*International Journal of advancement in electronics and computer engineering ( IJAECE)*
*Volume 15, Issue 8, Aug. 2026, pp. 105-109, ISSN 2278 -1412*
*Copyright © 2012: IJAECE*

# VoiceSwitch: Smartphone Voice-Controlled Home Automation using Arduino Uno and HC-05 Bluetooth with Acknowledgement Feedback

Dakksheya Iyer[1]
*[1]Independent Researcher, iyerdakk15@gmail.com, India*

***Abstract –*** Switching household appliances by hand is inconvenient for elderly people and persons with limited mobility, while commercial voice assistants need costly hubs, compatible appliances and a permanent internet connection. This paper presents VoiceSwitch, a low-cost voice-controlled home automation system in which an Android application converts the user's speech to text and sends a short command code over Bluetooth to an Arduino Uno, which drives a four-channel relay board and returns a text acknowledgement to the phone. The design uses only open hardware and a MIT App Inventor application, and its total component cost is about INR 1,400. The prototype was tested with five commands, 50 trials per command in a quiet room (35 dB) and in a noisy room (65 dB). Average recognition accuracy was 96.4% in the quiet room and 91.2% in the noisy room, the mean end-to-end response time was 0.91 s, and the Bluetooth link was reliable (at least 94% success) up to 10 m indoors. Error analysis shows that most failures arise from speech-to-text mistakes rather than from the Arduino or the wireless link.

***Keywords****: Home automation, voice control, Arduino Uno, HC-05, Bluetooth, relay module, Android, speech recognition, assistive technology.*

## I. INTRODUCTION

A home automation system lets residents control lights, fans and other appliances without walking to a wall switch. For elderly people, patients and persons with physical disabilities, this is not only a comfort feature but an accessibility need. Among the available control interfaces (wall panel, remote, mobile touch screen, gesture and voice), speech is the most natural because it needs no screen, no visual attention and no fine motor control.

Commercial smart speakers provide excellent voice control, but they require compatible smart plugs or bulbs, a cloud account and a continuous internet connection. These requirements raise the cost and make the system unusable during network outages. Many students, hobbyists and low-income households therefore need a simple local alternative that they can build, repair and extend themselves.

### *A. Problem statement*

The problem addressed in this work is to design and evaluate a low-cost, locally-controlled voice interface that can switch ordinary mains-powered appliances, gives the user confirmation of every action, and has measurable accuracy, delay and range.

### *B. Contributions*

- A complete Arduino Uno and HC-05 design with a documented pin map, level shifting and relay interface (Section III).
- A compact command protocol with terminator-based framing and acknowledgement messages that make every action verifiable on the phone (Section IV).
- A repeatable test protocol and measured results for recognition accuracy in two noise conditions, response-time breakdown, range and error causes (Sections V and VI).

The rest of the paper is organised as follows. Section II reviews related work, Section III describes the hardware, Section IV explains the methodology and software, Section V gives the experimental setup, Section VI presents and discusses the results, Section VII lists limitations, and Section VIII concludes the paper.

## II. RELATED WORK

Home automation systems are commonly classified by their communication medium. Bluetooth-based systems [1], [2] are cheap, need no router and have low power consumption, but their range is limited to a room or a floor. Wi-Fi based systems [3] and cloud-connected systems built on the Internet of Things [4], [5] provide worldwide access but depend on internet connectivity and introduce security concerns. GSM-based controllers allow control through calls or SMS but have recurring message costs and give little feedback.

Voice interfaces rely on automatic speech recognition (ASR). Modern ASR systems based on deep neural networks [6] are available free of charge inside Android through the SpeechRecognizer service [7], so a custom application does not need to implement its own recogniser. The present work combines this service with a minimal Bluetooth link to an Arduino platform [8], and concentrates on a clear protocol and a measured evaluation, which are often missing in project-style reports.

TABLE I
QUALITATIVE COMPARISON OF CONTROL APPROACHES FOR HOME APPLIANCES

| Approach | Internet needed | Voice | Cost | Feedback |
|---|---|---|---|---|
| Commercial smart speaker + hub | Yes | Yes | High | Yes |
| Wi-Fi / cloud relay (IoT app) | Yes | Optional | Low–Med | Yes |
| GSM / SMS controller | No (SIM) | No | Medium | SMS |
| Bluetooth + button app | No | No | Low | Optional |
| **Proposed VoiceSwitch** | No* | Yes | Low | Yes (ACK) |

*The Android speech service may use the internet unless an offline language pack is installed; the control path from phone to appliance is entirely local.

## III. HARDWARE DESIGN

### *A. Components and cost*

The prototype uses widely available modules (Table II). The Arduino Uno (ATmega328P, 5 V, 16 MHz) [8] is the

Copyright © 2012: IJAECE

controller. The HC-05 module provides a Bluetooth serial port profile (SPP) link [9] with a default data rate of 9600 baud. The four-channel relay board has opto-isolated inputs and contacts rated 10 A at 250 V AC and is used with the common (COM) and normally-open (NO) terminals.

TABLE II
COMPONENTS AND APPROXIMATE COST (INR)

| Component | Specification | Qty | Cost |
|---|---|---|---|
| Arduino Uno (clone) | ATmega328P, 5 V | 1 | 450 |
| HC-05 module | Bluetooth 2.0 SPP | 1 | 300 |
| Relay board | 4-ch, 5 V, opto-isolated | 1 | 180 |
| Resistors, wires | 1k, 2k, jumpers | – | 80 |
| 5 V / 2 A adapter | USB or barrel | 1 | 150 |
| Enclosure, terminals | Plastic box, connectors | 1 | 240 |
| **Total** | | | **1,400** |

### *B. Circuit connections*

Fig. 1 shows the connections. The HC-05 is connected to software-serial pins D2 (RX) and D3 (TX) rather than to the hardware serial pins 0 and 1, so that the USB port remains free for programming and debugging. Because the HC-05 data input is a 3.3 V logic input, the Arduino TX line (5 V) is passed through a resistive divider of 1 kΩ and 2 kΩ, which gives 3.3 V. The relay inputs IN1 to IN4 are driven from pins D8 to D11. The relay board used is active-LOW; the pins are therefore set HIGH at start-up so that every appliance is OFF after a reset or a power failure.

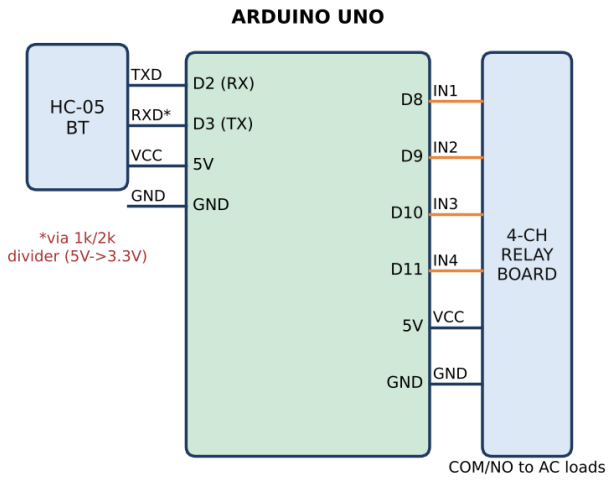



**Fig. 1.** Pin-level connections between Arduino Uno, HC-05 module and the four-channel relay board.

### *C. Electrical safety*

Mains wiring is hazardous. In the prototype, the low-voltage electronics and the AC terminals are in separate compartments of an insulated enclosure, the live conductor is switched by the relay, each load circuit is protected by a fuse or miniature circuit breaker, and no exposed mains conductor is accessible. The relay contact rating (10 A) must not be exceeded; inductive loads such as fan motors should be derated, and heavy loads should be switched through a contactor.

### *D. HC-05 configuration*

The module is placed in AT mode (38400 baud) once to set its name and PIN and to confirm the data baud rate of 9600. After configuration it is paired with the phone like any other Bluetooth device. Since the module has no encryption beyond the standard PIN-based pairing, the system is intended for short-range domestic use.

## IV. METHODOLOGY AND SOFTWARE

### *A. System architecture*

Fig. 2 shows the architecture. The user speaks into the phone. The application converts the speech to text using the Android SpeechRecognizer, matches the text with the command list and sends a short code over Bluetooth. The Arduino decodes the code, switches the relay and sends back an acknowledgement that is displayed in the application. Because the acknowledgement originates from the Arduino after the relay is actuated, the user can distinguish a recognised command from an executed command.

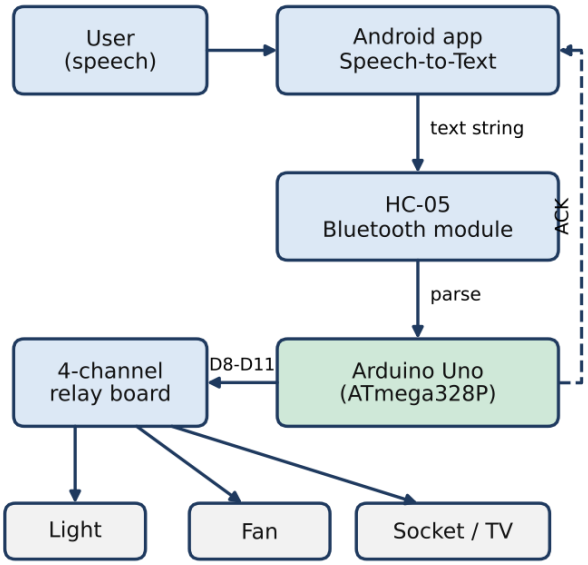



**Fig. 2.** Block diagram of the VoiceSwitch system. Dashed line: acknowledgement path.

### *B. Command protocol*

Each command is a two-character code followed by the terminator character '#' (Table III). The terminator makes the framing robust against partial or concatenated packets. Unknown codes are rejected with an "Unknown" message. The "All OFF" command switches off every relay and is meant as a safety command.

TABLE III
COMMAND SET AND ARDUINO PIN MAPPING

| Spoken phrase | Code | Pin | Action / ACK |
|---|---|---|---|
| "light on" | L1# | D8 | Relay 1 LOW → "Light ON" |
| "light off" | L0# | D8 | Relay 1 HIGH → "Light OFF" |
| "fan on" | F1# | D9 | Relay 2 LOW → "Fan ON" |
| "fan off" | F0# | D9 | Relay 2 HIGH → "Fan OFF" |
| "all off" | A0# | D8–D11 | All relays HIGH → "All OFF" |

### *C. Working flow*

The operation is summarised in Fig. 3. After initialisation (all relays OFF) the system waits for a spoken command. The app converts it to text and sends the code. If the code is valid, the Arduino switches the relay and sends the acknowledgement; otherwise the command is ignored and the system returns to waiting.

Copyright © 2012: IJAECE

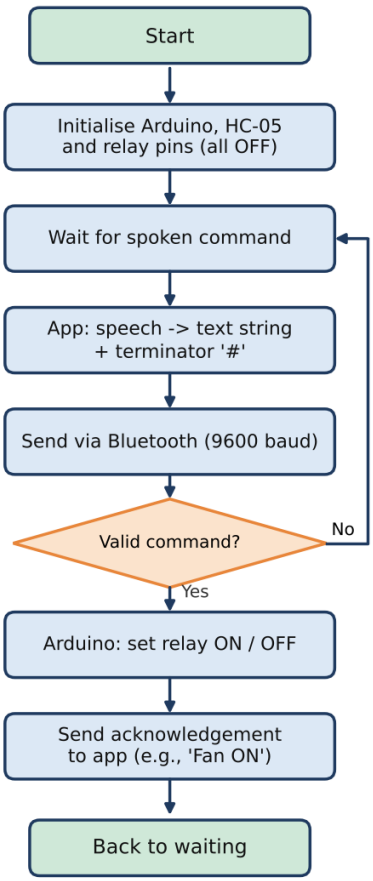



**Fig. 3.** Flowchart of VoiceSwitch operation.

### *D. Arduino algorithm*

**Algorithm 1.** Command parsing on the Arduino.

```
setup():
  bt.begin(9600)   // SoftwareSerial D2,D3
  for pin in 8..11:
    pinMode(pin, OUTPUT)
    write(pin, HIGH)   // all OFF
loop():
  while bt.available():
    ch = bt.read()
    if ch == '#':
      execute(buf); buf = ""
    elif len(buf) < 3: buf += ch
    else: buf = ""  // overflow guard
execute(cmd):
  L1: write(8,LOW);  ack "Light ON"
  L0: write(8,HIGH); ack "Light OFF"
  F1: write(9,LOW);  ack "Fan ON"
  F0: write(9,HIGH); ack "Fan OFF"
  A0: all HIGH;      ack "All OFF"
  other: ack "Unknown"
```

### *E. Android application*

The application was built with MIT App Inventor. A microphone button starts the SpeechRecognizer component; the returned text is converted to lower case and compared with the five phrases. A matching phrase is mapped to its code and sent through the BluetoothClient component. The application displays the recognised text, the connection status and the latest acknowledgement, and shows the state of each appliance (Fig. 4).

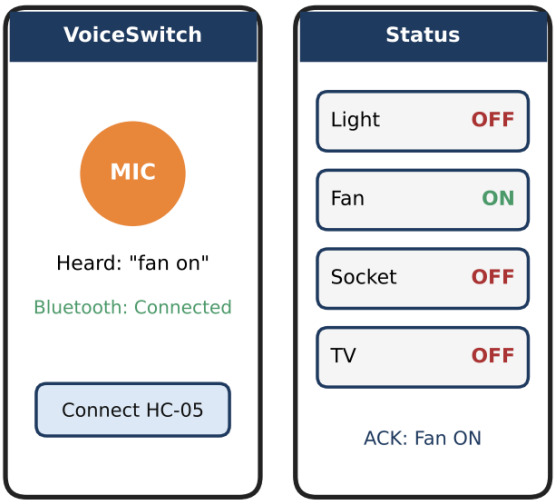



**Fig. 4.** User interface of the VoiceSwitch application: voice screen (left) and appliance status screen (right).

### *F. Performance metrics*

Recognition accuracy for a command is the fraction of trials in which the intended relay changed state:

$$Accuracy\ (\%) = (N_correct / N_trials) \times 100 \qquad (1)$$

The end-to-end response time is the sum of its components:

$$T_resp = T_stt + T_app + T_bt + T_ard \qquad (2)$$

where T_stt is the speech-to-text time, T_app the application processing and write time, T_bt the Bluetooth transmission time and T_ard the Arduino execution time.

## V. EXPERIMENTAL SETUP

The prototype was installed in a room of approximately 4 m × 4 m with a light, a table fan and two sockets connected to the relay board. Five adult volunteers (three male, two female) repeated each of the five commands ten times in each of two conditions, giving 50 trials per command and condition (250 trials per condition). The quiet condition had a background level of about 35 dB; in the noisy condition a television and a ceiling fan raised it to about 65 dB, measured with a sound-level meter application. The phone was held about 30 cm from the mouth and the same Android phone was used throughout.

A trial was counted as correct only if the intended relay changed state within 3 s. Response time was measured from the end of the utterance to the relay click, using a screen recording with audio and the acknowledgement timestamp. For the range test, the phone was moved away from the controller in steps of 2 m along a corridor with one wall between 8 m and 14 m, and 50 commands were sent at each distance. The three failure types counted were speech-to-text errors (wrong or no transcription), time-outs without speech detection, and Bluetooth link errors.

## VI. RESULTS AND DISCUSSION

### *A. Recognition accuracy*

Table IV and Fig. 5 show the results. In the quiet room, 241 of 250 commands were executed correctly (96.4%), and in the noisy room 228 of 250 (91.2%), so the 30 dB increase in noise lowered accuracy by 5.2 percentage points. “All off” had the lowest accuracy in both conditions because the two-word phrase was sometimes transcribed as “all of”, and a few “off” commands were transcribed as “of”.

Copyright © 2012: IJAECE

TABLE IV
CORRECT EXECUTIONS OUT OF 50 TRIALS PER COMMAND

| Command | Quiet | Quiet % | Noisy | Noisy % |
|---|---|---|---|---|
| Light ON | 49/50 | 98 | 47/50 | 94 |
| Light OFF | 49/50 | 98 | 46/50 | 92 |
| Fan ON | 48/50 | 96 | 46/50 | 92 |
| Fan OFF | 48/50 | 96 | 45/50 | 90 |
| All OFF | 47/50 | 94 | 44/50 | 88 |
| **Overall** | **241/250** | **96.4** | **228/250** | **91.2** |

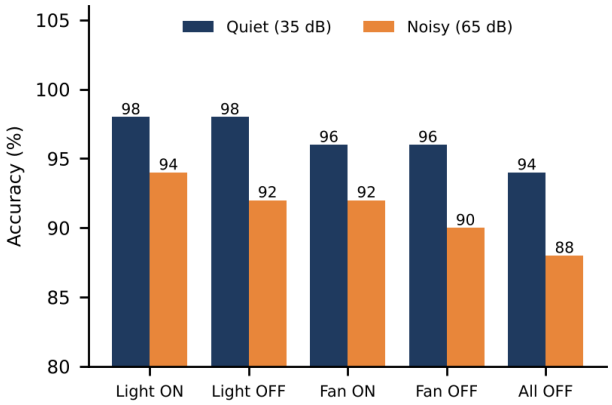



**Fig. 5.** Command recognition accuracy in quiet (35 dB) and noisy (65 dB) conditions.

### *B. Response time*

The mean response time over the five commands was 0.91 s. Fig. 6 shows that the speech-to-text stage takes 0.74 s, which is about 81% of the total; the application processing and Bluetooth write take 0.12 s, the Bluetooth link 0.04 s, and the Arduino with relay only 0.01 s. The command "All OFF" takes longer (1.10 s) because it is a longer phrase to transcribe, not because of relay switching. Table V gives the per-command values.




**Fig. 6.** Breakdown of the mean end-to-end response time.

TABLE V
MEAN RESPONSE TIME PER COMMAND (S)

| Command | Mean | Std. dev. |
|---|---|---|
| Light ON | 0.84 | 0.07 |
| Light OFF | 0.85 | 0.08 |
| Fan ON | 0.88 | 0.08 |
| Fan OFF | 0.89 | 0.09 |
| All OFF | 1.10 | 0.11 |

### *C. Bluetooth range*

Fig. 7 shows the success rate against distance. The success rate stayed at 94% or higher up to 10 m, then dropped to 78% at 12 m and 52% at 14 m, where the wall between phone and module attenuates the 2.4 GHz signal. A range of 10 m is adequate for a room or an apartment, and range can be extended by placing the controller centrally or by using a Wi-Fi module as future work.

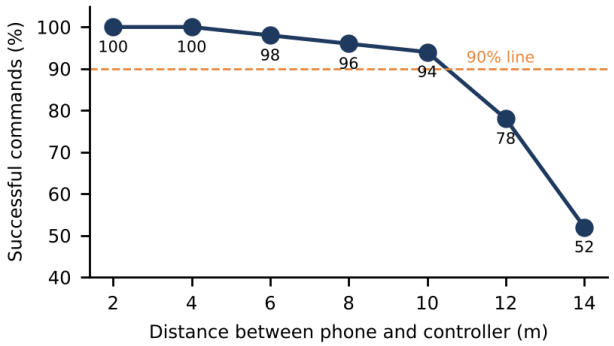



**Fig. 7.** Percentage of successful commands against distance (50 attempts per distance).

### *D. Error analysis*

Table VI classifies the failed trials. Nine failures occurred in the quiet room and 22 in the noisy room. In both rooms the majority were speech-to-text mistakes, so improving the microphone position or using a headset is the most effective way to improve accuracy. Bluetooth errors were rare (two in total) and no Arduino parsing errors were observed because every transmitted code was matched exactly.

TABLE VI
CLASSIFICATION OF FAILED TRIALS (OUT OF 250)

| Failure type | Quiet | Noisy |
|---|---|---|
| Wrong transcription | 7 | 15 |
| No speech detected (time-out) | 2 | 5 |
| Bluetooth link error | 0 | 2 |
| Arduino parsing error | 0 | 0 |
| **Total failures** | **9** | **22** |

### *E. Discussion*

The results show that a simple local architecture gives a response time of under one second and an accuracy above 91% even with 65 dB of background noise. The acknowledgement message was useful in practice: in all failed trials the absence or wrong text of the acknowledgement told the user at once that the command had not been executed, so the command could be repeated. Because phrases are matched exactly, the application does not trigger an appliance from unrelated speech. The main limitations are the dependence on the phone's speech service and the short Bluetooth range.

## VII. LIMITATIONS

The study used five volunteers and one phone model, so accuracy for other accents, languages and microphones may differ. The speech service may need internet unless an offline language pack is installed. The HC-05 link provides only basic pairing security. The prototype controls on/off switching only and does not measure current or detect a relay or load failure.

## VIII. CONCLUSION AND FUTURE WORK

VoiceSwitch, a voice-controlled home automation system using an Arduino Uno, an HC-05 module and a relay board, was designed and evaluated. It achieved 96.4% and 91.2% accuracy in quiet and noisy rooms, a mean response time of 0.91 s and a 10 m indoor range at a hardware cost of about INR 1,400. Future work includes offline keyword spotting on the phone, support for Hindi and other regional languages, a Wi-Fi gateway (ESP8266) for remote access, current sensing for energy monitoring, speaker verification for security, and dimming control with a TRIAC module.

*Copyright © 2012: IJAECE*

ACKNOWLEDGMENT

The author thanks the volunteers who took part in the voice tests.

REFERENCES

[1] R. Piyare, "Internet of Things: Ubiquitous home control and monitoring system using Android based smart phone," Int. J. Internet of Things, vol. 2, no. 1, pp. 5–11, 2013.

[2] N. Sriskanthan, F. Tan and A. Karande, "Bluetooth based home automation system," Microprocessors and Microsystems, vol. 26, no. 6, pp. 281–289, 2002.

[3] M. ElShafee and K. A. Hamed, "Design and implementation of a WiFi based home automation system," World Academy of Science, Engineering and Technology, vol. 68, pp. 2177–2180, 2012.

[4] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, 2013.

[5] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey," Computer Networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[6] G. Hinton et al., "Deep neural networks for acoustic modeling in speech recognition," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82–97, 2012.

[7] Android Developers, "SpeechRecognizer," Android API reference. [Online]. Available: https://developer.android.com/reference/android/speech/SpeechRecognizer

[8] Microchip Technology, "ATmega328P datasheet," and Arduino, "Arduino Uno Rev3 documentation." [Online]. Available: https://docs.arduino.cc

[9] Bluetooth SIG, "Bluetooth Core Specification," and Serial Port Profile (SPP) specification. [Online]. Available: https://www.bluetooth.com/specifications/