Smart Component Organizer using Google Assistant

Swathi P
Dept. of Electronics and Communication
GSSS Institute of Engineering and Technology for Women
Mysore, Karnataka, India

Sanjana G S
Dept. of Electronics and Communication
GSSS Institute of Engineering and Technology for Women
Mysore, Karnataka, India

Swathi R
Dept. of Electronics and Communication
GSSS Institute of Engineering and Technology for Women
Mysore, Karnataka, India

Dr. Rajendra R. Patil
Professor and Head,
Dept. of Electronics and Communication
GSSS Institute of Engineering and Technology for Women
Mysore, Karnataka, India

Priyanka P
Assistant Professor, Dept. of Electronics and Communication
GSSS Institute of Engineering and Technology for Women
Mysore, Karnataka, India

Abstract— With the rapid development of electronics technology, there are many types of electronic components than ever before. This leads to many problems in the management of components. Some of the most commonly faced problems are unnecessary repeat purchases, time-consuming storage, poor management of existing parts and mistakes that could have been avoided. Management of such a variety of components has been a typical issue among various organizations. Huge time and money are invested in the management of these items. However, there are many instances where these resources are wasted due to the lack of proper management system. Consider the following scenarios: 1.When the item is available, but its location is unknown. 2.When an item is available, but it cannot be searched due to time constraint. 3.When an item is available, but still the operator places an order. In the above listed scenarios, there is no full utilization of the available items. The proposed framework aims to design a smart component organizer which helps the operator to organize and manage a cluster of items using voice commands, enable hands free-searching and maintain a database of all the components, which keeps track of name, location, available quantity and search tags of each component.

Keywords— Google Assistant, Particle Photon, Microsoft Azure, IFTTT, SQL database, LED Strips, Components.

I. INTRODUCTION

The era of electronics began with the invention of the transistor in 1947 and silicon-based semiconductor technology. Seven decades later, we are surrounded by electronic devices and, much as we try to deny it, we rely on them in our everyday lives. Most of the electronic instruments and equipment would not have been possible without advances in electronics. In particular, the new, high volume applications has increased the pace of innovation in the development of a wide variety of components and devices.

There are a variety of container options to store electronic components as per the user requirement. Moulded plastic boxes that can be side locked and stacked either vertically or horizontally are available. Each compartment with a label for easy identification. The electronic components need to be sorted and stored in these compartments manually. A big problem is that the user cannot locate the required components immediately. Other problems include repeat purchases of existing components, time-consuming storage and poor management of the components.

In this paper, we propose a system to organize and digitally catalog each component to maintain a record of the available components and to mitigate the delay of searching the desired component. The objectives of the intended system are: to manage a cluster of components using voice commands, to enable hands-free searching, to eliminate duplication in ordering stocks and to maintain a record of each component.

REQUIREMENTS

The proposed project is an IoT - based component organizer which embraces the following elements: Google assistant, IFTTT (If This Then That), Particle Photon, Microsoft Azure, Azure SQL Database, 3.3 to 5V logic level converter and individually addressable LED strips. The block diagram of the intended project is as shown in figure 1.

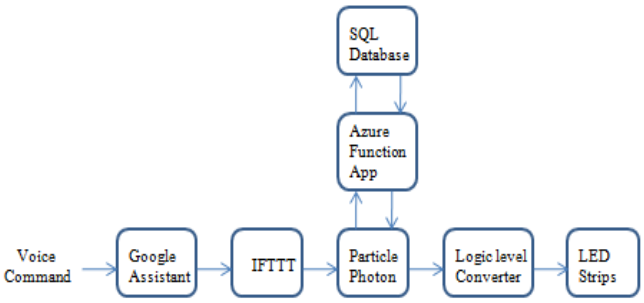

Figure 1: Block Diagram

A. *Google Assistant*

Google Assistant is an artificial intelligence-powered virtual assistant which is primarily available on mobile and smart home devices. Google Assistant can engage in two-way conversations. Google smart home lets users control the connected devices through the Google Home app[1]. In the proposed project, the operator gives voice commands to Google Assistant and those voice commands are converted to text. This text is used to trigger IFTTT.

B. *IFTTT*

If This Then That, also known as IFTTT, is a freeware web-based service that creates chains of simple conditional statements, called applets. In the proposed project, we have used IFTTT to integrate Google Home and Particle Photon[8]. We have created various applets which will enable the operator to search for the required components using suitable voice commands.

C. *Particle Photon*

Particle Photon is a Wi-Fi capable microcontroller, which is programmable Arduino style. Particle combines a powerful ARM Cortex M3 micro-controller with a Broadcom Wi-Fi chip in a minute module called the PØ (P-zero). Particle adds a 3.3VDC SMPS power supply, RF and user interface components to the PØ on a small single-sided PCB called the Photon[7]. The design is open source. In the proposed project Photon is used to publish an event to Azure Function App and to turn on the LED strip indicating the location of the desired component.

D. *Microsoft Azure Function*

Azure Function is a serverless compute service that lets the user run event-triggered code without having to explicitly provision or manage infrastructure[3]. Azure Functions allows the user to run small pieces of code (called "functions") without worrying about application infrastructure. With Azure Functions, the cloud infrastructure provides all the up-to-date servers the user needs to keep the application running at scale. A function is "triggered" by a specific type of event. Supported triggers include responding to changes in data, responding to messages, or as the result of an HTTP request.

E. *Azure SQL Database*

SQL Database is a cloud-computing database service (Database as a Service), that is offered by Microsoft Azure Platform which helps to host and use a relational SQL database in the cloud without requiring any hardware or software installation. Also, it provides various advanced features to its users and some of them are: long-term backup retention, geo-replication, automatic tuning, high-availability, scaling database resources and automated backups[3][4].

F. *Logic Level Converter*

A level shifter in digital electronics, also called logic-level shifter or voltage level translation, is a circuit used to translate signals from one logic level or voltage domain to another, allowing compatibility between ICs with different voltage requirements, such as CMOS(Complementary Metal Oxide Semiconductor) and TTL(Transistor-Transistor Logic). In the proposed project, we use the level shifter to convert 3.3V signal received from the microcontroller to 5V for driving the LED strip.

II. IMPLEMENTATION

A. *Circuit Diagram*

The circuit consists of Particle Photon, 3.3V to 5V logic level converter, 5V 4A power supply, terminal blocks, LEDs, relay, capacitors, resistors and diodes. A 5V 4A power supply is fed to the circuit with the help of a terminal block. The LED strips are connected to the four terminal blocks on the right side of the schematic. The pin 2 of the terminal blocks is connected to the data wire of the LED strip. The VIN pin of Particle Photon is connected to 5V, 4A supply and the GND pin is connected to ground. The D3 pin of Particle Photon is connected to LV(3.3V) of the logic level converter. The logic level converter considered is of a single channel. It is composed of 2N7000 N-channel E-MOSFET and 10k resistors. The 5V signal from the logic level converter drives the LED strip. The digital input/output pin D3 is used to publish the location of the component by turning on the LEDs of the corresponding tray. The relay is used to turn on/off the LED strip display. Figure 2 shows the schematic diagram of the project. Figure 3 shows the designed printed circuit board.

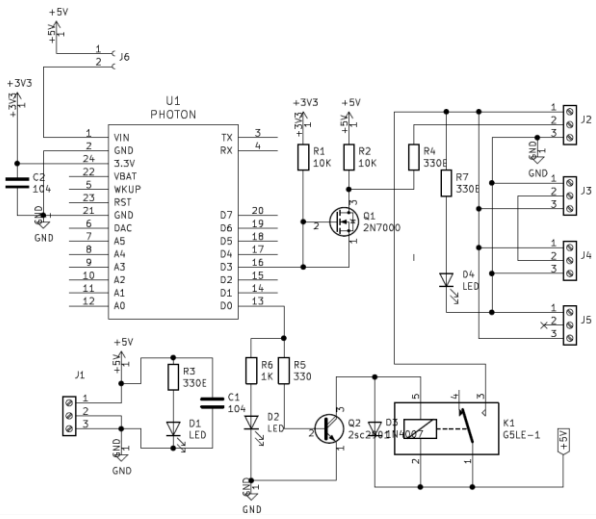

Figure 2: Schematic Diagram

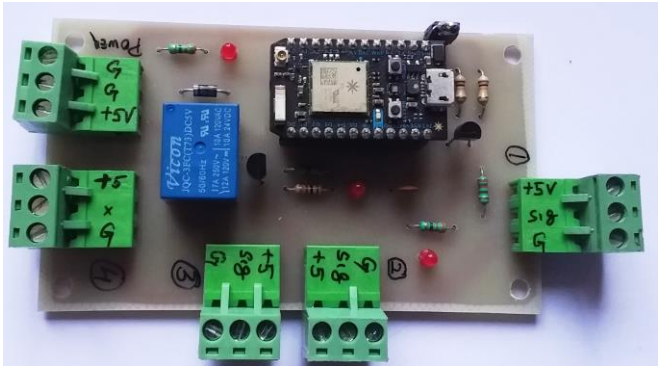

Figure 3: Designed PCB.

B. *Flow Diagram*

In this project, Google Assistant is used to receive the voice command regarding the required component from the operator. Google Assistant converts this voice command into text. This text then triggers IFTTT. IFTTT is an online platform which enables different applications to communicate with each other. IFTTT publishes an event via webhook to Particle Photon. Webhook is used to send or receive data from Particle devices and applications. Particle Photon sends the JSON payload via webhook to Microsoft Azure Function App. The Function App deserializes the JSON payload, generates a query and then queries the database. The database then sends the results of querying back to the Function App. The Function App sends results to Particle Photon via webhook. Finally, Photon publishes the location of the required component by turning on the LEDs of the storage unit containing it. Figure 4 shows the sequence of operation of the proposed project.

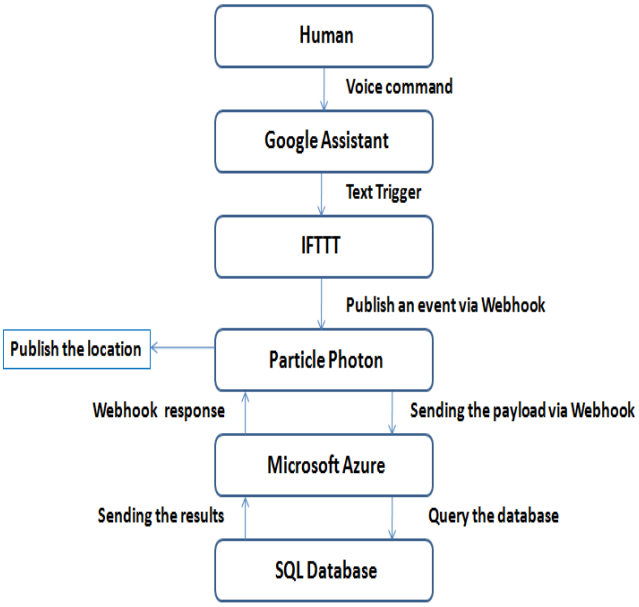

Figure 4: Sequence of Operation

The voice commands used to search for components are listed in the below table.

IFTTT Commands	Description
FindItem	Find the component
FindTags	Finds all the components matching the provided tags
InsertItem	Inserts a new component into the organizer
RemoveItem	Removes a component from the organizer
AddTags	Add tags to an existing component
SetQuantity	Set the quantity of the component to the said value
UpdateQuantity	Updated the quantity of an existing component

C. *SQL Database*

The SQL Database keeps track of each component in terms of names, quantities, locations, search tags. The database is queried to insert, remove or find a component, or update the component details. The SQL database consists of four tables. They are: dbo.Commands, dbo.HttpRequests, dbo.Items and dbo.Tags. The Items table stores the NameKey, Name, Quantity, Row, Column, Date of Creation information and last updated date and time information. NameKey is a unique identifier. The Tags table stores the Component Id, NameKey and the associated Tags for the component. The Commands table stores the Component Id, Date of Creation Information, associated Commands, DataIn and DataOut information. The HttpRequests table stores the component id, name of the published event and the date of creation information. The Commands and HttpRequests tables will show raw data logged from the Azure Function App, which may help debug malformed JSON input or bad speech-to-text translations from Google Assistant.

Figure 5 shows the Items table and figure 6 shows the Tags table. Visual Studio 2019 is used to view the contents of the SQL database.

dbo.Items [Data] | dbo.Tags [Data]

Max Rows: 1000

NameKey	Name	Quantity	Row	Col	IsSmallBox	DateCreated	LastUpdate
10k resistor	10k resistor	1	2	2	True	10-05-2020 14:39:02	10-05-2020
1k resistor	1k resistors	1	0	1	True	10-05-2020 14:20:52	10-05-2020
1mf capacitor	1mf capacitor	1	3	4	True	10-05-2020 14:46:29	10-05-2020
1n5819	1N5819	1	0	5	True	10-05-2020 14:34:32	10-05-2020
680e resistor	680E resistor	1	2	3	True	10-05-2020 14:39:57	10-05-2020
arduino uno	arduino uno	1	2	5	True	07-05-2020 16:33:10	07-05-2020
bolt	bolt	1	3	0	True	10-05-2020 14:41:03	10-05-2020
breadboard	breadboard	1	2	4	True	26-06-2020 11:24:05	26-06-2020
fuse	fuse	1	3	5	True	26-06-2020 07:43:08	26-06-2020
ic base	IC base	1	0	4	True	07-05-2020 16:32:18	07-05-2020
insulation tape	insulation tape	1	3	3	True	10-05-2020 14:45:31	10-05-2020
irf 540 n	IRF 540 N	1	1	3	True	07-05-2020 16:47:15	07-05-2020
irf z44n	IRF Z44N	1	1	0	True	10-05-2020 14:35:37	10-05-2020
jumper	jumper	10	2	1	True	07-05-2020 16:31:08	07-05-2020
ldr	LDR	1	1	1	True	10-05-2020 12:02:41	10-05-2020
lm 35	lm 35	1	1	4	True	10-05-2020 12:39:59	10-05-2020
male header	male header	1	1	5	True	07-05-2020 16:31:30	07-05-2020
nodemcu	NodeMCU	1	3	2	True	10-05-2020 14:44:03	10-05-2020
pir sensor	PIR sensor	1	3	1	True	10-05-2020 14:43:37	10-05-2020
pushbutton	pushbutton	1	0	3	True	07-05-2020 16:30:38	07-05-2020
red led	red led	7	0	0	True	07-05-2020 16:28:50	07-05-2020
rgb led	RGB LED	1	1	2	True	10-05-2020 14:36:28	10-05-2020
seven segment	seven segment	1	0	2	True	07-05-2020 16:29:47	07-05-2020
yellow led	yellow LED	1	2	0	True	08-05-2020 16:01:35	08-05-2020
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 5: SQL Database: Items table.

dbo.Tags [Data] | SQL Server Object Explorer

Max Rows: 1000

Id	NameKey	Tag
79	red led	red
80	red led	led
82	seven segment	seven
83	seven segment	segment
84	breadboard	breadboard
85	pushbutton	pushbutton
86	jumper	jumper
87	male header	male
88	male header	header
89	ic base	ic
90	ic base	base
92	arduino uno	arduino
93	arduino uno	uno
102	irf 540 n	irf
103	irf 540 n	540n
134	red led	diode
135	yellow led	yellow
136	yellow led	led
139	1k resistor	1k
140	1k resistor	resistor
141	1n5819	5819
142	irf z44n	irf
143	irf z44n	z44n
145	rgb led	rgb
146	rgb led	led
148	10k resistor	10k
149	10k resistor	resistor
150	680e resistor	680
152	680e resistor	resistor
153	bolt	bolt
154	pir sensor	pir
155	pir sensor	sensor
156	nodemcu	node
158	nodemcu	mcu
159	insulation tape	insulation
160	insulation tape	tape
161	1mf capacitor	1mf
162	1mf capacitor	capacitor
163	fuse	fuse
164	ldr	ldr
167	lm 35	lm
168	lm 35	35

Figure 6: SQL database: Tags table.

III. OPERATION

The sequence of operation is discussed in this section.

- First, the user gives the voice command. In this case, the voice command is Find NodeMCU.
- This command triggers IFTTT. IFTTT then publishes 2 events: Google_FindItem and callAzureFunctionEvent. Since we are finding an item, Google_FindItem event is generated. The callAzureFunctionEvent is published to Particle Photon.
- Particle Photon then sends the JSON payload to the Azure function app via webhook.
- The function app deserialises the payload, and queries the database.
- The database then sends the results of querying back to function app.
- The function app sends the webhook response to Particle Photon.
- Finally, Particle Photon gives a visual indication of the location of NodeMCU. Figure 7 shows the sequence of operation.

Figure 7: Sequence of operation

IV. APPLICATIONS

The proposed system find applications in the following fields:

1. *In Industries*
2. *In Pharmacies*
3. *In laboratories*

V. ADVANTAGES

- *Efficient management and Storage:* The organizer provides an effective method to store and manage the components, and ensure that the user acquires the desired components by ejecting the right storage units.
- *Easy to incorporate:* The organizer caters to many fields such as industries, laboratories, pharmacies etc.
- *Precision mechanisms and user-friendly:* All that the user requires is a voice assistant to organize the components.
- *Zero-delay:* There is zero delay in seeking the desired components.
- *Easy to handle and operate:* As the system is voice-controlled, the user does not have to put too much work in organizing and managing the components.

- *Collective Documentation:* The component name, quantity and location is maintained in a database and it updates itself when the new components are inserted or when existing components are removed.

VI. DISADVANTAGES

- The proposed system requires it to be connected to the internet when in use.
- The user should have knowledge of the correct tags before using the organizer.

VII. CONCLUSION AND FUTURE WORK

In this paper, it is shown how Smart Component Organizer organizes and manages the components significantly with the use of voice assistant. We believe that the derived smart organizer model is important not only because it helps to improve the efficiency of the operations in the specific domain but it also has good implications to a more general domain. Figure 8 shows the Smart Component Organizer. In future research, we plan to extend our analysis to a more challenging case when the storage space is a decision variable. We also plan to use actuators to push the storage unit of the most promising location identified for the searched component.

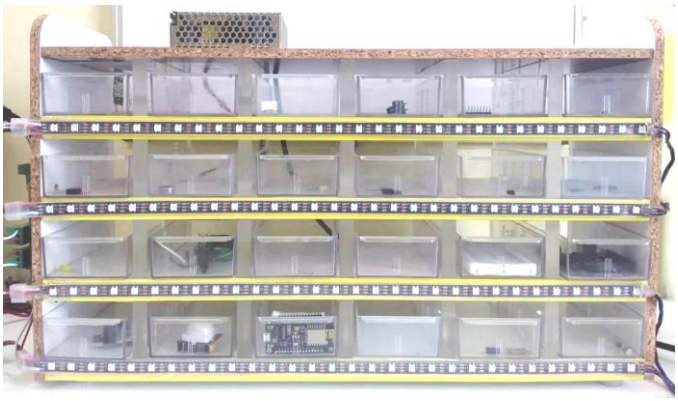

Figure 8: Setup of Smart Component Organizer

ACKNOWLEDGMENT

We consider it is a privilege and honour to express our sincere thanks to Dr. Rajendra R Patil, Professor and Head, Department of Electronics Communication Engineering for his support and invaluable guidance throughout the tenure of this project.

We would like to thank our internal guide Dr. Rajendra R. Patil, Professor and Head, Department of Electronics Communication Engineering and Priyanka P, Assistant Professor, Department of Electronics Communication Engineering for their support, guidance and motivation for the successful completion of this project.

REFERENCES

[1] "Preliminary Study of a Google Home Mini", Min Jin Park, Joshua I James, Journal of Digital Forensics (2018)

[2] Giuseppe Ghiani, Marco Manca, Fabio Paterno, And Carmen Santoro, " Personalization of Context-Dependent Applications Through Trigger-Action Rules ", ACM Transactions on Computer human Interaction (2017).

[3] Pratiksha P. Nikam , Ranjeetsingh S. Suryawanshi, " Microsoft Windows Azure: Developing Applications for Highly Available Storage of Cloud Service ", International Journal of Science and Research (2014).

[4] Shweta Dinesh Bijwe, P. L. Ramteke, " Database in Cloud Computing - Database-as-a Service with its challenges ", International Journal of Computer Science and Mobile Computing (2015).

[5] S.Nagaprasad, & A.VinayaBabu, & K.Madhukar, & Verghese, D.Marlene & V.Mallaiah, & A.Sreelatha, "Reviewing some platforms in cloud computing", International Journal of Engineering and Technology (2010).

[6] Arabolu Chandra Sekhar, Dr. R. Praveen Sam, " Architecture of SQL Azure", International Research Journal of Engineering and Technology (2015)

[7] Particle Photon: https://docs.particle.io/photon/

[8] IFTTT applets: https://ifttt.com/