

Article

Control of Smart Home Operations Using Natural Language Processing, Voice Recognition and IoT Technologies in a Multi-Tier Architecture

George Alexakis, Spyros Panagiotakis *, Alexander Fragkakis, Evangelos Markakis  and Kostas Vassilakis

Department of Electrical and Computer Engineering, Hellenic Mediterranean University, GR71004 Heraklion, Greece

* Correspondence: spanag@hmu.gr

Received: 15 April 2019; Accepted: 14 June 2019; Published: 1 July 2019



Abstract: The Internet of Things (IoT) is an emerging Internet-based architecture, enabling the exchange of data and services in a global network. With the advent of the Internet of Things, more and more devices are connecting to the Internet in order to help people get and share data or program actions. In this paper, we introduce an IoT Agent, a Web application for monitoring and controlling a smart home remotely. The IoT Agent integrates a chat bot that can understand text or voice commands using natural language processing (NLP). With the use of NLP, home devices are more user-friendly and controlling them is easier, since even when a command or question/command is different from the presets, the system understands the user's wishes and responds accordingly. Our solution exploits several available Application Programming Interfaces (APIs), namely: the Dialogflow API for the efficient integration of NLP to our IoT system, the Web Speech API for enriching user experience with voice recognition and synthesis features, MQTT (Message Queuing Telemetry Transport) for the lightweight control of actuators and Firebase for dynamic data storage. This is the most significant innovation it brings: the integration of several third-party APIs and open source technologies into one mash-up, highlighting how a new IoT application can be built today using a multi-tier architecture. We believe that such a tiered architecture can be very useful for the rapid development of smart home applications.

Keywords: Internet of Things; IoT; smart home; chatbot; natural language processing; NLP; Web Speech API; Dialogflow API; MQTT; multi-tier architecture

1. Introduction

Mobile phones, home electrical appliances, cars, simple light bulbs, airplanes—indeed, almost every device we use in our daily lives—is connected to the internet, sending and receiving data in order to communicate with the outside world. The growth of the Internet of Things (IoT) has been rapid. It is commonly used term and we all use it in some way. The Internet of Things has the potential to change the way we interact in our daily life. Everyday objects with a physical structure carry sensors and actuators able to undertake actions, or even more, hold ambient data. At the same time, with the use of the Internet, these data can be stored in databases, shared via any application imaginable and be ready for users, at any time without cost [1,2].

The IoT has several usages as mentioned, from industrial infrastructure to plain light bulbs. One of the most common applications is what is known as the smart home. A smart home is a house equipped with every day appliances that can hold data and connect to the Internet. Moreover, the residents can control remotely these devices or monitor their states in real time. With this ability,

someone authorized can access those data and program actions according to their needs, e.g., when the time is 8 p.m., turn the lights on. It is as easy as that [3].

The IoT can be made even more robust and easy, with the use of natural language processing (NLP). People that are not so familiar with the technology, such as elderly people or people with disabilities, but also people that want a simple way to interact with their devices, could be assisted by NLP systems to use existing technology [4]. In particular, an application used in conjunction with NLP could handle voice commands to turn on or off every device in a home, monitor the home's environmental conditions, or even control a smart car. This could be generalized to every interconnected device. NLP is a subfield of artificial intelligence (AI), which wishes to create a basis for computers to understand and use human natural language [5]. While computers are suited to handling structured data, natural language uses exactly the opposite: "unstructured data". Hence, it is difficult for computers to translate natural language and extract data out of it. At the early stages of NLP, machine-learning algorithms and preset hand-written rules were used to make the "translation". However, today, more advanced techniques, based on neural network algorithms, e.g., deep neural networks (DNN), make NLP more efficient than ever. NLP has now gained a huge interest and is used in a multitude of ways. Many applications, such as in email spam detection, question answering machines and machine translation, chatbots, information extraction and more [6], use NLP. Thus, it has significant importance in our everyday life.

In this paper, we introduce an application for controlling a smart home remotely. In our application we obtain the sensor values and control the actuators through natural language. A user can give orders to the application through free-styled speech, which are then executed by the application. In addition, anyone can also use a chatbot in the main page of our application to interact with it via free text. For training our NLP system, we used the Dialogflow Application Programming Interface (API), an Application Programming Interface (API) from Google for conversational interfaces that uses machine-learning techniques in order to train itself. This way, even if the questions/commands sent by the user differ to the ones preset during training, the system is able to "understand" and respond accordingly. We name our application the IoT Agent. The most significant innovation it brings is that it is totally based on third party APIs and open source technologies, so it highlights how a new IoT application can be built today using a multi-tier architecture without any compromise in performance. The rest of the paper is structured as follows: In the next section, we discuss similar implementations in the field. In Section 3, we present the architecture of our Web application, along with details about any subsystems it comprises. In Section 4, we continue by explaining the main logic of the NLP system of our application. In the Section 5, we discuss how our microcontrollers work and cooperate with the whole app, and in Section 6 we describe the logic and implementation of our front end. Section 7 discusses some critical issues related to the performance of our application, and finally, Section 8 concludes the paper.

2. Related Work

It is difficult to find implementations like ours in the literature, especially using natural language processing in IoT deployments. Most such implementations for the Internet of Things offer control of home devices through the integrated NLP system of Android or iOS smartphones [7], or of simpler devices like Alexa by Amazon [8]. Many other implementations develop custom NLP systems for smart home automation with the whole project based on the NLP system design and its final implementation [9–11]. In the current project, the NLP system used is an external online service. Hence, the challenge is to efficiently integrate this service to our project. This represents a remarkable difference from the current literature.

An interesting approach was introduced in 2017, by Cyril Joe Baby et al. in [12]. A home automation system was implemented, with few similarities to the system presented in the current paper, and which managed to control remotely some home devices via an application. Natural language processing techniques were also used for the "training" of the system. In addition, they used a local Web application from within the home network and an integrated NLP system in a Raspberry

Pi, with all processing taking place within the home network. This could potentially be a disadvantage for home automation, since the user is not able to control their house remotely from outside the home. In our implementation, the user client is a Web application that every authenticated user can access in the form of a website. In [13], an implementation with a Web application is presented, but does not use external services for NLP, voice synthesis, voice recognition and data storage.

Two other implementations focused on the communication and implementation of a smart home automation system, using a building control standard, the KNX technology. The first used a standalone application running only on a PC with an integrated NLP system [14]. The second presented an implementation of an IBM cloud service, based on a MQTT (Message Queuing Telemetry Transport) broker, which managed to exchange messages between the devices and the user. It also integrated natural language processing with the help of a custom service called Engagement, which “interacted in the language of humans, and answered with context” [15]. Although the latter implementation used an external NLP service and other external services, the whole implementation focused on the services implementation and the communication between services. In our development the effort is on the efficient integration of third-party services and data processing.

Other similar works to our implementation are those using online cloud services for voice recognition [16] or the Google Web Speech API for voice recognition and the MQTT protocol for controlling and monitoring the smart home [17]. Generally, these implementations use decentralized approaches and take advantage of cloud services, while distributing the work effort across different machines. Like the current implementation, these approaches employ multiple third-party services integration, but do not use either a Web application for user actions and home monitoring, or an NLP system as we do.

The most significant innovation that the current implementation presents in comparison to the existing research is that it is totally based on third party APIs and open source technologies, so it is simple, modern and functional. It is a mash-up of third-party services that designates how a new IoT application can be built today using a multi-tier architecture. One tier is devoted to voice recognition via the Web Speech API, a second tier to NLP via the Dialogflow API, and a third to storage via the Firebase API. A final tier is devoted to MQTT communication via the Eclipse MQTT Broker API. To the best of our knowledge, it is the first work that attempts such an integration. Furthermore, our user agent is a Web application developed with an angular framework, so it runs totally on the client-side without the need for a server. Hence, all the processing takes place in the client app. Considering that the only servers we use run behind the cloud services of the third parties mentioned above, this makes our system highly scalable, so it can support many concurrent users. Apart from this, the user interface of our app makes the interaction with the system friendly and easy. In total, our implementation exploits the typical IoT infrastructures by providing an enhanced user experience and the usability of an underlying IoT system with the integration of NLP, voice recognition, MQTT and many other technologies.

3. System Architecture

Figure 1 illustrates the overall architecture of our ecosystem, comprised of various subparts. The user, via the IoT Agent, a Web application that hosts all the functionality, can control and monitor sensors and actuators, which are integrated into the smart home. Critical for our decision to develop the IoT Agent as a web application was the ability of the latter to run easily on all devices, so it can be potentially accessed from everywhere.

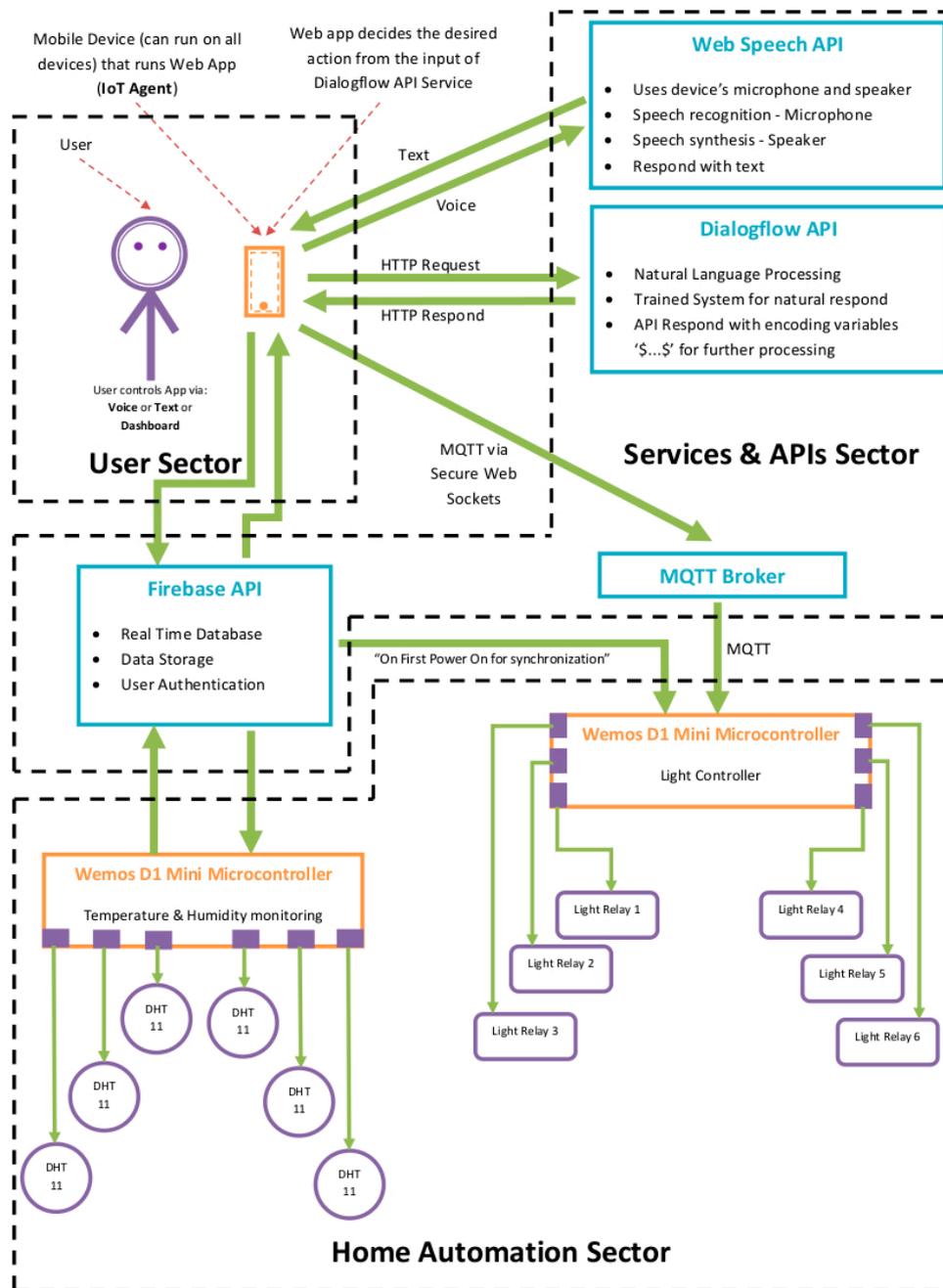


Figure 1. Main system architecture.

When the user is logged in to the IoT Agent app, the first thing he faces is the welcome page with the chatbot and, in the lower part of the page (i.e., the footer), a menu for navigation. Figure 2 illustrates the welcome page of the IoT Agent. From the first tab (chatbot), the user is offered two choices: either to type in the chatbot message field the command he wants, e.g., "Turn on the kitchen lights" or to hit the microphone button and speak the appropriate command with natural language. For implementing the voice feature of our system, we use the Web Speech API by W3C [18], which is supported currently by Chrome and Firefox. The Web Speech API defines a JavaScript API for both voice recognition and voice synthesis that enables web developers to use natural language as an input or output method for websites' components, such as forms or actions from the system. Generally, the API uses the default speech recognition system available on the device for speech recognition and synthesis. Most modern OSes incorporate such a speech system. In our application, voice recognition

“understands” the user’s voice commands and translates them to text, ready to be sent to the NLP system. Voice synthesis is responsible for the reproduction of the human voice in the user’s device when the application responds to the user [19]. So, if the microphone solution is chosen, a call is made to the Web Speech API and the voice is sent. The Web Speech API returns the sent voice message converted to text. Subsequently another call is made, this time to the Dialogflow API. To the latter is sent the text returned from the Web Speech API. The Dialogflow API, depending on the text that is sent, responds with the appropriate answers preset by us. The answers are then handled by the IoT Agent, so the appropriate commands are sent via MQTT to the microcontrollers located in the house for execution.

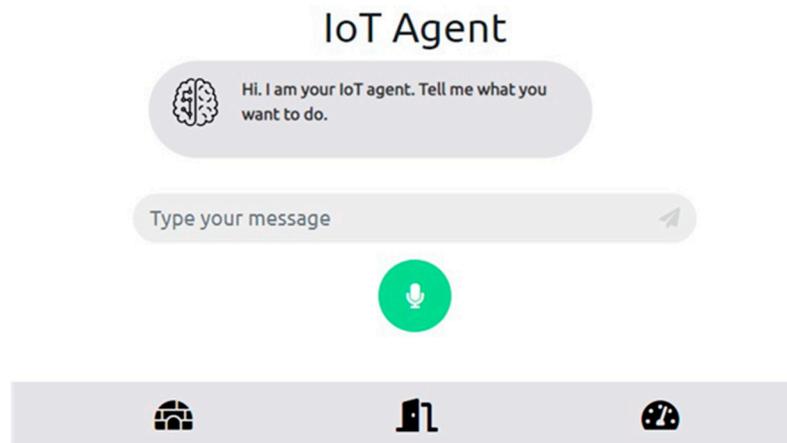


Figure 2. Welcome page of the Internet of Things (IoT) Agent.

For the smart home, we used as microcontrollers several Wemos D1 Mini V3.0.0 [20,21]. The reason we chose this microcontroller is its small size, its low power consumption and the fact it has a Wi-Fi chip integrated onboard. At the same time, it contains enough pins for use in our implementation. The Wemos D1 mini works with both 3.3 and 5 V DC. One microcontroller is placed in the electrical panel of the house for controlling the relays for the house lights. The relays used are the SRD-05VDC-SL-C, which operate at 5 V DC, and can switch on or off the 220 V AC circuits for lighting. The other microcontrollers are distributed in various rooms for gathering values from the temperature and humidity sensors. We use the DHT11 humidity and temperature sensors, which operate at 3.3 V DC.

The interconnection of actuators and sensors with microcontrollers is illustrated in Figure 3. The house we used for our real-life trial has six rooms. Hence, we used in total seven microcontrollers: one for the control of the lights and six for sensing the environmental conditions in the rooms. The microcontroller that is programmed to control the home lights in Figure 3 receives MQTT messages by the IoT Agent to handle their states. It is synchronized with the database the first time it boots and, after that, only uses the received MQTT messages for the whole process. As a result, it is working efficiently and very fast.

In order to hold and use the values of temperature and humidity from the DHT11 sensors, the light states and the user ID that is logged in to the system, a Firebase database was chosen [22]. Firebase is a real-time NoSQL database, provided by Google, used for rapid interaction. In Firebase, “data is stored as JSON objects and synchronized in real-time with every connected client”. This way, we can always have the temperature and humidity values stored in our database, in order to present them inside our Web application and to avoid downtimes. This means that even when hardware is offline, the database is always on and can send the latest stored data to the Web application. Firebase synchronizes with the microcontrollers that relate to the DHT11 sensors every 5 s, which keeps the results up to date. Moreover, the microcontrollers compare the present sensor values with the previous ones, so only updated measurements are sent to the database. As a result, the microcontroller does not overload the whole network with unnecessary traffic.

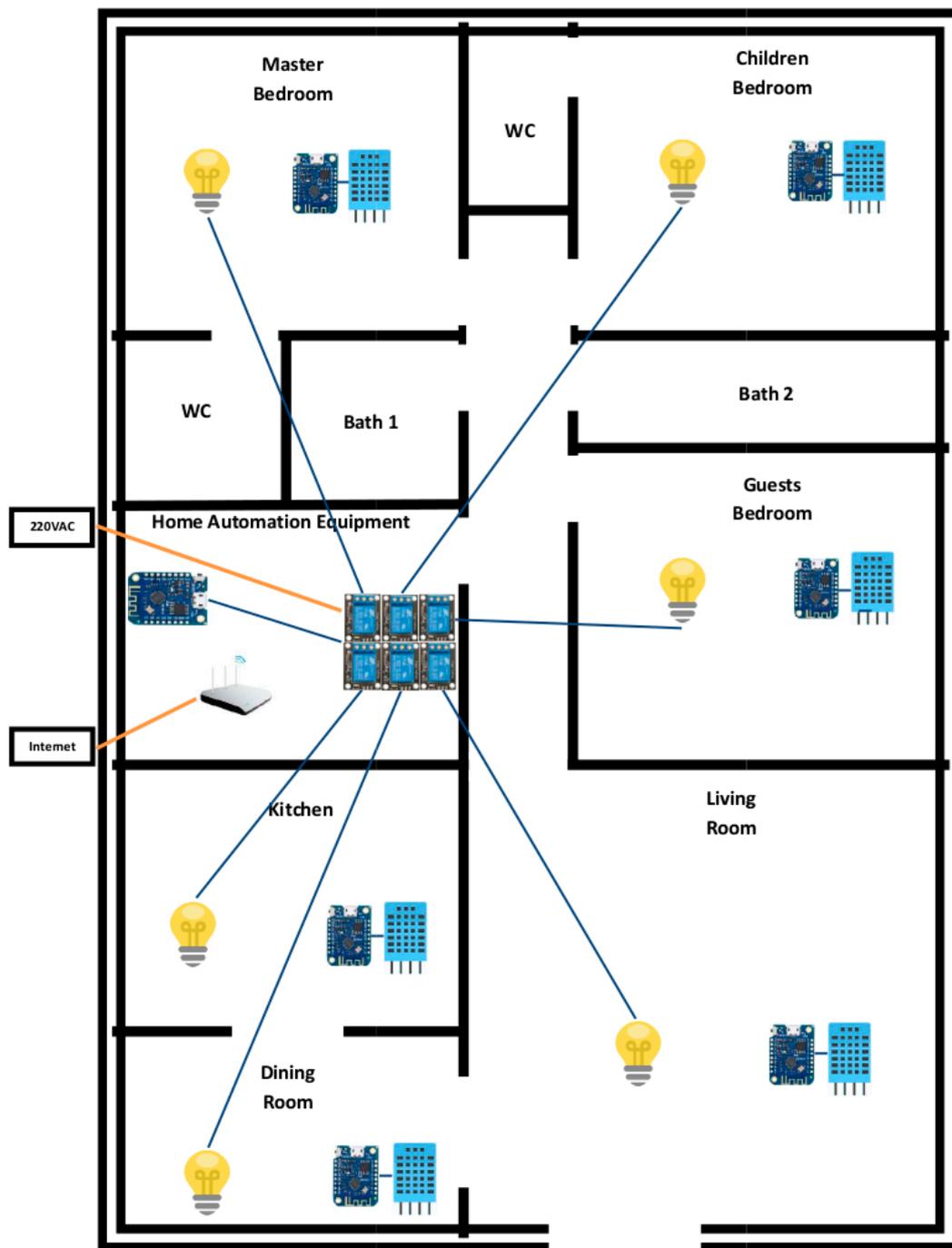


Figure 3. Generic interconnection of actuators and sensors with microcontrollers.

4. Natural Language Processing (NLP) System

Our system is further with the use of natural language processing. NLP is a subfield of artificial intelligence that uses computational techniques to understand, analyze and recreate human natural speech [23,24]. The IoT Agent uses NLP for its voice-chat feature, but also for its text-chat system [25]. Dialogflow is a very good solution for the convenient integration of NLP into an IoT system. The Dialogflow API, provided also by Google, is a natural language processing API, which uses machine-learning techniques for the training process [26]. With that in mind, even if the question/command is different from the presets, the API understands what is being asked by using keywords, and responds accordingly [26,27]. On the other hand, if the chat solution is chosen, the app again sends the text from the input to the Dialogflow API, which, using keywords from the trained

system, responds with the appropriate answer. In general, the Dialogflow API is the major component in our application, as it connects both the natural language feature and the chatbot feature. Moreover, the training of the Dialogflow API makes our system more efficient.

After the Dialogflow API responds with the answer in a text format, depending on the keyword returned, the IoT Agent automatically chooses the next move. If the returned keyword relates to the temperature or humidity, the system understands and fetches the value from the Firebase database. However, if the returned keyword relates to the lights, the application at first sends a message to the appropriate microcontroller via an MQTT broker and then updates the database. MQTT is a publish/subscribe protocol with “low network overhead and can be implemented on low power devices such as microcontrollers” [28]. It is specified in [29], is easy to use, open and lightweight, so is ideal for applications like ours [30]. Moreover, we used a broker via Secure WebSockets, in order to keep the levels of security as high as possible. The broker we used, provided by Eclipse, is called Mosquitto Broker, and is open source.

Below are presented some example training phrases we imported into the Dialogflow system, along with the desired responses. Tables 1 and 2 only present phrases and desired responses for humidity questions and light state actions, but the system has also been trained for temperature questions, light state questions/commands, general questions, microcontroller questions and navigation commands.

Table 1. Dialogflow humidity requests and responses.

Training Phrases (That System Expects)	Responses (That System Responds)
I would like to know the humidity of my bedroom.	
I want the humidity of kitchen.	
What is the humidity of kitchen?	<ul style="list-style-type: none"> • The humidity of \$Rooms is \$humidity\$ %.
Give me the humidity of kitchen.	<ul style="list-style-type: none"> • The \$Rooms has humidity \$humidity\$ %.
I want to know the humidity of my bedroom.	<ul style="list-style-type: none"> • The average humidity of \$Rooms is \$humidity\$ %.
Please, give me the humidity of guests’ bedroom.	<ul style="list-style-type: none"> • Be more specific. From which room?
Give me the humidity of all rooms.	<ul style="list-style-type: none"> • \$humidity\$ Celsius: \$Rooms.
The humidity of my bedroom.	
The humidity of all rooms.	

Table 2. Dialogflow lights requests and responses.

Training Phrases (That System Expects)	Responses (That System Responds)
Set the light on in kitchen.	
Light on in kitchen.	
Please, turn off the light of dining room.	
House lights on.	
House lights off.	<ul style="list-style-type: none"> • \$light\$Done. The lights of \$Rooms are \$Lights.
Turn off the lights of house.	<ul style="list-style-type: none"> • \$light\$Ok. All lights are \$Lights in \$Rooms.
Turn on the lights of house.	
Turn off the light of my bedroom.	
Turn on the light of my bedroom.	

Table 3 presents the room entities that the NLP system matches if the question/command includes one of them. For instance, if the command is “Turn off the light of my bedroom”, the system identifies the entity “my bedroom” and the response for that entity is “\$room1\$”. The Web application is programmed to identify the words inside “\$” and then to trigger the right action for the user’s question/command. As a result, when the user asks for room humidity, the system must always receive an input that contains “\$humidity\$” and the room entity properly parameterized.

Table 3. Dialogflow room entity examples.

\$room1\$	Kitchen, Kitchen, kitchen room, Kitchen room
\$room2\$	my bedroom, My bedroom, Your bedroom, My room, my room, master bedroom, master room, Master bedroom, my bedroom, My bedroom, Your bedroom, My room, my room, master bedroom, master room, Master bedroom
\$room3\$	child bedroom, Child bedroom, children room, Children room
\$room4\$	guests bedroom, guests bedroom, guests room, Guests room
\$room5\$	living room, Living room
\$room6\$	dining room, Dining room, eating room, Eating room
\$all-rooms\$	All rooms, house rooms, my home, my house, my place, house, home, all house rooms, Home, House, All rooms, My place, My home, My house

5. Microcontroller Implementation

This section underlines the main logic of the microcontrollers' code at the extreme edge of our system. The code for both microcontrollers is written in the Arduino IDE. We divided our code into two separate blocks of code, one for the temperature and humidity sensors, and one for the actuators and lights code, which we describe thoroughly in the paragraphs below.

Starting with the temperature/humidity sensor's code, in the loop() function we first check network and MQTT availability, respectively (for debugging purposes). After that, every 5 s we run a function called updateDB() that is responsible for checking and inserting every room's temperature and humidity into the Firebase database. If there are any differences to the later values, it refreshes the database with the new values. The main functions that update values are the setRoomsTemperature() and setRoomHumidity() functions. Figures 4–6 present the Arduino code written for the functions mentioned above.

We continue with the code for the actuators/lights. The function getCurrentLightStates() does what it describes. It connects to the Firebase database once on the setup and gets the state of the lights. This is critical for the synchronization of the lights.

Maybe the most important function of this block of code, is the callback() function. This changes the state of the lights depending on the MQTT messages published by the Web application. In the function the requested answer is in bytes. Figure 7 presents the latter functionality.

```

void updateDB() {
  int error = 0;
  for (int i = 0; i < 6; i++) {
    int chk = DHT.read11(dhtPin[i]);
    digitalWrite(LED_BUILTIN, HIGH);
    if (chk == DHTLIB_OK) {
      if ((abs(DHT.temperature - temperatureHistory[i]) > 1) && DHT.temperature >= 0 && DHT.temperature <= 60) {
        setRoomsTemperature(DHT.temperature, i);
      }
      if ((abs(DHT.humidity - humidityHistory[i]) > 3) && DHT.humidity >= 10 && DHT.humidity <= 100) {
        setRoomsHumidity(DHT.humidity, i);
      }
    } else if (chk == DHTLIB_ERROR_CHECKSUM) {
      error++;
    } else if (chk == DHTLIB_ERROR_TIMEOUT) {
      error++;
    }
  }
  if (error > 0) {
    client.publish(MQTTOutTopic, "02");
  }
}

```

Figure 4. Microcontroller updated() function.

```

void setRoomsTemperature(float temperature, int counter) {
  String temperatureId = "temperature/room";
  temperatureId += String(counter + 1);
  temperatureId += "/value";
  Firebase.setFloat(temperatureId, temperature);
  if (Firebase.failed()) {
    client.publish(MQTTOutTopic, "03");
    ESP.reset();
  }
  delay(500);
  temperatureHistory[counter] = temperature;
}

```

Figure 5. Microcontroller setRoomsTemperature() function.

```

void setRoomsHumidity(float humidity, int counter) {
  String humidityId = "humidity/room";
  humidityId += String(counter + 1);
  humidityId += "/value";
  Firebase.setFloat(humidityId, humidity);
  if (Firebase.failed()) {
    client.publish(MQTTOutTopic, "04");
    ESP.reset();
  }
  delay(500);
  humidityHistory[counter] = humidity;
}

```

Figure 6. Microcontroller setRoomHumidity() function.

```

void callback(char* topic, byte* payload, unsigned int length) {
  if (length == 2) {
    int state = 0;
    char c = (char)payload[0];
    int id = c - '0';
    if ((char)payload[1] == '1') {
      state = 1;
      digitalWrite(lightPin[id - 1], HIGH);
    } else {
      state = 0;
      digitalWrite(lightPin[id - 1], LOW);
    }
  } else if (length == 3) {
    if ((char)payload[0] == '0' && (char)payload[1] == '0' && (char)payload[2] == '1') {
      ESP.reset();
    }
  }
}

```

Figure 7. Microcontroller callback() function.

Table 4 shows the messages from the publisher and the expected results. The main characteristic of every message is that the first digit expresses the room id and the second the state of the light. For example, if the received number is 10, it means, for room1 turn the lights Off. In the same sense, 11 means, for room 1 turn the lights On.

Table 4. MQTT publisher responses depending on message.

10/11	Room1: OFF/ON
20/21	Room2: OFF/ON
30/31	Room3: OFF/ON
40/41	Room4: OFF/ON
50/51	Room5: OFF/ON
60/61	Room6: OFF/ON

6. Web Application Implementation

The main Web application is developed using the Angular framework. Angular is a framework also developed by Google for creating single-page applications that can be used for web, desktop or even mobile. The main characteristic of a Web application that is developed with the Angular framework is that it is a client-side web application. The development does not need the use of a web server, including the PHP language, to execute commands for manipulating the database, for user authentication, or other commands that a web server traditionally executes. In addition, the fact that the database and the main application framework are provided from the same manufacturer makes our application even more reliable. In Figure 8, the case function for handling every expected response of the NLP system is presented.

```

caseHandling(input) {
  if (input.includes('$temperature$')) {
    this.temperatureMeasuring(input);
  } else if (input.includes('$humidity$')) {
    this.humidityMeasuring(input);
  } else if (input.includes('$light$')) {
    this.lightHandling(input);
  } else if (input.includes('$light-state$')) {
    this.lightMeasuring(input);
  } else if (input.includes('$exit$')) {
    this.speakAndNotify('Bye!');
    this.authService.doLogout();
    this.ngZone.run(() => this.router.navigate(['/login'])).then();
  } else if (input === '') {
    this.speakAndNotify('Something went wrong. Try again.');
```

Figure 8. The IoT Agent caseHandling() method.

Overall, the IoT Agent is an application that gives the user the capability to control a smart home, but most of all, it is a Web application that runs on the client side. It uses technologies like Ajax and JavaScript to achieve this functionality. Ajax and JavaScript provide a more interactive approach to our system while keeping it running with good performance [31]. A user, after a successful log in (Figure 9), can interact with the chatbot with two different ways (Figure 2). The first is to type the command or a question and then press the send button. The second is to press the microphone button, then give a voice command or a question, and again press the button to stop recording. In both circumstances, the IoT Agent responds appropriately with text and voice through speakers.

Figure 9. The IoT Agent login screen.

A third option available through the IoT Agent application is the dashboard environment. The Dashboard is an efficient way to control completely the smart home, through toggle buttons for handling the lights or through a temperature and humidity panel for monitoring ambient conditions in

every room of the home. Moreover, there are extra settings for the user and the microcontrollers. Below, we present the user interfaces for the Chatbot tab (Figure 10) and the main Dashboard (Figure 11).

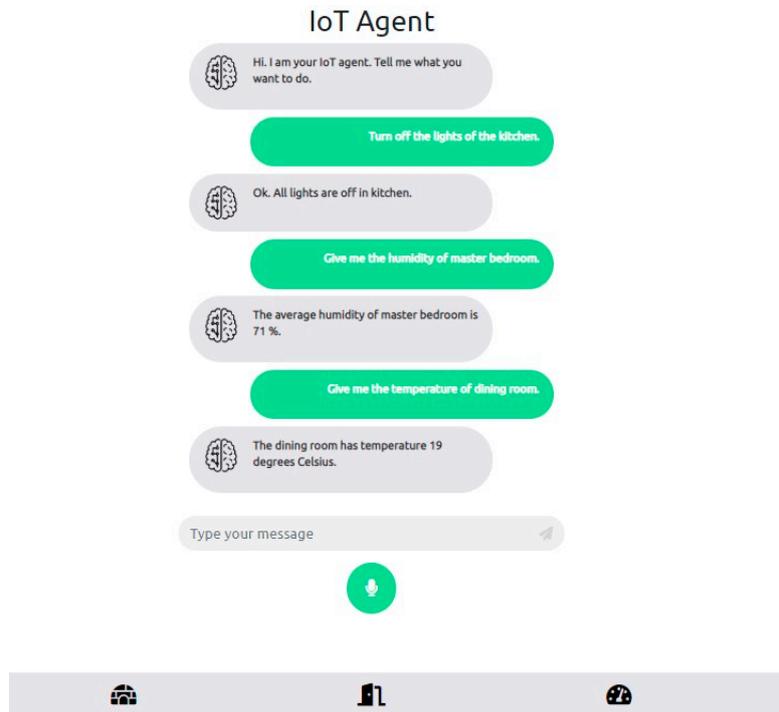


Figure 10. The IoT Agent Chatbot environment.

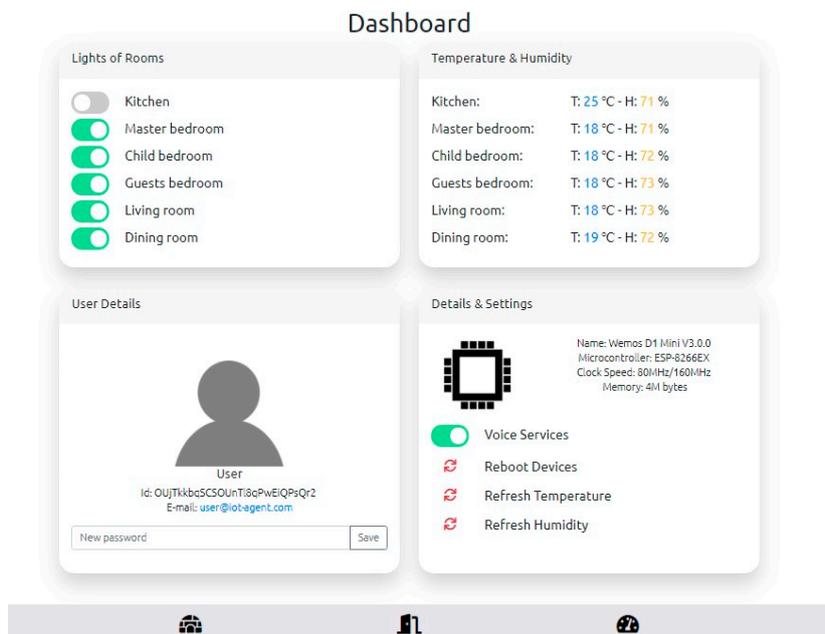


Figure 11. The IoT Agent Dashboard.

7. Discussion

As mentioned earlier, our application utilizes a multi-tier architecture integrating several available services into one functional and modern mash-up for smart home control. In this context, it is useful to evaluate the performance of our application to determine if such mash-ups can be useful in the IoT world. Hence, we attempted to evaluate the response time it takes from the moment a user executes

a command or asks a question via the User Interface (GUI) to the moment the system successfully responds. Moreover, we attempted to measure how this time is distributed among the three major tasks comprising an end-to-end session. These comprise the NLP engine response time, the processing time in the Web application and the microcontroller’s response time. We also attempted to separately measure the time it takes for the Web Speech API to respond but its response was so instant that it was impossible to measure any such latency. This is since the Web Speech API exploits the underlying voice engine of the mobile device. Hence, part of this procedure is executed locally in the device. For the evaluation tests, we measured the response time for the most representative operations in our system, such as handling the home’s lights or the receipt of temperature/humidity measurements. In addition, due to the unpredictable nature of mobile data networks, we emulated the network bandwidth of the interconnecting links so that representative network throughputs could be considered and realistic results received. Table 5 includes the evaluated test commands.

Table 5. Sample of test commands.

Id	Test Commands
C. 1	Turn off the lights of dining room.
C. 2	Turn on the lights of dining room.
C. 3	Give me the temperature of the dining room.
C. 4	Give me the humidity of the dining room.
C. 5	Turn off the house lights.
C. 6	Turn on the house lights.
C. 7	Give me the temperature of the house.
C. 8	Give me the humidity of the house.
C. 9	Give me information about microcontrollers.
C. 10	How are you?

Table 6 includes the response time from the NLP system (Dialogflow API) for each of the sample test commands of Table 5 under three different network throughputs. This measurement includes the time it takes from the moment a user presses the button in the IoT Agent app to send a text command, or presses the stop recording button to send a voice command, to the moment the app receives the final response from the NLP engine. So, these measurements also include the time required by the Web Speech API to convert voice-to-text. It is obvious from the results that the response time is severely affected by the available network bandwidth since it includes communication with a third-party server. Hence, the faster the access network, the lower the response time. Figure 12 illustrates the results.

Table 6. Response time of the natural language processing (NLP) engine under different access networks.

C./B.	ADSL (D.:3.93 Mbps/U.:0.57 Mbps)	Fast 3G (D.:1.51 Mbps/U.:0.45 Mbps)	Slow 3G (D.:0.36 Mbps/U.:0.26 Mbps)
C. 1	274 ms	570 ms	2001 ms
C. 2	245 ms	570 ms	2001 ms
C. 3	283 ms	572 ms	2002 ms
C. 4	289 ms	570 ms	2001 ms
C. 5	228 ms	570 ms	2003 ms
C. 6	227 ms	571 ms	2003 ms
C. 7	270 ms	572 ms	2002 ms
C. 8	273 ms	571 ms	2002 ms
C. 9	230 ms	573 ms	2002 ms
C. 10	442 ms	574 ms	2001 ms

Table 7 shows the data processing and decision-making time of the IoT Agent app for each of the sample test commands of Table 5 under three different network bandwidths. This measurement includes the time from the moment the app receives the response from the NLP engine to the moment

it sends a command to microcontrollers for execution or to the GUI for illustration. For instance, the commands that concern light handling include analysis of the input data from the NLP engine and decision making before sending MQTT messages to the microcontrollers. Similarly, the commands that concern sensor monitoring, apart from analysis of the NLP input, include retrieval of data from the local copy of Firebase. As was expected, the results reveal that this measurement is totally independent from the available network bandwidth, since data processing and decision-making takes place on the client-side. Figure 13 illustrates the results.

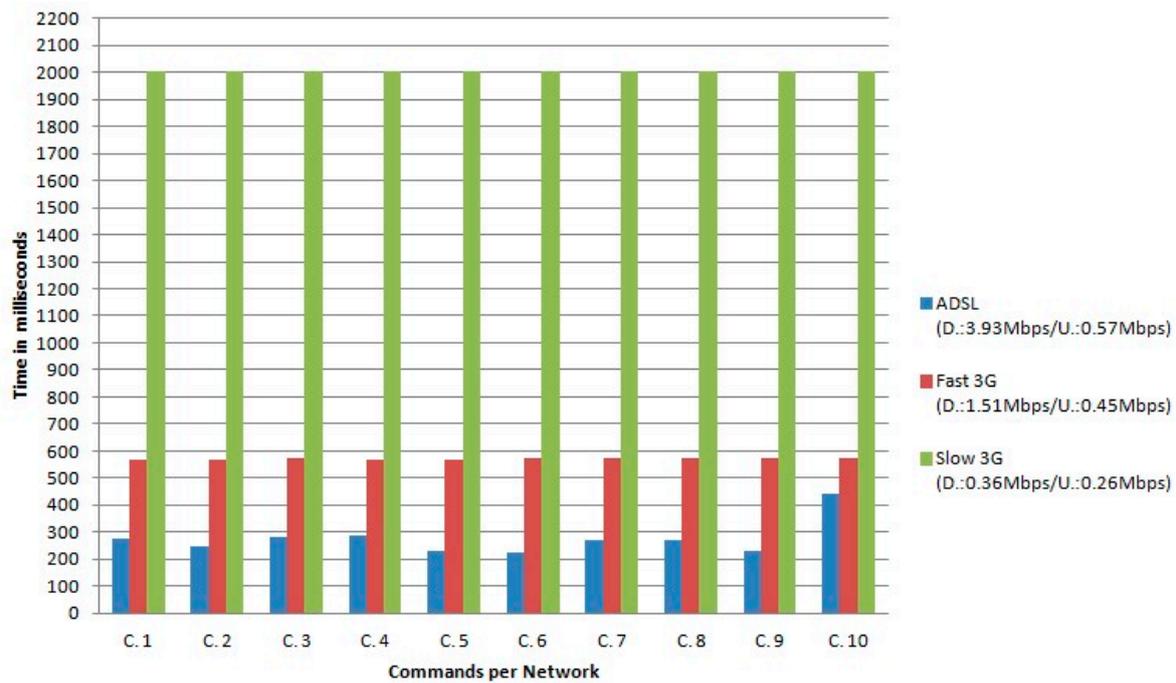


Figure 12. Response time of the NLP engine per network bandwidth.

Table 7. Data processing and decision-making time of the IoT Agent under different access networks.

C./B.	ADSL (D.:3.93 Mbps/U.:0.57 Mbps)	Fast 3G (D.:1.51 Mbps/U.:0.45 Mbps)	Slow 3G (D.:0.36 Mbps/U.:0.26 Mbps)
C. 1	4 ms	7 ms	5 ms
C. 2	2 ms	2 ms	2 ms
C. 3	1 ms	1 ms	1 ms
C. 4	1 ms	1 ms	1 ms
C. 5	4 ms	4 ms	6 ms
C. 6	3 ms	3 ms	7 ms
C. 7	1 ms	1 ms	1 ms
C. 8	1 ms	1 ms	1 ms
C. 9	0.5 ms	0.5 ms	0.5 ms
C. 10	0.5 ms	0.5 ms	0.5 ms

Table 8 includes the response time of the microcontroller system for the sample test commands, again considering different access networks for the IoT Agent. This measurement includes the time from the moment the IoT Agent app sends a command to the microcontrollers via MQTT to the moment they execute it. As was expected, the faster the access network of the IoT Agent, the lower the response time. We assumed microcontrollers are connected to the home’s WLAN network. This involves MQTT communication of the IoT Agent with the microcontrollers via the public MQTT broker we use. Furthermore, Table 8 includes only the sample commands 1, 2, 5 and 6, which concern communication with the microcontrollers, since the commands 3, 4, 7, 8, 9 and 10 concern sensor and

data reading from the data base, which take place locally via the local copy of Firebase. The required time for the latter actions has been already measured in Table 7. Figure 14 illustrates the results.

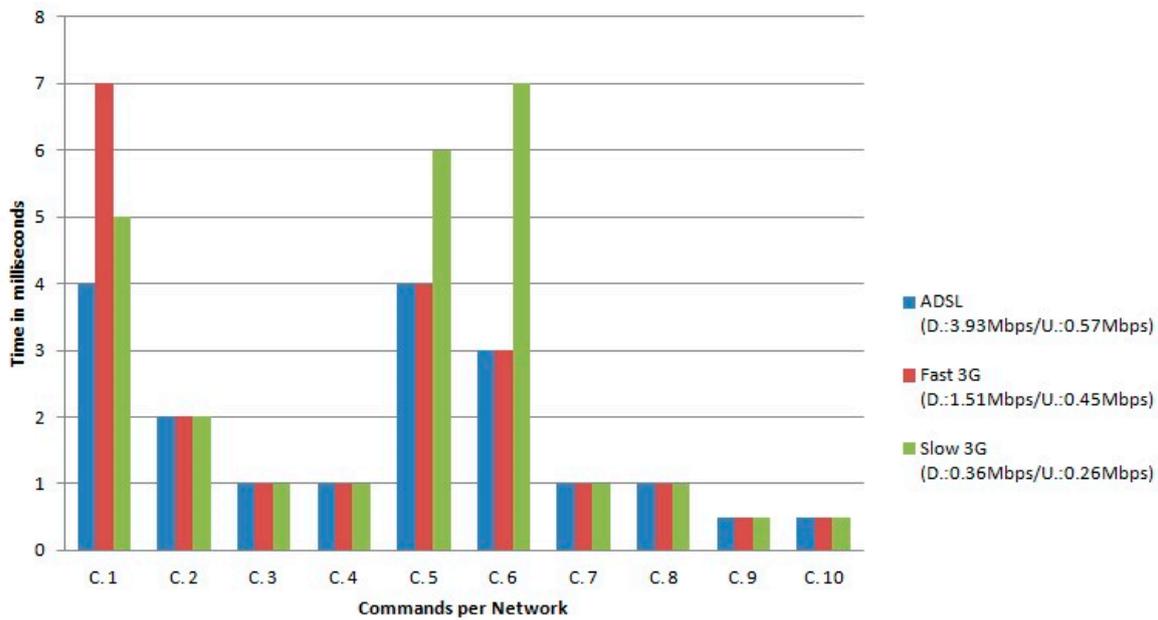


Figure 13. Data processing and decision-making time of the IoT Agent.

Table 8. Response time of microcontrollers under different access networks.

C./B.	ADSL (D.:3.93 Mbps/U.:0.57 Mbps)	Fast 3G (D.:1.51 Mbps/U.:0.45 Mbps)	Slow 3G (D.:0.36 Mbps/U.:0.26 Mbps)
C. 1	20 ms	70 ms	730 ms
C. 2	30 ms	100 ms	720 ms
C. 5	40 ms	50 ms	156 ms
C. 6	30 ms	30 ms	720 ms

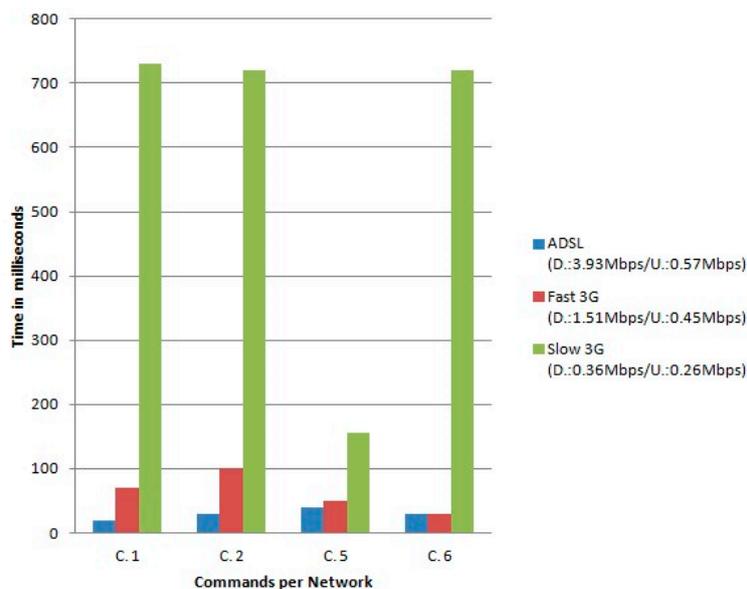


Figure 14. Response time of microcontrollers under different access networks.

Table 9 and Figure 15 summarize the above results, so the end-to-end distribution of time in our system for execution of the sample commands is illustrated. For illustration purposes we have chosen

the use case of an IoT Agent user that is connected to the Internet via a Slow 3G Network (D.:0.36 Mbps/U.:0.26 Mbps). We notice that the NLP response time accounts for almost 70% of the total time, while the communication with microcontrollers accounts for the remaining 30%. However, even in this worst-case scenario, seen from the throughput point of view, the total required time is not significant. This underlines that our selection of technologies is viable and can offer a feasible solution for efficient and low-cost IoT deployments.

Table 9. Total response time.

C./S.	NLP Response Time	Data Processing Time	Microcontroller Response Time
C. 1	2001 ms	5 ms	730 ms
C. 2	2001 ms	2 ms	720 ms
C. 3	2002 ms	1 ms	0 ms
C. 4	2001 ms	1 ms	0 ms
C. 5	2003 ms	6 ms	156 ms
C. 6	2003 ms	7 ms	720 ms
C. 7	2002 ms	1 ms	0 ms
C. 8	2002 ms	1 ms	0 ms
C. 9	2002 ms	0.5 ms	0 ms
C. 10	2001 ms	0.5 ms	0 ms

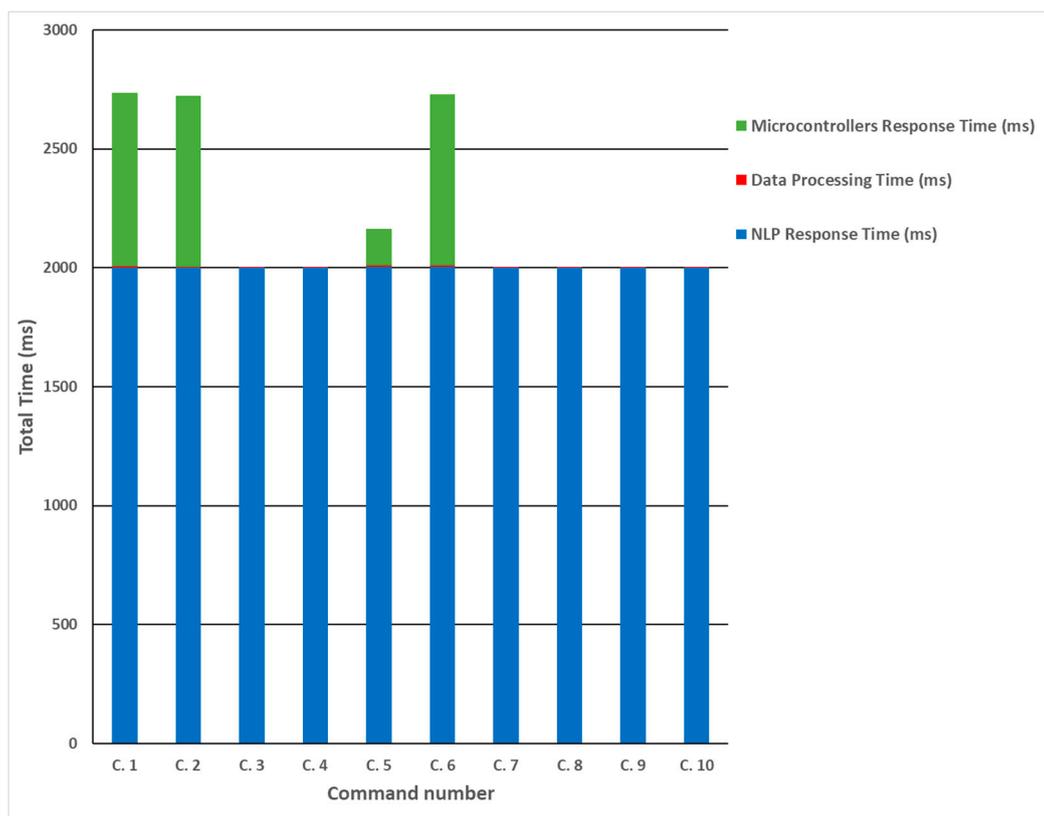


Figure 15. End-to-end response time for the sample commands.

As described in Section 3, we tested the viability and the performance of our application in a real-life testbed that was set up in an apartment. The testbed included the use of our application by the residents of the house for one week, each from his device. The testbed did not reveal any incompatibility of the application, which operated with the expected consistency. Despite the ambient noise of the house and the electromagnetic interference caused by other house equipment, the microcontrollers and the sensors operated very well. Apart from this testbed, we also demonstrated our application at a

digital festival of school Information Technology (IT) creativity, organized since 2008 in Greece, with the co-participation of local academic institutes. During this demonstration, students were offered the chance to interact in natural language with our application hosted on a tablet, in order to switch on/off some electric bulbs located in the demo area, or to monitor the ambient conditions in the demo room. Despite the extreme ambient noise in the demo area, the voice recognition system operated with consistency, provided that the user clearly and loudly articulated the voice commands. Furthermore, we noticed that the voice engine operated better when the students spoke with an English accent. In general, the real tests we have performed with our application so far show that our design is robust without revealing any remarkable issues.

8. Conclusions

The Internet of Things field undoubtedly offers a revolution in the way we interact with everyday objects and devices, still has a lot to offer, and is changing the way we interact with everyday devices. Even though our implementation is still in its early stages, we strongly believe that it can contribute in this direction and offer even more possibilities. Furthermore, with the use of natural language as an input method, usability is enhanced. People with visual or moving disabilities, or elderly people who are not so accustomed to technology, can be assisted to make every-day actions more easily with the use of our IoT Agent. Furthermore, natural language can also help people with special needs in the matter of safety.

Summarizing, the most significant innovations that the current implementation presents, in comparison to the literature, is that it is totally based on third party APIs and open source technologies, so it is simple, modern and functional. It is a mash-up of third-party services that highlights how a new IoT application can be built today using a multi-tier architecture. One tier is devoted to voice recognition via the Web Speech API, a second tier to NLP via the Dialogflow API, and a third to storage via the Firebase API. A final tier is devoted to MQTT communication via the Eclipse MQTT Broker API. To the best of our knowledge, it is the first work that attempts such an integration. Such a multitier architecture simplifies the development of IoT applications, enabling rapid prototyping, reduced development cycles and faster time-to-market. Furthermore, our user agent is a web application developed with the Angular framework, so it runs totally on the client-side without the need for a server. Hence, all processing takes place in the client app. Considering that the only servers we use run behind the cloud services of the third parties mentioned above, this makes our system highly scalable, without any effort from us, so it can support many concurrent users. Apart from this, the user interface of our app makes the interaction with the system friendly and easy. Overall, our implementation exploits the typical IoT infrastructures by providing an enhanced user experience and usability of the underlying IoT system with the integration of NLP, voice recognition, MQTT and many other technologies.

As for the expansion of our application, we plan to add a signup feature for new users. Additionally, more rooms and sensors can be added through an automatic procedure, using voice commands or through the control panel. Moreover, the IoT Agent Web application can be modified to a hybrid mobile application so is able to work natively for every single device without dependency on a web browser. Lastly, we can use more training phrases in the Dialogflow API to make the system as efficient and user-friendly as possible.

Author Contributions: Conceptualization, G.A., S.P., A.F., E.M. and K.V.; Methodology, G.A., S.P., A.F., E.M. and K.V.; Resources, G.A., S.P., E.M. and K.V.; Software, G.A. and A.F.; Supervision, E.M. and K.V.; Validation, G.A. and S.P.; Writing—original draft, G.A. and A.F.; Writing—review & editing, S.P.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ashton, K. That “Internet of Things” thing. *RFID J.* **2009**, *22*, 97–114.
2. Wortmann, F.; Flüchter, K. Internet of Things. *Bus. Inf. Syst. Eng.* **2015**, *57*, 221–224. [[CrossRef](#)]
3. Xiaoyi, C. The Internet of Things. In *Ethical Ripples of Creativity and Innovation*; Palgrave Macmillan: London, UK, 2016; pp. 61–68.
4. Rizvi, S.; Sohail, I.; Saleem, M.M.; Irtaza, A.; Zafar, M.; Syed, M. A Smart Home Appliances Power Management System for Handicapped, Elder and Blind People. In Proceedings of the 4th International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, Malaysia, 13–14 August 2018.
5. Verspoor, K.M.; Cohen, K.B. Natural Language Processing. In *Target Hub—Encyclopedia of Systems Biology*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 1495–1498.
6. Khurana, D.; Koli, A.; Khatter, K.; Singh, S. Natural Language Processing: Sate of The Art, Current Trends and Challenges. **2017**, arXiv:1708.05148.
7. Tharaniya Soundhari, M.; Brilly Sangeetha, S. Intelligent Interface Based Speech Recognition for Home Automation using Android Application. In Proceedings of the 2nd International Conference on Innovations in Information Embedded and Communication Systems (ICIIECS’15), Coimbatore, India, 19–20 March 2015.
8. Milivojša, S.; Ivanović, S.; Erić, T.; Antić, M.; Smiljković, N. Implementation of Voice Control Interface for Smart Home Automation System. In Proceedings of the 2017 IEEE 7th International Conference on Consumer Electronics, Berlin, Germany, 3–6 September 2017.
9. Han, Y.; Hyun, J.; Jeong, T.; Yoo, J.-H.; James, W.-K.H. A Smart Home Control System based on Context and Human Speech. In Proceedings of the 2016 18th International Conference on Advanced Communication Technology (ICACT), Pyeong Chang, Korea, 31 January–3 February 2016.
10. Jasmin Rani, P.; Bakthakumar, J.; Kumaar, B.P.; Kumaar, U.P.; Kumar, S. Voice Controlled Home Automation System using Natural Language Processing (NLP) and Internet of Things (IoT). In Proceedings of the 2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM), Chennai, India, 23–24 March 2017.
11. Jivani, F.D.; Manohar, M.; Radha, S. A Voice Controlled Smart Home Solution with a Centralized Management Framework Implemented Using AI and NLP. In Proceedings of the 2018 IEEE International Conference on Current Trends toward Converging Technologies, Coimbatore, India, 1–3 March 2018.
12. Cyril, J.; Faizan, K.; Swathi, J. Home Automation using IoT and a Chatbot using Natural Language Processing. In Proceedings of the Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, India, 21–22 April 2017.
13. Baby, C.J.; Munshi, N.; Malik, A.; Dogra, K.; Rajesh, R. Home Automation using Web Application and Speech Recognition. In Proceedings of the 2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS), Vellore, India, 10–12 August 2017.
14. Vanus, J.; Smolon, M.; Koziorek, J.; Martinek, R. Voice Control of Technical Functions in Smart Home with KNX Technology. *Lect. Notes Electr. Eng.* **2015**, *330*, 455–462.
15. Petnik, J.; Vanus, J. Design of Smart Home Implementation within IoT with Natural Language Interface. In Proceedings of the Conference on Programmable Devices and Embedded Systems (PDeS 2018), Ostrava, Czech Republic, 23–25 May 2018.
16. Erić, T.; Ivanović, S.; Milivojša, S.; Matic, M.; Smiljković, N. Voice Control for Smart Home Automation: Evaluation of Approaches and Possible Architectures. In Proceedings of the 2017 IEEE 7th International Conference on Consumer Electronics, Berlin, Germany, 3–6 September 2017.
17. Nan, E.; Radosavac, U.; Matic, M.; Stefanović, I.; Papp, I.; Antić, M. One Solution for Voice Enabled Smart Home Automation System. In Proceedings of the 2017 IEEE 7th International Conference on Consumer Electronics, Berlin, Germany, 3–6 September 2017.
18. Web Speech API. W3C. Available online: <https://w3c.github.io/speech-api/> (accessed on 21 March 2019).
19. MDN web docs. Mozilla. Available online: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API (accessed on 21 March 2019).
20. D1 mini v3.0.0(Retired). Wemos Electronics. Available online: https://wiki.wemos.cc/products:retired:d1_mini_v3.0.0 (accessed on 21 March 2019).

21. Kodali, R.K.; Sahu, A. An IoT based Weather Information Prototype Using WeMos. In Proceedings of the 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, India, 14–17 December 2016.
22. Firebase Documentation. Google Inc. Available online: <https://firebase.google.com/docs/database/> (accessed on 21 March 2019).
23. Hirschberg, J.; Manning, C.D. Advances in natural language processing. *Science* **2015**, *349*, 261–266. [[CrossRef](#)] [[PubMed](#)]
24. Ramos, C.; Augusto, J.C.; Shapiro, D. Ambient Intelligence—the Next Step for Artificial Intelligence. *IEEE Intell. Syst.* **2008**, *23*, 15–18. [[CrossRef](#)]
25. Rosruen, N.; Samanchuen, T. Chatbot Utilization for Medical Consultant System. In Proceedings of the 3rd Technology Innovation Management and Engineering Science International Conference (TIMES-iCON), Bangkok, Thailand, 12–14 December 2018.
26. Dialogflow API. Google Inc. Available online: <https://dialogflow.com/> (accessed on 21 March 2019).
27. Canonico, M.; de Russis, L. A Comparison and Critique of Natural Language Understanding Tools. In Proceedings of the CLOUD COMPUTING 2018: The Ninth International Conference on Cloud Computing, GRIDs, and Virtualization, Barcelona, Spain, 18–22 February 2018.
28. Light, A.R. Mosquitto: server and client implementation of the MQTT protocol. *J. Open Source Softw.* **2017**, *2*, 265. [[CrossRef](#)]
29. ISO/IEC 20922:2016—Information technology—Message Queuing Telemetry Transport (MQTT) v3.1.1. International Organization for Standardization. Available online: <https://www.iso.org/standard/69466.html> (accessed on 21 March 2019).
30. Tang, K.; Wang, Y.; Liu, H.; Sheng, Y.; Wang, X.; Wei, Z. Design and Implementation of Push Notification System Based on the MQTT Protocol. In Proceedings of the 2013 International Conference on Information Science and Computer Applications, Hu Nan, China, 8–9 November 2013.
31. Woychowsky, E. *AJAX: Creating Web Pages with Asynchronous Javascript and XML*; Prentice Hall: Upper Saddle River, NJ, USA, 2006.



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).