required.

A setup consisting of sensors and actuators is needed for the application, and the arrangement is shown in Figure 3. Here, the setup is used in a farm for a crop under test: (a) soil moisture sensor: gets the soil water content, (b) temperature and humidity sensor: fetches the ambient data, (c) the controller: controls the sensors, actuators, and works as per Algorithm 1, (d) motor that controls water to the crop, (e) relay module that controls the motor. The detailed working of this stand-alone device for intelligent farming irrigation-based applications (Nawandar 2019) has been done.

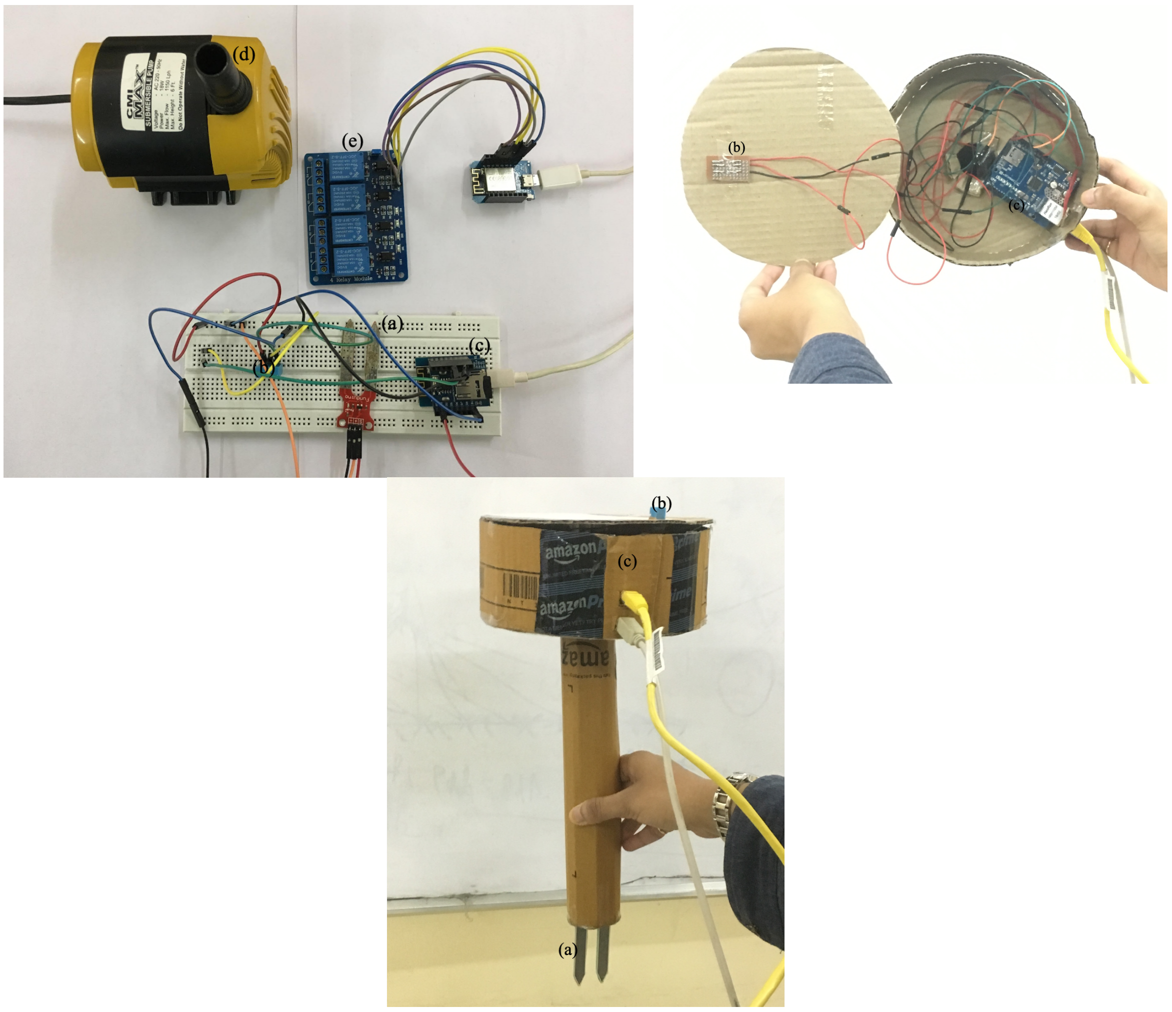
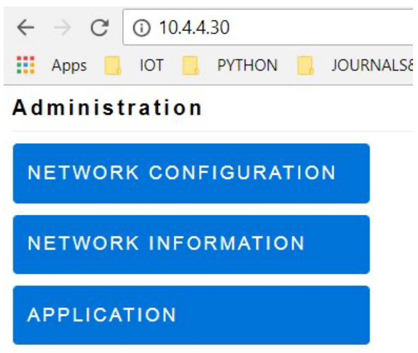


Figure 3. Stand-alone device: components used, intermediate connections, and a temporary prototype



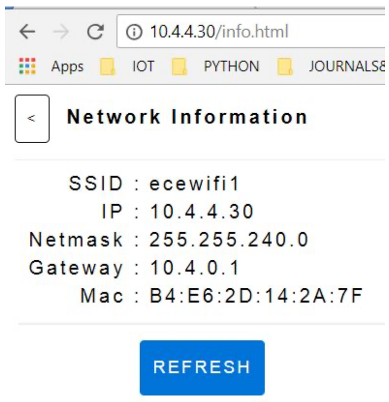
Administration

NETWORK CONFIGURATION

NETWORK INFORMATION

APPLICATION

(a) Admin page

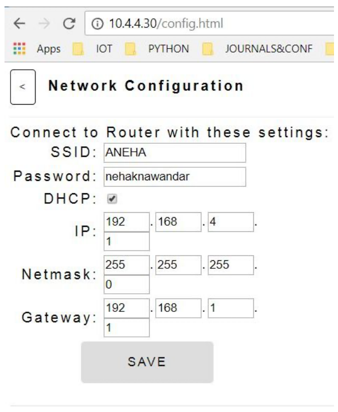


Network Information

```
    SSID : ecewifi1
      IP : 10.4.4.30
 Netmask : 255.255.240.0
 Gateway : 10.4.0.1
     Mac : B4:E6:2D:14:2A:7F
```

REFRESH

(b) Network information



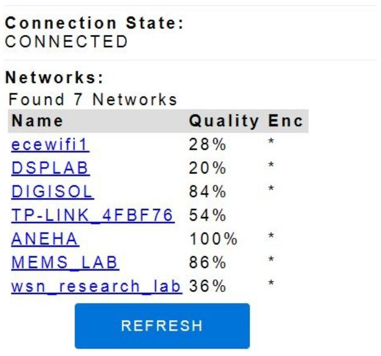
Network Configuration

Connect to Router with these settings:

```
    SSID: ANEHA
Password: nehaknawandar
    DHCP: ☑
      IP: 192 . 168 . 4 . 1
 Netmask: 255 . 255 . 255 . 0
 Gateway: 192 . 168 . 1 . 1
```

SAVE

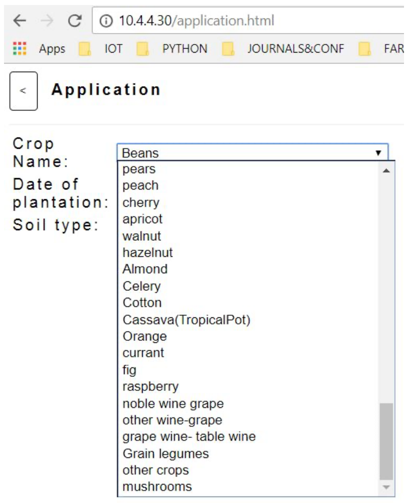
(c) Details of the network



```
Connection State:
CONNECTED

Networks:
Found 7 Networks
Name             Quality Enc
ecewifi1         28%     *
DSPLAB           20%     *
DIGISOL          84%     *
TP-LINK_4FBF76   54%
ANEHA            100%    *
MEMS_LAB         86%     *
wsn_research_lab 36%     *
```

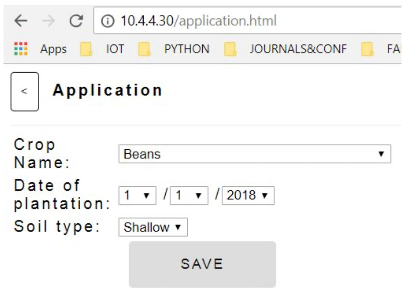
(d) List of available networks



Crop Name: Beans
Date of plantation:
Soil type:

Beans, pears, peach, cherry, apricot, walnut, hazelnut, Almond, Celery, Cotton, Cassava(TropicalPot), Orange, currant, fig, raspberry, noble wine grape, other wine-grape, grape wine- table wine, Grain legumes, other crops, mushrooms

(e) Crop options available

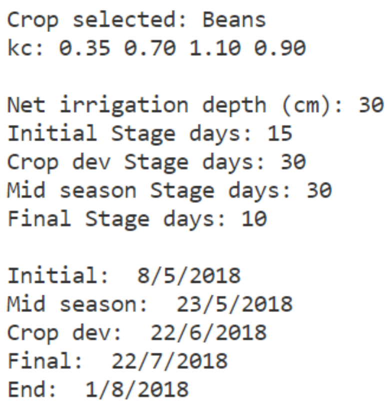


Crop Name: Beans
Date of plantation: 1 / 1 / 2018
Soil type: Shallow

(f) Inputs provided by the user

Figure 4. User-device communication HTTP webpage

An application option on the webpage, as shown in Figure 4 (a), is available, which on selection, generates fields for user input consisting of the crop, plantation date, and soil type. A list of yields seen from Figure 4 (e) is available from which a selection is made for a crop that needs to be planted. Similarly, dates and soil types are selected and stored on the device for further usage, as seen in Figure 4 (f). Based on these selections and the weather data history, some crop-related parameters are selected and computed to help determine the IN needs and schedule. The device is intelligent enough to tackle harsh climatic conditions and update IN needs accordingly. Some sample results can be seen in Figure 5. Figure 5(a) shows the number of days in every growth stage and the starting dates for the same, and Figure 5(b) gives the IN depth, water runtime, and dates (highlighted). Using this data provided by the device, users can grow plants whenever needed.

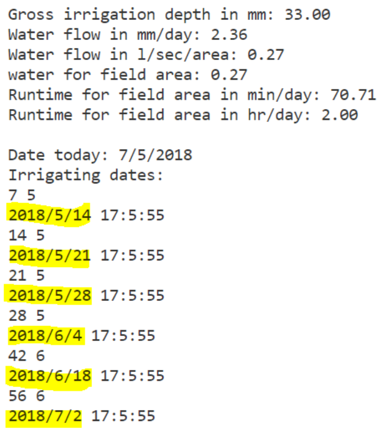


```
Crop selected: Beans
kc: 0.35 0.70 1.10 0.90

Net irrigation depth (cm): 30
Initial Stage days: 15
Crop dev Stage days: 30
Mid season Stage days: 30
Final Stage days: 10

Initial:  8/5/2018
Mid season:  23/5/2018
Crop dev:  22/6/2018
Final:  22/7/2018
End:  1/8/2018
```

(a) Stage-wise details of the selected crop



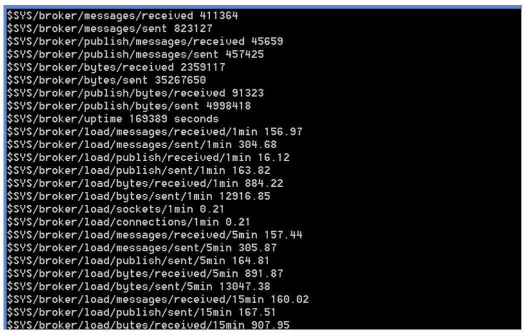
```
Gross irrigation depth in mm: 33.00
Water flow in mm/day: 2.36
Water flow in l/sec/area: 0.27
water for field area: 0.27
Runtime for field area in min/day: 70.71
Runtime for field area in hr/day: 2.00

Date today: 7/5/2018
Irrigating dates:
7 5
2018/5/14 17:5:55
14 5
2018/5/21 17:5:55
21 5
2018/5/28 17:5:55
28 5
2018/6/4 17:5:55
42 6
2018/6/18 17:5:55
56 6
2018/7/2 17:5:55
```

(b) Irrigation information for the selected crop

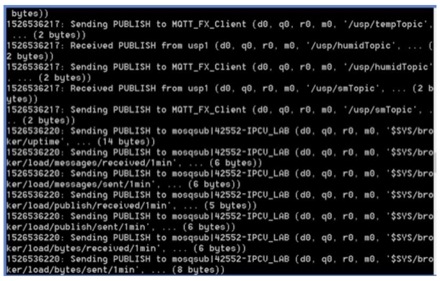
Figure 5. Computed parameters for Beans

As mentioned earlier, the data fetched via the sensors connected to the device can be remotely monitored using MQTT based server, generally referred to as a broker, and Figure 6 shows corresponding results. The Mosquitto broker status is seen in Figure 6 (a), which shows the received and sent data information, number of connections to the broker, and published messages information. Figure 6 (b) shows information about the subscribed topics, while the subscribed topics' sensor data values and timings are seen in Figure 6 (c). This data can also be viewed from an app on a smartphone, as shown in Figure 7. Here, the MQTT dashboard is first connected to the host Mosquitto broker, an IP address rather than a hostname. After establishing a successful connection, it subscribes to topics the broker hosts, i.e., temperature, humidity, and soil moisture topics (in this application). This keeps the user updated with the latest data and actuator status. An additional advantage is that if the device fails, it cannot update; thus, the user learns that it has forgotten/ stopped working and can rectify its issues.



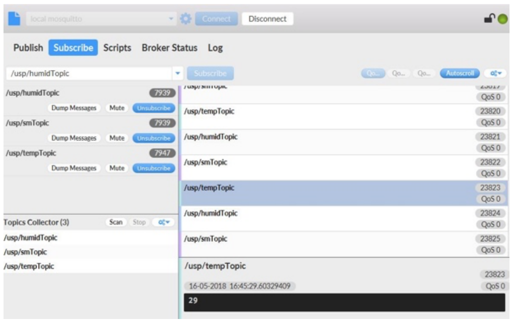
```
$SYS/broker/messages/received 411364
$SYS/broker/messages/sent 823127
$SYS/broker/publish/messages/received 45659
$SYS/broker/publish/messages/sent 457425
$SYS/broker/bytes/received 2359117
$SYS/broker/bytes/sent 35267650
$SYS/broker/publish/bytes/received 91323
$SYS/broker/publish/bytes/sent 4998418
$SYS/broker/uptime 169389 seconds
$SYS/broker/load/messages/received/1min 156.97
$SYS/broker/load/messages/sent/1min 304.68
$SYS/broker/load/publish/received/1min 16.12
$SYS/broker/load/publish/sent/1min 163.82
$SYS/broker/load/bytes/received/1min 884.22
$SYS/broker/load/bytes/sent/1min 12916.85
$SYS/broker/load/sockets/1min 0.21
$SYS/broker/load/connections/1min 0.21
$SYS/broker/load/messages/received/5min 157.44
$SYS/broker/load/messages/sent/5min 305.87
$SYS/broker/load/publish/sent/5min 164.81
$SYS/broker/load/bytes/received/5min 891.87
$SYS/broker/load/bytes/sent/5min 13047.38
$SYS/broker/load/messages/received/15min 160.02
$SYS/broker/load/publish/sent/15min 167.51
$SYS/broker/load/bytes/received/15min 907.95
```

(a) Broker status after turning it ON



```
 bytes))
1526536217: Sending PUBLISH to MQTT_FX_Client (d0, q0, r0, m0, '/usp/tempTopic',
 ... (2 bytes))
1526536217: Received PUBLISH from usp1 (d0, q0, r0, m0, '/usp/humidTopic', ... (
2 bytes))
1526536217: Sending PUBLISH to MQTT_FX_Client (d0, q0, r0, m0, '/usp/humidTopic'
, ... (2 bytes))
1526536217: Received PUBLISH from usp1 (d0, q0, r0, m0, '/usp/smTopic', ... (2 b
ytes))
1526536217: Sending PUBLISH to MQTT_FX_Client (d0, q0, r0, m0, '/usp/smTopic', .
.. (2 bytes))
1526536220: Sending PUBLISH to mosqsub|42552-IPCU_LAB (d0, q0, r0, m0, '$SYS/bro
ker/uptime', ... (14 bytes))
1526536220: Sending PUBLISH to mosqsub|42552-IPCU_LAB (d0, q0, r0, m0, '$SYS/bro
ker/load/messages/received/1min', ... (6 bytes))
1526536220: Sending PUBLISH to mosqsub|42552-IPCU_LAB (d0, q0, r0, m0, '$SYS/bro
ker/load/messages/sent/1min', ... (6 bytes))
1526536220: Sending PUBLISH to mosqsub|42552-IPCU_LAB (d0, q0, r0, m0, '$SYS/bro
ker/load/publish/received/1min', ... (5 bytes))
1526536220: Sending PUBLISH to mosqsub|42552-IPCU_LAB (d0, q0, r0, m0, '$SYS/bro
ker/load/publish/sent/1min', ... (6 bytes))
1526536220: Sending PUBLISH to mosqsub|42552-IPCU_LAB (d0, q0, r0, m0, '$SYS/bro
ker/load/bytes/received/1min', ... (6 bytes))
1526536220: Sending PUBLISH to mosqsub|42552-IPCU_LAB (d0, q0, r0, m0, '$SYS/bro
ker/load/bytes/sent/1min', ... (8 bytes))
```

(b) List of subscribed topics on the broker



/usp/humidTopic

/usp/humidTopic 7939

/usp/smTopic 7939

/usp/tempTopic 7947

Topics Collector (3)

/usp/humidTopic
/usp/smTopic
/usp/tempTopic

/usp/tempTopic	23820	QoS 0
/usp/humidTopic	23821	QoS 0
/usp/smTopic	23822	QoS 0
/usp/tempTopic	23823	QoS 0
/usp/humidTopic	23824	QoS 0
/usp/smTopic	23825	QoS 0

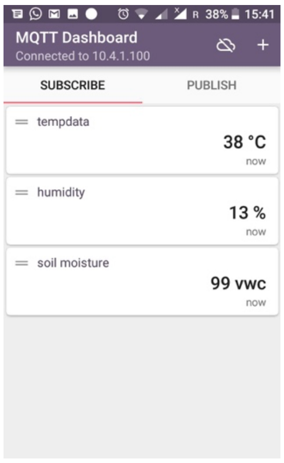
/usp/tempTopic 23823

16-05-2018 16:45:29.60329409 QoS 0

29

(c) Sensor data as seen from the dashboard accessed using a Windows PC

Figure 6. Sensor data on MQTT broker



R 38% 15:41

MQTT Dashboard

Connected to 10.4.1.100

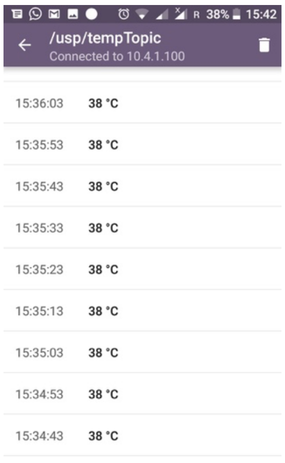
SUBSCRIBE PUBLISH

tempdata 38 °C now

humidity 13 % now

soil moisture 99 vwc now

(a) MQTT dashboard on a mobile app



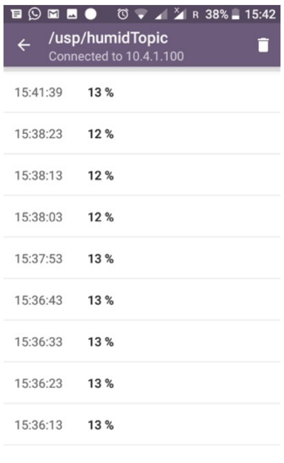
R 38% 15:42

/usp/tempTopic

Connected to 10.4.1.100

15:36:03	38 °C
15:35:53	38 °C
15:35:43	38 °C
15:35:33	38 °C
15:35:23	38 °C
15:35:13	38 °C
15:35:03	38 °C
15:34:53	38 °C
15:34:43	38 °C

(b) Subscription to the temperature topic



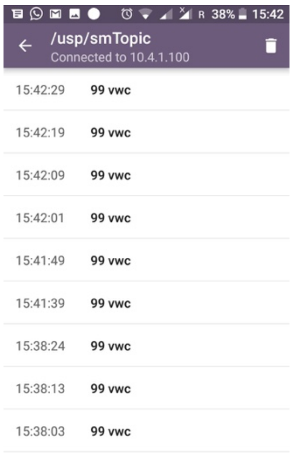
R 38% 15:42

/usp/humidTopic

Connected to 10.4.1.100

15:41:39	13 %
15:38:23	12 %
15:38:13	12 %
15:38:03	12 %
15:37:53	13 %
15:36:43	13 %
15:36:33	13 %
15:36:23	13 %
15:36:13	13 %

(c) Subscription to the humidity topic



R 38% 15:42

/usp/smTopic

Connected to 10.4.1.100

15:42:29	99 vwc
15:42:19	99 vwc
15:42:09	99 vwc
15:42:01	99 vwc
15:41:49	99 vwc
15:41:39	99 vwc
15:38:24	99 vwc
15:38:13	99 vwc
15:38:03	99 vwc

(d) Subscription to the soil moisture topic

Figure 7. Sensor data on remote device

The device could also be used for another smart application by connecting the proper sensors, and output tool, providing corresponding inputs, and creating an application set up for analysis.

4. CONCLUSION

This paper proposes a standalone device with (i) a device-user interaction mode, HTML webpage hosting, and application-specific user requirements, (ii) data sensing, decision-making, and actuation, and (iii) data publishing to server capabilities. It enables remote access to sensor data and actuator control and eliminates the need for manual operation by providing remote access to sensor data and actuator control. A home gardening environment was created using the proposed device and some agriculture-related sensors for verification. In addition, the device's overall irrigation requirement and schedule for various plants have been described. It demonstrates versatility for various cutting-edge applications, including intelligent homes, organic farming, indoor gardening, and organics. With the optimal combination of sensors, the device could be used for various other applications.

References

Anagnostopoulos, T., Zaslavsky, A., Kolomvatsos, K., Medvedev, A., Amirian, P., Morley, J. and Hadjieftymiades, S., 2017. Challenges and opportunities of waste management in IoT-enabled smart cities: A survey. *IEEE Transactions on Sustainable Computing*, *2*(3), pp.275-289.

Ashton, K., 2009. That 'internet of things' thing. *RFID journal*, *22*(7), pp.97-114.

Kelly, S.D.T., Suryadevara, N.K. and Mukhopadhyay, S.C., 2013. Towards the implementation of IoT for environmental condition monitoring in homes. *IEEE sensors journal*, *13*(10), pp.3846-3853.

Nawandar, N.K. and Satpute, V.R., 2019. IoT based low cost and intelligent module for smart irrigation system. *Computers and electronics in agriculture*, *162*, pp.979-990.

Said, A.M., Kamal, A.E. and Afifi, H., 2021. An intelligent parking sharing system for green and smart cities based IoT. *Computer Communications*, *172*, pp.10-18.

Scarpato, N., Pieroni, A., Di Nunzio, L. and Fallucchi, F., 2017. E-health-IoT universe: A review. *management*, *21*(44), p.46.

Sequeiros, H., Oliveira, T. and Thomas, M.A., 2021. The impact of IoT smart home services on psychological well-being. *Information Systems Frontiers*, pp.1-18.

Velez, J., Trafford, R., Pierce, M., Thomson, B., Jastrzebski, E. and Lau, B., 2018, March. IEEE 1451-1-6: Providing common network services over MQTT. In *2018 IEEE Sensors Applications Symposium (SAS)* (pp. 1-6). IEEE.

Zamora-Izquierdo, M.A., Santa, J., Martínez, J.A., Martínez, V. and Skarmeta, A.F., 2019. Smart farming IoT platform based on edge and cloud computing. *Biosystems engineering*, *177*, pp.4-17.