ISSN : 0973 - 8355

www.ijmmsa.com

INTERNATIONAL JOURNAL OF MATHEMATICAL, MODELLING, SIMULATIONS AND APPLICATIONS

IJMMSA

E-MAIL
editor.ijmmsa@gmail.com
editor@ijmmsa.com

INTERNATIONAL JOURNAL OF MATHEMATICAL, MODELLING, SIMULATION AND APPLICATIONS
IJMMSA
ISSN; 0973-8355 www.ijmmsa.com
Vol 17 Issue 3 Aug 2024

Design and Modelling of an IoT-Based Office Automation System Using ESP32 with Smartphone and Voice Control

Prof. Adarsh J. Mehta[1], Mr. Shekhar D. Dhale[2], Mr. Sumit B. Badure[3], Mr. Shrinath S. Desai[4], Mr. Vinit V. Bhandare[5], Mr. Harshal N. Talbhandare[6], Mr. Vijay M. Patade[7]

Assistant Professor, Department of Electrical Engineering, Nagesh Karajagi Orchid College of Engineering & Technology, Solapur[1]

***Abstract*—**Office buildings waste electricity when lights, fans and other appliances stay switched on in unoccupied spaces, and manual switching gives no remote visibility. This paper presents the design, modelling and prototype implementation of a low-cost Internet of Things (IoT) system that lets a user switch and monitor office appliances from anywhere. An ESP32 microcontroller with integrated Wi-Fi drives a four-channel opto-isolated relay module that switches AC loads and reports ambient temperature and humidity from a DHT11 sensor. Users command the system either from a Blynk smartphone dashboard (switch mode) or by speaking to Google Assistant, with an IFTTT applet relaying the phrase to the Blynk cloud (voice mode). Local switches remain functional and stay synchronised with the dashboard. The paper documents the architecture, the GPIO and virtual-pin mapping, the active-low relay interface, the power budget and the firmware structure, and compares the design with related smart-office proposals. The prototype, with an AC fan and an 8 W LED lamp as loads, costs Rs. 1335. Security limitations of cloud-based control and extensions such as occupancy sensing and energy metering are discussed.

Keywords*—Internet of Things, office automation, ESP32, relay module, Blynk, Google Assistant, IFTTT, energy conservation.*

I. Introduction

Offices concentrate many electrical loads—lighting points, fans, computers, printers, photocopiers, microwave ovens and socket outlets—and every appliance left switched on in an unoccupied room adds to energy consumption and to the electricity bill. Energy conservation, understood as reducing consumption by eliminating waste and improving efficiency, therefore starts with making sure that appliances run only when they are needed. Office automation, the use of computing and communication technologies to operate and support routine office functions with less human effort, is one route to that goal [1], [2].

The Internet of Things (IoT) makes such automation practical by giving physical devices network connectivity, so that they can be monitored and switched remotely from a smartphone or by voice. A manager responsible for several rooms or offices can then check and correct the state of appliances without being present. The barrier to adoption is often cost and complexity: many commercial systems are proprietary, need a dedicated hub and are priced beyond the reach of small offices, laboratories and classrooms.

This paper describes a low-cost design that addresses this gap. The system is built around an ESP32 microcontroller [7] and a four-channel relay module, and is controlled either from a Blynk smartphone dashboard [8] or by voice through Google Assistant and IFTTT [9]. The objectives are (i) to control hardware devices through the internet, (ii) to control and monitor office appliances, (iii) to exchange data between the devices, the cloud and the user, and (iv) to reduce wasted energy. The main contributions are:

(1) a layered architecture that combines smartphone (switch-mode) and voice-mode control on a single Wi-Fi-connected controller;
(2) a firmware design in which local switches stay fully functional and are kept synchronised with the cloud dashboard, so the office can still be operated manually;
(3) a documented hardware interface—GPIO and virtual-pin mapping, active-low relay drive and power budget—that allows the prototype to be reproduced; and
(4) a bill of materials of Rs. 1335 for the prototype, and a comparison with related smart-office systems.

Section II reviews related work. Sections III to V present the architecture, hardware and software design. Section VI reports the implementation, cost and discussion, Section VII covers advantages, limitations and security, Section VIII outlines applications and future work, and Section IX concludes.

II. Related Work

Vishwakarma *et al.* [1] describe a smart, energy-efficient office automation controller that turns ordinary appliances into remotely accessible devices. Their design uses a NodeMCU as the controller, an Adafruit MQTT broker for messaging, IFTTT to interpret voice commands and the Arduino IDE for programming; the office is controlled from Google Assistant and from a web application. To keep the system reachable, the controller is programmed to reconnect automatically to an always-available Wi-Fi network and is fed from a backup supply.

Somani *et al.* [2] combine office automation with security. A Raspberry Pi acts as the server, an Android application turns the smartphone into a remote control for the appliances, and a camera module with motion sensing at the entrance sends the owner a notification containing a real-time photograph, so that the owner can raise an alarm or, for a guest, operate the door remotely.

Other studies approach the problem from different angles: a dynamic distributed energy-management algorithm for office sensor networks [3], enhanced IoT-based smart-office systems [4], visual machine intelligence for office automation [5], and IoT-based smart home and office automation more generally [6].

Compared with these works, the present design concentrates on minimum cost and reproducibility. It keeps physical switches alive alongside cloud control, uses one ESP32 for both switching and sensing, and documents the

INTERNATIONAL JOURNAL OF MATHEMATICAL, MODELLING, SIMULATION AND APPLICATIONS
IJMMSA
ISSN; 0973-8355 www.ijmmsa.com
Vol 17 Issue 3 Aug 2024

interface in enough detail to be rebuilt. Table I summarises the differences from [1] and [2].

TABLE I
COMPARISON WITH TWO RELATED SYSTEMS

Feature	[1]	[2]	This work
Controller	NodeMCU	Raspberry Pi	ESP32
Server / cloud	Adafruit MQTT broker	Raspberry Pi as server	Blynk cloud
Voice control	Google Assistant + IFTTT	Not described	Google Assistant + IFTTT
Other features	Web app; auto Wi-Fi reconnect; backup power	Camera; motion-sensor alerts	Local switch override; temperature and humidity monitoring

III. System Architecture

Fig. 1 shows the overall architecture. The ESP32 is the central unit: it receives commands from the Blynk cloud over Wi-Fi and switches the appliances through a relay module. A DHT11 sensor provides ambient temperature and relative humidity, four local switches allow manual operation, and a 5 V adapter supplies the ESP32 and the relay coils. Users can issue commands in two modes.

Switch mode. The user toggles a button widget on the Blynk dashboard. The message travels to the Blynk cloud, which finds the target device through its unique authentication token and forwards the command to the ESP32. The same path works in the opposite direction, so changes made locally are reflected on the dashboard.

Voice mode. Google Assistant cannot address an ESP32 directly, and it does not understand device-specific phrases such as “turn on relay one” on its own. An IFTTT applet therefore acts as intermediary: the spoken phrase is recognised by Google Assistant, matched by the applet and converted into a request to the Blynk cloud, after which it follows the same path as a dashboard command. A command such as “OK Google, turn on the light” thus ends as an ON signal on the corresponding relay channel.

Wi-Fi was chosen as the communication medium because it gives infrastructure-level access to the internet without a gateway. Bluetooth is essentially a short-range, point-to-point technology with a data rate of the order of 720 kbit/s, whereas IEEE 802.11n Wi-Fi offers physical-layer rates of up to 150 Mbit/s, which leaves headroom for later extensions such as camera-based security [2]. The ESP32 integrates both radios, so the choice remains a firmware decision.

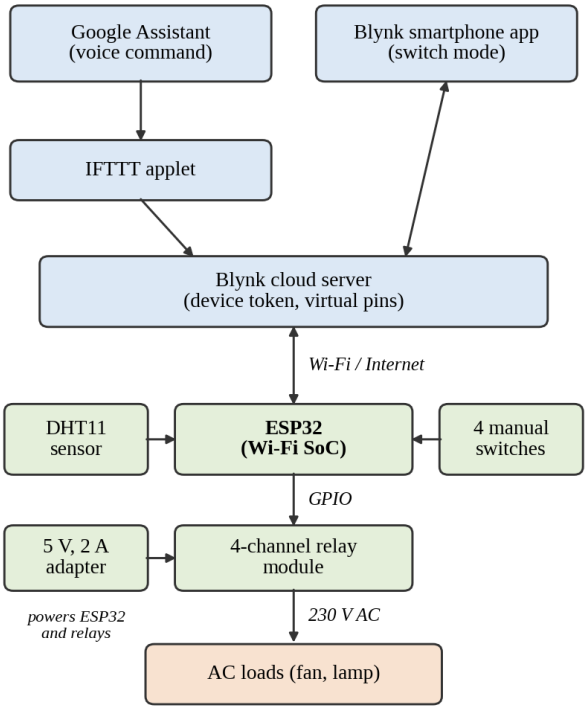

Fig. 1. Architecture of the proposed IoT-based office automation system.

IV. Hardware Design

A. ESP32 Controller

The ESP32 is a series of low-cost, low-power system-on-chip microcontrollers from Espressif Systems with integrated 2.4 GHz Wi-Fi and dual-mode Bluetooth [7]. Depending on the variant, it uses a single- or dual-core Xtensa LX6 or LX7 processor or a single-core RISC-V processor, and integrates antenna switches, an RF balun, a power amplifier, a low-noise receive amplifier, filters and power-management modules. It is the successor of the ESP8266 used on NodeMCU boards. The integrated Wi-Fi removes the need for an external modem, and the I/O operates at 3.3 V.

B. Four-Channel Relay Module

Appliances are switched by a four-channel 5 V relay module (Table II). Each relay has a 5 V coil and contacts rated 250 V AC / 30 V DC at 10 A. Transistor stages buffer the coil current, so each input needs only about 5 mA of trigger current, which an ESP32 GPIO can supply. Freewheeling diodes suppress the voltage spike when a coil is de-energised, an LED per channel shows which relay is active, and optocouplers provide optional galvanic isolation between the control inputs and the switched load, selected with the VCC/JD-VCC jumper. Screw terminals make the mains connections easy to wire and change.

ISSN; 0973-8355 www.ijmmsa.com
Vol 17 Issue 3 Aug 2024

TABLE II
FOUR-CHANNEL RELAY MODULE SPECIFICATIONS

Parameter	Value
Supply voltage	3.75–6 V
Trigger current	5 mA
Coil current (one / four relays on)	≈ 70 mA / ≈ 300 mA
Contact rating	10 A, 250 V AC / 30 V DC
Isolation	Optocoupler, jumper-selectable
Indication	One LED per channel

The module inputs are active-low: a relay is energised when its input is driven LOW. The firmware therefore writes the logical complement of the desired state to the relay pin (relay pin = NOT state) and drives all inputs HIGH at start-up so that no appliance is energised while the controller boots. The contact rating is far above the load current of the prototype: an 8 W lamp on a 230 V supply draws roughly 35 mA. Motor loads such as fans have starting surges and should be derated accordingly.

C. Power Supply and Loads

A 5 V, 2 A AC/DC adapter supplies the ESP32 and the relay coils. The supply current is

$$I_{sup} = I_{ESP32} + n \cdot I_{rel} \quad (1)$$

where $n \leq 4$ is the number of energised relays and $I_{rel} \approx 70$ mA. With all four relays on, the coils draw about 300 mA; adding the ESP32 and its Wi-Fi transmission peaks still leaves ample margin below the 2 A rating. The prototype loads are an AC fan and an 8 W LED lamp, connected through a socket and holder. The relay terminal blocks are the only mains-side connections; the mains wiring should be enclosed and protected by a suitable fuse or miniature circuit breaker.

D. Pin Assignment

Table III lists the GPIO assignment and the corresponding Blynk virtual pins used in the firmware. Virtual pins V1–V4 carry the relay states, V6 and V7 carry temperature and humidity, and a further virtual pin serves as a master "all off" command.

TABLE III
ESP32 GPIO AND BLYNK VIRTUAL-PIN MAPPING

Function	ESP32 GPIO	Blynk pin
Relay channels 1–4	23, 22, 21, 19	V1–V4
Local switches 1–4	13, 12, 14, 27	—
DHT11 data	16	V6 (temp.), V7 (RH)
Wi-Fi status LED	2	—

V. SOFTWARE AND FIRMWARE DESIGN

A. Blynk Platform

Blynk is an IoT platform with iOS and Android applications for controlling hardware over the internet [8]. It has three parts: the mobile application, in which a dashboard is built by dragging and dropping widgets; the server, which handles all communication between the smartphone and the hardware; and client libraries for the hardware, which process incoming and outgoing messages. A device template and a per-device authentication token bind the ESP32 to its dashboard. Each button press travels to the cloud, which locates the device by its token and forwards the value to the corresponding virtual pin.

B. Voice Path

Google Assistant is a virtual assistant that uses natural-language processing to carry out spoken requests on phones, smart speakers and displays. IFTTT ("If This, Then That") is a web service that links devices and services through applets, each pairing a trigger with an action [9]. Here the trigger is a phrase recognised by Google Assistant and the action is a request that sets the matching Blynk virtual pin. Voice control therefore needs only a phone or smart speaker with Assistant and an IFTTT account, with no extra hardware.

C. Firmware

The firmware is written in the Arduino IDE with the ESP32 Arduino core, the Blynk library and a DHT library. Its structure is shown in Fig. 2.

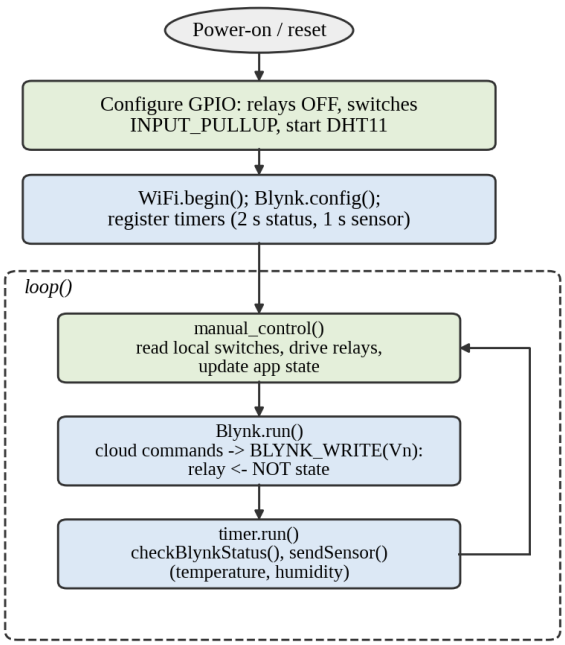

Fig. 2. Firmware structure of the ESP32 controller.

Initialisation. setup() configures the relay pins as outputs and drives them to the OFF level, configures the switch pins as inputs with internal pull-up resistors, starts the DHT11 driver and calls WiFi.begin() with the network credentials. Blynk.config() is used instead of the blocking Blynk.begin(), so start-up does not wait for the network. Two software timers are registered: a connectivity check every 2 s and a sensor upload every 1 s.

Main loop. Each pass of loop() calls manual_control(), Blynk.run() and timer.run(). Because manual_control() runs independently of the cloud state, the local switches remain in the control path when Wi-Fi or the cloud is unavailable.

Cloud commands. A BLYNK_WRITE handler is registered for each switch virtual pin. It stores the received value as the toggle state of the channel and writes its complement to the relay pin. A handler for the master

ISSN; 0973-8355 www.ijmmsa.com
Vol 17 Issue 3 Aug 2024

command turns all channels off and writes the new states back so that the dashboard stays consistent.

Local switches. manual_control() implements maintained-switch behaviour with edge detection. When a switch input goes LOW while its stored switch state is LOW, the relay is energised, the toggle state is set, the dashboard is updated with Blynk.virtualWrite() and the switch state is latched; the opposite transition switches the relay off in the same way. The physical switches and the app therefore stay in agreement.

Reconnection and sync. checkBlynkStatus() tests Blynk.connected() every 2 s and drives a Wi-Fi status LED that is on while the cloud connection is up. When the connection is (re)established, BLYNK_CONNECTED() requests the last stored values of the switch and sensor virtual pins, so the relays follow the dashboard state; a configuration flag selects the alternative in which the controller pushes its own state to the server.

Sensing. sendSensor() reads the DHT11 and pushes temperature and humidity to V6 and V7; a failed read is discarded and the previous values are kept. Two values per second is well below the Blynk limit of ten values per second. The DHT11 has a typical accuracy of about ±2 °C and ±5 % relative humidity, which suits indication of comfort conditions but not precise control. The Wi-Fi credentials and device token are compiled into the firmware; the security implications are discussed in Section VII.

VI. Implementation and Results

A. Prototype

The prototype was assembled from the components in Table IV: an ESP32 board, a four-channel relay module, a 5 V, 2 A adapter, an AC fan and an 8 W LED lamp connected through a socket and holder, and jumper wires between the controller and the relay inputs. The ESP32 and relay board were powered from the adapter, and the fan and lamp were switched by relay channels.

B. Functional Behaviour

The prototype operates in both control modes described in Section III. The implemented functions are: (i) switching each relay from the Blynk dashboard; (ii) switching by spoken commands relayed through Google Assistant, IFTTT and Blynk; (iii) switching by the local switches with the dashboard state updated accordingly; (iv) display of temperature, humidity and channel states on the dashboard; (v) a master command that switches all channels off; and (vi) a Wi-Fi status LED giving local indication of cloud connectivity.

C. Cost

TABLE IV
Bill of Materials of the Prototype

Sr.	Component	Qty	Rate (Rs.)
1	ESP32 microcontroller	1	400
2	4-channel relay module	1	180
3	Adapter (5 V, 2 A)	1	130
4	AC fan	1	325
5	LED bulb (8 W)	1	80
6	Jumper wires	10	20
7	Wire, socket, holder, etc.	—	200
	Total		**1335**

The total cost is Rs. 1335. The controller, relay board, adapter and jumper wires account for Rs. 730 of this; because they are shared by all channels, the fixed cost per controlled appliance falls to about Rs. 183 when all four channels are used. The reported bill does not include the DHT11 sensor and the local switches used by the firmware.

D. Discussion

The prototype shows that cloud-connected switching and voice control of AC appliances can be achieved with off-the-shelf parts at low cost. Energy is saved indirectly: appliances that would otherwise be left running can be switched off from any location, by hand, app or voice. This paper does not report measured energy savings, command latency or long-term reliability; these depend on office usage, the Wi-Fi network and cloud performance, and are left for a field study.

VII. Advantages, Limitations and Security

A. Advantages

Convenience: appliances are managed from one dashboard or by voice. *Flexibility:* further channels and appliances can be added by extending the relay stage and the virtual-pin map. *Energy efficiency and cost:* remote switch-off reduces idle consumption and hence the electricity bill. *Monitoring:* temperature and humidity data give insight into office conditions. *Affordability:* the whole chain is built from inexpensive, widely available modules.

B. Limitations

Cloud control depends on the availability of power, Wi-Fi and the third-party Blynk and IFTTT services; the local switches mitigate this for basic operation. Integration and setup need some technical skill, and the time and money needed for a large deployment can be considerable. Because commands and data pass through external services, security and privacy must be addressed. Finally, convenience should not lead to over-reliance: the office must remain operable by hand if the system fails.

C. Security Recommendations

The prototype firmware stores the Wi-Fi credentials and the Blynk token as constants. For a deployment we recommend: a WPA2/WPA3 network with a strong passphrase, separated from the main office network; provisioning credentials at run time instead of compiling them into the source; regenerating the device token if the

ISSN; 0973-8355 www.ijmmsa.com
Vol 17 Issue 3 Aug 2024

source is shared; two-factor authentication on the Google, IFTTT and Blynk accounts; and encrypted (TLS) connections to the cloud.

VIII. Applications and Future Work

The system suits lighting and fan control in offices, classrooms and laboratories, multi-room monitoring, and remote shutdown of appliances at the end of the working day. The same hardware can be extended with sensors for environment monitoring and access control.

Future work includes: (i) PIR or mmWave occupancy sensing and scheduling for automatic switch-off; (ii) current or energy metering to quantify savings; (iii) MQTT over TLS, over-the-air firmware updates and a local fallback when the cloud is unreachable; (iv) solid-state relays or phase-control for fan-speed regulation; (v) camera and face-recognition door access as in [2]; and (vi) a field trial measuring latency, reliability and kWh savings.

IX. Conclusion

This paper presented the design and prototype of a low-cost IoT-based office automation system in which an ESP32 switches AC appliances through a four-channel relay module and reports temperature and humidity. Control is possible from a Blynk dashboard, by voice through Google Assistant and IFTTT, and from local switches that stay synchronised with the cloud. The documented hardware interface, pin mapping and firmware structure make the design reproducible, and the prototype cost of Rs. 1335 makes it attractive for small offices and educational laboratories. Quantitative evaluation of energy savings and responsiveness, and stronger security provisioning, are the next steps.

Acknowledgment

The authors thank Dr. B. K. Sonage (I/C Principal), Prof. V. B. Bhosale (Head, Department of Electrical Engineering) and Prof. A. B. Auti (Project Coordinator) of NKOCET, Solapur, for their support and for the facilities provided.

References

[1] S. K. Vishwakarma, P. Upadhyaya, B. Kumari, and A. K. Mishra, “Smart energy efficient office automation system using IoT.”

[2] S. Somani, P. Solunke, S. Oke, P. Medhi, and P. P. Laturkar, “IoT based smart security and office automation.”

[3] T.-Y. Yang, C.-S. Yang, and T.-W. Sung, “A dynamic distributed energy management algorithm of office sensor network for office automation system,” in *Proc. 3rd Int. Conf. Computing, Measurement, Control and Sensor Network*, 2016.

[4] T. Churasia and P. K. Jain, “Enhance smart office automation system based on Internet of Things,” in *Proc. 3rd Int. Conf. I-SMAC (IoT in Social, Mobile, Analytics and Cloud)*, IEEE, 2019, ISBN 978-1-7281-4365-1.

[5] Suraj, I. Kool, D. Kumar, and S. Barman, “Visual machine intelligence for office automation.”

[6] M. Arunkumar, V. Archana, S. Anjana Devi, and S. Nithya Sri, “Advanced smart home and office automation using IoT,” in *Proc. 5th Int. Conf. Smart Systems and Inventive Technology (ICSSIT)*, IEEE, 2023, pp. 462–468, doi: 10.1109/ICSSIT55814.2023.10061148.

[7] Espressif Systems, “ESP32 series datasheet.” [Online]. Available: https://www.espressif.com

[8] Blynk, “Blynk IoT platform documentation.” [Online]. Available: https://docs.blynk.io

[9] IFTTT, “IFTTT: Every thing works better together.” [Online]. Available: https://ifttt.com